Insect-Inspired Optic Flow Estimation and Obstacle Avoidance

Thesis Report
July 2022

Yvonne Eggers

Faculty of Aerospace Engineering

TU Delft Delft University of Technology

Challenge the future

Cover Image: House fly red eye and green background (2022) by Amal biju Unni under Unsplash License.
Available at: https://unsplash.com/photos/z-h45uj_fmY

Insect-Inspired Optic Flow Estimation and Obstacle Avoidance

Thesis Report

by

Yvonne Eggers

to obtain the degree of Master of Science
at Delft University of Technology,
to be defended publicly on Wednesday July 13, 2022 at 13:00 PM.

Student number:	4444922	
Thesis committee:	Prof. Dr. G.C.H.E de Croon,	TU Delft, supervisor
	Dr. J. C. van Gemert,	TU Delft
	Dr. E. J. J. Smeur,	TU Delft

TU Delft

Preface

This report summarizes the findings of my MSc thesis within the Control and Simulation department at the Aerospace Engineering faculty at Delft University of Technology. I would like to thank my supervisors Guido de Croon and Julien Dupeyroux for their guidance and support throughout the thesis process. A big thanks also goes to my friends Killian Swannet and Michael Westheim. Our weekly game nights were a welcome distraction from work when things were not going well. A special word of appreciation also goes to Miha Zupanič. Thank you for planning all these little trips that made life a little brighter and for constantly taking some of the load off my shoulders. Finally, I would like to thank my parents for their understanding and patience.

Yvonne Eggers
July 11, 2022

Abstract

Event cameras and spiking neural networks (SNNs) allow for a highly bio-inspired, low-latency and power efficient implementation of optic flow estimation. Just recently, a hierarchical SNN was proposed in which motion selectivity is learned from raw event data in an unsupervised manner using spike-timing-dependent plasticity (STDP). However, real-life applications of this SNN are currently still limited by the fact that the exact choice of neuron parameters depends on the spatiotemporal properties of the input. Furthermore, tuning the network is a challenging task due to the high degree of coupling between the various parameters. Inspired by neurons in biological brains that modify their intrinsic parameters through a process called intrinsic plasticity, this research proposes update rules which adapt the voltage threshold and maximum synaptic delay during inference. This allows applying the already trained network to a wider range of operating conditions and simplifies the tuning process. Starting with a detailed parameter analysis, primary functions and undesired side effects are assigned to each parameter. The update rules are then designed in such a way as to eliminate these side effects. Unlike existing update rules for the voltage threshold, this work does not attempt to keep the firing activity of output neurons within a specific range, but instead aims to adjust the threshold such that only the correct output maps spike. In particular, the voltage threshold is adapted such that output spikes occur in no more than two maps per retinotopic location. The maximum synaptic delay is adapted such that the resulting apparent pixel velocities of the input match those of the data used during training. A sensitivity analysis is presented which illustrates the effects of newly introduced parameters on the network performance. Furthermore, the adapted network is tested on real event data recorded onboard a drone avoiding obstacles. Due to the difficulties in matching the output of the adapted SNN to the ground truth data, quantitative results are inconclusive. However, qualitative results show a clear improvement in both the density and correctness of output spikes. The complementary code for this research can be found here: `https://github.com/tudelft/IP-STDP-FlowNet`.

Contents

Nomenclature

List of Abbreviations

ANN	Artificial Neural Network
ATIS	Asynchronous Time-Based Image Sensor
CFA	Color Filter Arrays
CNN	Convolutional Neural Network
CoG	Center of Gravity
COMANV	Center-Of-Mass Average Nearness Vector
CW	Clockwise
DAVIS	Dynamic and Active-Pixel Vision Sensor
DCMD	Descending Contralateral Movement Detector
DL	Drive Left Population
DMWTA	Decision-Making Winner Take All
DNN	Deep Neural Network
DR	Drive Right Population
DVS	Dynamic Vision Sensor
eDVS	Embedded Dynamic Vision Sensor
EMD	Elementary Motion Detector
exc	Excitatory Speed Population
FAITH	FAst ITerative Half- plane
FCN	Fully Convolutional Network
FoE	Focus of Expansion
FoV	Field of View
FPGA	Field-Programmable Gate Array
IF	Integrate-and-Fire
IMU	Inertial Measurement Unit
IP	Intrinsic Plasticity
LGMD	Lobula Giant Movement Detector
LIF	Leaky Integrate-and-Fire
LST	Lateral Sound Transmitter
LTP	Long-Term Depression
LTP	Long-Term Potentiation

MAV	Micro Aerial Vehicle
meDVS	Miniature Embedded Dynamic Vision Sensor
NAS	Neuromorphic Auditory Sensor
OA	Obstacle Avoidance
OFE	Optic Flow Encoder
OF	Optic Flow
OL	Obstacle Left Population
OR	Obstacle Right Population
RL	Reinforcement Learning
ROI	Region of Interest
RQ	Research Question
sEMD	Spiking Elementary Motion Detector
SNN	Spiking Neural Network
sp	Speed Population
SRM	Spike-Response-Model
SRQ	Sub-Research Question
SSD	Sound Source Direction
STDP	Spike-Timing-Dependent Plasticity
SURF	Speeded Up Robust Features
TTC	Time-to-Contact
UAV	Unmanned Aerial Vehicle
VLNP	Ventrolateral Neuropils
VLSI	Very-Large-Scale-Integration

List of Roman Symbols

$\dot{X}$	Velocity component along x-axis
$\dot{x}$	Retinal velocity along x-axis of retinal plane
$\dot{Y}$	Velocity component along y-axis
$\dot{y}$	Retinal velocity along y-axis of retinal plane
$\dot{Z}$	Velocity component along z-axis
$\hat{t}$	Time of last output spike
I^{ext}	External driving current
p_0	Principal point and origin of retinal plane coordinate system
$t^{(f)}$	Firing time of presynaptic neuron
v_{rest}	Neuron resting membrane potential
$w_{\max}$	Maximum weights
x_0	x-component of focus of expansion

y_0	y-component of focus of expansion
x	Pixel location
A	Observer's rotation along x-axis
a	Parameter controlling time scale of recovery variable
B	Observer's rotation along y-axis
b	Parameter controlling sensitivity of recovery variable
c	Reset value of membrane potential
C (p. 31)	Capacitance
C (p. 6)	Observer's rotation along z-axis
D	Divergence
d	Reset value of recovery variable
E (p. 10)	Difference equation of pixel brightness values between two subsequent frames
E (p. 21)	Excitatory
e(x,y,t)	Event at pixel location (x,y) and time t
f	Focal length
F(**x**)	Brightness values at pixel location **x** in the first of two subsequent frames
F(w)	Dependence of the STDP update on the current synapse weights w
G(**x**)	Brightness values at pixel location **x** in the second of two subsequent frames
h	Disparity vector
I	Inhibitory
I(t)	Neuron current at time t
O	Origin of the coordinate system
p	Projection of point P onto the retinal plane
P (p. 21)	Photoreceptor
P (p. 6)	Location of texture element within the visual field
R	Resistance
R(t)	Output of EMD at time t
S	Summing
t	Time
U	Observer's velocity along x-axis
u	Retinal velocity along x-axis of retinal plane
u(t)	Recovery variable at time t
V	Observer's velocity along y-axis
v	Retinal velocity along y-axis of retinal plane
v(t)	Neuron membrane potential at time t
W	Observer's velocity along z-axis
x	Normalized image coordinate in x-direction
y	Normalized image coordinate in y-direction

List of Greek Symbols

α	Weighing factor
Δt	Time step
Δw	Change in synaptic strength
ϵ (p.31)	Time course of membrane potential upon receiving presynaptic spike
ϵ (p.38)	Elevation
ϵ_b	Brightness constancy constraint
ϵ_c	Smoothness constraint
η (p.20)	Adaptation gain
η (p.31)	Time course of membrane potential after input spike
κ	Variation of membrane potential for external driving current
λ	Time constant
μ (p. 20)	Parameter controlling amount of leakage
μ (p. 38)	Nearness
ν	Normalized velocity of observer
ω	Ventral flow
ϕ	Azimuth
τ	Delay in elementary motion detector

List of Subscripts

+	Long-Term Potentiation
-	Long-Term Depression
hp	High-pass filter
i	i^{th} neuron
j	j^{th} presynaptic neuron
lp	Low-pass filter
x	Along x-axis
y	Along y-axis
z	Along z-axis

List of Superscripts

+	Long-Term Potentiation
-	Long-Term Depression
R	Rotational component
T	Translational component

List of Figures

List of Tables

1 Introduction

Flying robots have the ability to operate in hostile, cluttered environments and hard-to-reach places such as tall structures or isolated locations. This makes them suitable for many tasks that are either too tedious, difficult, or dangerous for human beings. Examples of this include looking for victims in crisis areas (Arnold et al., 2018), inspections of windmills (Shafiee et al., 2021) and bridges (Chen et al., 2014), agricultural field surveying (Christiansen et al., 2017), and predictive maintenance of gas pipes (Duisterhof et al., 2021). These tasks frequently require the employed drones to autonomously perform complex navigational tasks such as take-off and landing, path-planning, and obstacle avoidance, which is commonly done with the help of computer vision. Furthermore, the typically cluttered operating environments pose size restrictions on the flying robots, leading to an increased interest in tiny drones or micro air vehicles (MAVs). However, computer vision applications onboard MAVs are often still limited by the relatively heavy weight, high power consumption, and computational cost of the required hardware and processing algorithms.

To overcome this issue, inspiration can be drawn from flying insects as they perform fast maneuvers in unpredictable, cluttered environments despite their tiny brains that come with highly limited neural and sensory resources. Honey bees, for instance, forage for food for up to 13 km, at high speeds of around 24 km/s, while avoiding obstacles like trees and tussocks encountered during the journey (Srinivasan, 2011). Flying insects are known to primarily depend on visual input to perform navigational tasks, such as landing and obstacle avoidance. In particular, they have been found to utilize optic flow, which quantifies the perceived motion of features in the visual field of an observer. A great effort has been made to mimic this behavior and compute optic flow using imaging sensors. However, the majority of research utilizes conventional frame-based cameras, which provide full pixel arrays with brightness information at a fixed and often low (30 - 60 fps) frame rate. This stands in stark contrast to photoreceptors in flying insects, which react *asynchronously* to *changes* in brightness. Consequently, the use of frame-based cameras inherently limits the temporal accuracy of optic flow estimates. This can lead to motion blur when recording fast-moving objects and redundant information processing in the absence of motion within the visual field.

Inspired by biological retinas, a new class of vision sensors has recently emerged that do not output a sequence of static images, but rather a spike-like stream of data that only contains information about the time, location, and polarity of pixels in which the log-illumination has changed by a certain threshold. These sensors are called Dynamic Vision Sensors (DVS), event cameras or silicon retinas (Lichtsteiner et al., 2008; Posch et al., 2011; Brandli et al., 2014). Event cameras achieve very high temporal resolutions, a wide dynamic range, as well as a low power consumption. This makes them promising for the computation of optic flow onboard MAVs. However, due to the asynchronous nature of these sensors, their output varies greatly from that of conventional cameras. Consequently, established frame-based methods for optic flow computation cannot directly be applied to event-based input. First efforts have already been made to translate frame-based algorithmic methods (Benosman et al., 2011; Benosman et al., 2014; Hordijk et al., 2018), but also methods using artificial neural networks (ANNs) (Zhu et al., 2018) to the event-based domain. Since ANNs mimic neural computations in biological brains, they already represent a step towards bio-inspired optic flow estimation. However, they still synchronously transmit information at each propagation cycle and thus do not fully utilize the high temporal resolution of event-based cameras. This limitation can be overcome with spiking neural

networks (SNNs). Unlike conventional neural networks, they imitate the asynchronous, spike-like information transmission in biological neurons (Pfeiffer and Pfeil, 2018). This way, they can fully benefit from the advantages of event-based vision sensors. Furthermore, they allow for efficient implementation on neuromorphic processors (Furber et al., 2013; Akopyan et al., 2015; Davies et al., 2018; Moradi et al., 2018; Neckar et al., 2019) which directly reflect the structure of neural computations on a hardware level.

However, the complex internal dynamics and discrete nature of spiking neurons increase the difficulty of optic flow estimation, and there are only few end-to-end spiking solutions to date. In these approaches, motion selectivity arises either directly from the chosen architecture of the network (Salt et al., 2020; Milde et al., 2018; Haessig et al., 2018) or through learning which adapts the strength of synaptic connections (Paredes-Valles et al., 2020; Paredes-Vallés et al., 2021). Since learning approaches bear the promise of innovative solutions and higher levels of robustness, they are of great interest for optic flow estimation onboard MAVs. Remembering the efficiency with which flying insects perform navigational tasks, especially the approach presented in (Paredes-Valles et al., 2020) is promising since it provides local and global optic flow estimates in a highly biologically plausible manner. Inspired by Reinhardt detectors (Reichardt and Rosenblith, 1961), which model motion detection in flying insects, it applies delays to several synaptic delays, and training is performed through an unsupervised, bio-inspired learning rule called spike-timing plasticity (STDP). However, currently the choice of neuron parameters still largely depends on the spatiotemporal properties of the input, which inherently limits the range of possible operating environments. Furthermore, the effects of the various parameters on the network performance are coupled, which makes tuning the network a challenging task. To overcome this issue, once again, inspiration can be drawn from nature.

In addition to changing the strength of synaptic connections, neurons in the lobula plate of insects have been shown to adapt their intrinsic parameters through a process called intrinsic plasticity (Borst and Egelhaaf, 1987; de Ruter van Steveninck et al., 1986). Several authors have proposed approaches that utilize intrinsic plasticity to adapt neuron parameters such as the voltage threshold, the membrane potential or the time constant in real time, to keep the spiking activity of neurons within a desired operating range. However, these approaches simply aim to drive the activity of neurons to a desired target value without directly incorporating information about the network performance. This can lead to the amplification of noise within the input signal, and thus to erroneous network outputs. Furthermore, to the best of this author's knowledge, there currently exist no method which adapts the maximum synaptic delay in approaches using multisynaptic connections, such as the one presented in (Paredes-Valles et al., 2020).

1.1. Motivation and Research Question

The goal of this research is to design intrinsic plasticity update rules which adapt the maximum synaptic delay and the voltage threshold within the spiking architecture presented in (Paredes-Valles et al., 2020) in real time. The update rules shall decrease the amount of coupling between parameters to simplify the tuning process. Furthermore, the proposed voltage threshold update rule shall not only consider the frequency but also the correctness of output spikes. This shall increase the robustness of the network and allow applying an existing set of trained weights to a wider range of operating conditions. This is useful since the SNN in question can compute optic flow from raw event data in a fully spiking manner. Consequently, it can provide lightweight, energy-efficient optic flow estimates at a significantly higher temporal resolution than existing approaches. Furthermore, it can potentially be implemented on neuromorphic hardware. Overcoming its current limitations would thus open the doors to high-speed visual navigation onboard MAVs which is currently still limited by weight and power restrictions. With these considerations in mind, this report aims to address the following research question.

> Can intrinsic plasticity reduce the amount of coupling between neuron parameters and increase the robustness of the SNN for optic flow estimation presented in (Paredes-Valles et al., 2020)?

1.2. Structure of this Work

This report is divided into two parts. Part I is a scientific paper presenting the main contributions of the research presented in this work. It can be read as a stand-alone document and contains background information about the most important concepts related to this thesis as well as a summary of related work. Furthermore, it provides a thorough analysis of the neural parameters of the SNN presented in (Paredes-Valles et al., 2020) and a description of the proposed intrinsic plasticity update rules. The paper is concluded with an evaluation of the update rules and recommendations for future work.

While the scientific paper specifically focuses on optic flow estimation, part II focuses more on its practical applications. In particular, it presents a detailed literature review on optic flow estimation in the context of insect-inspired obstacle avoidance onboard MAVs. First, chapter 2 provides a more detailed introduction to optic flow including a mathematical definition and several directly related quantities. In addition, conventional frame-based approaches to optic flow estimation are presented and it is explained why they are not suitable for the use onboard MAVs. To draw inspiration for possible improvements, the working principle of motion detection and obstacle avoidance in insects is explained in chapter 3. Subsequently, in chapter 4 technological advancements are presented which help to make optic flow estimation more power-efficient and resilient. Furthermore, existing optic flow-related engineering applications are presented for each of these technological advancements. In chapter 5 an overview of existing insect-inspired obstacle avoidance methods is provided and the presented literature is synthesized in chapter 6.

I

Scientific Paper

Intrinsic Plasticity for Robust Event-Based Optic Flow Estimation

Yvonne Eggers, Julien Dupeyroux, and Guido C.H.E. de Croon

Abstract—Event cameras and spiking neural networks (SNNs) allow for a highly bio-inspired, low-latency and power efficient implementation of optic flow estimation. Just recently, a hierarchical SNN was proposed in which motion selectivity is learned from raw event data in an unsupervised manner using spike-timing-dependent plasticity (STDP). However, real-life applications of this SNN are currently still limited by the fact that the exact choice of neuron parameters depends on the spatiotemporal properties of the input. Furthermore, tuning the network is a challenging task due to the high degree of coupling between the various parameters. Inspired by neurons in biological brains that modify their intrinsic parameters through a process called intrinsic plasticity, this research proposes update rules which adapt the voltage threshold and maximum synaptic delay during inference. This allows applying the already trained network to a wider range of operating conditions and simplifies the tuning process. Starting with a detailed parameter analysis, primary functions and undesired side effects are assigned to each parameter. The update rules are then designed in such a way as to eliminate these side effects. Unlike existing update rules for the voltage threshold, this work does not attempt to keep the firing activity of output neurons within a specific range, but instead aims to adjust the threshold such that only the correct output maps spike. In particular, the voltage threshold is adapted such that output spikes occur in no more than two maps per retinotopic location. The maximum synaptic delay is adapted such that the resulting apparent pixel velocities of the input match those of the data used during training. A sensitivity analysis is presented which illustrates the effects of newly introduced parameters on the network performance. Furthermore, the adapted network is tested on real event data recorded onboard a drone avoiding obstacles. Due to the difficulties in matching the output of the adapted SNN to the ground truth data, quantitative results are inconclusive. However, qualitative results show a clear improvement in both the density and correctness of optic flow estimates. The complementary code for this research can be found here: https://github.com/tudelft/IP-STDP-FlowNet.

Index Terms—Optic flow estimation, spiking neural networks, spike-timing dependent plasticity, intrinsic plasticity, dynamic vision sensors, computer vision, neuron adaptation

1 Introduction

Flying robots have the ability to operate in hostile, cluttered environments and hard-to-reach places such as tall structures or isolated locations. This makes them suitable for many tasks that are either too tedious, difficult, or dangerous for human beings. Examples of this include looking for victims in crisis areas [1], inspections of windmills [2] and bridges [3], agricultural field surveying [4], and predictive maintenance of gas pipes [5]. These tasks frequently require the employed drones to autonomously perform complex navigational tasks such as take-off and landing, path-planning, and obstacle avoidance, which is commonly done with the help of computer vision. Furthermore, the typically cluttered operating environments pose size restrictions on the flying robots, leading to an increased interest in tiny drones or micro air vehicles (MAVs). However, computer vision applications onboard MAVs are often still limited by the relatively heavy weight, high power consumption, and computational cost of the required hardware and processing algorithms.

To overcome this issue, inspiration can be drawn from flying insects since they perform fast maneuvers in unpredictable, cluttered environments despite their tiny brains that come with highly limited neural and sensory resources. Honey bees, for instance, forage for food for up to 13 km, at high speeds of around 24 km/s, while avoiding obstacles like trees and tussocks encountered during the journey [6]. Flying insects are known to primarily depend on visual input to perform navigational tasks, such as landing and obstacle avoidance. In particular, they have been found to utilize optic flow, which quantifies the perceived motion of features in the visual field of an observer. A great effort has been made to mimic this behavior and compute optic flow using imaging sensors. However, the majority of research utilizes conventional frame-based cameras, which provide full pixel arrays with brightness information at a fixed and often low (30 - 60 fps) frame rate. This stands in stark contrast to photoreceptors in flying insects, which react *asynchronously* to *changes* in brightness. Consequently, the use of frame-based cameras inherently limits the temporal accuracy of optic flow estimates. This can lead to motion blur when recording fast-moving objects and redundant information processing in the absence of motion within the visual field.

Inspired by biological retinas, a new class of vision sensors has recently emerged that do not output a sequence of static images, but rather a spike-like stream of data that only contains information about the time, location, and polarity of pixels in which the log-illumination has changed by a certain threshold. These sensors are called Dynamic Vision Sensors (DVS), event cameras or silicon retinas [7], [8], [9]. Event cameras achieve very high temporal resolutions, a

• *Y. Eggers (MSc student), J. Dupeyroux (supervisor), and G.C.H.E. de Croon (supervisor) are with the Control & Simulation Department at Delft University of Technology, Delft, The Netherlands*

wide dynamic range, as well as a low power consumption. This makes them promising for the computation of optic flow onboard MAVs. However, due to the asynchronous nature of these sensors, their output varies greatly from that of conventional cameras. Consequently, established frame-based methods for optic flow computation cannot directly be applied to event-based input. First efforts have already been made to translate frame-based algorithmic methods [10], [11], [12], but also methods using artificial neural networks (ANNs) [13] to the event-based domain. Since ANNs mimic neural computations in biological brains, they already represent a step towards bio-inspired optic flow estimation. However, they still synchronously transmit information at each propagation cycle and thus do not fully utilize the high temporal resolution of event-based cameras. This limitation can be overcome with spiking neural networks (SNNs). Unlike conventional neural networks, they imitate the asynchronous, spike-like information transmission in biological neurons [14]. This way, they can fully benefit from the advantages of event-based vision sensors. Furthermore, they allow for efficient implementation on neuromorphic processors [15], [16], [17], [18], [19] which directly reflect the structure of neural computations on a hardware level.

However, the complex internal dynamics and discrete nature of spiking neurons increase the difficulty of optic flow estimation, and there are only few end-to-end spiking solutions to date. In these approaches, motion selectivity arises either directly from the chosen architecture of the network [20], [21], [22] or through learning which adapts the strength of synaptic connections [23], [24]. Since learning approaches bear the promise of innovative solutions and higher levels of robustness, they are of great interest for optic flow estimation onboard MAVs. Remembering the efficiency with which flying insects perform navigational tasks, especially the approach presented in [23] is promising since it provides local and global optic flow estimates in a highly biologically plausible manner. Inspired by Reinhardt detectors [25], which model motion detection in flying insects, it applies delays to several synaptic delays, and training is performed through an unsupervised, bio-inspired learning rule called spike-timing plasticity (STDP). However, currently the choice of neuron parameters still largely depends on the spatiotemporal properties of the input, which inherently limits the range of possible operating environments. Furthermore, the effects of the various parameters on the network performance are coupled, which makes tuning the network a challenging task. To overcome this issue, once again, inspiration can be drawn from nature.

Neurons in the lobula plate of insects have been shown to adapt their intrinsic parameters through a process called intrinsic plasticity [26], [27]. Several authors have utilized intrinsic plasticity to adapt neuron parameters such as the voltage threshold, the membrane potential or the time constant, to keep the spiking activity within a desired operating range. However, these approaches simply aim to drive the activity of neurons to a desired target value without directly incorporating information about the network performance. This can lead to the amplification of noise within the input signal, and thus to erroneous network outputs. Furthermore, to the best of this author's knowledge, there currently exist no method which adapts the maximum synaptic delay in approaches using multisynaptic connections, such as the one presented in [23].

This research contains *three main contributions*. First of all, we provide a detailed analysis of the effects of the neuron parameters on the network performance during inference. Based on this analysis, we propose two intrinsic plasticity update rules which adapt the voltage threshold and the maximum synaptic delay within the spiking architecture presented in [23]. The update rules increase the robustness of the network and allow applying an existing set of trained weights to a wider range of operating conditions. Furthermore, they decrease the amount of coupling between the various parameters, which simplifies the tuning process. This is useful since the SNN in question can compute optic flow from raw event data in a fully spiking manner. Consequently, it can provide lightweight, energy-efficient optic flow estimates at a significantly higher temporal resolution than existing approaches. Due to its spiking nature, it also bears the potential for implementation on neuromorphic hardware. Overcoming its current limitations thus opens the door to high-speed visual navigation onboard MAVs, which is currently still limited by weight and power restrictions. Finally, we provide a Python implementation of the network proposed in [23] (originally implemented in C++), which allows for quicker prototyping and easier interfacing with other Python-based SNN libraries.

The remainder of this paper is structured as follows: section 2 provides more detailed background information about event cameras, spiking neural networks, intrinsic plasticity, and optic flow estimation. Subsequently, the SNN from [23], which represents the foundation for this research, is briefly introduced in section 3 and its current limitations are highlighted to better illustrate the motivation behind this paper. In section 4 the analysis of the neural parameters is performed, while section 5 - section 7 introduce the update rules for the voltage threshold and the maximum synaptic delay. To highlight the effects of the newly introduced parameters, a sensitivity analysis is performed in section 8 and finally, the proposed update rules are evaluated on real event sequences in section 9.

2 Background

2.1 Dynamic Vision Sensors

Dynamic vision sensors mimic the working principle of biological retinas. Rather than sampling the visual scene at a fixed frame rate like conventional frame-based cameras, each of their pixels asynchronously generates so-called events in response to changes in the perceived brightness. This makes their working principle similar to vision in flying insects. Event-based vision sensors generate output events whenever the logarithmic change in brightness exceeds a predefined threshold. The changes in image intensity are measured with respect to a reference log-illumination saved during the last event. This information is communicated in the form of an event stream containing information about the location (x, y), the time stamp t, and the polarity (ON/OFF) of the brightness change where ON and OFF events represent increases and decreases in the log-illumination, respectively [28].

2.2 Spiking Neural Networks

Spiking neural networks (SNN), are artificial neural networks (ANN) that imitate the working principle of biological neural networks more accurately than conventional ANNs. While ANNs transmit information at each propagation cycle, SNNs mimic the spike-like information processing in real biological processes by operating asynchronously and in a parallel fashion. This way, information can quickly pass through multiple network layers, resulting in an initial output estimate as soon as the first input spikes arrive. Consequently, neural computations boil down to the timing of the spikes and the identity of the used synapses [14]. This is also referred to as *pseudo-simultaneous* information processing [29], [30] and is highly efficient since computations only have to be performed in the active parts of the network. These properties allow SNNs to fully utilize the benefits of event-based vision sensors, which makes them promising for the use of visual navigation onboard MAVs.

2.2.1 Neuron Models

To simulate the spike-like communication of neurons in SNNs, a multitude of different models have been proposed with varying levels of computational complexity and biological plausibility. Generally speaking, it can be differentiated between *conductance-based* models that strive to quantify the actual electrophysiological processes occurring in real neurons, and *phenomenological* models which simply aim to model the outputs of real neurons. The Hodgkin-Huxley Model [31] e.g. is considered one of the most realistic conductance-based models. While it is not commonly used in engineering applications, it has served as an inspiration for more efficient, practical approaches such as the Morris-Lecar [32] and FitzHugh-Nagumo Model [33]. However, the most frequently used models include the Izhikevich [34], (Leaky) Integrate-and-Fire (LIF) [35] and Spike-Response-Model (SRM) [36].

Despite the large number of available formulations, most of them share the same fundamental working principle. Generally speaking, a generic neuron fulfills three functions. These include receiving signals, integrating them over time, and communicating information to subsequent cells. Individual neurons are connected through so-called *synapses*. Neurons transmitting through the synapses are referred to as *presynaptic*, while the receiving ends are referred to as *postsynaptic*. The strength of synaptic connections is called the *synaptic efficacy* and is often represented in terms of weights that are convolved with incoming spikes. Each neuron integrates the incoming electric signals over time, resulting in the internal state variable called the *membrane potential* $v(t)$. Signals which increase the value of the membrane potential are named *excitatory*, while signals which decrease it are referred to as *inhibitory*. If the membrane potential exceeds a certain voltage threshold v_{th}, the postsynaptic neuron produces an output spike, and the membrane potential is set back to the reset value v_{reset}. Furthermore, the neuron enters a refractory period Δt_{refr} during which incoming spikes do not affect the magnitude of the membrane potential. In the absence of input spikes, the membrane potential decays to its resting value v_{rest}. This process is illustrated in Figure 1 using the LIF model as an example.

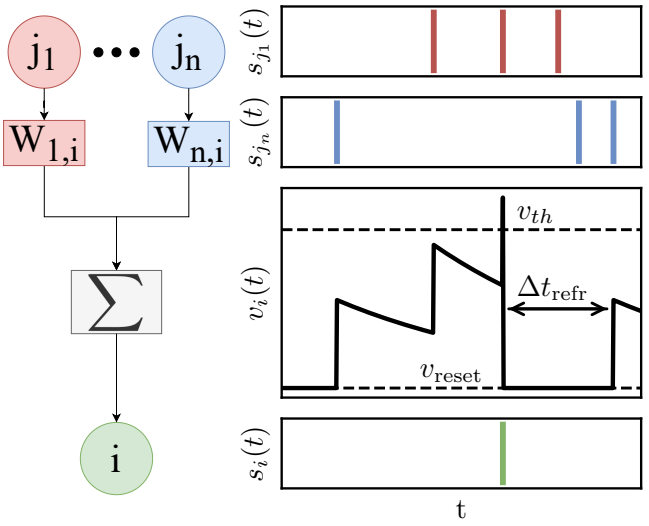


Fig. 1: Illustration of spike propagation in the LIF model. The spike trains s_{j_1} to s_{j_n} of presynaptic neurons j_1 to j_n are multiplied with their respective synaptic weights $W_{1,i}$ to W_{n_i}. The weighted spikes are then integrated over time, which drives changes in the membrane potential $v_i(t)$. Once the membrane potential crosses the threshold v_{th}, the postsynaptic neuron i produces an output spike $s_i(t)$. At the same time its membrane potential is reset to $v_i = v_{\text{reset}}$ and it enters into the refractory period Δt_{refr}. In the absence of input spikes, the membrane potential decays to its resting value v_{rest}.

2.2.2 Synaptic Plasticity

The strengths of synaptic connections are not constant but vary through a mechanism called *synaptic plasticity* which represents the basis for learning and the forming of memories in biological neural networks [37]. To translate this form of learning to engineering applications, a wide range of update rules for the synaptic efficacies have been proposed. These approaches can be divided into supervised and unsupervised methods.

In unsupervised learning, no ground truth is available and there is no notion of specific adaptations being 'good' or 'bad' [38]. Consequently, changes in synaptic efficacy merely emerge locally from the spatiotemporal patterns of the neural input. Synaptic connections between neurons are strengthened or weakened based on the relative timing of their output spikes. This kind of learning is referred to as *Hebbian learning* after Hebb's postulate "neurons that fire together, wire together" [39]. The most commonly used form of Hebbian learning is *Spike-Timing-Dependent Plasticity* (STDP). Using this approach, the synaptic connections are strengthened if presynaptic cells persistently activate nearby postsynaptic cells through a process called *Long-Term Potentiation* (LTP). Reversely, synaptic connections are weakened if postsynaptic spikes persistently occur just before the presynaptic spikes. This process is called *Long-Term Depression* (LTD). A wide range of STDP methods have been proposed, which vary in their exact implementation of the weight update rules. It can e.g. be distinguished between *additive* rules, which only consider the relative

timing of pre- and postsynaptic spikes, and *multiplicative* rules, which also account for the current synaptic efficacy [40]. A comprehensive overview of STDP in SNNs for pattern recognition is provided in [40]. While traditional STDP formulations have already shown great success in computer vision applications, such as image classification or optic flow computation [23], [41], [42], most of them still utilize static training approaches. This means they do no longer update their weights to adapt to new input statistics once the training phase has finished. This makes them less robust to changes in the operating environment. In an effort to overcome this issue, several authors have proposed STDP rules that utilize *controlled forgetting* [43], [44], [45]. This means that weights retain life-long plasticity without the risk of catastrophic forgetting[1] by introducing a leaky term in the traditional STDP formulation.

In supervised learning, the output of the network is compared to a previously established ground truth and the synaptic weights are modified to minimize the difference between the actual and the desired output. However, the discontinuous nature of spiking neuron models makes the translation of commonly used methods, such as backpropagation, to SNNs challenging [46]. A multitude of approaches aim to overcome this issue by performing backpropagation on continuous approximations of the spiking neuron dynamics, which are often referred to as surrogate gradients [47], [48], [49], [50]. Another common strategy is to binarize the activation functions of conventional ANNs for efficient inference [51], [52], [53]. While binarized networks resemble the spike-like nature of SNNs, they maintain synchronous layer-by-layer information processing. However, binarization still leads to a more energy-efficient computation on neuromorphic systems due to the sparse activations. Other approaches avoid the issue of gradient descent in SNNs altogether by performing the training on conventional ANNs and subsequently converting them into SNNs [54], [55], [56], [57]. The obtained SNNs show similar performances as their underlying conventional neural networks in benchmark tests, but require a large number of spikes to accurately model the network dynamics [58], [59]. Finally, reinforcement learning has also been applied to learning in SNNs. While the majority of these approaches focus on reward-modulated STDP [60], [61], there are also practical applications including robotics tasks, such as navigation and obstacle avoidance [62], [63].

2.2.3 Intrinsic Plasticity

In several experiments investigating the dynamic response properties of motion detection in flying insects [26], [27], neurons in the insect brain have been found to show adaptation. Rather than changing the strength of synaptic connections, this form of adaption modifies intrinsic electrical neuron parameters and is thus referred to as *intrinsic plasticity*. It is useful in the context of SNNs since spiking neurons only have a limited range of responses available and modifying the neuron parameters can help in keeping the average activity of neurons within a desired operating range [64].

Several authors have utilized this mechanism to adapt the voltage threshold within SNN architectures. For this purpose, the majority of approaches consider postsynaptic information. In [64] e.g. the threshold is adapted to make LIF neurons spike an average k out of N times or similarly [65] adapt the neuron's excitability in the Izhikevich model to make the output firing activity match a specific goal. Furthermore, the method proposed in [66] utilizes a low pass filter capturing the past output activity of neurons and adjusts the threshold accordingly.

However, experiments performed in [24] have shown that postsynaptic approaches are often too slow to capture the typically rapidly changing input dynamics of event cameras. In [23] this issue is avoided by introducing a presynaptic trace that keeps track of the recent input spike history. By subtracting the presynaptic trace from the neuron current, highly active neurons are penalized. However, this also leads to an overall decrease in spiking activity, which makes this approach less suitable for very sparse inputs. Aiming to combine the best of both worlds, [24] have proposed a crossover model which adapts the voltage threshold based on presynaptic activity. However, their experiments have shown no clear improvements in performance when compared to other neuron models. Moreover, all the models above simply aim to regulate the spiking activity, but do not specifically aim to maximize the information content. [67] showed that the output response of neurons in the visual cortical follows an approximately exponential distribution. Furthermore, they argue that this exponential distribution yields maximal information entropy[2]. Consequently, several authors have proposed neuron models with adaptive parameters aiming to achieve an exponential output distribution. The issue of the discontinuous nature of spiking neurons is circumvented by either directly substituting spiking neuron parameters into an existing model for Sigmoid neurons [68], assuming a constant input firing rate allowing for a continuous formulation of the output firing rate [69], [70], or by employing a soft-reset spiking neuron which resets the neuron membrane potential in a mathematically continuous manner [70].

However, all of these approaches are derived for specific neuron models and cannot easily be translated to other approaches. Furthermore, they only consider the spiking *activity* of neurons and not the *'quality'* of output spikes. This means that they do not directly account for the network performance in the adaptation of the neuron parameters. Finally, to the best of this author's knowledge, there currently exists no update rule which adapts multiple synaptic delays in architectures inspired by Hassenstein-Reichhardt correlators such as the one presented in [23].

2.3 Event-Based Optic Flow Estimation

Optic flow is defined as the vectors describing the displacements of features in the visual field induced by the relative motion between an observer and its environment. Since event cameras bear the promise of low-latency, light-weight, and power-efficient image processing, several event-based approaches to optic flow estimation have been proposed in

1. Catastrophic forgetting refers to the tendency of artificial neural networks to forget previously learned patterns when confronted with new input.

2. Information entropy describes the average level of information or uncertainty contained in a random variable's possible outcomes.

recent years. These approaches can be divided into algorithmic and neural methods. Examples of algorithmic solutions include the gradient-based approach in [10] which translates the well-known frame-based Lucas-Kanade method [71] to event-based applications. Furthermore, several authors utilize plane-fitting algorithms which estimate optic flow from the gradients of spatiotemporal event surfaces [11], [12], [72], [73]. Drawing inspiration from biology, spatiotemporal filters or frequency-based methods are also often used for algorithmic event-based optic flow estimation [74], [75], [76]. Finally, [77], [78], [79] utilize correlation-based methods which predict optic flow by warping event images to identify correspondences between different frames of accumulated events.

With regard to non-spiking neural approaches, the authors from [13] proposed the first convolutional neural network (CNN) which learns optic flow in a self-supervised manner from the photometric error of subsequent grayscale images. They further improved on this approach in [80] where they presented a method that no longer relies on grayscale images as a supervisory signal, but directly utilizes events for this task. In [81] an ANN similar to the one in [13] was proposed. However, this network utilizes the image depth and camera pose to warp successive event slices [82]. Finally, several authors have proposed smaller ANNs with the intention to fully utilize the high temporal frequency of event cameras. [83], [84].

In a step toward an end-to-end spiking implementation of optic flow estimation, the authors in [85], [86] proposed a hybrid network that implements the ANN presented in [13] with spiking neurons in the encoder, while still using conventional neurons in the decoder. Furthermore, numerous fully spiking neural networks have been proposed that mimic specific neuron structures in the insect brain. These include e.g. the elementary motion detector (EMD) [21], [22], [87], [88] and lobula giant movement detector [20], [89]. While these models provide fast optic flow estimates and have also been implemented on neuromorphic hardware, they do not involve learning. To the best of this author's knowledge, there currently only exist two learning SNNs for optic flow estimation. The first one is presented in [83] and consists of a spiking implementations of the network architectures EV-FLowNet [13] and FireNet [90]. The second one is the previously mentioned hierarchical spiking neural network for local and global optic flow estimation presented in [23]. As explained in section 1, this SNN is especially promising since it utilizes local update rules, which makes it highly biologically plausible and thus amenable to on-chip learning. However, currently, this network does not generalize well to input data with spatiotemporal properties different from those encountered during training, which limits its use for practical applications. A more detailed description of this network is provided in the following section.

3 Hierarchical Spiking Neural Network for Optic Flow Estimation

This research aims to increase the robustness of the SNN presented in [23]. Hence, this section provides a brief summary of its key components, architecture and the employed neuron model (for a more detailed explanation, please refer to [23]). Furthermore, an overview of the implementation performed in this research is given and the current limitations of the SNN are highlighted to illustrate why the use of intrinsic plasticity is promising to increase its robustness.

To perform optic flow estimation, the SNN utilizes velocity-tuned filters which are learned in an unsupervised fashion from raw event data using a novel multiplicative STDP learning rule. Motion selectivity arises with the help of several temporal delays, which are applied to multisynaptic connections between neurons and mimic the working principle of insect-inspired Hassenstein-Reichhardt correlators [25]. These delays are denoted as τ and are linearly spaced between the minimum delay $\tau_1 = 1$ ms and a maximum delay $\tau_{\max}$, i.e. $\tau \in [1, \tau_{\max}]$. Furthermore, the proposed SNN makes use of a modified LIF neuron model, which regulates the spiking activity of individual cells by penalizing highly active neurons. The following sections explain the network architecture and adaptive neuron model in more detail.

3.1 SNN Architecture

As illustrated in Figure 2, the SNN comprises of six layers. The Input Layer encodes the event-based data into two retinotopically arranged, two-dimensional neural maps (one each for ON- and OFF-events). Next, the input is filtered by extracting visual features in the Single-Synaptic Convolutional (SS-Conv) Layer. To reduce the number of required convolutional kernels, the ON- and OFF-channels are combined into a single map in the Merge Layer. Subsequently, in the Multi-Synaptic Convolutional (MS-Conv) Layer, local motion estimates of the features extracted in the SS-Conv layer are provided with the help of multisynaptic delays. As a first step towards global motion estimation, the Pooling Layer reduces the spatial dimensionality of the MS-Conv layer. Finally, the Dense Layer, consisting of fully connected neurons, provides a global motion estimate. The SS-Conv, MS-Conv, and Dense layers comprise plastic neuron connections which are trained with the proposed STDP rule. The remaining layers mainly serve to merge information and reduce the dimensionality of the network. Accordingly, they utilize constant weights. As this work primarily focuses on navigational tasks such as obstacle avoidance, which heavily rely on local optic flow, only the first four layers up to the MS-Conv layer providing local motion estimates are considered.

3.2 Adaptive Neuron Model

Since event-based vision sensors show rapidly changing input statistics, an adaptive neuron model was introduced in [23]. It is based on the LIF neuron and accordingly its internal dynamics are characterized by Equation 1, where v represents the membrane potential, λ_v the voltage time constant, v_{rest} the resting membrane potential, and i the forcing function. The subscript $i = 1, 2, ..., n^l$ refers to the i^{th} neuron in the postsynaptic layer l.

$$\lambda_v \frac{dv_i(t)}{dt} = -(v_i(t) - v_{\text{rest}}) + i_i(t) \tag{1}$$

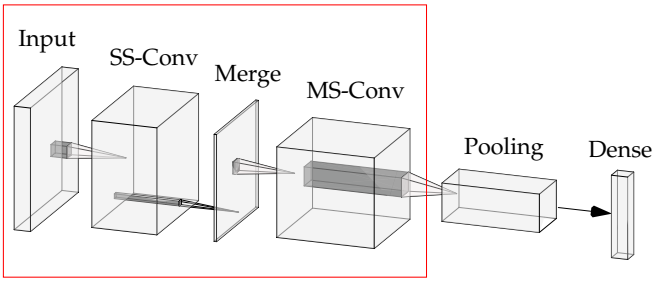


Fig. 2: Illustration of the SNN architecture presented in [23]. The black arrow represents the fully connected synapses and the red box indicates the part of the network considered in this research.

Similarly to the LIF neuron, the modified model produces output spikes once a specific voltage threshold v_{th} has been reached and subsequently enters into a refractory period Δt_{refr} during which incoming spikes do not affect its membrane potential. However, in addition to the weighted input spikes, the forcing function includes a *homeostasis* parameter, penalizing highly active neurons. Denoting the presynaptic neurons from layer $l-1$ as $j = 1, 2, ..., n^{l-1}$ and the delays of the multisynaptic connections as $d = 1, 2, ..., m$, the forcing function is defined as:

$$i_i(t) = \sum_{j=1}^{n^{l-1}} \sum_{d=1}^{m} \left(W_{i,j,d}\, s_j^{l-1}(t-\tau_d) - X_{i,j,d}(t) \right) \tag{2}$$

The synaptic efficacies are denoted by $\mathbf{W} \in \mathbb{R}^{n^l \times n^{l-1} \times m}$ and the binary variable $\mathbf{s}^{l-1}(t-\tau_d) \in \mathbb{R}^{n^{l-1}}$ represents incoming spikes delayed by the synapse specific delay $\tau \in \mathbb{R}^m$. Finally, the homeostasis parameter is defined in terms of the presynaptic trace $\mathbf{X} \in \mathbb{R}^{n^l \times n^{l-1} \times m}$ which quantifies the recent history of transmitted spikes. The temporal dynamics of this parameter are defined in Equation 3, where λ_X represents the time constant of the presynaptic trace, and α a scaling factor regulating the impact of incoming spikes.

$$\lambda_X \frac{dX_{i,j,d}(t)}{dt} = -X_{i,j,d}(t) + \alpha s_j^{l-1}(t-\tau_d) \tag{3}$$

Since the presynaptic trace is subtracted from the weighted input spikes, it reduces the value of the forcing function for highly active neurons to a higher degree than for less active ones and thus regulates the spiking activity. It should be noted that the formulation of the forcing functions for the SS-Conv and MS-Conv layers varies slightly from the general formulation shown in Equation 2. Rather than subtracting the neuron-specific presynaptic trace from the weighted input spikes, the maximum presynaptic trace within the direct neural neighborhood $\mathbf{N}_{i,k}$ of neuron i is considered. This way it is ensured that filters specialize in the visual features themselves rather than their leading edges (please refer to [23] for a more detailed explanation). Denoting the channels in the Input layer as $ch = 1, 2, ..., f^{(0)}$ and the convolutional window size as r, yields the following formulation for the forcing function in the SS-Conv layer:

$$\begin{aligned} i_{i,k}(t) = &\sum_{j=1}^{r} \sum_{ch=1}^{f^{(0)}} W_{j,ch,k} s_{j,ch}^{(0)}(t-\tau) \\ &- \max_{\forall b \in \mathbf{N}_{i,k}} \sum_{j=1}^{r^2} \sum_{ch=1}^{f^{(0)}} X_{b,j,ch}(t) \end{aligned} \tag{4}$$

The forcing function of the MS-Conv layer consists of excitatory weights which are positive at filter locations which correspond to feature edges and inhibitory weights which are negative everywhere else. This way input features that only partially match the excitatory component of the weights are penalized and incorrect outputs spikes are prevented. This is shown in Equation 5 where the superscripts exc and inh refer to the excitatory and inhibitory parts of the weights, respectively, and $\beta \in [0, 1]$ determines the impact of the inhibitory synapses on the weights.

$$\begin{aligned} i_{i,k}(t) = &\sum_{j=1}^{r} \sum_{d=1}^{m} \left(W_{j,d,k}^{\text{exc}} + \beta W_{j,d,k}^{\text{inh}} \right) s_j^{(2)}(t-\tau_d) \\ &- \max_{\forall b \in \mathbf{N}_{i,k}} \sum_{j=1}^{r} \sum_{d=1}^{m} X_{b,j,d}(t) \end{aligned} \tag{5}$$

3.3 Implementation

In this research, the presented SNN was implemented using the SNN deep learning library NORSE [91] and the open-source machine learning framework PyTorch [92]. For training, we purposely chose a single event sequence that only contains constant velocities and no 3D motion. This way, we aim to illustrate that employing intrinsic plasticity allows generalizing generic weights to input data with entirely different spatiotemporal dynamics. Training was performed in a layer-by-layer manner with the event sequence presented in [72]. It comprises a circular disk with eight compartments of different gray levels, rotating at a constant angular velocity. The sequence has a resolution of 240 x 180 pixels and was recorded with a DAVIS DVS [9].

As illustrated in Figure 3, this sensor produces more events in odd-numbered than even-numbered pixel columns. To prevent the occurrence of vertical stripes in the weights, and to reduce the computational load, the input was consequently downsampled by a factor of two for both training and inference. To obtain a more circular distribution of optic flow vectors and thus allow for a better generalization of the weights, the rotating disk was cropped into a square format and rotated by $\pm$ 90 degrees during training. Furthermore, random flips in polarity and spatial orientation were used as data augmentation mechanisms. For evaluation, we use the ODA dataset [93] created by our research team in the Cyber Zoo at TU Delft. It contains more realistic sequences, including 3D motion recorded onboard a drone avoiding indoor obstacles.

Figure 4 illustrates the rotating disk as well as an example from the ODA sequences. Table 1 provides an overview of the parameters used during training and inference. The trained SS-Conv and MS-Conv weights are illustrated in Figure 5 and Figure 6, respectively. For the latter, only the weights corresponding to the first and last

TABLE 1: Overview of parameters used during training (Tr.) and inference (Inf.). The delays are linearly spaced within the range $[1, \tau_{\max}]$ and all simulations were performed with a time step of Δt = 1 ms.

				Rotating disk					ODA sequences		
				SS-Conv		Merge	MS-Conv		SS-Conv	Merge	MS-Conv
	Parameter	Symbol	Unit	Inf.	Tr.	Inf.	Inf.	Tr.	Inf.	Inf.	Inf.
Neural	Voltage threshold	v_{th}	[-]	0.3	0.3	0.001	0.2	0.2	0.2	0.001	0.4
	Resting/reset membrane potential	$v_{\text{rest}}/v_{\text{reset}}$	[-]	0	0	0	0	0	0	0	0
	Presynaptic trace scaling factor	α	[-]	0.3	0.5	-	0.3	0.4	0.3	-	0.3
	Presynaptic trace time constant	λ_X	[ms]	5	5	5	30	30	5	5	30
	Voltage time constant	λ_v	[ms]	5	5	5	30	30	5	5	30
	Refractory period	Δt_{refr}	[ms]	1	1	1	1	1	1	1	1
	Maximum synaptic delay	$\tau_{\max}$	[ms]	1	1	1	200	200	1	1	120
Architecture	Kernel size	r	[-]	5		1	5		5	1	5
	Stride	s	[-]	2		1	2		2	1	2
	Padding	p	[-]	0		0	0		0	0	0
	Number of output maps	f	[-]	32		1	64		32	1	64
	Number of delays	m	[-]	1		1	10		1	1	10
Training	Initialization weights	w_{init}	[-]	0.5		-	$\pm$ 0.5		0.5	-	$\pm$ 0.5
	Learning rate	η	[-]	$5 \cdot 10^{-4}$		-	$5 \cdot 10^{-4}$		$5 \cdot 10^{-4}$	-	$5 \cdot 10^{-4}$
	Convergence threshold	L_{th}	[-]	0.05		-	0.05		0.05	-	0.05
	Scale of inhibitory synapses	β	[-]	-		-	0.5		-	-	0.5
	Weight distribution factor	a	[-]	0		-	0		0	-	0

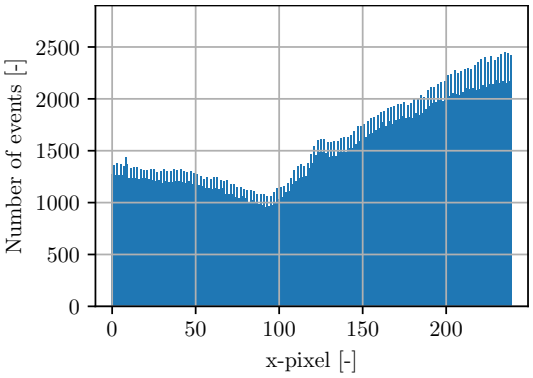
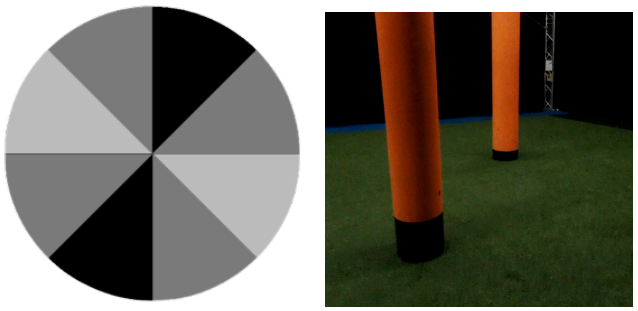


Fig. 3: Number of events occurring along the horizontal axis of the event array during one sequence of the rotating disk recorded with the DAVIS DVS.



(a) Rotating disk [72] (b) Example ODA sequence

Fig. 4: Motives used in the generation of the event sequences employed for training (left) and inference (right).

delay are depicted due to the large number of kernels in this layer. This allows retracing how the respective features move through the delays and thus provides an intuitive representation of the learned motion estimates. The optic flow vectors depicted on the right side of Figure 6 were computed from the MS-Conv weights following the same approach as presented in [23]. In particular, a variation of the histogram matching method EdgeFlow [94] was employed. It provides optic flow estimates by summing the brightness of the weights at two different delays in the horizontal and vertical direction, respectively. Subsequently, a linear curve is fitted to the difference between the resulting histograms of the two chosen delays for both the horizontal and vertical direction. The slopes of the lines fitted to the horizontal and vertical histograms then represent the horizontal and vertical optic flow components, respectively. It should be noted that the resulting estimates are dimensionless and normalized w.r.t. the largest optic flow magnitude. Figure 7 provides an example of the network output for the rotating sequence during inference. Unless specified otherwise, the parameters, units and weights presented in this section are used for all analyses and experiments conducted in this research.

3.4 Limitations and Motivation

While the proposed SNN has been shown to successfully provide optic flow estimates, it is still subject to several limitations. These are explained hereafter to highlight the motivation behind this research.

The first limitation concerns the high degree of coupling between the various parameters and network layers, which makes it very challenging to tune the SNN. Ideally, every parameter should only affect one aspect of the network performance. From now on, we will refer to this as the *primary function*. However, in reality, most parameters affect the inference process in more than one way such that the primary function is overshadowed by other changes which we will refer to as *side effects*. The parameter α e.g. regulates

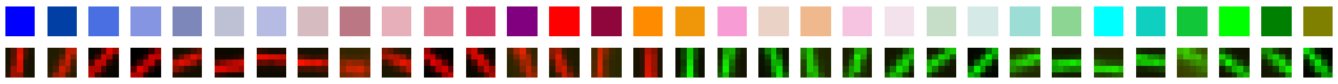
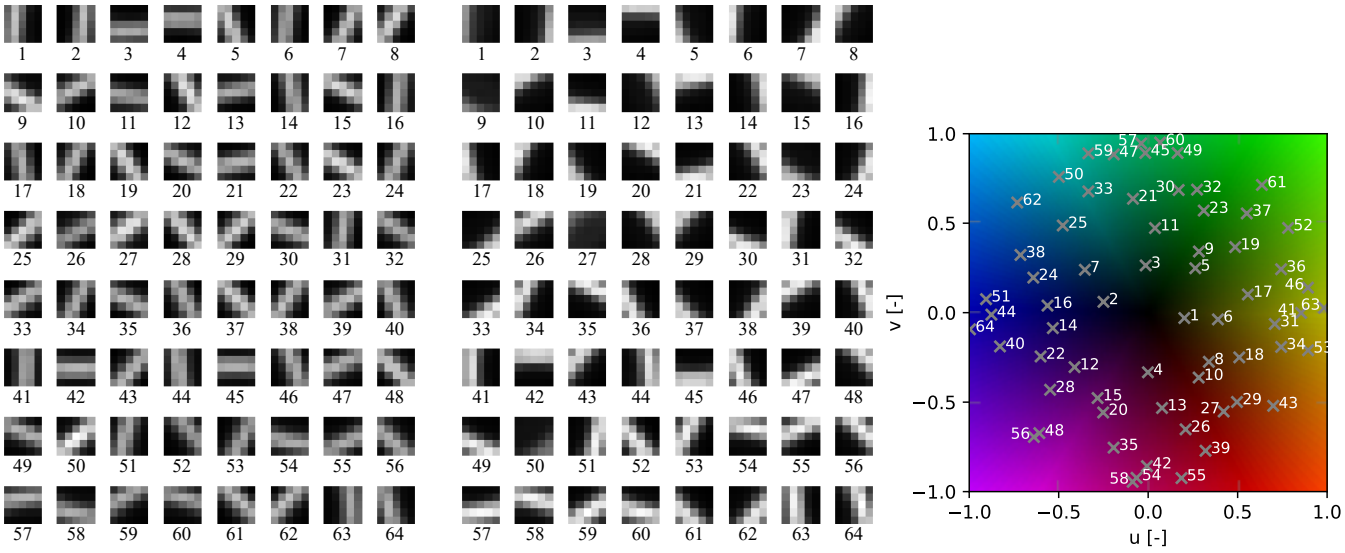


Fig. 5: SS-Conv weights learned from the rotating disk sequence. The first row depicts the color legend used to encode the various output maps and the second row shows the shape of the learned kernels. ON and OFF channels are represented in green and red, respectively.



(a) $\tau = 1$ ms (b) $\tau = 200$ ms (c) Optic flow vectors

Fig. 6: MS-Conv kernels obtained from the rotating disk sequence. The left and middle images show the kernels corresponding to the first and last delay of the weights, respectively. Each square represents one kernel and the kernels are sorted in increasing order of optic flow magnitude from the top left to the bottom right. The right image shows the color legend of the optic flow vectors corresponding to the respective kernels. The directions and magnitudes are encoded in hue and brightness, respectively, and all vectors are normalized w.r.t. the vector with the largest magnitude.

the spiking activity by penalizing the forcing function of neurons with large presynaptic traces. Consequently, its main function could be defined as regulating the difference in spiking activity between various neurons. However, this effect is overshadowed by the overall increase/decrease in the number of output spikes that occurs, since the forcing functions of all active neurons are penalized. This means that the actual effect of changing α can only be seen if the voltage threshold is also adjusted accordingly. In addition, the tuning of the various layers is coupled. This means that any changes made in the first layers of the network require that subsequent layers are re-tuned.

The second limitation concerns the dependency of the parameters on the spatiotemporal properties of the input data. A first indicator for this issue is the fact that different sets of parameters are needed for different kinds of input data, as can be seen in Table 1. Furthermore, optic flow is encoded in discrete maps whose magnitude is scaled by a constant value of the maximum synaptic delay. This greatly limits the range of optic flow magnitudes that can be captured by the network. Ideally, it should be possible to use one set of weights and parameters for a wide range of operating conditions. However, as will be shown in section 9, the MS-Conv layer provides poor local motion estimates when simply applying the rotating disk parameters to the ODA sequences. While this could mean that the rotating disk weights are insufficient to capture the more complex dynamics of the ODA sequences, simply adjusting the inference parameters already greatly improves the optic flow estimates. This implies that one set of weights can be used for a wider range of input data if the inference parameters are adjusted accordingly. It can be concluded that both the issue of the high degree of coupling between the parameters and the parameter's dependency on the input can be overcomed by adjusting the intrinsic neuron parameters. This process corresponds exactly to the concept of intrinsic plasticity introduced in subsubsection 2.2.3.

4 Parameter Analysis

To establish parameter update rules, it is crucial to understand how the various parameters affect the network performance. Consequently, this section presents a detailed parameter analysis. For this purpose, we perform test runs using the rotating disk data and slightly vary one parameter at a time. This way, we aim to identify the primary functions and side effects of each parameter, as well as their dependencies on the input. In doing so, we focus on *neural* parameters (as opposed to parameters related to *training* or the network *architecture*). The obtained insights are then synthesized to evaluate how they can be utilized to propose intrinsic plasticity update rules.

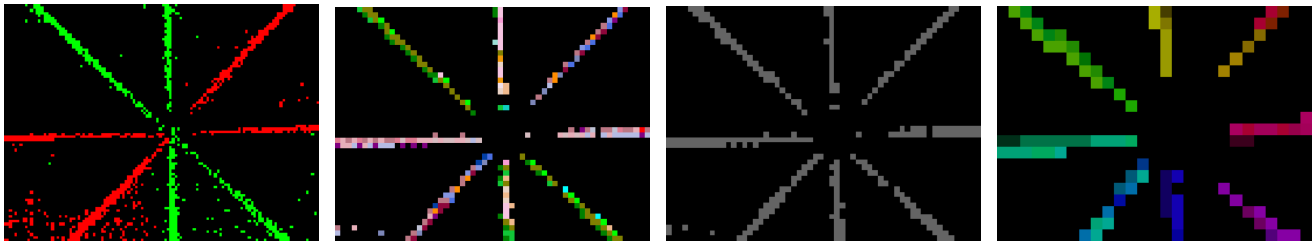


Fig. 7: Example of the outputs within the several layers of the SNN for the rotating disk sequence. ON and OFF events within the Input layer are represented in green and red, respectively. The colour encoding schemes for the SS-Conv and MS-Conv layers are shown in Figure 5 and Figure 6, respectively.

4.1 Analysis

For the analysis, we consider the output of the MS-Conv layer, since the effect of the parameter $\tau_{\max}$ can only be observed there. However, please note that the results can also be applied to the SS-Conv layer. Figure 8 provides an overview of the MS-Conv output using the input depicted in Figure 7 for varying values of the network parameters. Each of these parameters will be discussed in the following subsections.

4.1.1 Voltage Threshold

The first row of Figure 8 shows that the output becomes noisy for low values of the voltage threshold, v_{th}. This comes as no surprise, since fewer input spikes are required to make the neurons in the subsequent layer spike. As a result, even the noise in the input is sufficient to drive the membrane potential over the voltage threshold. In contrast, not all moving features are captured if the threshold is too high. Considering that the disk is rotating clockwise and comparing the output colors for $v_{\text{th}} = 0.1$ to the color legend depicted in Figure 6, it can also be seen that choosing a threshold that is too low results in output spikes in maps that do not correctly represent the motion of the input features. Consequently, the voltage threshold's primary function is to regulate how closely the presynaptic trace has to match the weights to produce output spikes. Furthermore, it appears that the primary function of the voltage threshold is not overshadowed by any other side effects. This means that adjusting the threshold directly yields the expected impact on the output without the need to adjust any of the other parameters. The voltage threshold does, however, depend on the input. Inputs with higher spike frequencies require higher thresholds and vice versa, which means that the threshold has to be adjusted to the dynamics of the input.

4.1.2 Presynaptic Trace Scaling Factor

As explained in subsection 3.4 the primary function of the presynaptic trace scaling factor α is to balance out differences in the spiking activity. Looking at the input depicted in Figure 7, it can be seen that the neurons in the center of the image spike less frequently than those at the edges. We would expect this difference to decrease for increasing values of α. However, in the second row of Figure 8, it can be seen that this is not the case. Instead, the overall number of spikes decreases. This happens since an increase in α leads to an increase in the presynaptic trace, which in return reduces the value of the forcing function and thus also the membrane potential, which ultimately leads to a decrease in the number of output spikes. Consequently, the balancing influence of α is overshadowed by the side effect of an overall decrease in spikes. To counteract this effect and achieve the desired result, the voltage threshold needs to be adjusted accordingly. Furthermore, the value of α depends on the range of spiking activity in the input. Inputs with a wide range of firing frequencies require large values of α to reduce the activity of frequently spiking neurons. However, applying the same large value of α to an input with a smaller range of spike frequencies decreases the activity of the more active neurons drastically. In fact, their spiking frequency can reduce so much that they become less active than the previously less frequently spiking neurons. This effect can even drive the neuron activity below the noise level.

4.1.3 Presynaptic Trace Time Constant

Figure 9 depicts the schematic solution to a differential equation equivalent to the ones shown in Equation 3 and Equation 1 for varying magnitudes of the time constant λ. It can be seen that changing this parameter has two effects: larger values of λ slow down the decay of the output and lead to an overall decrease in its magnitude, while the reverse is true for smaller values of λ. Since changes in the output magnitude can simply be compensated for by adjusting the voltage threshold, we propose that the primary function of λ_X, is to control how quickly the presynaptic trace decays. For a time constant that is too small for the temporal dynamics of the input, the presynaptic trace fully decays before the arrival of the next spike, similarly to the graph corresponding to the small time constant in Figure 9. This means that only one spike at a time can contribute to the growth of the presynaptic trace, such that frequently spiking neurons cannot actually be penalized. For a time constant that is too large, on the other hand, the presynaptic trace decays so slowly that even input spikes which have occurred a long time ago still contribute to an overall increase in the presynaptic trace. This phenomenon can also be observed in Figure 9, where the graph corresponding to a large λ value has not fully decayed yet by the time of the arrival of the fourth input spike. As a consequence, the output is driven to a value larger than the one after the first spike. This means that previous spikes still contribute to increasing the output magnitude, despite the long break

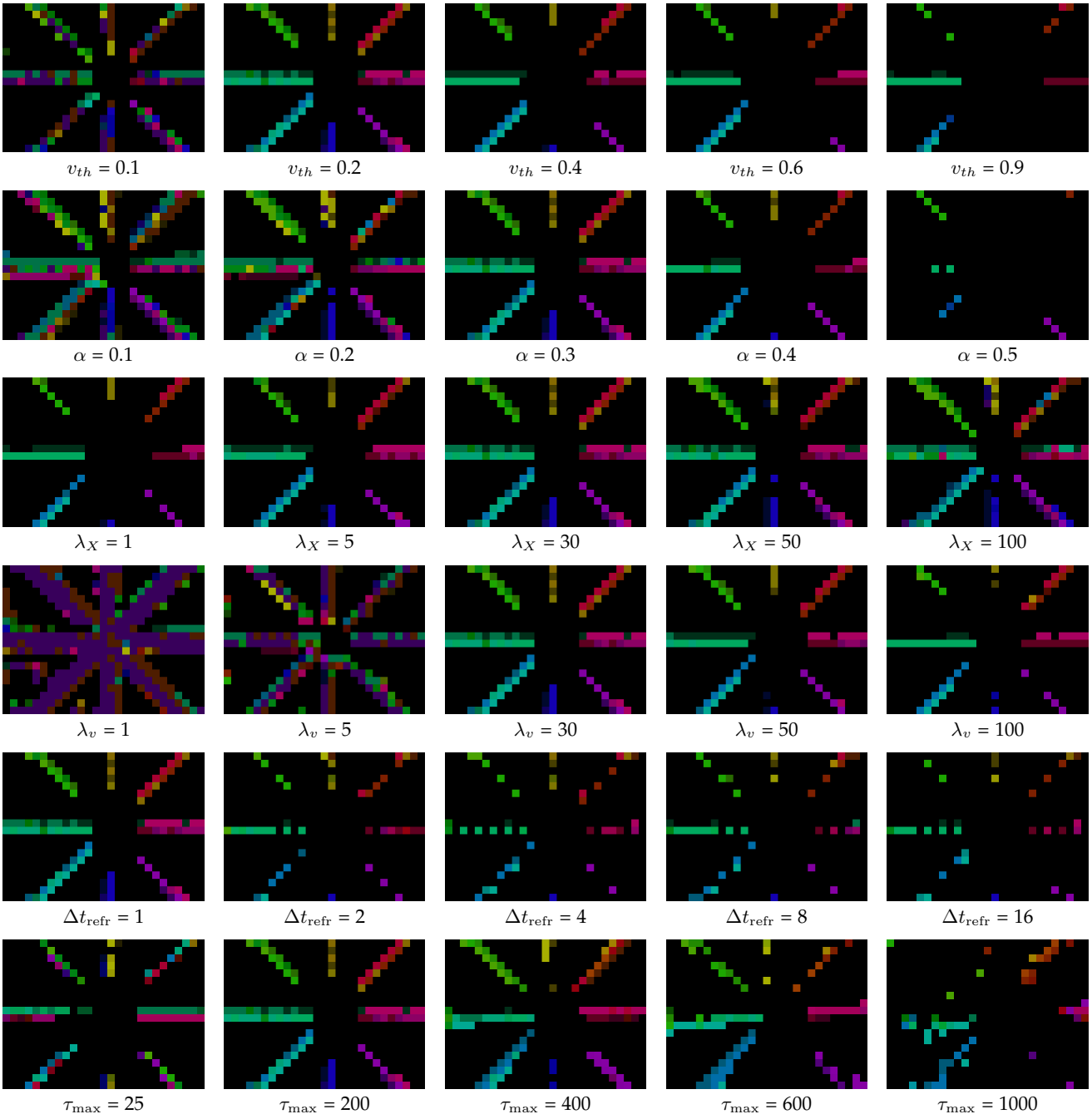


Fig. 8: MS-Conv output for the rotating disk for varying values of the network parameters.

between spikes s_3 and s_4. For the presynaptic trace this would mean that spiking neurons are penalized too much, leading to an overall decrease in spiking activity. However, this effect cannot be observed in the third row of Figure 8. Instead, the overall spiking activity increases for increasing values of λ_X. The reason for this becomes clear when remembering that smaller values of λ_X lead to larger presynaptic trace magnitudes, which in return leads to smaller values in the forcing function and thus fewer output spikes. Furthermore, inputs with varying spike frequencies require different values of λ_X which means that this parameter also depends on the spatiotemporal properties of the input.

4.1.4 Voltage Time Constant

Similarly to the presynaptic trace time constant λ_X, the primary function of the voltage time constant λ_v, is to control how quickly the membrane potential decays. In particular, the membrane potential decays more slowly in the absence of input spikes when increasing λ_v and vice versa. As explained in subsubsection 4.1.3, this increases the maximum amount of time between two spikes, during which the second spike still contributes to an increase in the membrane potential. Consequently, we would like to see the occurrence of output spikes produced by less frequently spiking neurons when increasing λ_v. However, increasing

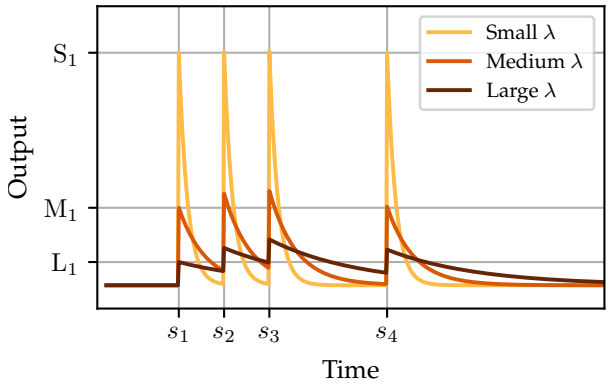


Fig. 9: Schematic representation of the solution to a generic differential equation of the same form as Equation 1 and Equation 3. The output is presented for three different magnitudes of the time constant λ and the parameters s_1 - s_4 represent the time of input spikes. For easier comparison, the magnitudes of the output at the time of the first input spike s_1, are indicated as S_1, M_1 and L_1 for the small, medium, and large time constant, respectively.

the time constant also leads to the side effect of an overall decrease in the membrane potential, which reduces the number of output spikes. This effect can be seen in the fourth row of Figure 8. Consequently, the voltage threshold would have to be lowered for this effect to become visible. Similarly to the presynaptic trace time constant, the spiking frequency of the input needs to be taken into account when tuning λ_v.

4.1.5 Refractory Period

The primary function of the refractory period Δt_{refr} is to temporally separate output spikes by controlling for how long neurons remains inactive after firing. If the duration of this period increases, the density of output spikes is expected to decrease since at each moment in time, neurons that have previously spiked are still in the refractory period and are thus unable to produce more output spikes. Looking at the third row of Figure 8 it can be seen that the density of output spikes does indeed decrease for increasing values of Δt_{refr} and that there are no side effects coupling this parameter to other parameters.

4.1.6 Maximum Synaptic Delay

Looking at the last row of Figure 8, it can be seen that the width of the MS-Conv output increases for increasing values of the maximum synaptic delay $\tau_{\max}$ (the remaining delays are adjusted such that they are still linearly spaced within the range [1, $\tau_{\max}$]) and that the relative number of spikes in the center increases while it decreases at the edges. Furthermore, maps representing local motion estimates of the same orientation but in the opposite direction start spiking for very small values of $\tau_{\max}$. The increased width can be explained by the fact that inputs contributing to the generation of an output spike reach further into the past, such that spikes can also occur at locations further away from the leading edge of features. If the maximum delay used during inference is smaller than the delay that was used during training, smaller feature displacements are mapped onto the same weights. This means that motion, which has previously triggered spikes in maps corresponding to larger optic flow vectors, now causes spikes in maps corresponding to slower motion. As a result, only the parts of the image with the fastest motion still have sufficient overlap with the weights to produce output spikes. The reverse is true for maximum delays larger than the one used during training. This explains why the number of spikes in the center, corresponding to slower image velocities, increases for increasing maximum delays while it decreases in the outer parts. Finally, the use of very small delays results in features that do effectively not move across the different delays within the presynaptic trace. This means that they trigger output spikes in maps that represent features of the same orientation but moving in opposite directions. This effects can be seen in the last row of Figure 8 for a maximum synaptic delay of $\tau_{\max}$ = 25 ms.

4.2 Evaluation

Table 2 provides an overview of the identified primary functions, side effects, and dependencies on the input for each parameter. It can be seen that the side effects of three parameters consist of changing the overall spiking activity. Furthermore, the choice of all parameters but the maximum synaptic delay depends on the spike frequency of the input. Since adjusting the voltage threshold can compensate for very high or low spiking activities, v_{th} appears to be a promising candidate for the adaption during inference. While adjusting the threshold cannot decrease the dependency on the input for presynaptic parameters (v_{th} does not affect the presynaptic trace), it can counteract the undesired side effects and thus decrease the amount of coupling. Consequently, we aim to replace the voltage threshold with a new parameter that does not depend on the dynamics of the input and whose value can thus be kept constant for a wide range of different inputs. The voltage threshold will then be adapted during inference such as to achieve this constant value of the new parameter. The synaptic delays represent the centerpiece of motion selectivity in the investigated SNN. However, the analysis in this section has shown that a suitable value of $\tau_{\max}$ is crucial for the proper functioning of the network. Since the magnitude of this value also depends on the spatiotemporal properties of the input, we follow the same approach as for the voltage threshold and introduce a new parameter for which this is not the case. We will then adapt the maximum synaptic delay such as to achieve a previously established constant target value of the new parameter.

5 Intrinsic Plasticity Update Rules

In the previous section, it was shown that the majority of neuron parameters are coupled to the voltage threshold and that a suitable value for the maximum synaptic delay is crucial for the correct functioning of the SNN presented in [23]. Consequently, we propose update rules for these two parameters in the following subsections.

TABLE 2: Summary of parameter analysis

Parameter	Primary function	Side effects	Dependency on input
v_{th}	Regulates how closely the presynaptic trace must match the weights to produce output spikes	-	Spike frequency of input
α	Balances differences in spiking activity between neurons	Decreases overall spiking activity	Range of spike frequencies in input
λ_X	Controls the rate of decay in the presynaptic trace	Increases overall spiking activity	Spike frequency of input
λ_v	Controls the rate of decay in the membrane potential	Decreases overall spiking activity	Spike frequency of input
Δt_{refr}	Controls the temporal separation of output spikes	-	Spike frequency of input
$\tau_{\max}$	Controls the temporal spacing between presynaptic trace samples	Increases the number of active output maps for small values of $\tau_{\max}$	Pixel velocity of input

5.1 Voltage Threshold Update Rule

In section 4 we established that tuning the voltage threshold can effectively increase or decrease the number of output spikes and can thus help regulate the spiking activity of output neurons. As shown in subsubsection 2.2.3 this effect has been utilized by several authors to propose intrinsic plasticity update rules. However, these approaches do not directly account for the correctness of the network output. To fill this gap, we propose an update rule which does not explicitly regulate the spiking activity, but instead, aims to control to what degree the presynaptic trace has to match the filters to produce output spikes. For this purpose, we introduce a new parameter called the *synaptic stiffness* S which quantifies in how many different maps the same input produces output spikes. Unlike the voltage threshold, it does not depend on the spatiotemporal properties of the input and can thus be set to a constant value. This will be explained in more detail in section 6. We then propose the following update rule, which adapts the voltage threshold at neuron i such as to achieve a previously established target value S^{tar} of the synaptic stiffness.

$$\begin{aligned}\Delta v_{th_i}(t) = \eta_{v_{\text{th}}}\Big((1-c_i(t))(1-\lambda_{v_{\text{th}}})(v_{\text{th}_{\text{rest}}} - v_{\text{th}}(t)) \\ -c_i(t)\lambda_{v_{\text{th}}} v_{\text{th}}(t)\frac{1}{\sum c_k(t)}\sum_{k=1}^{N_{th}} c_k(t)(S^{\text{tar}} - S_k(t))\Big)\end{aligned} \tag{6}$$

The update rule consists of a term decreasing the voltage threshold in the absence of output spikes (first term of Equation 6) and a term increasing it when there are output spikes (second term of Equation 6). The latter is computed from the difference between the actual and the target value of the synaptic stiffness averaged over a neural neighborhood of size $N_{v_{\text{th}}}$. In the average calculation, only the synaptic stiffness of spiking neurons is taken into account. Whether a neuron is spiking is indicated with the parameter c_k which takes on the value $c_k = 1$ for spiking neurons and $c_k = 0$ for non-spiking neurons. Updates are only performed on neurons that have at least one spiking neuron within the averaging neighborhood. This is quantified by the parameter c_i which takes on the value $c_i = 1$ if at least one neuron within the neighborhood is spiking and $c_i = 0$ otherwise. Choosing a window equal to the image size results in a single voltage threshold update for all neurons, while choosing a window size of $N_{v_{th}} = 1$ results in a local rule with independent updates for all neurons. While the voltage threshold can vary across different retinotopic locations of the image, it is constant across different maps since the quantification of the synaptic stiffness is not map-specific (see section 6). Since the proposed update rule translates errors in the dimensionless synaptic stiffness to changes in the voltage threshold, we multiply it with the current value of the voltage threshold to avoid the need for gain scheduling. In the absence of output spikes within the averaging window ($c_i = 0$), the voltage threshold is driven to a small resting value $v_{\text{th}_{\text{rest}}}$. The ratio between decay in the absence of spikes and increases in the presence of spikes is regulated with the parameter $\lambda_{v_{\text{th}}}$. Finally, $\eta_{v_{\text{th}}}$ represents the learning rate of the update rule.

5.2 Maximum Synaptic Delay Update Rule

In section 4 it was shown that an existing set of spatiotemporal filters can only produce meaningful optic flow estimates if the spatiotemporal properties of the input are similar to the ones of the data used during training. To overcome this limitation, we propose an update rule which adapts the maximum synaptic delay $\tau_{\max}$. Since the delays are linearly spaced within the range $[1, \tau_{\max}]$, this increases/decreases their temporal spacing and effectively slows down/speeds up the input. By choosing an appropriate value for $\tau_{\max}$, this way the presynaptic trace of the input can be matched to the shape of the existing synaptic weights. This process is illustrated in Figure 10. On the left-hand side, the history of the schematic presynaptic trace of a bar moving vertically through space is depicted for varying pixel velocities. On the right side, a schematic representation of an existing spatiotemporal filter is shown. It can be seen that the presynaptic traces can be matched to the existing filter by increasing the maximum delay for the slowly moving bar and decreasing it for the fast bar. This way, an existing set of weights can still be used on input data with spatiotemporal properties different from those of the training data. With this adaptation, motion estimates are no longer only encoded in the identity of the spiking output maps, but also in the current value of $\tau_{\max}$.

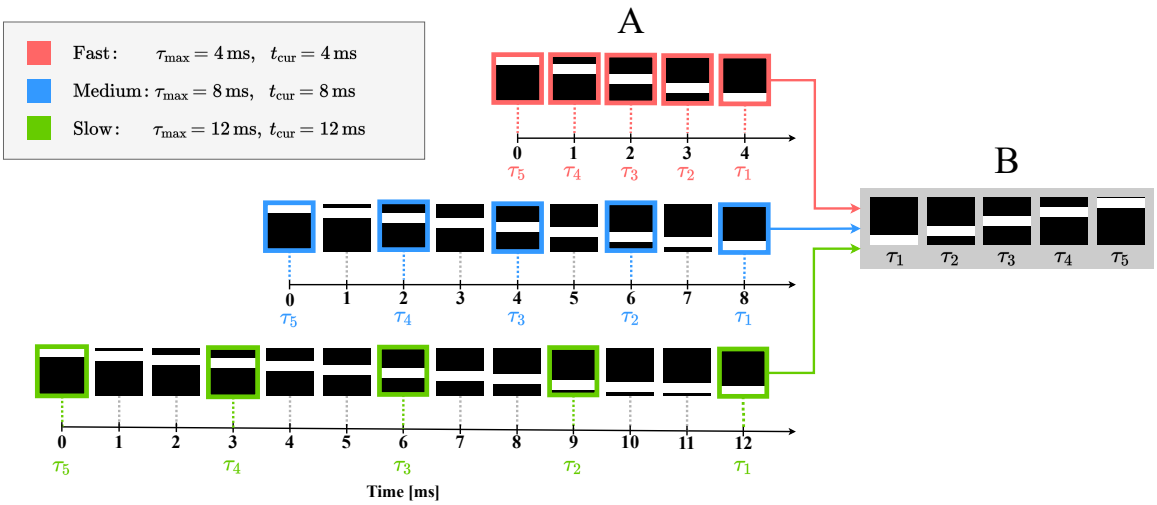


Fig. 10: Illustration of the suggested delay adaptation for one established spatiotemporal filter with $m = 5$ delays and inputs with varying image velocities. A: schematic representation of the presynaptic trace of a bar moving vertically through space at varying pixel velocities. Larger maximum delays are applied to slower motion and vice versa to achieve the same apparent motion across the delays. The colorful boxes highlight the linearly spaced delays $\in [0, \tau_{\max}]$. The presynaptic traces are compressed/stretched out in time to match the existing spatiotemporal filter (B). The current time t_{cur}, i.e. the time relative to which the delays are applied, is set to the time at which the bar is about to leave the respective windows. B: schematic representation of an established spatiotemporal filter to be matched. It should be noted that the order of the windows appears to be reversed, since τ_1 corresponds to the smallest delay and thus to the most recent presynaptic trace appearing at the latest time step.

To adapt the maximum synaptic delay we introduce a new parameter, the *relative optic flow magnitude* A, which is proportional to the product of the actual optic flow magnitude and the maximum synaptic delay, i.e.

$$A = \left(\sqrt{u^2 + v^2}\right) \cdot (\tau_{\max} - \tau_1) \tag{7}$$

where u and v represent the horizontal and vertical components of the optic flow vectors, respectively. The relative optic flow magnitude is an indicator for how far a feature travels across the window of a spatiotemporal filter during the time period $\Delta t = \tau_{\max} - \tau_1$. As shown in Figure 10 features moving at different optic flow magnitudes can have the same *relative* optic flow magnitudes if they utilize different maximum delays. Similarly to the synaptic stiffness, the relative optic flow magnitude does not depend on the properties of the input and can thus be kept at a constant value. To match the dynamics of any input signal to a set of existing weights, we thus propose a simple controller which drives the relative optic flow magnitude A at neuron i to a target value A^{tar}. This results in the following update rule for the change in the maximum synaptic delay $\tau_{\max}$:

$$\Delta\tau_{\max_i}(t) = c_i(t)\eta_{\tau_{\max}}\tau_{\max_i}(t)\frac{1}{\sum c_k(t)}\sum_{k=1}^{N_{\tau_{\max}}} c_k(A_i^{\text{tar}} - A_i(t)) \tag{8}$$

The overall structure of this update rule is equivalent to the one proposed for the voltage threshold. However, since this rule does not rely on output spikes, the decaying term is not required. The update is computed based on the difference between the current relative optic flow magnitude A and the target value A^{tar} averaged over a window of size $N_{\tau_{\max}}$. Since the true optic flow is not available within the network the actual value of A cannot directly be computed. Consequently, we will present several approaches to approximate it in section 7. Choosing a window equal to the image size results in a single $\tau_{\max}$ update for all neurons, while choosing a window size of $N_{vth} = 1$ results in a local rule with independent updates for all neurons. While the maximum synaptic delay can vary across different retinotopic locations of the image, it is constant across different maps. Similarly to Equation 6, the parameter $c_k \in [0, 1]$ indicates whether the presynaptic trace of neuron k is active and the parameter $c_i \in [0, 1]$ indicates if at least one of the presynaptic traces within the averaging window is active. Again, updates are only performed at neurons for which $c_i = 1$ and only neurons with active presynaptic traces are accounted for in the averaging computation ($c_k = 1$). Furthermore, the update is multiplied with the current value of $\tau_{\max}$ to compensate for the mismatch in units between errors in the relative optic flow magnitude (as will be shown in section 7 the proposed approximations of A are dimensionless) and increments in the maximum synaptic delay. Finally, $\eta_{\tau_{\max}}$ represents the learning rate.

6 Quantifying the Synaptic Stiffness

Given a set of distinct weights, the number of maps in which the same input produces an output spike can be controlled by adjusting the voltage threshold. If the voltage threshold is set to a very low value, even a very small amount of overlap between the presynaptic trace and the weights of

any map will be sufficient to drive the membrane potential of the corresponding neurons over the spiking threshold. Therefore, the same input will produce output spikes in a number of different maps. However, this implies that not all of the resulting output spikes actually classify the properties of the input well. The opposite is true if the voltage threshold is set to a very large value. Only if the input matches the filters extremely well, does the membrane potential increase enough to produce an output spike. Consequently, adjusting the voltage threshold does not only allow regulating the neuron spiking activity, but also the accuracy of the network output. Furthermore, the number of maps in which the same input produces output spikes is a direct indicator of this accuracy.

An example of this is shown in Figure 11. It depicts the number of maps in which the same input produces output spikes for varying values of the voltage threshold during one sequence of the rotating disk data. It can be seen that there is indeed a direct correlation between the number of spiking output maps and the voltage threshold. Hence, we use this quantity to define the synaptic stiffness S. Given a distinct set of fully trained weights, we propose that spiking neurons increase or decrease their voltage threshold until output spikes occur in on average $S = S^{\text{tar}}$ output channels within their neural neighborhood of size $N_{v_{\text{th}}}$. We do this, to assure that only those velocity-tuned filters, which correctly represent the motion within the image, are spiking. Using this idea, the complete update rule for the voltage threshold can then be defined as

$$\Delta v_{th_i}(t) = \eta_{v_{\text{th}}}\Big((-c_i(t))(1-\lambda_{v_{\text{th}}})(v_{\text{th}_{\text{rest}}} - v_{\text{th}}(t)) \\ -c_i(t)\lambda_{v_{\text{th}}} v_{\text{th}}(t)\frac{1}{\sum c_k(t)}\sum_{k=1}^{N_{th}} c_k(t)(S_k(t) - S^{\text{tar}})\Big) \tag{9}$$

$$S_k = \sum_{ch=1}^{f^{(l)}} s^l_{k,ch}(t - \tau_d) \tag{10}$$

where $s^l_{k,ch} \in [0,1]$ indicates whether a spike has occurred in map number ch at neuron location k. It should be noted that small values of the maximum synaptic delay also increase the number of maps in which output spikes occur (see subsubsection 4.1.6). Consequently, it is also coupled to the synaptic stiffness, and it cannot necessarily be distinguished whether increases in S have been caused by low values of v_{th} or low values of $\tau_{\max}$. While this could theoretically lead to errors in the v_{th} updates, the maximum synaptic delay is also adapted to a suitable level. Thus, this issue is only prevalent in the transient time until the value of $\tau_{\max}$ has converged. In the design of the $\tau_{\max}$ update rule, it is then however crucial to avoid coupling with the voltage threshold.

7 Quantifying the Relative Optic Flow Magnitude

In this section, we present several approaches to quantify the relative optic flow magnitude introduced in subsection 5.2. While not all of them have proven fruitful, we still include them hereafter for future reference. The explored

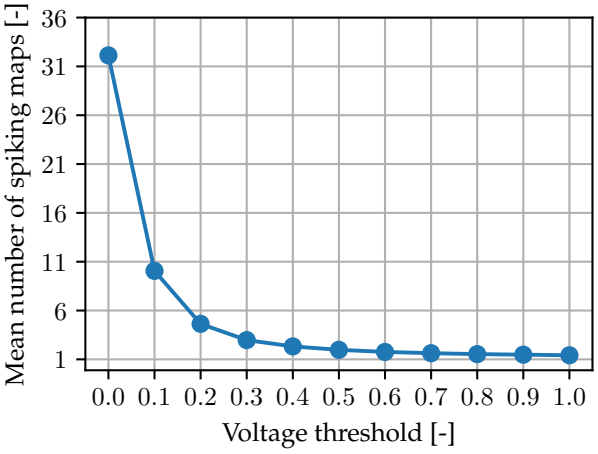


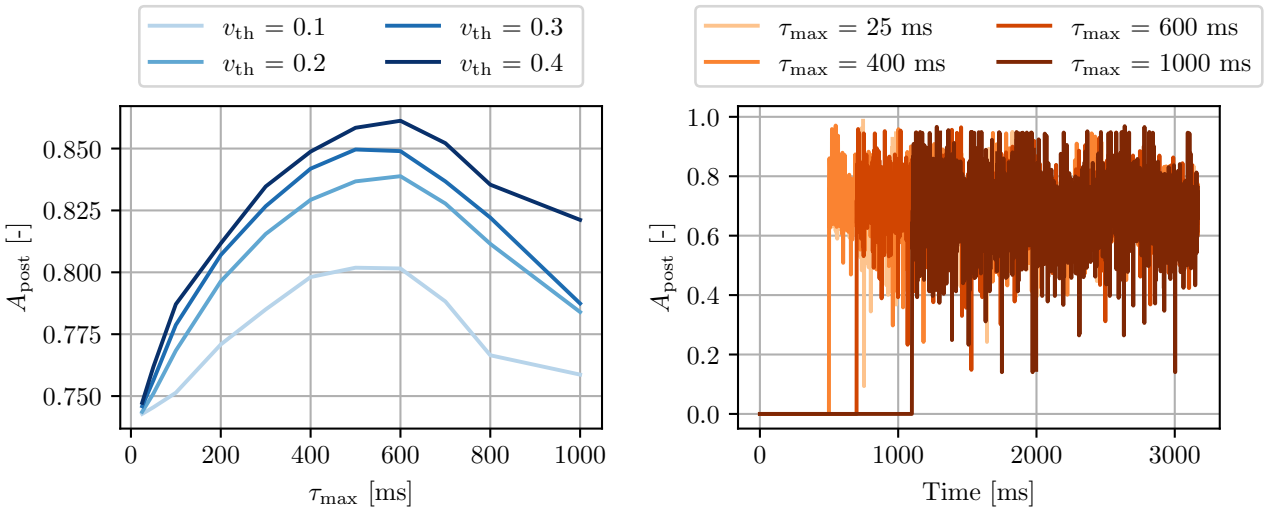
Fig. 11: Number of MS-Conv maps in which spikes occur per retinotopic location. The values are averaged over the entire image and duration of the rotating disk sequence for varying values of the voltage threshold.

approaches can be divided into post- and presynaptic, as well as combined methods. While the postsynaptic methods rely on information about the output spikes to quantify A, the presynaptic methods only consider the presynaptic trace, which makes them inherently faster. However, using the postsynaptic approaches, computations only need to be performed for spiking neurons while the presynaptic methods are applied to all neurons. Finally, the combined methods merge the two approaches by considering the presynaptic traces of spiking neurons.

For a fair comparison between the various approaches, we perform test runs using the same input, namely the rotating disk sequence, for all methods. Consequently, the real optic flow magnitude $\sqrt{u^2+v^2}$ is kept constant across all runs. We then vary the maximum synaptic delay, and adjust all delays such that they are linearly spaced within the range [1, $\tau_{\max}$] to achieve changes in the relative optic flow magnitude, A (see Equation 7).

7.1 Postsynaptic Approach

In order to quantify A, we need to establish how far features travel across the presynaptic trace window within the time frame $\Delta t = \tau_{\max} - \tau_1$. If the chosen value of $\tau_{\max}$ is too large for the temporal dynamics of the input, features travel a far distance within this time, which predominantly triggers output spikes in maps corresponding to fast motion and vise versa. The perhaps most obvious way of quantifying A is thus to consider the magnitudes of the motion represented by the spiking maps. In subsection 3.3 we have already assigned optic flow vectors to each output map, resulting in the distribution depicted on the right side of Figure 6. Since A is simply the product of the optic flow magnitude and the time difference between the first and last synaptic delay (see Equation 7), we can assign a *relative* optic flow magnitude A_{ch} to each map. For this purpose, we should theoretically multiply the computed optic flow magnitudes with the value of $(\tau_{\max} - \tau_1)$ used during training. However, we omit this step since the same constant value of $(\tau_{\max} - \tau_1)$



(a) A_{post} vs τ_{max} (b) A_{post} vs time

Fig. 12: Approximated relative optic flow magnitude A_{post}, during one sequence of the rotating disk data. The left image shows A_{post} averaged over the entire image and sequence duration for different voltage thresholds. The right image shows the history of A_{post} only averaged over the entire image for varying values of the maximum synaptic delay and a constant voltage threshold of $v_{\text{th}} = 0.2$.

was used for all maps and since the optic flow vectors are normalized to be within the range [0,1]. Consequently, we simply define A_{ch} as the optic flow magnitude assigned to map ch. It should be noted that the resulting values are not indicated in pixels, since the method described in subsection 3.3 returns dimensionless estimates of the optic flow vectors. Unfortunately, it is not straight-forward to determine the optic flow vectors in a spiking manner. Consequently, we assume the presence of a fully trained set of weights. This allows us to only perform this computation once and save the results such that they can be accessed during inference. We then propose that the relative optic flow magnitude A within the presynaptic trace of neuron i can be approximated as the average value of A_{ch} of the spiking maps at the same neuron location. Denoting this approximation as A_{post}, the following relationship holds

$$A_{\text{post}_i}(t) = \frac{1}{\sum s^{(3)}_{i,ch}(t-\tau_d)} \sum_{ch=1}^{f^{(3)}} s^{(3)}_{i,ch}(t-\tau_d) A_{i,ch} \qquad (11)$$

where the binary variable $\mathbf{s}^{(3)}_{i,ch}(t-\tau_d)$ represents output spikes in the ch^{th} map at neuron i, delayed by the synapse specific delay $\tau_{\mathbf{d}}$, and $f^{(3)}$ the number of output maps in the MS-Conv layer.

Ideally, we would like to see a linear increase in A_{post} for increasing values of A since this would indicate that A_{post} does indeed approximate the relative optic flow magnitude within the presynaptic trace well. To investigate if this is the case, we compute A_{post} during several test runs with the rotating disk sequence and vary the relative optic flow magnitude by adjusting τ_{max}. Furthermore, test runs are performed for various values of v_{th} to identify a possible coupling with the voltage threshold, which would feed into the issue described in section 6. The result of this analysis is depicted on the left side of Figure 12. It can be seen that A_{post} does indeed increase in an approximately linear manner for values of up to $\tau_{\text{max}} \approx 400$ ms. Subsequently, the slope of A_{post} starts decreasing until it finally becomes negative. Consequently, from this point on, inputs with larger relative optic flow magnitudes trigger spikes in output maps which capture slower motion. This is the case since the duration $\Delta t = \tau_{\text{max}} - \tau_1$ becomes so long, that features captured in one delay have already moved out of the presynaptic trace window at the time of the next delay. Instead, neighboring features have entered the same window, and consequently, the resulting pattern in the presynaptic trace triggers output spikes in maps representing slower motion. When thinking about the temporal delays as samples from the presynaptic trace, and the amount of time it takes for a feature to cross the presynaptic trace window as the signal period, this is similar to aliasing in digital signal processing. When the sampling frequency (the inverse of the temporal spacing between the delays) is too low for the frequency of the considered input (inverse of the signal period), the original signal cannot be reconstructed unambiguously. Due to this similarity, we will refer to the observed phenomenon as *temporal delay aliasing*. For the purpose of quantifying A, this is an undesirable property, since any value of A_{post} cannot be clearly assigned to one individual value of A. When trying to match the relative optic flow magnitude to a previously established target value as proposed in subsection 5.2, it can consequently not be determined how τ_{max} needs to be adjusted to meet this target.

This issue is further amplified by the fact that A_{post} also depends on the voltage threshold, as can be seen on the left side of Figure 12. This is the case since lower threshold values allow for presynaptic traces, which do not match the weights as closely, to produce output spikes. Consequently, output maps corresponding to slower motion continue spiking when increasing τ_{max} which results in a reduced increase in A_{post} when compared to larger threshold values.

The same value of $\tau_{\max}$ thus leads to different values of A_{post} and, consequently, different approximations of A for varying values of v_{th}. Since the proposed update rule drives A to a desired value by changing $\tau_{\max}$, this would lead to different updates for different threshold values. However, since v_{th} does not affect the temporal dynamics of the input, this should not be the case. Furthermore, as explained in section 6, the update rule of the voltage threshold is coupled to $\tau_{\max}$ since small values of the maximum synaptic delay can also increase the synaptic stiffness S. If the update rule of $\tau_{\max}$ is also coupled to v_{th}, neither parameter will be able to converge.

Finally, the identity of spiking output maps does not only depend on the relative optic flow magnitude, but also largely on the motion direction. Consequently, specific output maps might continue spiking after A has changed, since the overall motion direction still matches the corresponding filters well. This effect is further amplified by the fact that not all optic flow directions are represented at multiple magnitudes. Looking at the right side of Figure 6, it can be seen that there is e.g. a lack of MS-Conv filters which capture motion towards the bottom left at larger pixel velocities. While maps 56 and 48 identify motion which is at an angle of approximately 45 degrees with the horizontal, motion to the bottom left at smaller or larger angles can only be captured for smaller pixel velocities. Accordingly, increasing/decreasing A does not necessarily yield output spikes in maps corresponding to smaller/larger values of A_{ch}, but might just result in no output spikes whatsoever. All of the above, leads to fluctuations in A_{post} which are not necessarily caused by changes in A. This results in a poor real-time coupling between A_{post} and A as shown on the right side of Figure 12 (A is again varied by adjusting $\tau_{\max}$). It can be seen that the variance in A_{post} is larger than the difference in the mean value for varying maximum delays. Knowing A_{post} at any given moment in time does thus not provide sufficient information to make conclusions about the real value of A.

In conclusion, the proposed method is not suitable for an intrinsic plasticity update rule for $\tau_{\max}$ due to the occurrence of temporal delay aliasing, the coupling of the output spikes with the voltage threshold, and the discrete nature of the output maps which all allow spikes to occur in maps which do not accurately represent the true relative optic flow magnitude of the input. Furthermore, this method relies on the presence of an existing set of weights, which means that it cannot be expanded to be used during training in future research.

7.2 Combined Approaches

The shortcomings of the postsynaptic approach presented in the previous section can mainly be attributed to discrepancies between the real relative optic flow magnitude, A, and its approximation, A_{post}, which is based on the identity of spiking output maps. In this section, we thus present approaches that do not solely focus on output spikes, but instead combine post- and presynaptic information by looking at the presynaptic traces of spiking neurons. In particular, we propose that the *activity* of delays within the presynaptic trace of spiking neurons is directly connected to the relative optic flow magnitude. For this purpose, we define the activity $c_{i,d}$ of the presynaptic trace at neuron i and synaptic delay d as follows

$$c_{i,d}(t) = \begin{cases} 1, & \text{if } \frac{1}{n^{l-1}} \sum_{j=1}^{n^{l-1}} c_{i,j,d}(t) \geq c_{i_{th}} \\ 0, & \text{if } \frac{1}{n^{l-1}} \sum_{j=1}^{n^{l-1}} c_{i,j,d}(t) < c_{i_{th}} \end{cases} \tag{12}$$

$$c_{i,j,d}(t) = \begin{cases} 1, & \text{if } \hat{X}_{i,j,d}(t-\tau_d) \geq c_{j_{th}} \\ 0, & \text{if } \hat{X}_{i,j,d}(t-\tau_d) < c_{j_{th}} \end{cases} \tag{13}$$

where $j = 1, 2, ..., n^{l-1}$ refers to the presynaptic neurons locally connected to neuron i (see subsection 3.2). The delay d of the presynaptic trace of neuron i is considered active if the average number of active individual connections between neuron i and its presynaptic neurons exceeds the threshold $c_{i_{th}}$. This threshold was introduced to ensure that the proposed definition of activity captures features and not just active individual neurons, which could correspond to noise. Individual synaptic connections are considered active ($c_{i,j,d} = 1$) if their normalized presynaptic trace $\hat{X}_{i,j,d}$ exceeds the threshold $c_{j,th}$. This threshold was introduced since the value of the presynaptic trace does not directly reset to zero in the absence of spikes, but slowly decays and only *approaches* a value of zero. The presynaptic trace is normalized to be within the range [0,1] to avoid the need for varying threshold values for inputs with varying spike frequencies. Using this definition of activity, we investigate two approaches to quantify A, including one considering the *distribution* of activity across the delays and one considering the *number* of active delays.

7.2.1 Distribution of Active Delays

At first thought, one might expect that smaller delays within the presynaptic trace are active more frequently for small values of $\tau_{\max}$ and vice versa. To investigate this idea, we have recorded the frequency of the delay activity ($c_{i,d}$ = 1) within the MS-Conv layer during one sequence of the rotating disk data. The result of this is depicted in Figure 13. It can be seen that the frequency of activity decreases across all delays for increasing values of $\tau_{\max}$. This is the case since the delays take longer to fill up and stay full for a shorter period of time for input features moving at the same pixel velocity. This results in a lower number of output spikes.

However, the *distribution* of activity across the delays does not vary much for different values of $\tau_{\max}$. It can be seen that it stays approximately constant for maximum delays up to $\tau_{\max} \approx 200$ ms. Only if the maximum delay is larger than the amount of time it takes for input features to cross the presynaptic trace window, do some delays become inactive. Predominantly, the last delays are affected by this. A possible reason for this becomes apparent when looking at Figure 6. There, it can be seen that the features within the MS-Conv filters are more centered in the first than in the last delay and as a result, inputs with the last delay inactive are more likely to match the filters than inputs with the first delay inactive. However, either way, there appears to be a stronger correlation between the number of active

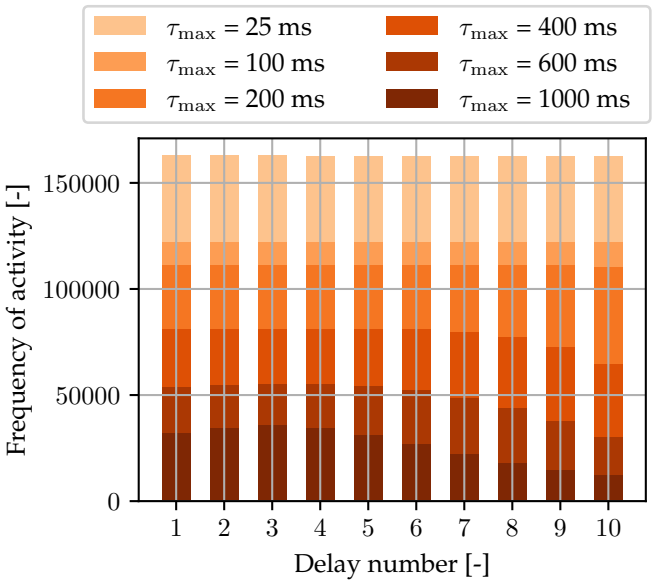


Fig. 13: Distribution of active delays within the presynaptic traces of spiking neurons in the MS-Conv layer during one sequence of the rotating disk data for varying values of $\tau_{\max}$. Delays are considered active if $c_{i,d}$ = 1 and the activity of delays is summed over the entire image and sequence.

delays and changes in $\tau_{\max}$ than between the distribution of activity across the delays and $\tau_{\max}$. Consequently, we investigate the number of active delays in the following subsection.

7.2.2 Number of Active Delays

The specific combination of the pixel velocity of the input data and the employed maximum synaptic delay determines within how many delay steps a feature travels across the presynaptic trace. If the maximum synaptic delay is very large, features can travel a long distance within the time between two subsequent delays. As a consequence, they may pass the entire presynaptic trace window before the time of the last delay. This means that, at no point in time, all delays become active. A similar effect can be observed for features traveling at a large pixel velocity, while the opposite is true for small values of $\tau_{\max}$ or small pixel velocities. Consequently, the number of active delays is proportional to the product of the pixel velocity and $\tau_{\max}$ and thus directly correlated to the relative optic flow magnitude. Consequently, we propose to approximate A at any spiking neuron i with the parameter A_{comb} which represents the number of active delays within the presynaptic trace of spiking neurons i:

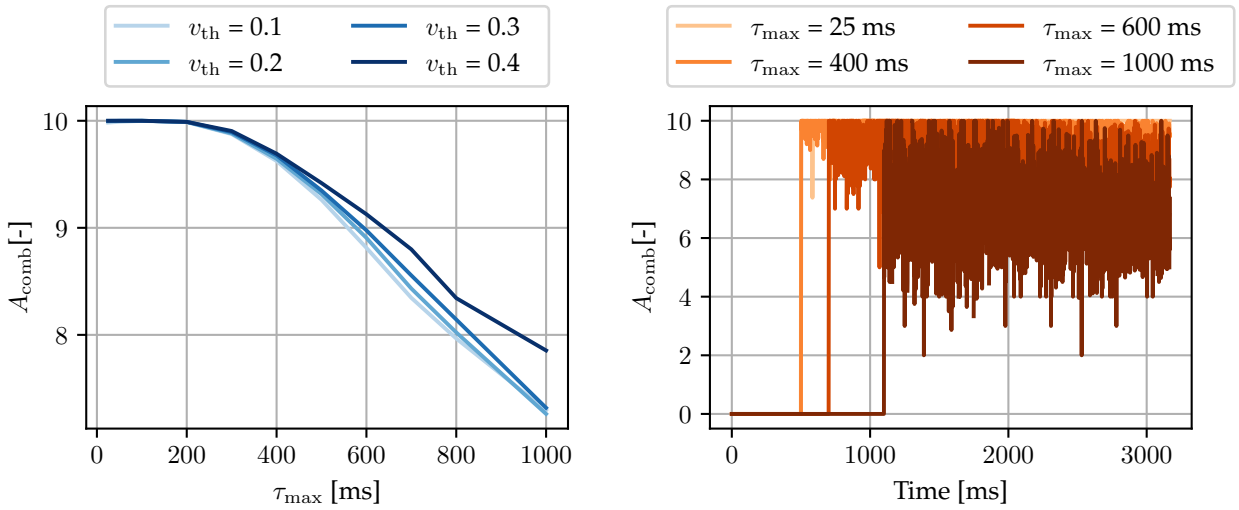
$$A_{\text{comb}_i}(t) = \sum_{d=1}^{m} c_{i,d}(t) \tag{14}$$

Following the same reasoning as in subsection 7.1, we would like to see a linear relationship between A and A_{comb}. However, since larger values of A result in a smaller number of active maps, we would like to see decreasing values of A_{comb} for increasing values of A. The left side of Figure 14 depicts A_{comb} as a function of $\tau_{\max}$ for varying values of v_{th}. It can be seen that after an initial plateau, the number of active delays indeed decreases in an approximately linear fashion. The plateau is caused by the fact that the number of active delays only starts decreasing once $\tau_{\max}$ is larger than the amount of time it takes for features to travel across the presynaptic trace. Unlike the postsynaptic approach, this method does not appear to provoke temporal delay aliasing and shows the desired trend even for larger values of $\tau_{\max}$. For the postsynaptic method, the presence of just one neighboring feature within the first presynaptic delay is sufficient to trigger spikes in output maps corresponding to much lower values of A_{ch}. For the combined method, however, this only increases the number of active delays by one and thus has a much smaller effect on the approximated relative optic flow magnitude. Consequently, the influence of temporal delay aliasing is greatly reduced for this approach.

Furthermore, the approximated relative optic flow magnitude varies less for different values of v_{th} when compared to the postsynaptic approach. While the identity of the spiking output maps still depends on the voltage threshold, we are now considering the presynaptic trace of the spiking neurons rather than the value of A_{ch} assigned to the maps in which they occurred. A poorly-chosen value of the voltage threshold can still lead to erroneous output spikes, but they now 'link' to their presynaptic traces, which do not depend on v_{th}. Consequently, the voltage threshold no longer directly controls the approximated value of A, but only determines which presynaptic traces are considered. This results in more consistent estimates of the relative optic flow magnitude, and thus makes this approach more suitable for the proposed update rule of $\tau_{\max}$.

Moreover, the number of active delays can take on any integer value and is not restricted to the discrete optic flow magnitudes assigned to the output maps. This leads to an improved real-time coupling between A_{comb} and the maximum synaptic delay, as shown on the right side of Figure 14. While there is still a large amount of overlap, the means of the curves are more 'pulled apart' when compared to the right side of Figure 12. This makes it easier to draw conclusions about $\tau_{\max}$ based on the number of active delays. In addition, this method does not rely on an existing set of weights and consequently also has the potential to be applied during training in future research.

However, a major drawback of this method becomes apparent when remembering that the presynaptic trace is supposed to match the shape of the trained weights. To capture as many motion directions and magnitudes as possible with the smallest possible set of weights, all delays within the weights need to be utilized. Consequently, the target value of A_{comb} should be set to $A_{\text{comb}}^{\text{tar}} = m$. However, this corresponds exactly to the plateau region in Figure 14 and does thus once again create ambiguity, since different values of $\tau_{\max}$ result in the same number of active delays. This issue can be circumvented by setting the desired value of A_{comb} just below the maximum number m. For $m = 10$, this value could e.g. be $A_{\text{comb}}^{\text{tar}}$ = 9.9. However, this results in $\tau_{\max}$ values which are too large for the input. Furthermore, it requires averaging the number of active delays over several neurons to drive the integer values of individual neurons to the desired non-integer value.



(a) A_{comb} vs τ_{max} (b) A_{comb} vs time

Fig. 14: Approximated relative optic flow magnitude A_{comb} of spiking neurons in the MS-Conv layer during one sequence of the rotating disk. The left image shows A_{comb} averaged over the entire image and sequence duration for varying values of the voltage threshold. The right image shows the history of A_{comb} only averaged over the entire image for varying values of the maximum synaptic delay and a constant voltage threshold of $v_{\text{th}} = 0.2$.

7.3 Presynaptic Approaches

While the combined approach presented in the previous section has already addressed some issues of the method presented in subsection 7.1, it still relies on postsynaptic information, which makes it inherently slower. In this section we thus aim to translate this approach into a purely presynaptic one which allows providing quicker estimates of A using only the presynaptic trace.

However, this involves the challenging task of defining which parts of the presynaptic trace to consider. For the mixed approach, we simply looked at the presynaptic traces of spiking neurons, but this is no longer possible since we have no information about the output spikes. To illustrate this issue, we consider an artificial sequence of a horizontal bar moving vertically through space. To simplify this process, the exact dynamics of the presynaptic trace as shown in Equation 3 are omitted and the value of the presynaptic trace is simply set to $X_{i,j,d} = c_{i,j,d}$ with $c_{j_{th}} = 0$. For illustrative purposes, we also introduce the concept of the *folded presynaptic trace*. It refers to a representation of the presynaptic trace in which its dimensions are reduced from $\mathbf{X} \in \mathbb{R}^{n^l \times n^{l-1} \times m}$ to $\mathbf{X} \in \mathbb{R}^{n^l \times m}$, where n^l, n^{l-1} and m refer to the number of postsynaptic neurons i, the number of presynaptic neurons j locally connected to any neuron i, and m the number of delays, respectively (see subsection 3.2). Effectively, this means that the presynaptic traces of all postsynaptic neurons are represented in a 2D image (when representing n^l in 2D) at each delay. The presynaptic trace of individual neurons i can then be obtained from this image by extracting a window of the size of the convolutional kernel, r from the folded presynaptic trace at the corresponding retinotopic position.

As illustrated in Figure 15, we analyze the folded presynaptic trace of the moving bar in two separate ways. First, it is recorded within the same convolutional window but at different times. Subsequently, it is recorded at one specific time, but the presynaptic trace window is translated along the direction of motion. Consequently, the relative position of the bar within the presynaptic trace window moves due to the actual movement of the bar over time for the first scenario and due to the movement of the presynaptic trace window in space for the second scenario. For both cases, there are four parameters of interest. These include the reference time/distance, t_0/d_0, at which the moving bar first enters the presynaptic trace window, the time/distance, $t_{\text{ma}}/\ d_{\text{ma}}$, at which the maximum number of delays become active, the time/distance, $t_d/\ d_d$, at which the number of active delays starts decreasing again, and the period of time/distance, $t_f = t_d - t_{\text{ma}}/\ d_f = d_d - d_{\text{ma}}$, during which all delays are active (not indicated in Figure 15).

We consider six different combinations of the pixel velocity, v_p, of the bar and the maximum synaptic delay. Furthermore, we introduce the parameter N_{AD} which indicates how many delays are active within the folded presynaptic trace at any neuron i and any given time:

$$N_{\text{AD}_i} = \sum_{d=1}^{m} c_{i,d} \tag{15}$$

This definition stands in contrast to A_{comb} which only refers to the number of active delays within the presynaptic traces of spiking neurons. We do however employ the same definition of activity as shown in Equation 12. Figure 16 depicts the history of N_{AD} as a function of both time and pixel distance for the six different combinations of v_p and τ_{max}. Furthermore, for each scenario, the full presynaptic trace history is provided for one combination of parameters to provide a better understanding of how features travel across the presynaptic trace delays. Please refer to the Appendix for the full presynaptic trace histories of the remaining parameter combinations.

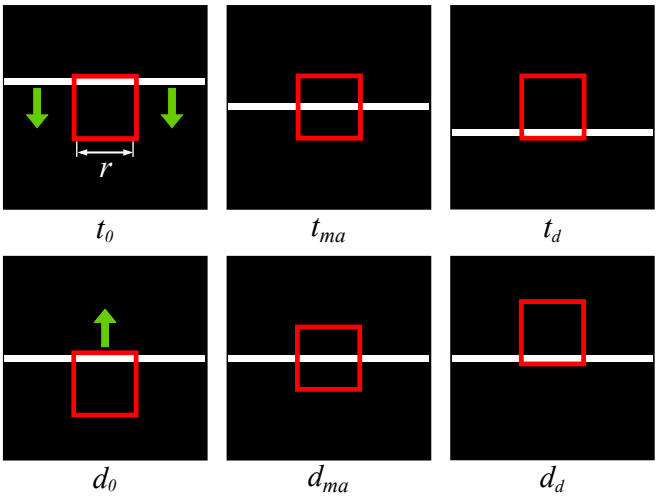


Fig. 15: Simplified folded presynaptic trace of an artificial bar sequence moving vertically through space at a pixel velocity of $v_p = 1px/ms$. The first delay of a total of $m = 5$ delays is depicted. The delays are linearly spaced within the range [1, 5] ms. The red boxes indicate convolutional windows of size $r = 9$ px and the bar has a thickness of 1 px. In the upper image, the convolutional window is fixed in space and the bar moves over time as indicated by the green arrows. In the lower image, the location of the bar is fixed and the window moves in space. Please refer to the text for an explanation of the remaining symbols.

Looking at Figure 16, it can be seen that the presynaptic trace undergoes periods during which it 'fills up' and 'empties out'. During this process, only few delays are active. Considering the number of active delays at these points, would thus give the false impression of a very large relative optic flow magnitude. It is therefore crucial to define which parts of the presynaptic trace to consider when computing A using only presynaptic information. In the following two sections, we propose two approaches to dealing with this issue. The first one considers the *maximum* number of active delays within a specified window, while the second one considers the *average* number of active delays. A more detailed analysis of Figure 16 will be performed in subsection 7.3.

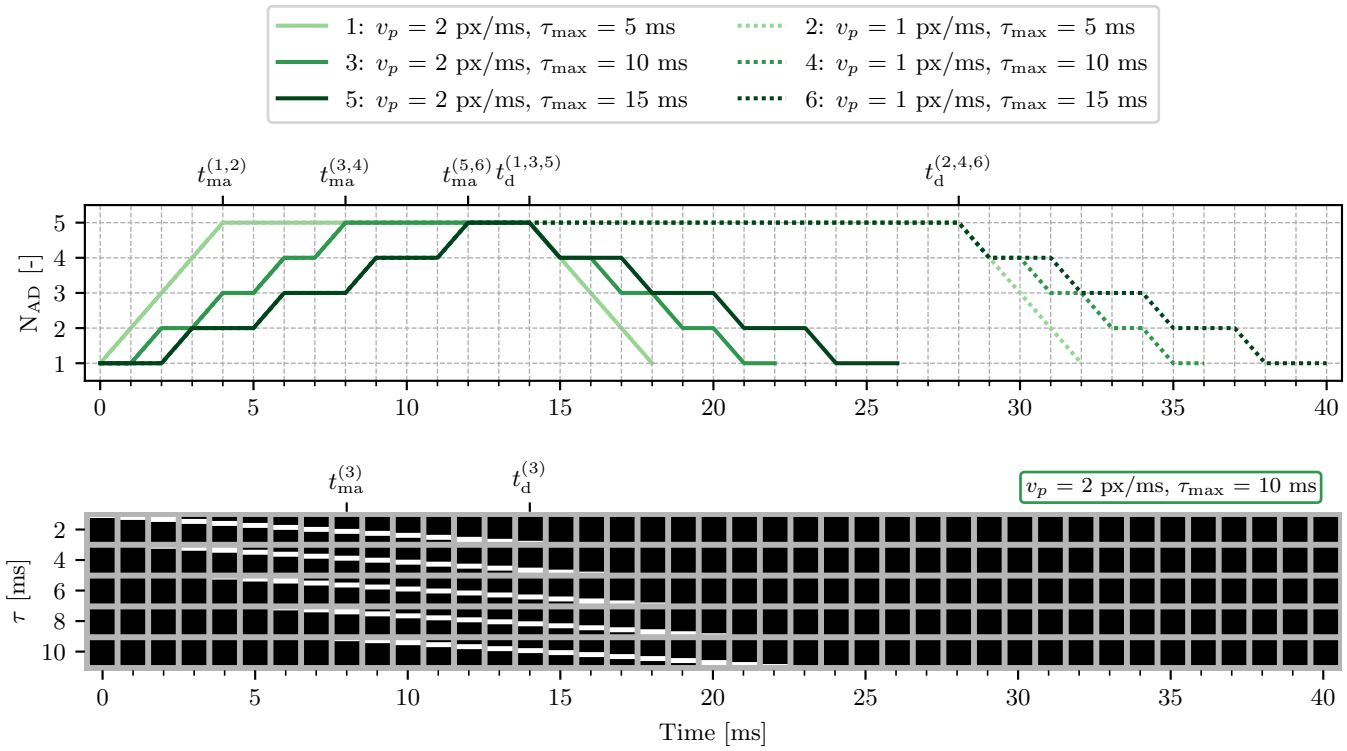
7.3.1 Maximum Number of Active Delays

The number of active delays within the presynaptic trace only provides useful information about the relative optic flow magnitude when the delays have fully 'filled up'. On first thought, one might expect the fill-up time between individual delays to contain valuable information about the relative optic flow magnitude. However, in subsubsection 7.3.2 it will be shown that this quantity only depends on the maximum synaptic delay and not on the pixel velocity of the observed features. Consequently, we could simply consider the presynaptic trace when the number of active delays is at its maximum. However, this would introduce large delays since we would have to wait for features to fully move into the delays before we can start computations. This is where the alternative approach depicted in the lower part of Figure 15 comes in handy. Rather than waiting for features to move through the presynaptic window, we can simply look at the presynaptic trace of neighboring neurons in the opposite direction of motion. For them, features have already traveled further across the presynaptic trace window, which allows us 'to see into the future'. Looking at Figure 16 it can be seen that these two approaches are not completely equivalent since they differ in the amount of time/ number of pixels it takes for the delays to fill up. However, the maximum number of active delays is the same for both of them such that it can be identified by searching in space rather than time, which avoids the introduction of lag. Consequently, we propose to approximate the relative optic flow magnitude at any neuron location i within the folded presynaptic trace with the parameter A_{pre_m}. It represents the maximum number of active delays within a window R of size $r_{\tau_{\max}}$ surrounding neuron i in the folded presynaptic trace. This results in the following definition:

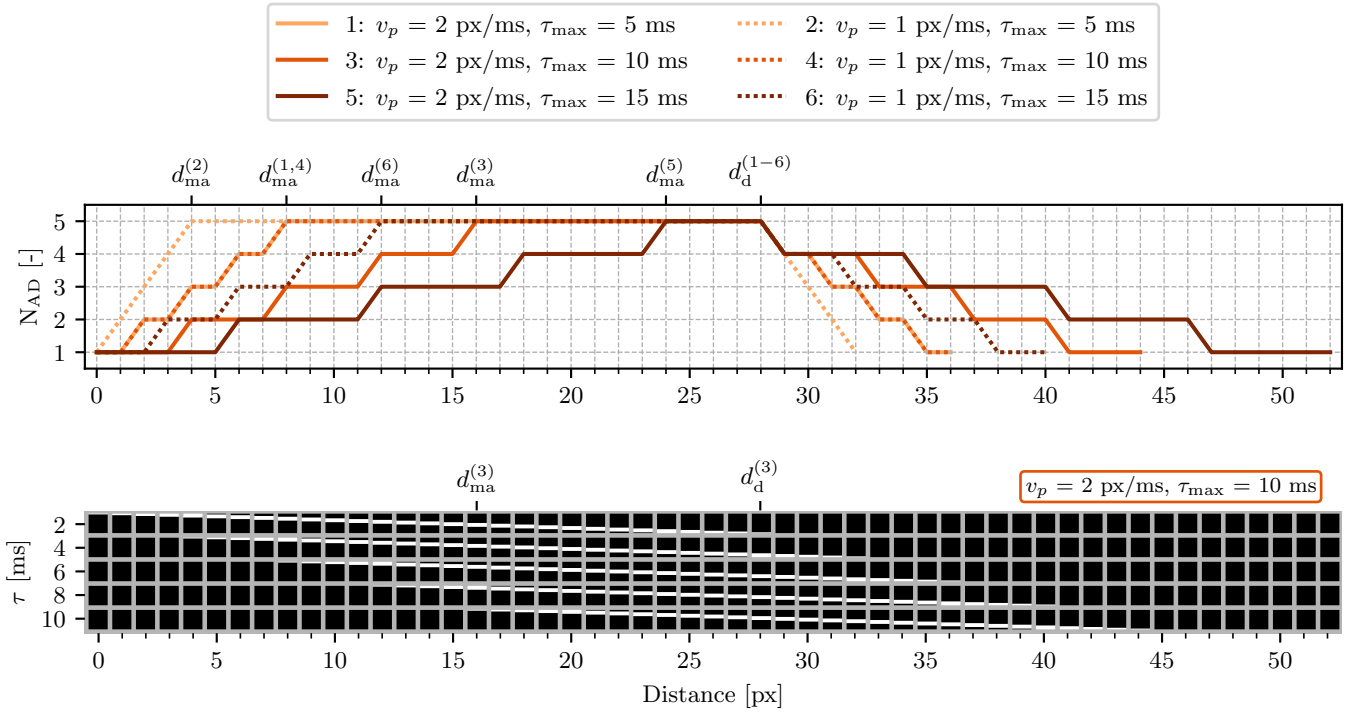
$$A_{\mathrm{pre}_{m\,i}} = \max_{\forall i \in R} N_{\mathrm{AD}_i} \tag{16}$$

Theoretically, it is sufficient to look for the maximum value of N_{AD} in the direction of motion. However, in practice, this is difficult since the whole purpose of the network is to identify optic flow which quantifies the motion direction. Consequently, we instead look for the maximum value of N_{AD} in the 2D-dimensional window of size $r_{\tau_{\max}}$. This is shown in Figure 17 which presents a 2D-illustration of the number of active delays within the folded presynaptic trace for the same artificial bar sequence as shown in Figure 16.

Once again, we investigate the correlation between the proposed quantity A_{pre_m} to approximate A and the actual value of the relative optic flow magnitude. We do this by performing several test runs with the rotating disk sequence at varying values of $\tau_{\max}$. Since we only rely on presynaptic information to compute A_{pre_m}, this parameter does not depend on the voltage threshold. We do, however, present results for varying values of the window size $r_{\tau_{\max}}$. The result of this is depicted on the left side of Figure 18. It can be seen that for large window sizes, the graphs take on the same overall shape as the equivalent combined approach. However, the smaller the window gets, the shorter the initial plateau becomes until it disappears completely, and the graphs take on a non-linear shape. The initial shortening of the plateau can be attributed to the fact that for very large window sizes, the maximum number of active delays is determined over a larger area. Consequently, it requires large increases in $\tau_{\max}$ before that maximum number starts decreasing. At some point the window size becomes so small that at locations at which the presynaptic trace is not 'filled up' the window does not reach the full regions. At these window sizes, the initial plateau completely disappears. For $r_{\tau_{\max}} = 1$ px (= 2 % in Figure 18) we are no longer computing the maximum number of delays which become active but are simply averaging the number of active delays. While this does not represent the initial intention for this approach, it appears promising since it avoids the issue of the initial plateau. However, to utilize this approach, we need to better understand the regions of the presynaptic trace in which the number of active delays is not at its peak. From now on we will refer to these regions as the *edges* of the presynaptic trace, and we will analyze them in more detail in subsubsection 7.3.2.



(a) History of the presynaptic trace as a function of time



(b) History of the presynaptic trace as a function of the pixel distance

Fig. 16: Evolution of the presynaptic trace as a function of time (upper image) and pixel distance (lower image). In each image, the number of active delays N_{AD} is shown in the top half for six different combinations of the pixel velocity v_p and the maximum synaptic delay $\tau_{\max}$. The bottom parts present an example of the full presynaptic trace history for one of these combinations. The figures were created with a presynaptic trace windows size of $r = 25$ px, a bar thickness of $\mathrm{th}_X = 5$ px and $d = 5$ delays. For illustrative purposes, the delays are not spaced within $[1, \tau_{\max}]$ ms, but within the range $[\tau_{\max}/m, \tau_{\max}]$ ms.

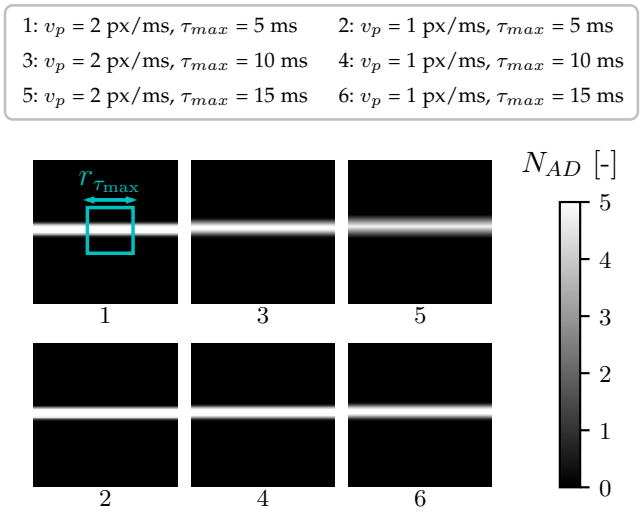


Fig. 17: 2D-illustration of the number of active delays N_{AD} within the folded presynaptic trace of the artificial bar sequence depicted in Figure 16. The blue window indicates the size $r_{\tau_{\max}}$ of the window R within which the maximum value of N_{AD} is determined in Equation 16.

Looking at the right side of Figure 18, it can be seen that the real-time coupling between the number of active delays and $\tau_{\max}$ is also greatly improved when compared to the approach presented in subsection 7.2. Since the estimates of A_{pre_m} are averaged over all neurons rather than just the spiking ones, there is very little overlap between the graphs corresponding to the different delays. This way, more accurate conclusions can be drawn about the relative optic flow magnitude based on the current value of A_{pre_m}.

7.3.2 Average Number of Active Delays

To better understand the edges of the presynaptic trace, in this section, we do not only look at the maximum number of delays that become active when features travels across the presynaptic trace but its entire history. Since we are mainly concerned with overcoming the issue of the 'plateau region' in Figure 14, we only consider scenarios in which all delays become active at some point. If this is not the case, it can easily be determined that the applied maximum synaptic delay is too large, and it can be reduced accordingly until it hits the 'plateau region'. To see what information the edges of the presynaptic trace contain about the relative optic flow magnitude, we once again consider Figure 16.

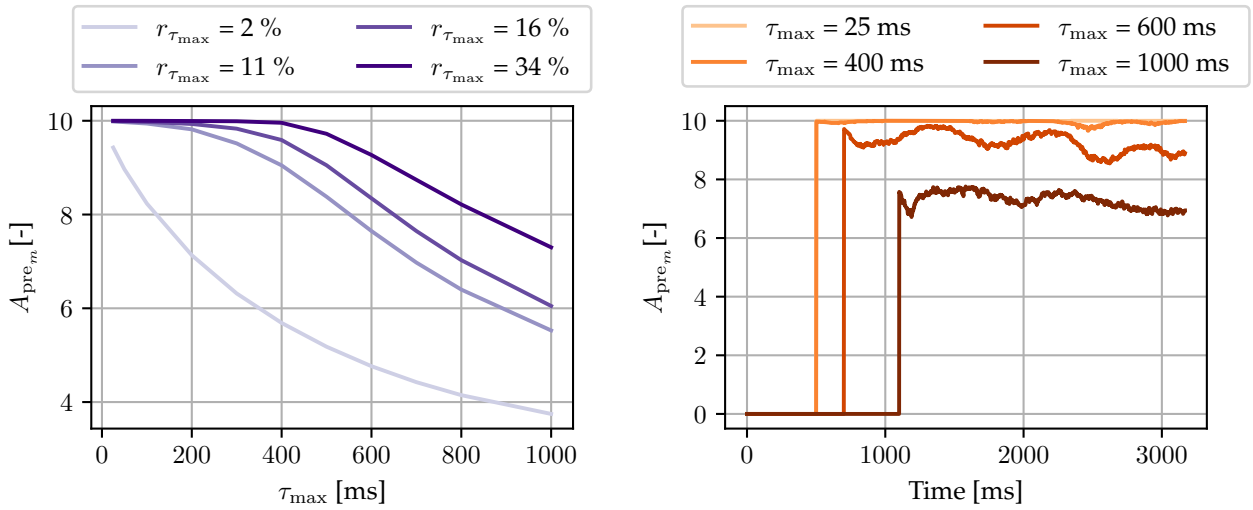
For the time history, t_{ma} only depends on $\tau_{\max}$ and not on the pixel velocity of the input. This makes sense since t_{ma} corresponds to the time at which the last delay first becomes active and this happens $\Delta t = \tau_{\max} - \tau_1$ ms after the first delay has become active. Since t_{ma} does not depend on the pixel velocity, it thus contains no useful information about the relative optic flow magnitude. A similar observation can be made for t_d, the time at which the number of active delays starts decreasing again. Looking at the first row of the full presynaptic trace history in the upper image of Figure 16, it can be seen that t_d corresponds to the amount of time it takes the feature to travel completely through the first delay. It thus only depends on the size of the window, the thickness of the feature and the pixel velocity. Since the first two parameters are kept constant in Figure 16, it appears to only depend on the pixel velocity. Consequently, this parameter is not useful either when trying to quantify the relative optic flow magnitude.

While the period of time during which all delays are active does depend on both $\tau_{\max}$ and v_p, it is proportional to their difference rather than their product. Thus, once again, it does not serve well as a means of quantifying A. This can also be seen by comparing graphs 1 and 4 in the time history of N_{AD}. While they have the same relative optic flow magnitude, they do not have the same values of t_f. This shows that the time histogram of presynaptic traces is not a promising tool for the adaptation of the maximum delay.

The parameters of interest behave differently when considering the number of active delays as a function of space rather than time. Similarly to t_d, d_d corresponds to the pixel distance within which the feature travels across the first delay. However, since the presynaptic traces are not considered in time but in space, it is the displacement of the windows which makes the feature travel across the delays. Consequently, d_d does not depend on the pixel velocity but on the window size, the thickness of the bar and the stride with which the windows are moved. Since all of these parameters are held constant in Figure 16, all parameter combinations yield the same value for d_d.

However, the distance d_{ma} required to fill up all delays depends on both $\tau_{\max}$ and v_p. In fact, features moving at twice the pixel velocity take twice the number of steps to fill up all delays when compared to the time histogram. The reason for this becomes clear when realizing that we no longer have the same units on the x- and y-axis. For the time histogram the time t_{ma} it takes until $t_{ma} - (\tau_{\max} - \tau_1) = t_0$ is naturally just the time $(\tau_{\max} - \tau_1)$ itself. However, for the distance histogram, the equivalent distance d_{ma} also depends on the pixel velocity. This is the case since the displacement of the feature within the first delay only depends on the stride and not the chosen parameters (this can be seen when comparing the first delay of the full histograms shown in the Appendix). As a result, this distance is always the same and features that travel at a larger pixel velocity have to move further into the window until their delayed version first appears in the first delay. Consequently, d_{ma} is proportional to $d_{ma} \propto \tau_{\max} \cdot v_p/s$ which, for a constant stride, corresponds exactly to the relative optic flow magnitude! Consequently, considering the number of active delays as a function of pixel distance rather than time does not only avoid lag, but also captures the relative optic flow magnitude better. In fact, Figure 16 shows that the two lines (1,4) which correspond to the same optic flow magnitudes coincide. This implies that inputs with the same relative optic flow magnitude have the same values of d_{ma} and d_d.

Finally, the pixel distance within which all delays remain active is simply proportional to d_d - d_{ma}. This means that it is proportional to the negative of the relative optic flow magnitude scaled by the stride plus d_d which depends on the stride, the window size, and the thickness of the feature. Assuming that the thickness of the presynaptic traces is approximately constant, this entire term represents a constant. Consequently, both d_{ma} and d_f are promising parameters for the adaptation of $\tau_{\max}$. However, d_{ma} has the additional



(a) A_{pre_m} vs $\tau_{\max}$ (b) A_{pre_m} vs time

Fig. 18: Approximated relative optic flow magnitude A_{pre_m} of neurons in the folded presynaptic trace of the MS-Conv layer during one sequence of the rotating disk. The left image shows A_{pre_m} averaged over the entire image and sequence duration for varying values of the window size $r_{\tau_{\max}}$ (provided as percentage of the total image size). The right image shows the history of A_{pre_m} only averaged over the entire image for varying values of the maximum synaptic delay and a constant window size of $r_{\tau_{\max}}$ = 34 %.

advantage that it does not depend on any other constants. The established dependencies of the various parameters for both the time and the pixel distance histograms are summarized in Table 3.

Unfortunately, d_{ma} and d_f are difficult to quantify in a spiking manner, since they are defined in the direction of motion. However, assuming that d_f is indeed constant, two features with the same optic flow magnitude, will have the same values for d_{ma} and d_f. This means that the number of active delays N_{AD} averaged over a small window capturing the feature will be the same. This average number of active delays can easily be computed within the neural network. In fact, remembering the averaging window $N_{\tau_{\max}}$ introduced in Equation 8, this approach is equivalent to the one described in subsubsection 7.3.1 with a window size of $r_{\tau_{\max}} = 1$. Since this approach does not create ambiguity for small values of $\tau_{\max}$ we deem it the most suitable to quantify the relative optic flow magnitude. Denoting this approximation of A, as A_{pre_a}, the complete update rule then becomes:

$$\Delta\tau_{\max_i}(t) = c_i(t)\eta_{\tau_{\max}}\tau_{\max_i}(t)\frac{1}{\sum c_k(t)} \cdot \sum_{k=1}^{N_{\tau_{\max}}} c_k\left((A^{\mathrm{tar}}_{\mathrm{pre}_{a_i}} - A_{\mathrm{pre}_{a_i}}(t)\right) \tag{17}$$

$$A_{\mathrm{pre}_{a_i}} = N_{\mathrm{AD}_i} = \sum_{d=1}^{m} c_{i,d} \tag{18}$$

$$c_{i,d}(t) = \begin{cases} 1, & \text{if } \frac{1}{n^{l-1}}\sum_{j=1}^{n^{l-1}} c_{i,j,d}(t) \geq c_{i_{th}} \\ 0, & \text{if } \frac{1}{n^{l-1}}\sum_{j=1}^{n^{l-1}} c_{i,j,d}(t) < c_{i_{th}} \end{cases} \tag{19}$$

$$c_{i,j,d}(t) = \begin{cases} 1, & \text{if } \hat{X}_{i,j,d}(t-\tau_d) \geq c_{j_{th}} \\ 0, & \text{if } \hat{X}_{i,j,d}(t-\tau_d) < c_{j_{th}} \end{cases} \tag{20}$$

It should, however, be noted that this approach is limited by the feature density within the image. If the distance between features is smaller than the desired width of d_{ma}, their N_{AD}-histograms start merging into each other and the approach will start to produce erroneous results. Furthermore, this approach is based on the analysis of an artificial sequence containing continuous movement and a constant velocity. In addition, the temporal dynamics of the presynaptic trace were neglected. When translating this approach to real life applications, we consequently implicitly make the same assumptions about the employed real data.

8 Sensitivity Analysis

Since the intrinsic plasticity update rules proposed in the previous sections have introduced a number of new parameters, we perform a detailed sensitivity analysis in this section. This shall highlight how they affect the performance of the update rules. For this purpose, we once again consider the MS-Conv output using the rotating disk sequence as an input while varying one parameter at a time. Unless specified otherwise, the remaining parameters are kept at the values listed in Table 4.

8.1 Target Values of New Parameters

The synaptic stiffness target value S^{tar} specifies in how many output maps the same input is allowed to produce output spikes. The first row of Figure 19 shows how varying this parameter affects the motion estimates provided by the MS-Conv layer. It can be seen that small values result in very sparse output, since we only allow one output map

TABLE 3: Overview of the parameters used for the analysis of the presynaptic trace edges.

	Parameter	Symbol	Unit	Dependency
Time	Time at which maximum number of delays become active	t_{ma}	ms	f($\tau_{\max}$)
	Time at which number of active delays starts decreasing	t_d	ms	f(r, th_X, v_p, s)
	Period of time during which all delays are active	$t_f = t_{\mathrm{ma}} - t_d$	ms	f(r, th_X, v_p, s) - f($\tau_{\max}$)
Space	Distance at which maximum number of delays become active	d_{ma}	px	f($\tau_{\max} \cdot v_p \cdot \frac{1}{s}$)
	Distance at which number of active delays starts decreasing	d_d	px	f(r, th_X, s)
	Distance during which all delays are active	$d_f = d_{\mathrm{ma}} - d_d$	px	f(r, th_X, s) - f($\tau_{\max} \cdot v_p \cdot \frac{1}{s}$)

TABLE 4: Overview of parameters used during intrinsic plasticity update rules of the voltage threshold v_{th} and the maximum synaptic delay $\tau_{\max}$.

	Parameter	Symbol	Unit	SS-Conv	MS-Conv
v_{th}	Synaptic stiffness target value	S^{tar}	[-]	2	2
	Averaging window size	N_{th}	[% of image width]	100	100
	Learning rate	$\eta_{v_{\mathrm{th}}}$	[-]	0.05	0.05
	Decay Factor	$\lambda_{v_{\mathrm{th}}}$	[-]	0.5	0.5
	Resting voltage threshold	$v_{\mathrm{th}_{\mathrm{rest}}}$	[-]	0.01	0.01
$\tau_{\max}$	Relative optic flow magnitude target value	$A^{\mathrm{tar}}_{\mathrm{pre}_a}$	[-]	-	6
	Averaging window size	$N_{\tau_{\max}}$	[% of image width]	-	100
	Learning rate	$\eta_{\tau_{\max}}$	[-]	-	-0.05
	Presynaptic trace magnitude threshold	$c_{j_{\mathrm{th}}}$	[-]	-	0.02
	Presynaptic trace quantity threshold	$c_{i_{\mathrm{th}}}$	[-]	-	0

to spike per retinotopic location. This requires the presynaptic trace to match the weights very well. For increasing values of S^{tar} this requirement is loosened, and the output becomes denser. This does however come at the cost of a higher density of incorrect output spikes, as can be seen when remembering that the disk is rotating clockwise and comparing the color legend in Figure 6 to the output for S^{tar} = 5. Performing a trade-off between these two effects, we chose a value of $S^{\mathrm{tar}} = 2$. This value also makes sense on an intuitive level. While allowing two output maps to spike, limits the occurrence of contradictory estimates, it also accounts for the fact that the true optic flow value might take on a value located in between two of the discrete vectors assigned to the output maps. As explained in section 6 we hypothesize that this value only depends on the employed weights and not on the input data. Consequently, we keep it at the same, constant value regardless of the spatiotemporal properties of the input. While S^{tar} behaves similarly to the voltage threshold, in subsection 9.1 it will be shown that unlike v_{th} it is less coupled to the remaining parameters.

The target value of the estimated relative optic flow magnitude $A^{\mathrm{tar}}_{\mathrm{pre}_a}$ indicates how many delays we require to be active on average within a specific presynaptic trace window. Looking at the second row of Figure 19 it can be seen that choosing a too small value of $A^{\mathrm{tar}}_{\mathrm{pre}_a}$ leads to the same behavior as choosing a too large maximum synaptic delay and vise versa (see section 4). While the relative optic flow magnitude has the same effect on the network performance as the maximum synaptic delay, in section 9 it will be shown that it depends less on the temporal dynamics of the input. This implies that the same value of $A^{\mathrm{tar}}_{\mathrm{pre}_a}$ provides good results for different kinds of input data. Keeping this in mind, we chose a value of $A^{\mathrm{tar}}_{\mathrm{pre}_a} = 6$ since it provides good results for the rotating disk data.

8.2 Averaging Window Size

The averaging window size $N_{v_{\mathrm{th}}}$ specifies the size of the neighborhood within which individual values of the synaptic stiffness are averaged during the update rule. Figure 20 depicts the MS-Conv output and the resulting v_{th} values for varying values of $N_{v_{\mathrm{th}}}$ using an initial threshold of v_{th} = 0.1 (please refer to Figure 8 for an illustration of the initial output). Since smaller window sizes result in an overall reduction in v_{th}, we use a logarithmic scale, such that differences in the voltage threshold can still easily be identified in the corresponding images. No padding is applied to the windows, as this would yield inconsistent voltage threshold updates. While zero-padding does not directly affect the computation of the window average since only spiking neurons are taken into account, it does reduce the number of active neurons within windows at the edges of the image. Since the voltage threshold is reduced in the absence of output spikes, it is thus reduced more at the edges than in the center of the image if padding is applied. To counteract the change in image dimensions due to the lack of padding, the voltage threshold estimates are upsampled using a 'nearest' algorithm to match the original dimensions.

While larger window sizes better eliminate erroneous output spikes, they do not capture the less frequently spiking features in the center and bottom of the image. This is the case since one threshold is computed for the entire image, resulting in a value which is too low for some parts of the image and too high for others. This issue seems to improve for smaller window sizes, which allow the threshold to adapt to specific parts of the image. However, this leads to an increase in erroneous output spikes that can be attributed to two factors. The first one is that very small window sizes provide little time for the algorithm to adjust the threshold before an input feature has passed the window. Consequently, the number of spiking maps cannot

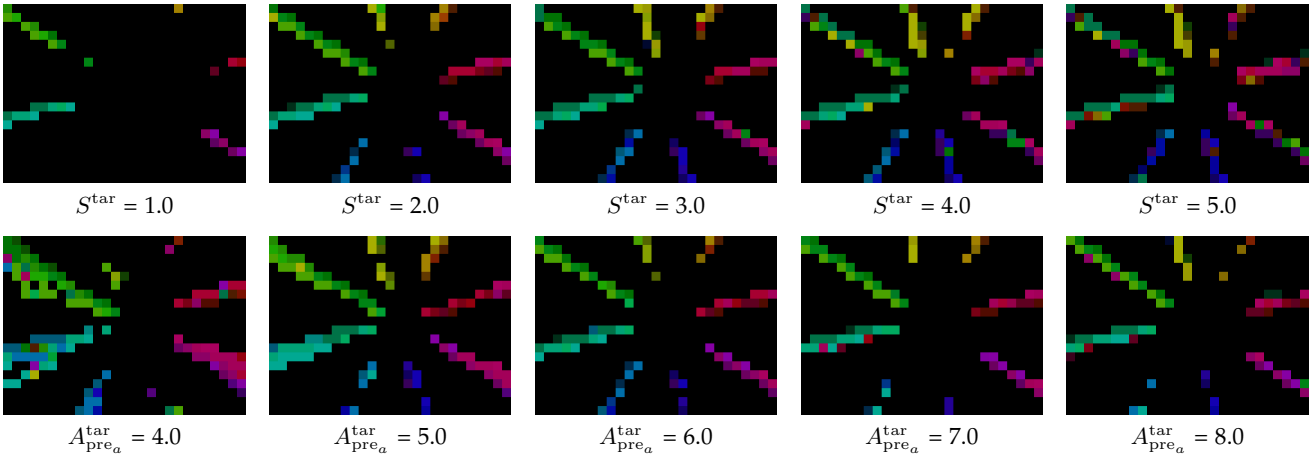


Fig. 19: MS-Conv output of the rotating disk sequence employing the intrinsic plasticity update rules for varying values of the synaptic stiffness target value S^{tar} (first row) and the relative optic flow magnitude target value $A^{\text{tar}}_{\text{pre}_a}$ (second row) .

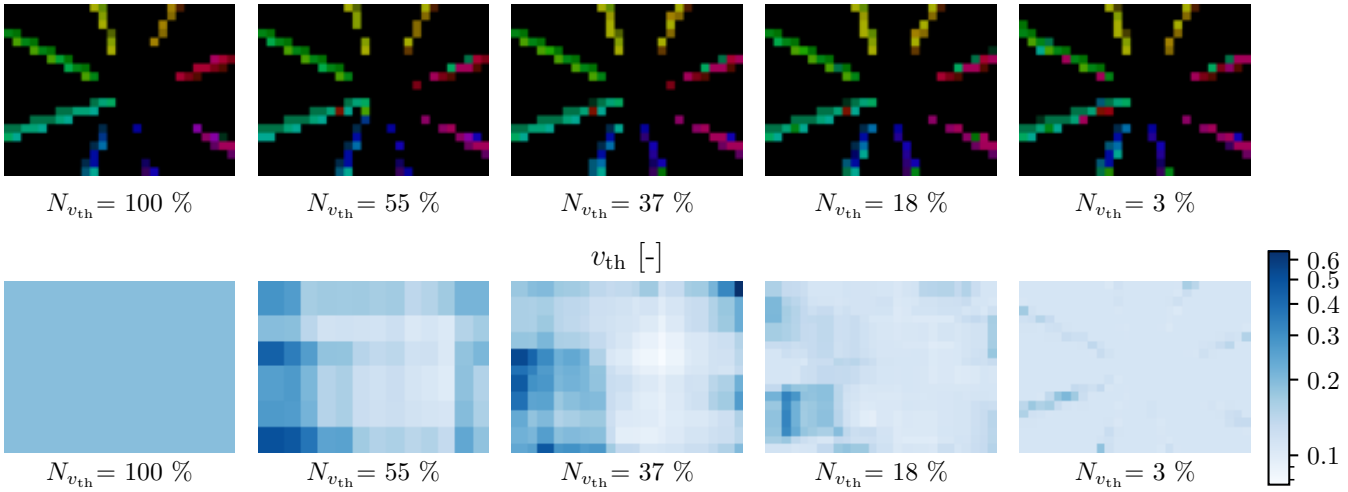


Fig. 20: Effect of the window size $N_{v_{\text{th}}}$ on the performance of the update rule for the voltage threshold v_{th}. The first row shows the MS-Conv output for varying window sizes and the second row the corresponding magnitudes of v_{th}. The window sizes are specified as a percentage of the total image width, and for all test runs an initial value of v_{th} = 0.1 was used. Please refer to Figure 8 for an illustration of the initial output.

be driven to the desired value on time. The second reason becomes clear when remembering the formulation of the update rule. To allow the algorithm to recover from v_{th} values which are too high to produce any output spikes, the term lowering the threshold in the absence of spikes was added. However, this term becomes more and more dependent on the input for decreasing window sizes. Assuming that v_{th} is too high if there are no output spikes in the entire image is a reasonable assumption. When only considering a small part of the image, however, a lack of output spikes does not necessarily mean that v_{th} is too high, but could merely imply that there is no movement in the corresponding part of the image. By choosing a very small window size, we ask the algorithm to decrease the threshold at every window until output spikes are produced, which might also correspond to noise. This explains the overall decrease in the voltage threshold and the occurrence of incorrect output spikes for very small window sizes.

While the described effects are quite subtle for the presented scenario, we considered the MS-Conv layer, in which the majority of noise has already been filtered out. We would thus recommend the use of a larger window size in the SS-Conv layer. Although the proposed update rule appears to be robust in the MS-Conv layer even for small window sizes, we still chose a value of $N_{v_{\text{th}}} = 100\%$ to isolate the performance of the proposed approach from the influence of the window size during the evaluation.

The window size $N_{\tau_{\text{max}}}$ defines the size of the neighborhood within which individual, neuron-specific estimates of the relative optic flow magnitude are averaged during the update rule. The effect of this parameter on the network performance is presented in Figure 21. For all test runs, a starting value of τ_{max} = 25 ms was used. It can be seen that for all window sizes, τ_{max} is driven to a higher, more suitable value. Accordingly, the undesired effects resulting from using a too small τ_{max} value (see subsubsection 4.1.6)

can no longer be observed in the MS-Conv output. However, for smaller window sizes, incorrect output maps start spiking and motion in the lower center of the image is no longer captured. The reason for this becomes apparent when looking at the $\tau_{\max}$ values. For $N_{\tau_{\max}}$ = 48/34 % the maximum synaptic delay is larger in the general central area of the image. This makes sense since the input contains slower motion there. However, for smaller window sizes a drastic increase in $\tau_{\max}$ can be observed which is contained to only the bottom left part of the center. This is the result of one of the limitations explained in subsubsection 7.3.2. There, we have introduced the *edge* of the folded presynaptic trace, which describes the region within the folded presynaptic trace in which the number of active delays N_{AD} is larger than zero but smaller than its maximum value. In the third row of Figure 21 this corresponds to the gray regions (as opposed to black or white) of the image. We furthermore established that the width of these edges in the direction of motion is directly proportional to the relative optic flow magnitude. However, in the lower-left center of the image, features are located so close together that their presynaptic traces overlap and the edges are no longer visible. This creates the illusion of a very large relative optic flow magnitude and accordingly, $\tau_{\max}$ is increased to compensate for this. As a consequence, the maximum synaptic delay becomes so large that motion can no longer be captured in the corresponding parts of the input. For $N_{\tau_{\max}}$ = 100 % this effect is averaged over the entire image and does thus not have a large impact. Looking at the corresponding image of N_{AD} it can be seen that the presynaptic traces in fact still overlap in this region. However, for $N_{\tau_{\max}}$ = 2 %, $\tau_{\max}$ is increased so much that the presynaptic traces are 'broken up' in the corresponding region. Finally, the effect of the changes in $\tau_{\max}$ on the folded presynaptic trace are shown in the last row of Figure 21. When using the same maximum delay across the entire image, the features in the last delay remain straight, since they are all delayed by the same amount. For smaller window sizes, however, this is not the case and the features in the last delay of the folded presynaptic trace start 'bending' since different parts are considered at different points in time. Once the windows become too small, features start falling apart since different delays are applied to neighboring neurons. This is the case since we approximate the width of the presynaptic trace edges as the proportion between full and non-full delays. However, once the window size becomes smaller than the width of the edges, this is no longer possible and separate estimates are obtained for the two regions, resulting in discontinuous $\tau_{\max}$ updates across the image.

This analysis shows that in principle, the proposed update rule allows computing local updates of $\tau_{\max}$ if a sufficiently large window size is chosen. In that case the optic flow magnitude is no longer only encoded in the identity of the spiking output maps but also in the intrinsic neuron state of $\tau_{\max}$. However, this analysis also shows that noise and overlapping edges can lead to inconsistent updates across the image. Since accurate information about the relative magnitudes of optic flow vectors across the image is crucial for navigational purposes, we thus chose to use one $\tau_{\max}$ value for all retinotopic locations. It should, however, be noted that most of the presented shortcomings are not inherent to the method utilizing the width of the presynaptic edges, but are a result of the way the width is estimated. For future research, it might thus be interesting to investigate alternative ways of quantifying this property.

8.3 Learning Rate

As the name reveals, the learning rates $\eta_{v_{\mathrm{th}}}$ and $\eta_{\tau_{\max}}$ determine how quickly the respective update rules respond to changes in the input. Naturally, we would like the response to be as quick as possible while not resulting in instability. Using trial and error we find that this can be achieved with learning rates of $\eta_{v_{\mathrm{th}}} = 0.05$ and $\eta_{\tau_{\max}}$ = -0.5, where the minus signs accounts for the inversely proportional relationship between A_{pre_a} and $\tau_{\max}$.

8.4 Decay Factor

Looking at Equation 9 it can be seen that the decay factor $\lambda_{v_{\mathrm{th}}}$ determines the relationship between the rate of increase in the voltage threshold if output spikes occur in too many maps and the decrease in the absence of output spikes. Figure 22 depicts the MS-Conv output for varying values of this parameter. Since $\lambda_{v_{\mathrm{th}}}$ is mainly of interest for small window sizes, we use a value of $N_{v_{\mathrm{th}}} = 3\%$. For $\lambda_{v_{\mathrm{th}}} = 0$ the update rule can only ever drive the threshold to the resting value which of course leads to a large number of erroneous output spikes. The reverse is true for $\lambda_{v_{\mathrm{th}}} = 1$. Since the update rule can now only increase the threshold, the output becomes more sparse. Accordingly, $\lambda_{v_{\mathrm{th}}}$ can be chosen in such a way as to either prioritize the correctness or density of output spikes. Since we have no specific preference in this research, we chose a value of $\lambda_{v_{\mathrm{th}}} = 0.5$.

8.5 Presynaptic Trace Thresholds

In section 7 it was shown that the number of active delays within the presynaptic trace 'fill up' and 'empty out' meaning that it consists of a full part and edges on both sides. However, in this analysis the temporal dynamics of the presynaptic trace were neglected and in practice, the presynaptic trace does not become zero as soon as an edge moves out of the visual field. Instead, it slowly decays, which effectively increases the amount of time for which all delays are active and thus the width of the 'full' part. To counteract this effect, the presynaptic trace magnitude threshold $c_{j_{\mathrm{th}}}$ was introduced in Equation 13. The effect of this threshold on the number of active delays N_{AD} in the folded presynaptic trace is shown in the first row of Figure 23. It can be seen that a threshold of $c_{j_{\mathrm{th}}} = 0$ causes all delays to be constantly active, since the presynaptic trace never fully decays to zero. For increasing values of the threshold however, the width of the full part decreases, and the edges become visible again. Since choosing a small threshold value results in a wider full part, the presynaptic traces of neighboring features are more likely to overlap. As shown in subsection 8.2 this can result in erroneous $\tau_{\max}$ updates. However, a threshold that is too high does not capture all input features. Performing a trade-off between these two considerations, we chose a value of $c_{j_{th}}$ = 0.02.

The second row of Figure 23 shows that with the help of this threshold the predicted relationship between N_{AD} and

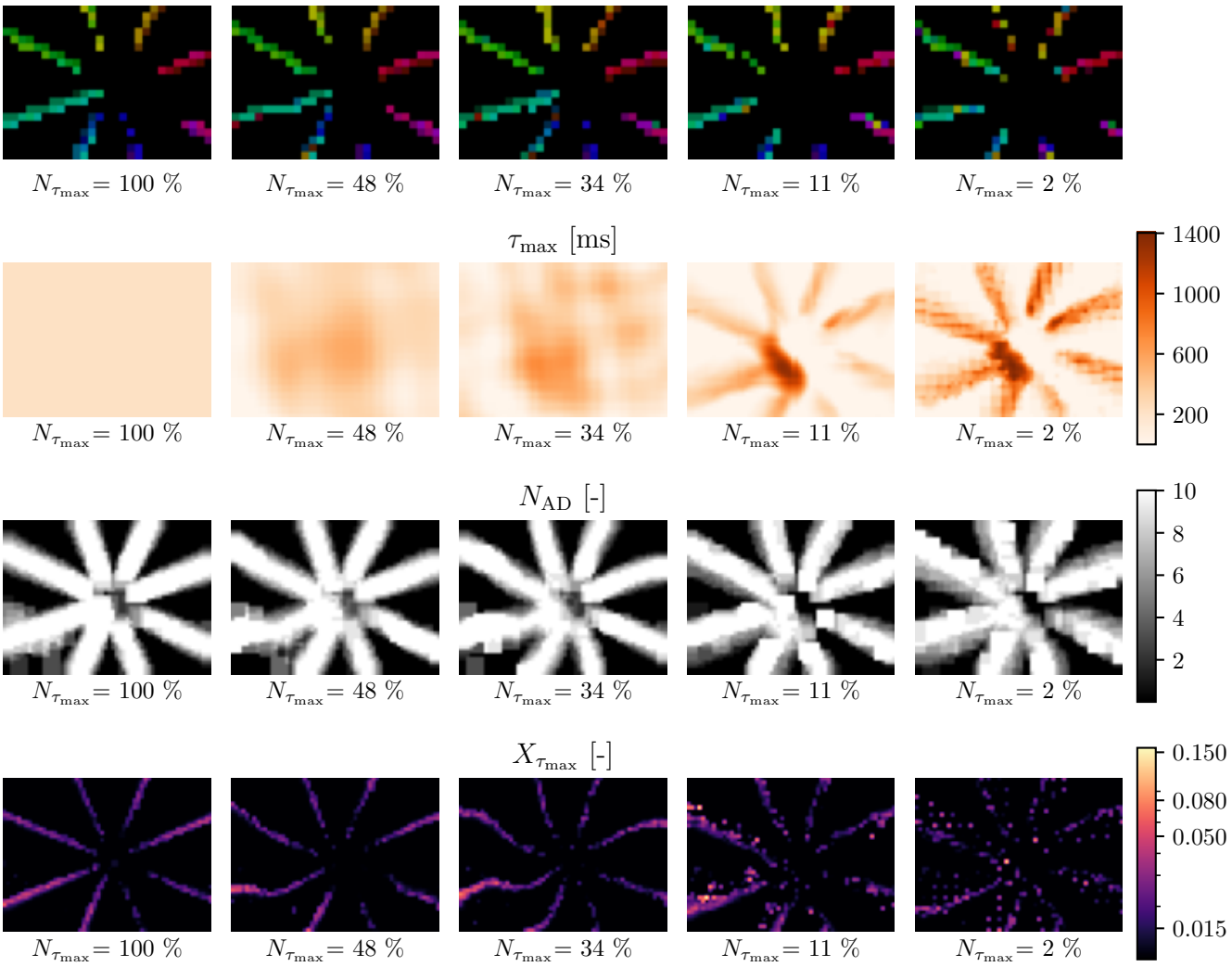


Fig. 21: Effect of the window size $N_{\tau_{\max}}$ on the performance of the update rule for the maximum synaptic delay $\tau_{\max}$. The first row shows the MS-Conv output for varying window sizes and the second row the corresponding magnitudes of $\tau_{\max}$. The third row shows the number of active delays N_{AD} and the fourth row the maximum delay within the folded presynaptic trace $X_{\tau_{\max}}$. The window sizes are specified as a percentage of the total image width, and for all test runs an initial value of $\tau_{\max}$ = 25 ms was used. Please refer to Figure 8 for an illustration of the initial output.

the relative optic flow magnitude still holds true for real data, although the temporal dynamics of the presynaptic trace were neglected in the derivation in subsubsection 7.3.2. In particular, a coupling between the number of active delays and the relative optic flow magnitude can be observed. The thickness of the edges increases for increasing values of $\tau_{\max}$ while the width of the full part decreases. Furthermore, the edges become wider towards the outer parts of the image which correspond to larger pixel velocities. However, the introduction of the threshold does provoke an undesired coupling with the presynaptic trace time constant λ_X. While changes in λ_X do not require adjustments in the maximum synaptic delay, they do influence the shape of N_{AD}. The presynaptic trace decays faster for smaller values of λ_X such that it reaches the threshold more quickly which results in a more narrow full part as illustrated in the third row of Figure 23. The opposite holds true for larger values of λ_X. This is an undesirable property as it can lead to inconsistent $\tau_{\max}$ updates for the same input and maximum synaptic delay. The consequences of this will be illustrated in subsection 9.1.

While $c_{j_{\mathrm{th}}}$ accounts for the fact that the presynaptic trace never fully decays, the quantity threshold $c_{i_{\mathrm{th}}}$ defined in Equation 13 was introduced to filter out noise. Rather than evaluating $X_{i,j,d}$ at individual synapses, it controls how *many* of them need to exceed the magnitude threshold for the delay to be considered active. This ensures that features, rather than individually firing neurons are considered, and is thus supposed to filter out noise from the presynaptic trace. Looking at the last row of Figure 23 it can be seen that the amount of noise does indeed decrease for increasing values of $c_{j_{\mathrm{th}}}$. However, values which are too large also filter out sparse input. To increase the robustness of the update rule w.r.t. variations in the input signal and keeping in mind that the SS-Conv layer filters out the majority of noise, we consequently set the quantity threshold to $c_{i_{\mathrm{th}}} = 0$.

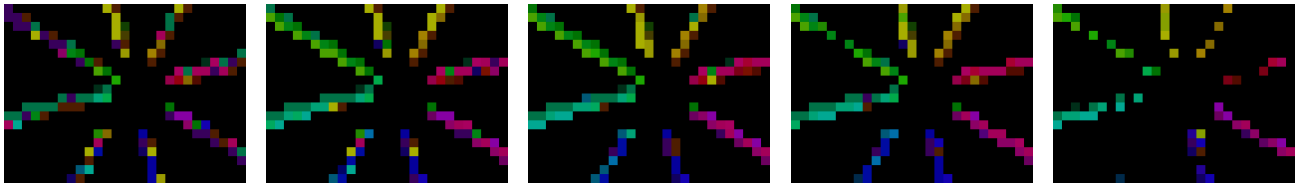


Fig. 22: Effect of the decay factor $\lambda_{v_{\text{th}}}$ on the MS-Conv output. All test runs were performed with a window size of $N_{v_{\text{th}}}$ = 3 % and an initial value of v_{th} = 0.1.

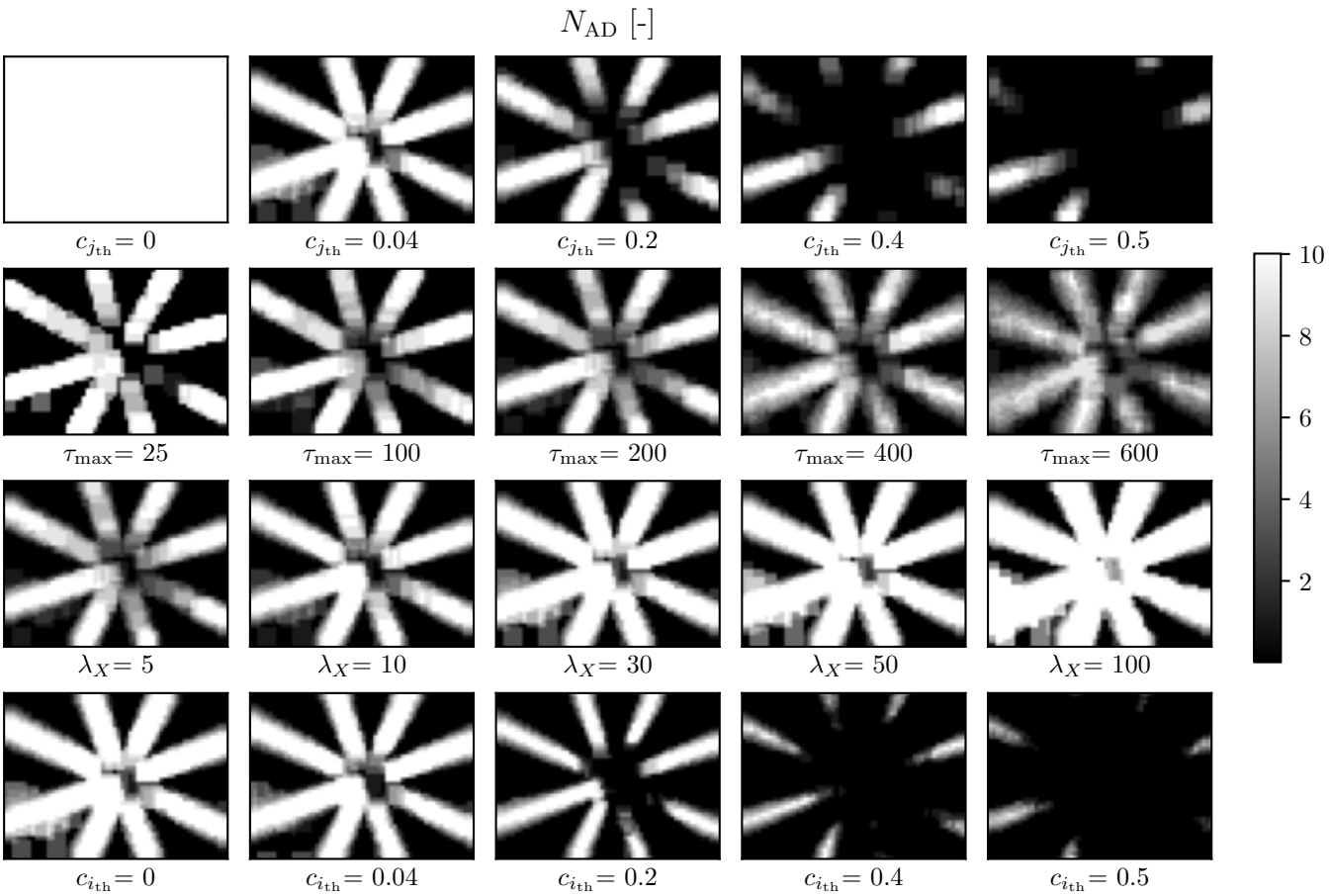


Fig. 23: Number of active delays N_{AD} within the folded presynaptic trace for varying values of the presynaptic trace magnitude threshold, $c_{j_{\text{th}}}$ (first row), the maximum synaptic delay, τ_{max} (second row), the presynaptic trace time constant, λ_X (third row), and the presynaptic trace quantitiy threshold, $c_{i_{\text{th}}}$ (fourth row).

9 Evaluation

To asses the proposed update rules this section will first demonstrate that their ability to decouple the neuron parameters from one another. Subsequently, both a qualitative and a quantitative evaluation using the ODA dataset will be performed.

9.1 Decoupling

In this section, the parameter analysis performed in section 4 is repeated utilizing the proposed update rules during inference. The corresponding snapshots of the MS-Conv output can be seen in Figure 24 and the time histories of the voltage threshold and the maximum synaptic delay are depicted in Figure 25.

9.1.1 Voltage Threshold

Looking at the first row of Figure 24 and Figure 25 it can be seen that changing the initial voltage threshold no longer affects the output. Since the target value of the synaptic stiffness is kept constant, the voltage threshold update rule always drives v_{th} to the same value. As shown in section 8 the density and accuracy of the output spikes can now be controlled by adjusting the synaptic stiffness. Furthermore, the update rule for the maximum synaptic delay appears to be unaffected by changes in v_{th}. This was expected, since the τ_{max} updates only rely on presynaptic information.

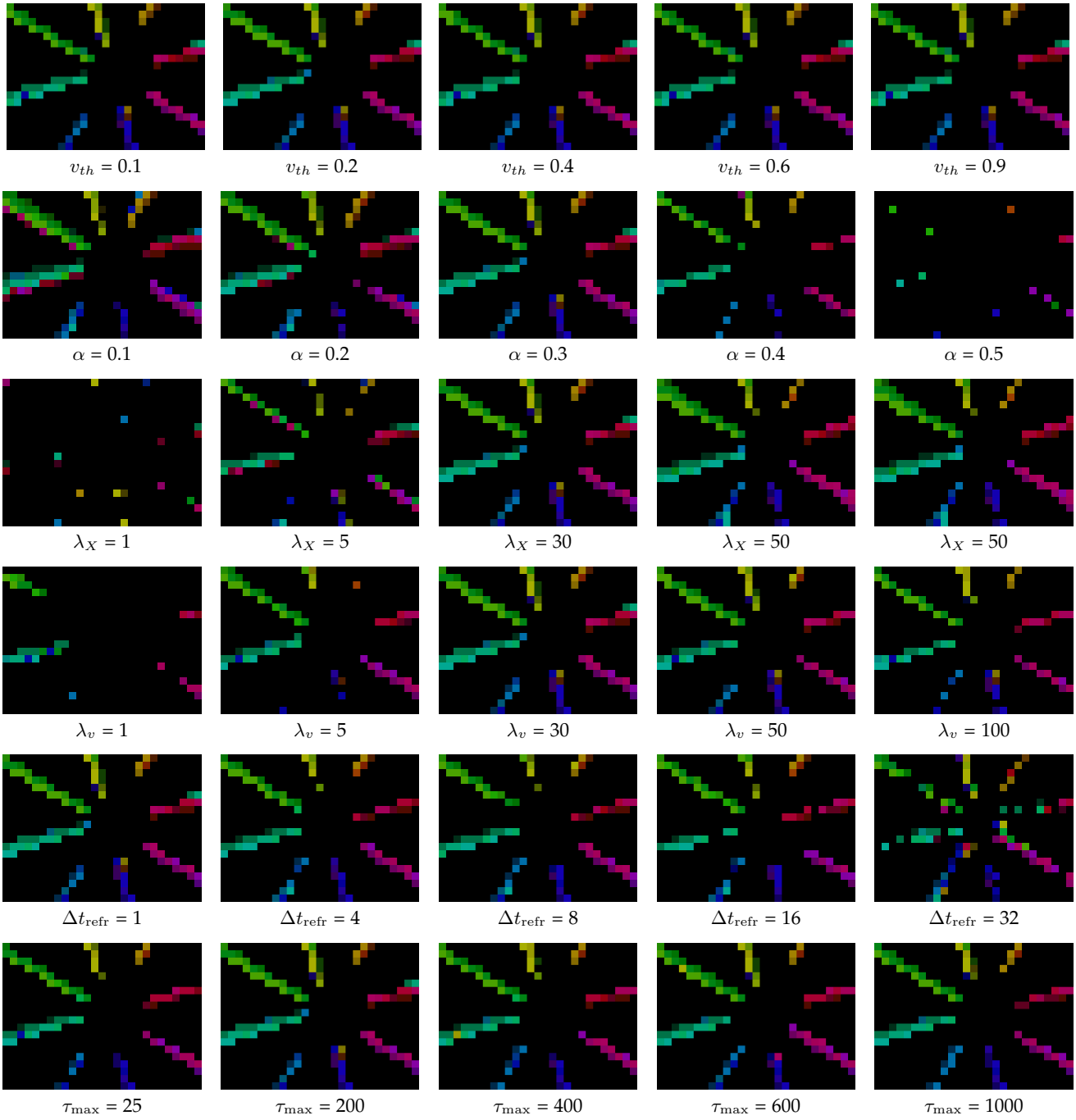


Fig. 24: MS-Conv output for the rotating disk employing the proposed update rules.

9.1.2 Presynaptic Trace Scaling Factor

In section 4 the main function of α was defined as balancing out differences in the spiking activity of neurons. Previously, this main function was overshadowed by the overall decrease in the number of output spikes caused by increases in α. Looking at the second row of Figure 25, it can be seen that the v_{th} update rule automatically counteracts this effect by decreasing v_{th}. The second row of Figure 24 shows that this uncovers the main function of the presynaptic trace scaling factor. For increasing values of α the spiking activity of the less active blue part of the output decreases less than for the more active green part. While an overall decrease in spiking activity can still be observed, this is only the case since at some point α decreases the spiking activity so much that v_{th} approaches zero and cannot be lowered any further. Although α affects the presynaptic trace, it does not appear to influence the τ_{max} update rule. This is the case since τ_{max} is adapted by considering the width of the edges within the folded presynaptic trace, while α only affects its magnitude. Using the normalized presynaptic trace in Equation 13 thus prevents any undesired coupling between the τ_{max}-update and α.

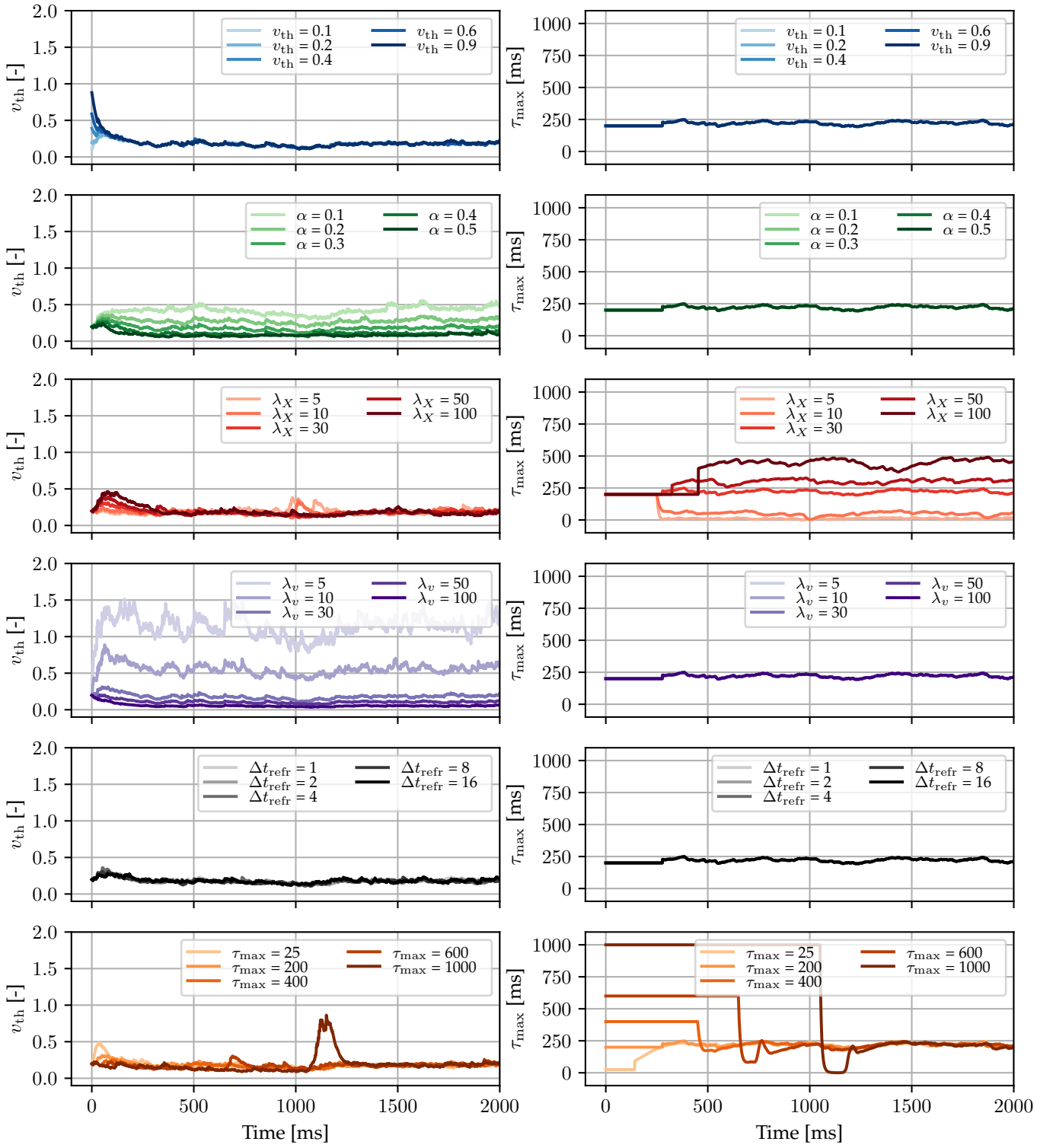


Fig. 25: History of the voltage threshold v_{th} and maximum synaptic delay $\tau_{\max}$ during one sequence of the rotating disk data employing the proposed update rules.

9.1.3 Presynaptic Trace Time Constant

In section 4 it was established that choosing a value of the presynaptic trace time constant λ_X which is too small for the temporal dynamics of the input, diminishes the penalizing effect of the presynaptic trace on frequently spiking neurons. Furthermore, it was explained that choosing a value that is too large would lead to an overall decease in spiking activity. However, these predicted effects cannot be observed in Figure 24, despite the applied voltage threshold update rule. Instead, the output appears to behave similarly to when the maximum synaptic delay is adjusted. The reason for this becomes clear when observing the history of $\tau_{\max}$ in the third row of Figure 25. Although λ_X does theoretically not affect the relative optic flow magnitude, $\tau_{\max}$ takes on different values for varying values of the maximum synaptic delay. This undesired effect is a result of the coupling between N_{AD} and λ_X as shown in subsection 8.5. Since the presynaptic trace takes longer to decay to the threshold $c_{j_{th}}$ for larger values of λ_X, the width of the full part of the presynaptic trace increases. This in return, leads to larger estimates of the relative optic flow magnitudes and thus to increases in $\tau_{\max}$. Consequently, the expected behavior of the output is overshadowed by the changes in $\tau_{\max}$. In practice this coupling can lead to erroneous $\tau_{\max}$ updates if the value of λ_X does not match the temporal dynamics of the input well.

9.1.4 Voltage Time Constant

The main function of λ_v is to control the temporal dynamics of the modified LIF neurons. In section 4 we assumed that smaller values of λ_v should thus result in outputs in which neurons with higher input spike frequencies should become more active and vice versa. However, this effect was completely overshadowed by the overall decrease in the spiking activity for increasing values of λ_v. Looking at the fourth row of Figure 25, it can be seen that once again the v_{th} update rule counteracts this effect by decreasing the voltage threshold for larger values of λ_v. Accordingly, the main effect of λ_v now becomes visible in the fourth row of Figure 24. For small values of λ_v the membrane potential decays so quickly that only neurons with fast input dynamics at the outer edges of the image can still produce output spikes. As the value of λ_v increases, the membrane potential decays less quickly and also neurons in the center parts of the image with slower input dynamics can produce output spikes. Since λ_v does not affect the presynaptic trace, there is no coupling between this parameter and the $\tau_{\max}$ update rule.

9.1.5 Refractory Period

In section 4 it was shown that there appears to be no major coupling between the refractory period and the remaining parameters. Accordingly, the update rules of both v_{th} and $\tau_{\max}$ are largely unaffected by changes in this parameter. While there are slight differences in the history of the voltage threshold, it always fluctuates around the same value. These small differences can be attributed to the fact the refractory period affects the density of the output spikes, which in return can have a minor effect on the synaptic stiffness.

9.1.6 Maximum Synaptic Delay

In the last rows of Figure 24 and Figure 25 it can be seen that, similarly to adapting the voltage threshold, adapting the maximum synaptic delay no longer affects the output. Regardless of the initial maximum synaptic delay, $\tau_{\max}$ is always driven to the same value. This is the case since the temporal spacing within the presynaptic trace is now controlled with the relative optic flow magnitude as was shown in section 8. In the history of $\tau_{\max}$ it can be seen that $\tau_{\max}$ stays constant for a while before it starts adapting. This is the case since the update rule is only performed once the delays have filled up at $t > \tau_{\max}$. Furthermore, initial oscillations can be observed for larger values of $\tau_{\max}$. These can be attributed to the fact the $\tau_{\max}$ update is multiplied with the current value of the maximum synaptic delay. This shall account for the fact that the dimensionless approximation of the relative optic flow magnitude is directly translated into a delay and shall avoid the need for different gains for different temporal input dynamics. However, in this case it effectively changes the learning rate since the same input is used and only the value of $\tau_{\max}$ is varied. The peaks in the history of v_{th} are a result of the coupling between the synaptic stiffness and the maximum synaptic delay at small values of $\tau_{\max}$ as explained in section 6.

Overall, it can be concluded that the proposed update rules achieve the desired decoupling for all parameters but the presynaptic trace time constant λ_X. Here, the coupling between the estimate of the relative optic flow magnitude and the presynaptic trace results in undesired changes of the maximum synaptic delay. When tuning the network, adjustments in λ_X will thus have to be accompanied with adjustments in A^{tar} to counteracts the undesired changes in $\tau_{\max}$.

9.2 Qualitative Results

As explained in subsubsection 7.3.2, the main limitation of the $\tau_{\max}$ update rule consists of the fact that it struggles with high spatial frequencies. However, commonly used event camera data sets such as MVSEC [82] typically contain very dense input. Consequently, we resort to using the ODA sequences presented in subsection 3.3 for evaluation. Since there is no official ground truth available for this dataset, we compute it with EV-FlowNet which has shown a high level of accuracy for event-based optic flow estimation [13]. We use the same weights and parameters as in all the previous analyses (see Table 4) but chose a stride of $s = 1$ in all network layers to increase the resolution for illustrative purposes.

Figure 26 depicts the time history of the parameters v_{th}, $\tau_{\max}$, S, and A_{pre_a} during one sequence of the ODA dataset. It can be seen that the update rules drive the voltage threshold of the SS-Conv layer down while driving the threshold in the MS-Conv layer up. Furthermore, the maximum synaptic delay is reduced when compared with the non-adapted value. Figure 27 and Figure 28 show the effect of these changes on the network output at several different times within the sequence. Most of the presented examples correspond to the end of the sequence when the drone approaches the truss structure depicted in the back of the right image in Figure 4. This is the case since at this

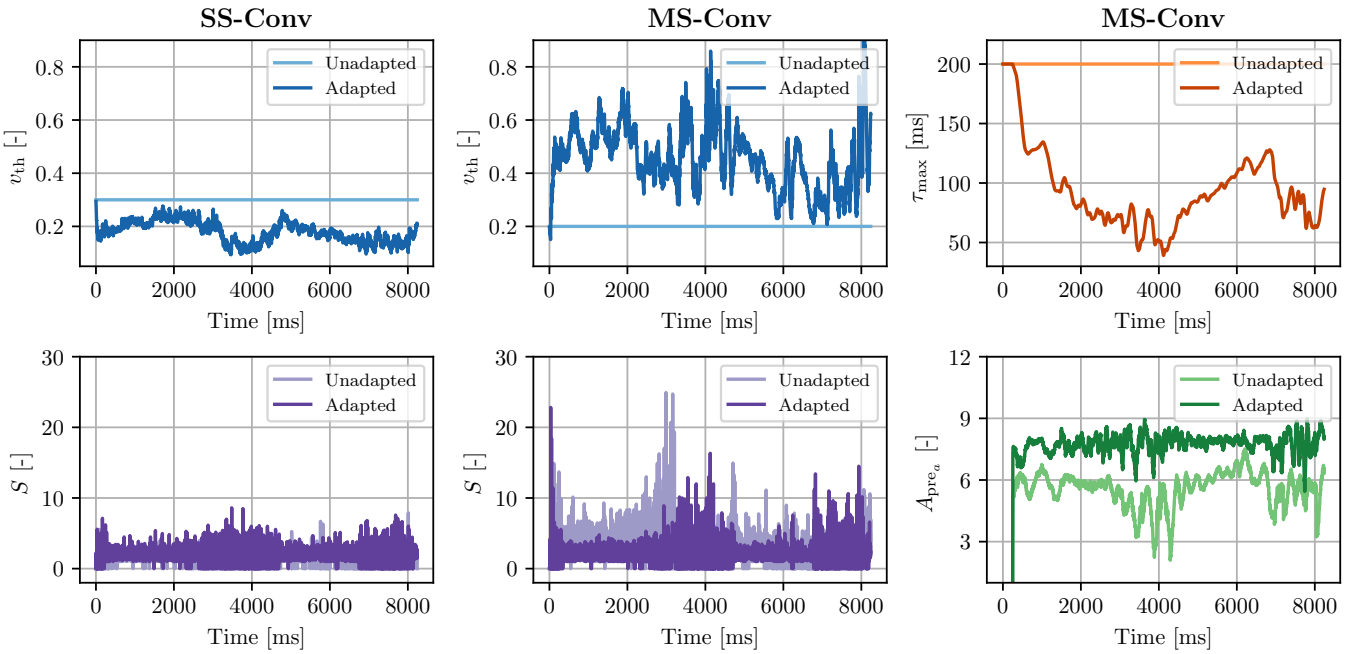


Fig. 26: History of the adapted and newly introduced parameters during one sequence of the ODA dataset.

point the density of input events reduces due to the lower contrast in the image. Consequently, the effect of the update rules can best be observed at this time. While the MS-Conv threshold is larger for the adapted sequences, the reduction of the threshold in the SS-Conv layer allows the network to capture features which it has previously missed. In fact, some features are even better captured than in the ground truth computed with EV-FlowNet. Reversely, towards the end of the sequence, the increase in the voltage threshold leads to a reduction in erroneous output spikes, as can be seen in the bottom image of Figure 27.

In section 4 it was shown that a maximum synaptic delay which is too large for the temporal dynamics of the input leads to a stretched out 'broken up' appearance of the edges in the output. This phenomenon can be observed in the non-adapted MS-Conv output images in Figure 28. Furthermore, the number of active delays shows very wide 'edges' which can also be attributed to a value of τ_{max} which is too large (see section 7). Both of these effects can no longer be observed in the images corresponding to the adapted output, implying that the IP update rules have successfully driven the maximum synaptic delay to a more suitable value.

9.3 Quantitative Results

Given the provided ground truth, a meaningful quantitative evaluation of the proposed update rules is a highly challenging task. However, to lay the foundation for future research, this section highlights the exact problems that arise and shows possible solutions that were investigated.

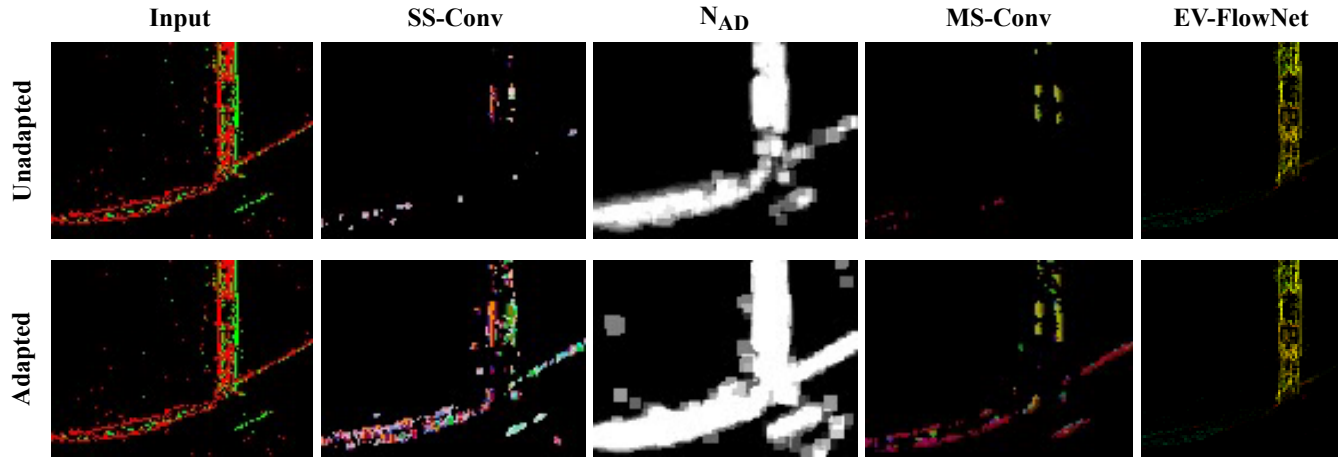
The quantitative analysis has identified that the proposed voltage threshold update rule performs especially well for sparse input data when compared to the unadapted model. However, for sparse input data EV-FlowNet provides close-to-zero optic flow estimates [13] which means that this strong point cannot be evaluated in a quantitative manner. While this limitation can be attributed to the choice of ground truth data, there are other issues which are inherent to the properties of the SNN analyzed in this research.

First of all, our SNN does not provide numerical optic flow estimates. Instead, motion direction is encoded in the identity of the spiking output maps. As explained in subsection 3.3 the optic flow vectors corresponding to the output maps were approximated using the variant of the EdgeFlow histogram matching method presented in [23]. However, unlike most available ground truth data, this method does not encode optic flow as pixel-wise displacements. Consequently, a correction factor, cf, was applied mapping our optic flow estimates to pixel-wise displacements. Assuming a constant ratio between the pixel displacement and the computed optic flow magnitude, we use the following correction factor:

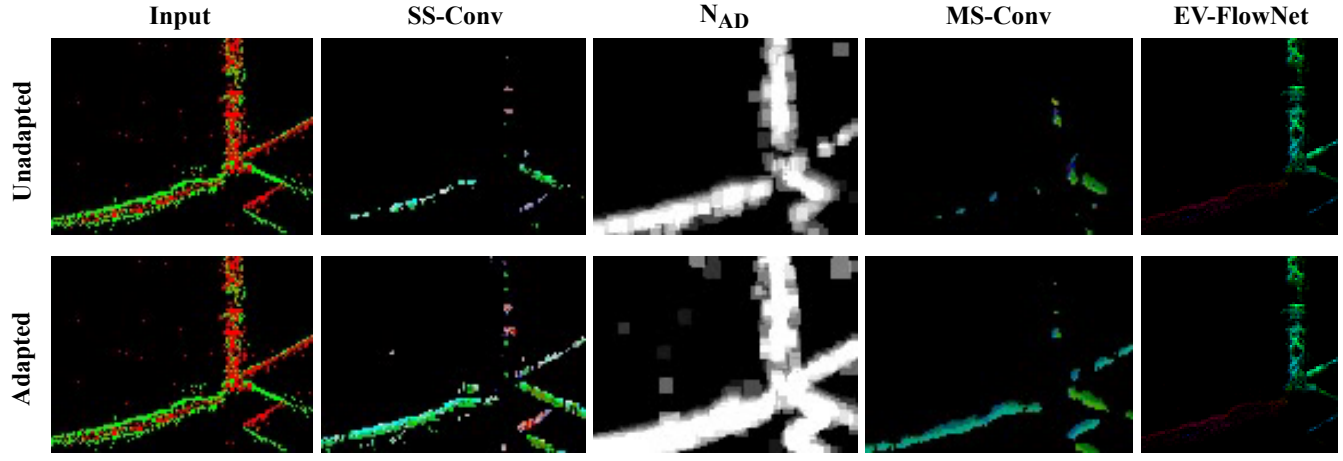
$$cf = \frac{N_{\text{px}} \cdot dt_{\text{EV}}}{\bar{OF} \cdot \tau_{\text{max}}} \tag{21}$$

where N_{px} represents the manually determined pixel displacement for one specific map, $\bar{OF}$, the corresponding computed optic flow magnitude and dt_{EV} the time step used for the evaluation with EV-FlowNet. For convenience, the pixel-wise displacement was determined for the third map, which primarily contains vertical motion. Visual inspection of Figure 6 gives $N_{\text{px}} \approx 1$ px.

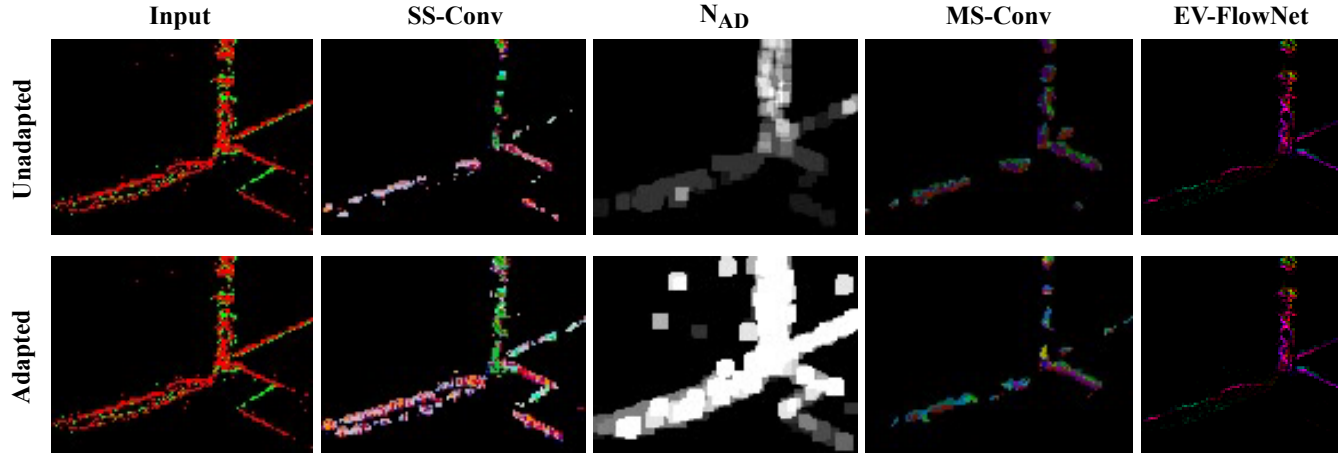
Another difficulty lies in the different time steps applied. Most existing ground truth data for event-based optic flow estimation provide estimates in frequencies which correspond to frame rates in conventional cameras. However, the network proposed in this work utilizes a much smaller time step of $dt = 1$ ms. Employing a larger time step would require the use of a different set of parameters and it would not be clear if the results could be translated back in a meaningful way. Buffering the output spikes on the other



(a) t = 5963 ms

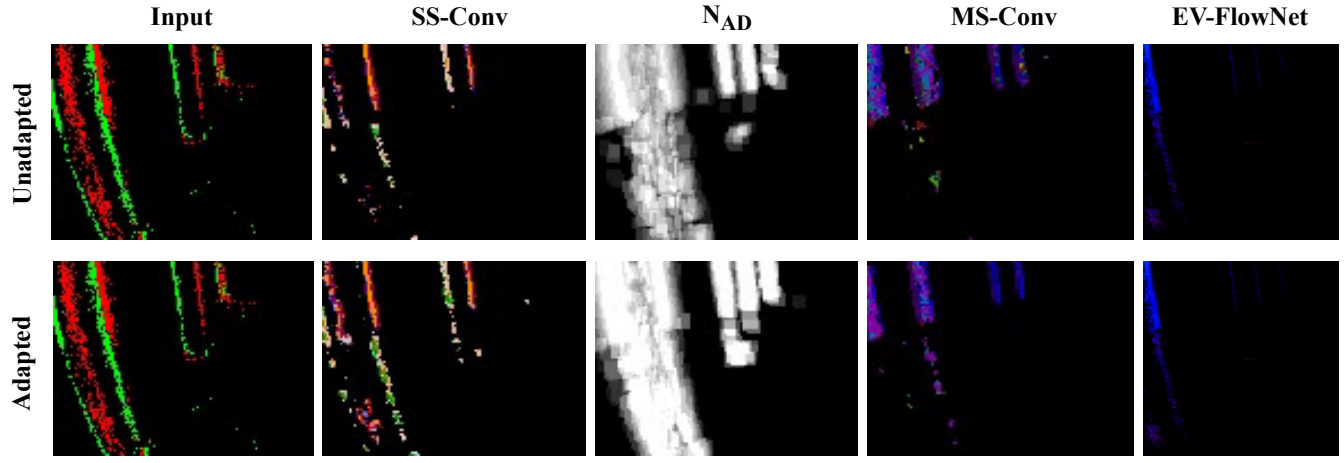


(b) t = 6355 ms

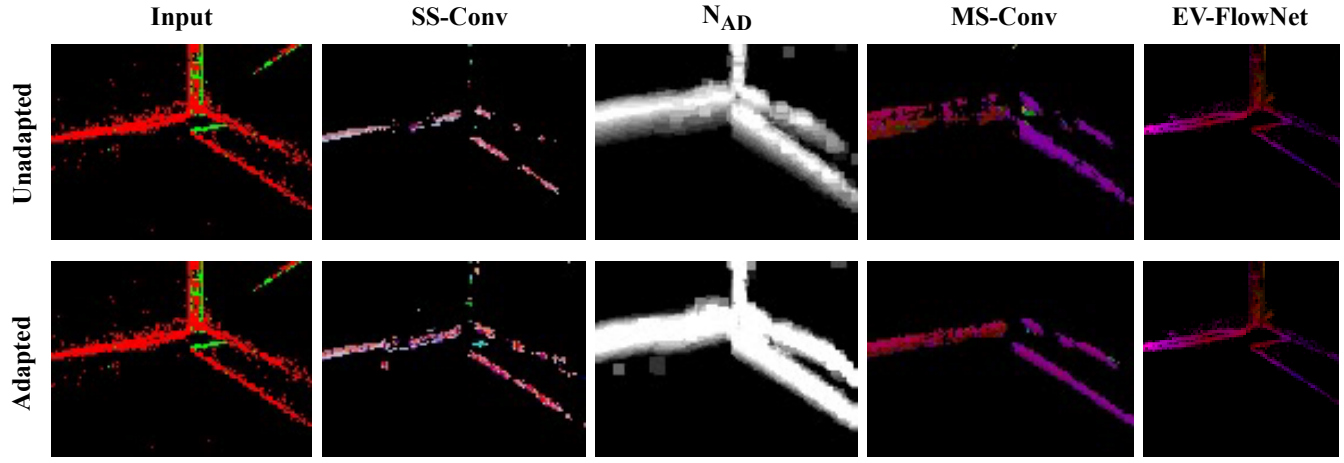


(c) t = 8070 ms

Fig. 27: SNN output employing the proposed intrinsic plasticity update rules at different times during one sequence of the ODA dataset. The first column shows the input to the network, the second one the SS-Conv output, the third one the number of active delays N_{AD} within the folded presynaptic trace of the MS-Conv layer (brighter pixels represent larger values), the fourth column the MS-Conv output and the fith column the output obtained with EV-FlowNet.



(a) t = 922 ms



(b) t = 4923 ms

Fig. 28: SNN output employing the proposed intrinsic plasticity update rules at different times during one sequence of the ODA dataset. The first column shows the input to the network, the second one the SS-Conv output, the third one the number of active delays N_{AD} within the folded presynaptic trace of the MS-Conv layer (brighter pixels represent larger values), the fourth column the MS-Conv output and the fith column the output obtained with EV-FlowNet.

hand, results in output features which are much thicker when compared to typical ground truth data which makes a pixel-wise comparison challenging.

In addition, existing networks for event-based optic flow estimation utilize some sort of back-propagation, which ensures that the output represents the input data in a pixel-wise manner. However, the SNN in this work only utilizes feed-forward information processing and several delays are introduced within the network architecture. This means that the output at any time t does not exactly represent the input at the same time. However, quantifying the exact shift is not straight-forward since the output is not only shifted in time but also depends on the history of the input due to the multisynaptic delays. In fact, as shown in section 4 an unsuitable choice of $\tau_{\max}$ can 'stretch out' the output and result in output spikes in retinotopic locations in which the input features have already passed.

Taking into account all of the above challenges, we do not believe that a quantitative pixel-by-pixel analysis can be performed in a meaningful way with the available ground truth. However, to illustrate some of the above issues, hereafter we present some examples of attempted comparisons. Figure 30 illustrates the average optic flow magnitude during one ODA sequence computed with EV-FlowNet and our network. On first sight, it can be seen that the magnitudes of the optic flow estimates are limited to a narrow range if the proposed update rules are not employed. While the magnitudes computed with the adapted maximum synaptic delay appear to show some bias which can probably be attributed to imperfection in the correction factor, they seem to follow the general trend of the ground truth up to $t \approx 2000$ ms. After that, the two lines dramatically diverge. The results from EV-FlowNet imply that the optic flow approaches zero from that time on. However, visual inspection of the input sequence shows that this is not the case and that the input merely becomes more sparse at this point in time. This

Input Difference MS-Conv Output

Fig. 29: Illustration of pixel-wise correspondences between the input and output image. The left and right image illustrate an example input and MS-Conv output, respectively. In the center image pixels that are active in the input, but not active in the output image are indicated in white.

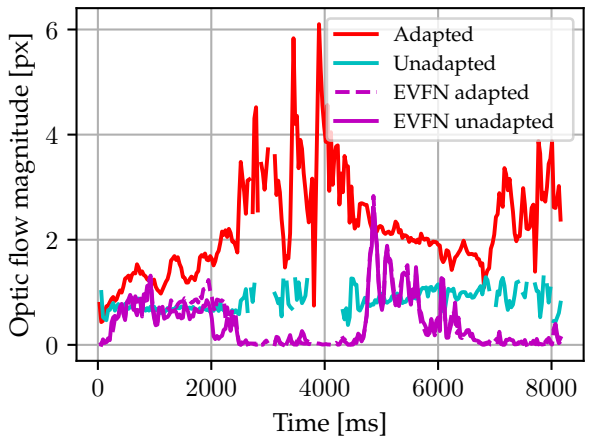


Fig. 30: Average optic flow magnitude during one ODA sequence computed with our network and EV-FlowNet. For comparison, only pixels for which estimates are available in our network are taken into account. EVFN adapted and EVFN unadapted represent the estimates provided by EV-FlowNet averaged over pixels which are active in the adapted and unadapted model, respectively. Discontinuities in the graphs correspond to times at which no estimates are available.

implies that the low estimates can be attributed to EV-FlowNets tendency to output zero-flows for sparse input data. The same observation can be made towards the end of the sequence. As a result, the unadapted model would perform better in a quantitative comparison. However, this stands in contrast to the qualitative results, which show a clear improvement when using the proposed update rules. It is however noteworthy that the adapted model appears to always produce higher optic flow magnitude estimates when EV-FlowNets estimates approach zero. While this could imply that the proposed update rules struggles with sparse input data, no clear conclusions can be made due to the lack of reliable ground truth for the corresponding segments.

Similarly, Figure 29 illustrates the difficulty of performing pixel-wise comparisons. To quantify the performance of the voltage threshold, we would like to compute what fraction of the neurons that are active in the input image are also active in the output. However, due to the changes in feature thickness and slight shift in time, the computed difference is much larger than anticipated. The bottom part of the pillar e.g. is clearly captured in the output image. However, since it is slightly shifted to the top, it is indicated as missing in the difference picture. This effect also makes pixel-wise comparisons of the optic flow value non-meaningful.

It could be attempted to overcome this issue by trying to identify the shift between the images or by performing a comparison which is not conducted in a pixel-wise manner. Furthermore, the performance of the update rules could be evaluated by computing global motion estimates, which are scalar values and would thus be easier to compare. However, the exact implementation of such an approach is beyond the scope of this paper.

10 Conclusion

In this paper we have presented a detailed analysis of the neuron parameters of the SNN for optic flow estimation presented in [23]. Based on this analysis, we have proposed novel update rules which adapt the voltage threshold and the maximum synaptic delay during inference. This reduces the amount of coupling between parameters, and makes the network more robust to changes in the input dynamics. Consequently, the proposed update rules ease the process of tuning the network and allow applying an existing set of weights to a larger range of operating environments. While quantitative results have shown a high level of robustness for the adaptation of the voltage threshold, the adaptation of the maximum synaptic delay is coupled to the presynaptic time constant and does not work well for inputs containing high spatial frequencies. However, both of these limitations are caused by the fact that the width of the presynaptic trace edge is approximated by averaging the number of active delays. For future research it might thus be interesting to investigate different means of approximating this width. Furthermore, developing measures for a quantitative evaluation of the proposed update rules could provide deeper insights into their exact effects on the network performance. Finally, the use of life-long learning for the synaptic weights could be investigated to achieve even higher levels of robustness.

REFERENCES

[1] R. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *Journal of International Humanitarian Action*, vol. 3, Dec. 2018. [Online]. Available: https://doi.org/10.1186/s41018-018-0045-4

[2] M. Shafiee, Z. Zhou, L. Mei, F. Dinmohammadi, J. Karama, and D. Flynn, "Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis," *Robotics*, vol. 10, p. 26, Feb. 2021.

[3] N. Hallermann and G. Morgenthal, "Visual inspection strategies for large bridges using unmanned aerial vehicles (uav)," in *IABMAS 2014, Shanghai*, Jul. 2014.

[4] M. P. Christiansen, M. Laursen, R. Jørgensen, S. Skovsen, and R. Gislum, "Designing and testing a UAV mapping system for agricultural field surveying," *Sensors*, vol. 17, p. 2703, Nov. 2017.

[5] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, and G. C. H. E. de Croon, "Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 9099–9106.

[6] M. V. Srinivasan, "Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics," *Physiological Reviews*, vol. 91, no. 2, pp. 413–460, 2011, pMID: 21527730. [Online]. Available: https://doi.org/10.1152/physrev.00005.2010

[7] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[8] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[9] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 2333–2341, 2014.

[10] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural networks : the official journal of the International Neural Network Society*, vol. 27, pp. 32–7, Nov. 2011.

[11] R. Benosman, C. Clercq, X. Lagorce, S. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, Feb. 2014.

[12] B. Hordijk, K. Scheper, and G. Croon, "Vertical landing for micro air vehicles using event-based optical flow," *Journal of Field Robotics*, vol. 35, pp. 69–90, Jan. 2018.

[13] A. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," Jun. 2018.

[14] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers in Neuroscience*, vol. 12, p. 774, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00774

[15] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[16] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[17] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[18] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[19] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144–164, Jan. 2019.

[20] L. Salt, D. Howard, G. Indiveri, and Y. Sandamirskaya, "Parameter optimization and learning in a spiking neural network for uav obstacle avoidance targeting neuromorphic processors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3305–3318, 2020.

[21] M. B. Milde, O. J. N. Bertrand, H. Ramachandran, M. Egelhaaf, and E. Chicca, "Spiking Elementary Motion Detector in Neuromorphic Systems," *Neural Computation*, vol. 30, no. 9, pp. 2384–2417, Sep. 2018. [Online]. Available: https://doi.org/10.1162/neco_a_01112

[22] G. Haessig, A. Cassidy, R. Alvarez-Icaza, R. Benosman, and G. Orchard, "Spiking optical flow for event-based sensors using ibm's truenorth neurosynaptic system," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, Oct. 2017.

[23] F. Paredes-Vallés, K. Y. W. Scheper, and 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*, vol. 42, no. 8, pp. 2051–2064, Mar. 2020.

[24] J. Hagenaars, F. Paredes-Valles, and G. de Croon, "Self-supervised learning of event-based optical flow with spiking neural networks," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 7167–7179. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/39d4b545fb02556829aab1db805021c3-Paper.pdf

[25] W. Reichardt, "Autocorrelation, a principle for evaluation of sensory information by the central nervous system," in *Sensory communication*, W. Rosenblith, Ed. Cambridge, MA, USA: MIT Press, 1961, pp. 303–317. [Online]. Available: http://hdl.handle.net/11858/00-001M-0000-0013-F2B6-B

[26] A. Borst and M. Egelhaaf, "Temporal modulation of luminance adapts time constant of fly movement detectors," *Biol. Cybern.*, vol. 56, no. 4, p. 209–215, Jun. 1987. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1007/BF00365215

[27] R. R. de Ruter van Steveninck, W. H. Zaagman, and H. A. K. Mastebroek, "Adaptation of transient responses of a movement-sensitive neuron in the visual system of the blowfly calliphora erythrocephala," *Biol. Cybern.*, vol. 54, no. 4–5, p. 223–236, Aug. 1986. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1007/BF00318418

[28] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[29] C. Farabet, R. Paz-Vicente, J. Pérez-Carrasco, C. Zamarreño-Ramos, A. Linares-Barranco, Y. Lecun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing," *Frontiers in Neuroscience*, vol. 6, pp. 1–12, Apr. 2012.

[30] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco, "Event-driven sensing and processing for high-speed robotic vision," in *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, Oct. 2014, pp. 516–519.

[31] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Bulletin of Mathematical Biology*, vol. 52, no. 1, pp. 25 – 71, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0092824005800047

[32] C. E. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber." *Biophysical journal*, vol. 35 1, pp. 193–213, 1981.

[33] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane." *Biophysical journal*, vol. 1 6, pp. 445–66, 1961.

[34] E. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[35] R. B. Stein, "A theoretical analysis of neuronal variability." *Biophysical journal*, vol. 5, pp. 173–94, 1965.

[36] W. M. Kistler, W. Gerstner, and J. L. v. Hemmen, "Reduction of the hodgkin-huxley equations to a single-variable threshold model," *Neural Computation*, vol. 9, no. 5, pp. 1015–1045, 1997.

[37] M. Baudry, "Synaptic plasticity and learning and memory: 15 years of progress," *Neurobiology of Learning and Memory*, vol. 70, pp. 113–118, 1998.

[38] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological cybernetics*, vol. 98, pp. 459–78, Jul. 2008.

[39] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. New York: Wiley, Jun. 1949.

[40] A. Vigneron and J. Martinet, "A critical survey of stdp in spiking neural networks for pattern recognition," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[41] T. Iakymchuk, A. Rosado, J. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Frances-Villora, "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 4, Dec. 2015. [Online]. Available: https://doi.org/10.1186/s13640-015-0059-4

[42] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks," *Pattern Recogn.*, vol. 94, no. C, p. 87–95, Oct. 2019. [Online]. Available: https://doi.org/10.1016/j.patcog.2019.05.015

[43] P. Panda, J. Allred, S. Ramanathan, and K. Roy, "Asp: Learning to forget with adaptive synaptic plasticity in spiking neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. PP, Mar. 2017.

[44] J. Allred and K. Roy, "Controlled forgetting: Targeted stimulation and dopaminergic plasticity modulation for unsupervised lifelong learning in spiking neural networks," *Frontiers in Neuroscience*, vol. 14, Jan. 2020.

[45] R. V. W. Putra and M. Shafique, "Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1057–1062.

[46] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3227–3235, Jul. 2018.

[47] J. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, Aug. 2016.

[48] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/82f2b308c3b01637c607ce05f52a2fed-Paper.pdf

[49] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00119

[50] Z. Wang, Y. Zhang, H. Shi, L. Cao, C. Yan, and G. Xu, "Recurrent spiking neural network with dynamic presynaptic currents based on backpropagation," *International Journal of Intelligent Systems*, vol. 37, no. 3, pp. 2242–2265, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22772

[51] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf

[52] M. Kim and P. Smaragdis, "Bitwise neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML), Workshop on Resource-Efficient Machine Learning*, Lille, France, Jul. 2015.

[53] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer International Publishing, 2016, pp. 525–542.

[54] J. 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 feed forward convnets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2706 – 2719, Nov. 2013.

[55] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/10a5ab2db37feedfdeaab192ead4ac0e-Paper.pdf

[56] D. Zambrano, R. Nusselder, H. S. Scholte, and S. M. Bohté, "Efficient computation in adaptive artificial spiking neural networks," *CoRR*, vol. abs/1710.04838, 2017. [Online]. Available: http://arxiv.org/abs/1710.04838

[57] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2017.00682

[58] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.

[59] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in Neuroscience*, vol. 13, Feb. 2018.

[60] R. Florian, "A reinforcement learning algorithm for spiking neural networks," in *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*, 2005, p. 8.

[61] Z. Bing, I. Baumann, Z. Jiang, K. Huang, C. Cai, and A. Knoll, "Supervised learning in snn via reward-modulated spike-timing-dependent plasticity for a target reaching vehicle," *Frontiers in Neurorobotics*, vol. 13, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00018

[62] T. Stewart, A. Kleinhans, A. Mundy, and J. Conradt, "Serendipitous offline learning in a neuromorphic robot," *Frontiers in Neurorobotics*, vol. 10, Feb. 2016.

[63] I. Krauhausen, D. A. Koutsouras, A. Melianas, S. T. Keene, K. Lieberth, H. Ledanseur, R. Sheelamanthula, A. Giovannitti, F. Torricelli, I. Mcculloch, P. W. M. Blom, A. Salleo, Y. van de Burgt, and P. Gkoupidenis, "Organic neuromorphic electronics for sensorimotor integration and learning in robotics," *Science Advances*, vol. 7, no. 50, p. eabl5068, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abl5068

[64] A. Lazar, G. Pipa, and J. Triesch, "Fading memory and time series prediction in recurrent networks with different forms of plasticity," *Neural Networks*, vol. 20, no. 3, pp. 312–322, 2007, echo State Networks and Liquid State Machines. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608007000469

[65] X. Li, W. Wang, F. Xue, and Y. Song, "Computational modeling of spiking neural network with learning rules from stdp and intrinsic plasticity," *Physica A: Statistical Mechanics and its Applications*, vol. 491, pp. 716–728, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437117307896

[66] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 795–805.

[67] R. Baddeley, L. Abbott, M. Booth, F. Sengpiel, T. Freeman, E. Wakeman, and E. Rolls, "Responses of neurons in primary and inferior temporal visual cortices to natural scenes," *Proceedings. Biological sciences / The Royal Society*, vol. 264, pp. 1775–83, Jan. 1998.

[68] C. Li and Y. Li, "A spike-based model of neuronal intrinsic plasticity," *IEEE Transactions on Autonomous Mental Development*, vol. 5, no. 1, pp. 62–73, 2013.

[69] W. Zhang and P. Li, "Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks," *Frontiers in Neuroscience*, vol. 13, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2019.00031

[70] A. Zhang, Y. Gao, Y. Niu, X. Li, and Q. Chen, "Intrinsic plasticity for online unsupervised learning based on soft-reset spiking neuron model," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2020.

[71] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of*

the 7th International Joint Conference on Artificial Intelligence - Volume 2, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 674–679.

[72] B. Rueckauer and T. Delbruck, "Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor," *Frontiers in Neuroscience*, vol. 10, p. 176, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2016.00176

[73] M. T. Aung, R. Teo, and G. Orchard, "Event-based plane-fitting optical flow for dynamic vision sensors in fpga," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[74] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Frontiers in neuroscience*, vol. 9, p. 137, Apr. 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2015.00137

[75] S. Tschechne, R. Sailer, and H. Neumann, "Bio-inspired optic flow from event-based neuromorphic sensor input," in *Artificial Neural Networks in Pattern Recognition*, N. El Gayar, F. Schwenker, and C. Suen, Eds. Cham, Switzerland: Springer International Publishing, 2014, pp. 171–182.

[76] F. Barranco, C. Fermuller, and Y. Aloimonos, "Bio-inspired motion estimation with event-driven sensors," in *Advances in Computational Intelligence*, I. Rojas, G. Joya, and A. Catala, Eds. Cham, Switzerland: Springer International Publishing, Jun. 2015, pp. 309–321.

[77] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4465–4470.

[78] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3867–3876.

[79] A. Mitrokhin, C. Fermuller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018. [Online]. Available: https://doi.org/10.1109%2Firos.2018.8593805

[80] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 989–997.

[81] 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 *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5831–5838.

[82] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3d perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[83] F. Paredes-Vallés and G. C. H. E. de Croon, "Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3445–3454.

[84] Z. Li, J. Shen, and R. Liu, "A lightweight network to learn optical flow from event data," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 1–7.

[85] C. Lee, A. K. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 366–382.

[86] C. Lee, A. K. Kosta, and K. Roy, "Fusion-FlowNet: energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures," 2021. [Online]. Available: https://arxiv.org/abs/2103.10592

[87] M. Giulioni, X. Lagorce, F. Galluppi, and R. Benosman, "Event-based computation of motion flow on a neuromorphic analog neural platform," *Frontiers in Neuroscience*, vol. 10, Feb. 2016.

[88] C. Richter, F. Röhrbein, and J. Conradt, "Bio-inspired optic flow detection using neuromorphic hardware," Sep. 2014. [Online]. Available: https://mediatum.ub.tum.de/doc/1281617/789727.pdf

[89] L. Salt, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance with lgmd neuron: Towards a neuromorphic uav implementation," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[90] C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. E. Mahony, and D. Scaramuzza, "Fast image reconstruction with an event camera," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 156–163.

[91] C. Pehle and J. E. Pedersen, "Norse - A deep learning library for spiking neural networks," Jan. 2021, documentation: https://norse.ai/docs/. [Online]. Available: https://doi.org/10.5281/zenodo.4422025

[92] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[93] R. Dinaux, N. Wessendorp, J. Dupeyroux, and G. C. H. E. d. Croon, "Faith: Fast iterative half-plane focus of expansion estimation using optic flow," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7627–7634, 2021.

[94] K. McGuire, G. de Croon, C. de Wagter, B. Remes, K. Tuyls, and H. Kappen, "Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3255–3260.

APPENDIX

This Appendix provides supplementary material to the analysis performed in section 7 investigating the presynaptic trace of an artificial event sequence of a horizontal bar moving vertically through space. In particular, the history of the number of active delays N_{AD} within the presynaptic trace was investigated. Figure 31 and Figure 32 present N_{AD} as a function of time and pixel distance, respectively. Unlike Figure 16, they present the full presynaptic trace histories for all six investigated parameter combinations. Please refer to section 7 for additional information.

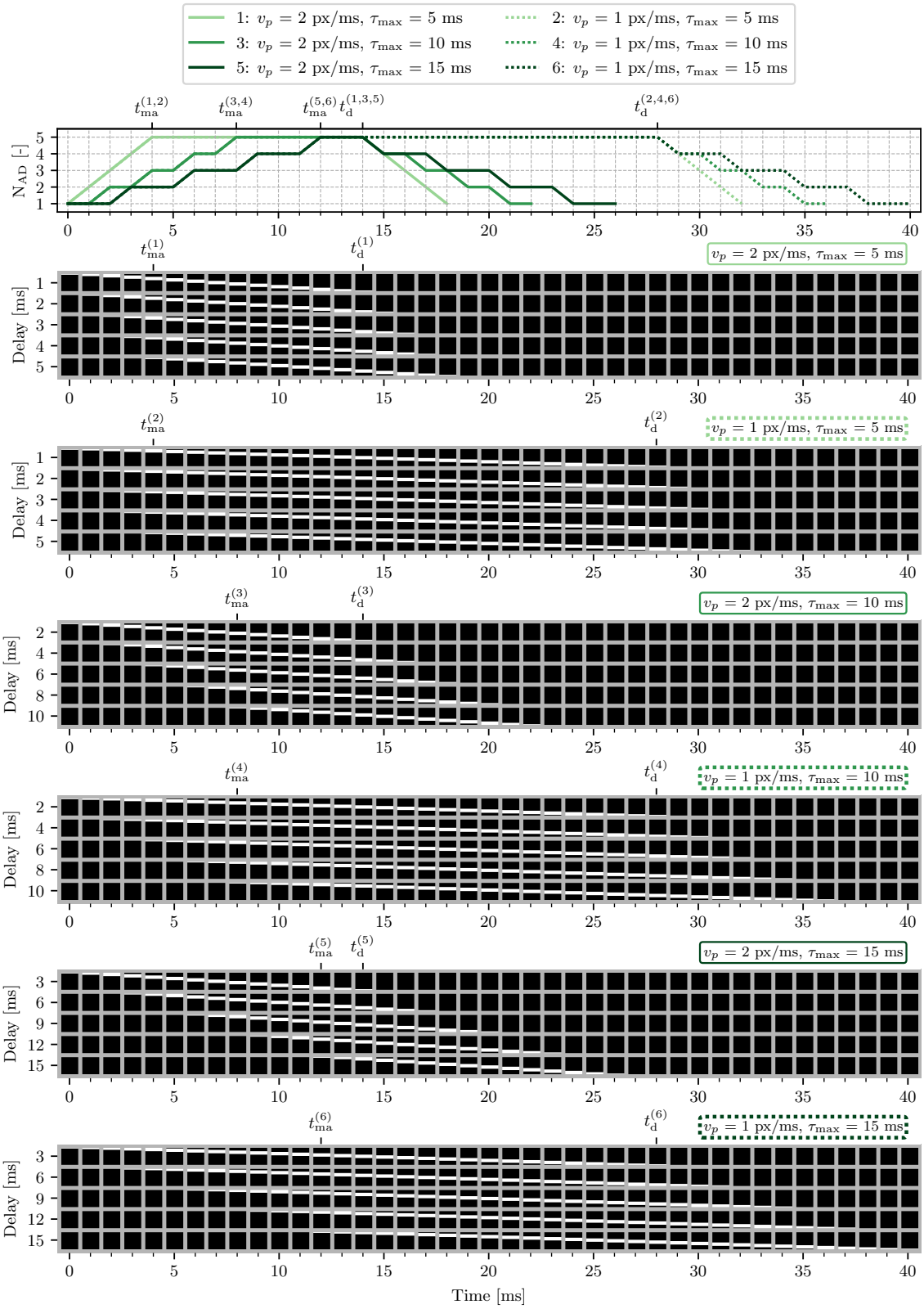


Fig. 31: Evolution of the presynaptic trace as a function of time. The number of active delays N_{AD} is shown in the top half for six different combinations of the pixel velocity v_p and the maximum synaptic delay $\tau_{\max}$. The bottom part illustrates the corresponding full presynaptic trace histories. The figures were created with a presynaptic trace windows size of $r = 25$ px, a bar thickness of $\mathrm{th}_X = 5$ px and $d = 5$ delays. For illustrative purposes, the delays are not spaced within $[1, \tau_{\max}]$ ms, but within the range $[\tau_{\max}/m, \tau_{\max}]$ ms.

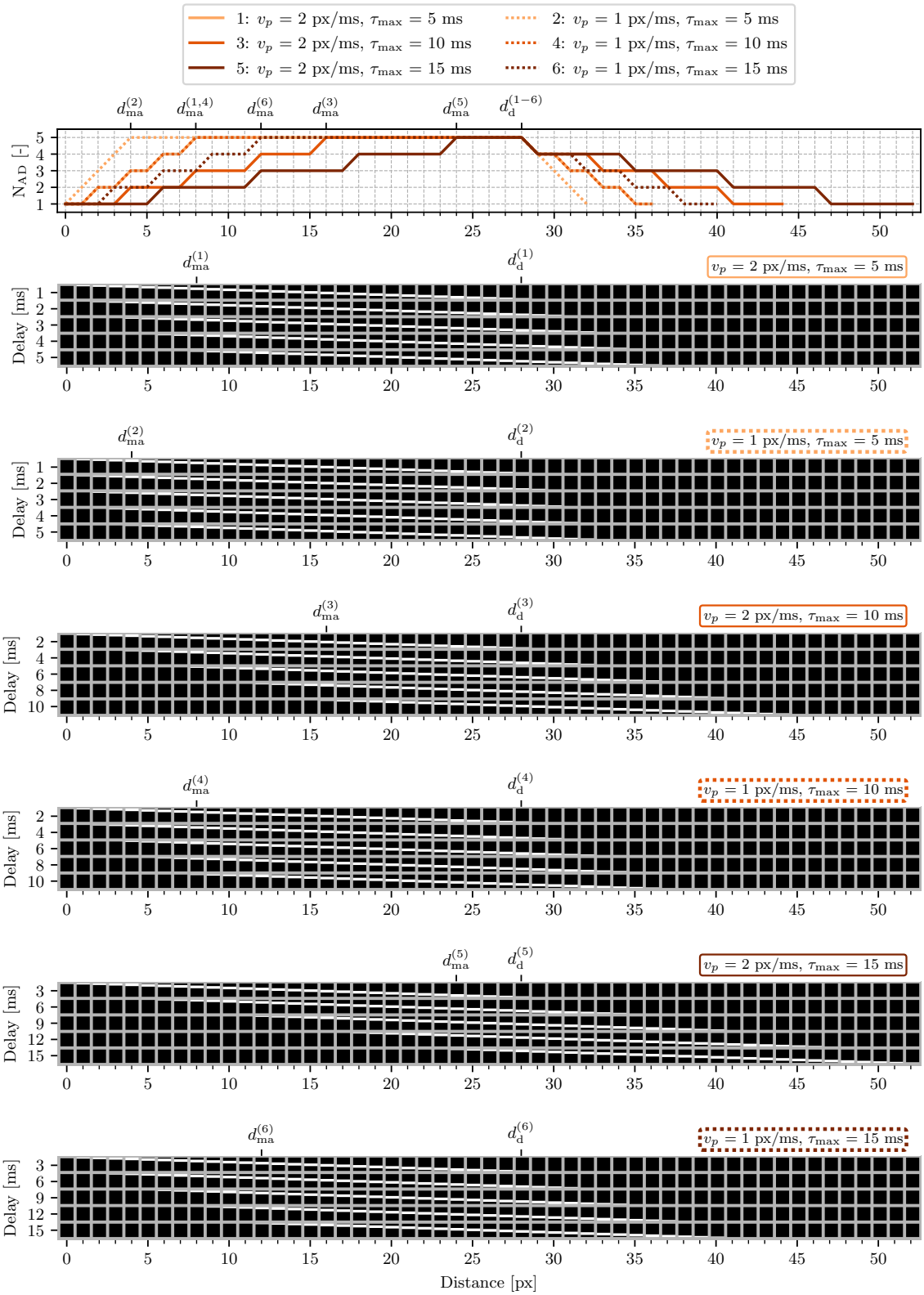


Fig. 32: Evolution of the presynaptic trace as a function of pixel distance. The number of active delays N_{AD} is shown in the top half for six different combinations of the pixel velocity v_p and the maximum synaptic delay $\tau_{\max}$. The bottom part illustrates the corresponding full presynaptic trace histories. The figures were created with a presynaptic trace windows size of $r = 25$ px, a bar thickness of $\mathrm{th}_X = 5$ px and d = 5 delays. For illustrative purposes, the delays are not spaced within $[1, \tau_{\max}]$ ms, but within the range $[\tau_{\max}/m, \tau_{\max}]$ ms.

II

Literature Study

2

Foundations of Optic Flow

Optic flow (OF) contains rich information about the ego-motion[1] of an observer and the 3D structure of their surroundings. Insects are known to heavily rely on this information when performing navigational tasks such as centering and wall-following, landing, or obstacle avoidance (OA). As this research aims to draw inspiration from insects to perform more robust OF estimation and ultimately OA onboard MAVs, the foundations of OF are presented in this chapter. First, in section 2.1 the concept of OF is explained in more detail. Subsequently, a mathematical description is provided in section 2.2 and the so-called OF-observables, are introduced in section 2.3. Finally, in section 2.4 commonly used OF estimation methods are presented and it is explained why they are not suitable for the use onboard MAVs.

2.1. Optic Flow Definition

Gibson (1979) proposed the idea that perception is a direct process that does not require transforming or supplementing the data received from the eye. Instead, he suggests that the eye perceives a complex pattern of light, the optical array, which constantly changes as we move. Consequently, perception and movement are intertwined and the changes in the optical array contain information about the environment which enables us to perceive depth directly without the help of other visual cues. The quantity describing these changes is called optic flow (OF). It is defined as the set of vectors quantifying the motion of texture elements projected from the 3D surroundings onto the 2D retinal image of an observer. This is shown in Figure 2.1 (retrieved from (Gibson, 1979)) depicting a pilot's visual field during landing. The arrows represent the OF-vectors showing the displacement of image features in the 2D plane with the movement of the observer.

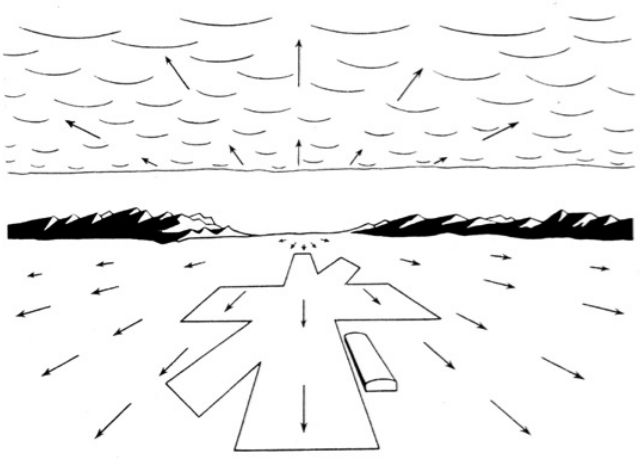


Figure 2.1: Illustration of optic flow in the visual field of a pilot during landing. Retrieved from Gibson (1979).

[1] Ego-motion refers to any spatial displacement of an observer within their environment.

2.2. Pinhole Camera Model

Longuet-Higgins and Prazdny (1980) proposed a mathematical model for the relationship between an observer's ego-motion and the corresponding induced OF on the retina. Their formulation models the projection of the 3D environment onto the retina using a pinhole camera model. This implies that the camera aperture [2] is represented as a single point and that the retina is modeled as a flat surface. It should be noted that this model cannot be applied to cameras with a wide field of view due to the introduced distortions. A visualization of the pinhole model can be seen in Figure 2.2 (retrieved from (Longuet-Higgins and Prazdny, 1980)). The origin O of the outer coordinate system OXYZ coincides with the location of a monocular observer's eye in a static environment. U, V, and W and A, B, and C represent the velocity and the rotation of the observer along and about the X, Y, and Z axes, respectively. The point P represents the location of a texture element in the visual field, while its projection onto the retinal plane p_0xy is denoted by p. The principal point $p_0 = (0,0,f)$ is the intersection between the optical axis OZ and the retinal plane. The focal length f is set to $f = 1$ in order to simplify the derivation.

Using these definitions, the velocity components, $\dot{X}$, $\dot{Y}$, and $\dot{Z}$ of P in the moving frame OXYZ induced by the observer's ego-motion can be computed by adding the contributions of the translational and rotational components of the movement. Doing this results in the expression shown in Equation 2.1.

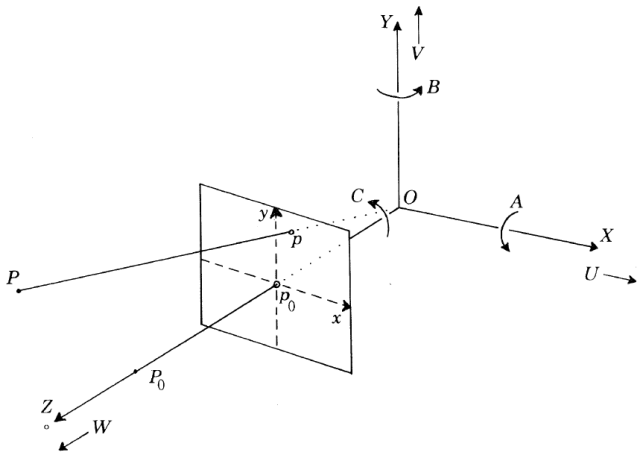


Figure 2.2: Illustration of the pinhole camera model. Retrieved from Longuet-Higgins and Prazdny (1980).

$$\begin{cases} \dot{X} = -U - BZ + CY \\ \dot{Y} = -V - CX + AZ \\ \dot{Z} = -W - AY + BX \end{cases} \tag{2.1}$$

OF can now be defined as the velocity (u, v) of the projected point p in the retinal plane as shown in Equation 2.2. Introducing the normalized image coordinates illustrated in Equation 2.3 and substituting Equation 2.1 and Equation 2.3 into Equation 2.2 an expression for the retinal velocity of the texture element can be obtained. The result of this is depicted in Equation 2.4, where it can be seen that the retinal velocities are comprised of a translational (u^T and v^T) and a rotational (u^R and v^R) component. These components are written out in Equation 2.5.

$$(u, v) = (\dot{x}, \dot{y}) \tag{2.2}$$

$$(x, y) = (X/Z, Y/Z) \tag{2.3}$$

$$\begin{cases} u = \dot{X}/Z - X\dot{Z}/Z^2 = (-U/Z - B + Cy) - x(-W/Z - Ay + Bx) = u^T + u^R \\ v = \dot{Y}/Z - Y\dot{Z}/Z^2 = (-V/Z - Cx + A) - y(-W/Z - Ay + Bx) = v^T + v^R \end{cases} \tag{2.4}$$

[2] Aperture is the opening through which light travels to enter a camera.

$$\begin{cases} u^T = (-U + xW)/Z, & v^T = (-V + yW)/Z, \\ u^R = -B + Cy + Axy - Bx^2, & v^R = -Cx + A + Ay^2 - Bxy \end{cases} \tag{2.5}$$

2.3. Optic Flow Observables

While it was explained that OF is a powerful tool to obtain information about an observer's ego-motion, it is not clear yet how this information can be retrieved from the formulation presented in section 2.2. The assumption of a static environment ensures that the ego-motion states (A, B, C, U, W, Z) are constant for all world points. However, this is not the case for the depth Z which can vary for different image locations. This means that attempting to solve a system of equations comprised of several OF-vectors for the ego-motion states, is a challenging and computationally expensive task which makes it unsuitable for the use onbard MAVs. However, by applying some simplifying assumptions, a new set of parameters, also called OF-observables can be derived. They describe quantities concerning the ego-motion of the observer and the structural properties of the environment. As these observables are widely used in the field of insect-inspired OA, their derivation and the required assumptions are presented in the remainder of this section.

2.3.1. Derotation

In section 2.2 it was established that OF comprises a translational and a rotational part. Considering once again Equation 2.5 it can be seen that the translational part does not depend on the observer's rotation A, B, C while the rotational component is independent of Z. Consequently, only the translational part is useful when trying to obtain depth information about objects in the environment. It is in fact common practice to correct the perceived OF onboard MAVs by removing the rotational component. This is often done with the help of light-weight inertial measurement units (IMU's) (e.g. (Beyeler et al., 2009; Zingg et al., 2010)). Other approaches separate the OF components physically by stabilizing the camera (e.g. (Meyer et al., 2016)) to counteract rotations or by employing a saccadic flight strategy (e.g. (Zufferey and Floreano, 2005; Bertrand et al., 2015)) which will be explained in more detail in chapter 3.

2.3.2. Orthogonal Flat Surface Assumption

Even after employing derotation to the OF vectors, extracting information about the observer's ego-motion from the OF formulation in Equation 2.5 is not a straightforward task. This is the case since every OF vector still introduces an additional unknown to the system of equations due to the varying values of Z. In order to circumvent this issue several authors assume orthogonal flat surfaces in the visual field (e.g. (Bertrand et al., 2015; Croon et al., 2013)). This way every additional OF vector provides more information about the ego-motion parameters without introducing additional unknowns. While the parameters U, V, and W can not be disentangled from the depth Z, the normalized quantities U/Z, V/Z, and W/Z, from now on denoted as ν_x, ν_y, and ν_z, can be estimated at a point i using Equation 2.6.

$$\begin{cases} u_i^T = -(U/Z) + x_i(W/Z) = -\nu_{x_i} + x_i\nu_{z_i} \\ v_i^T = -(V/Z) + y_i(W/Z) = -\nu_{y_i} + y_i\nu_{z_i} \end{cases} \tag{2.6}$$

2.3.3. Focus of Expansion

The Focus of Expansion (FoE) is the projection of the observer's motion direction onto the 2D retinal plane. The points x_0 and y_0 as defined in Equation 2.7 represent the x- and y-components of the FoE, respectively. Following the derivation in Longuet-Higgins and Prazdny (1980), Equation 2.5 can be rewritten by substituting x_0 and y_0. This yields the expression in Equation 2.8. By further dividing the vertical and horizontal components of the OF, as shown in Equation 2.9, it can be seen why the FoE is of interest in the context of OA. Looking at Equation 2.9 it becomes apparent that the translational flow follows straight lines which all converge at the point (x_0, y_0). This implies that the FoE is the point in which the vector describing the observer's motion direction intersects the image plane and that it is thus the vanishing point of the image. The vanishing point describes the only part within the image where the OF is zero regardless of the distance to the corresponding world point. This means that only little information is available about the 3D structure of the environment in this direction. This represents a problem for collision avoidance since it means that frontal obstacles are difficult to detect. In chapter 5 several approaches aiming to deal with this issue will be presented.

$$x_0 = U/W, \quad y_0 = V/W \tag{2.7}$$

$$u^T = (x - x_0)v_z, \quad v^T = (y - y_0)v_z \tag{2.8}$$

$$v^T/u^T = (y - y_0)/(x - x_0) \tag{2.9}$$

2.3.4. Time-to-Contact

The Time-to-Contact (TTC) provides an estimate for an agent's time until collision with an obstacle and represents the basis for many collision avoidance implementations. While no absolute distances can be retrieved from OF without additional information about the observer's velocity (see subsection 2.3.2), the ratio of the distance to an obstacle and the forward velocity is readily available. In Equation 2.10 it is shown that the TTC can be computed for every retinal position (x,y) if the FoE (x_0, y_0) and the translational OF components (u^T, v^T) are known.

$$TTC = Z/W = 1/v_z = (x - x_0)/u^T = (y - y_0)/v^T \tag{2.10}$$

2.3.5. Divergence

The divergence of a vector field is commonly described in terms of the partial spatial derivatives of the field vectors. Employing this definition to OF vectors, the flow divergence at the position (x, y) within the retinal image can be defined according to Equation 2.11. Assuming orthogonal flat surfaces in the visual field and derotated OF, this expression can be simplified by substituting Equation 2.6 yielding the relationship shown in Equation 2.12. It can be seen that there is a direct relationship between the flow divergence, the normalized ego-motion along the axis of movement, and the TTC. It should be noted that OF divergence is frequently simply defined as $D = v_z$ rather than $D = 2v_z$ in the scientific literature (e.g. (Croon et al., 2013; Xiao et al., 2021)).

$$D(x, y) = \frac{\partial u}{\partial x}(x, y) + \frac{\partial v}{\partial y}(x, y) \tag{2.11}$$

$$D = 2v_z = \frac{2}{\tau} \tag{2.12}$$

2.4. Frame-Based Optic Flow Estimation

In order to estimate OF as defined in section 2.1 some sort of visual sensor is required which can capture the pixel velocity of features in the visual field. The most commonly used sensors for this task are frame-based cameras. They provide a stream of consecutive images containing information about brightness (and color) at every pixel as opposed to event-based cameras which will be introduced in section 4.1. However, unfortunately, frame-based cameras show undesirable qualities when it comes to OF estimation onboard autonomous MAVs. The purpose of this section is to highlight these issues to illustrate why there is a need for alternative means of OF estimation.

The earliest attempts of estimating OF with the help of frame-based cameras were proposed by Horn and Schunck (1981) and Lucas and Kanade (1981) and represented the foundation for a multitude of different approaches. These approaches included gradient-based methods which estimate the pixel velocities from spatial and temporal derivatives of the image brightness (e.g. (Weber and Malik, 1992; Oron et al., 2014; Faisal and Barron, 2007; Menze et al., 2015)), correlation-based methods in which OF is computed as the displacement of specific features in the image within two consecutive frames (e.g.(Camus, 1995b; Wills and Belongie, 2004)) and frequency-based methods which utilize velocity tuned filters in the Fourier domain (e.g. (Adelson and Bergen, 1985; Heeger, 1988; Gautama and Van Hulle, 2002)). Furthermore, insect-inspired methods exist which aim to mimic motion detection in insects (see chapter 3). In recent years more and more approaches have been proposed which make use of neural networks to estimate OF (e.g. (Ilg et al., 2017; Zhao et al., 2017; Xiang et al., 2018)). In these approaches, pairs of subsequent frames and some sort of ground truth are commonly fed into the network which then updates its weights to minimize the error (Tu et al., 2019).

While neural networks have shown impressive results for OF estimation, there are still issues that are inherent to frame-based approaches in general. Since there is a large selection of extensive reviews papers about frame-based OF estimation (e.g. (Barron et al., 1994; Tu et al., 2019; Zhai et al., 2021)), this work does not dive deeper into the various individual methods. Instead, the two most influential ones, namely the approaches presented by Lucas and Kanade (1981) and Horn and Schunck (1981), are presented in more detail in subsection 2.4.2 and subsection 2.4.1, respectively in order to illustrate the issues of frame-based methods in the context of OA on-board MAVs in subsection 2.4.3.

2.4.1. Lucas-Kanade

In (Lucas and Kanade, 1981) OF is retrieved by providing an estimate for the disparity between two frames of a motion sequence obtained with a frame-based camera. If F(**x**) and G(**x**) describe the pixel values at each location $\mathbf{x} \in R^n$ of the two images, disparity can be defined as the vector h that minimizes a certain difference equation of the two functions. An example of such a difference equation is given in Equation 2.13 and a one-dimensional visual representation of the disparity vector h is depicted in Figure 2.3 (retrieved from (Lucas and Kanade, 1981)). The value of F(**x**) at $\mathbf{x}+\mathbf{h}$ can be estimated using the Taylor expansion shown in Equation 2.14.

$$E = \sum_{x \in R} [F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})]^2 \tag{2.13}$$

$$F(\mathbf{x}+\mathbf{h}) \approx F(\mathbf{x}) + \mathbf{h}\frac{\partial}{\partial \mathbf{x}} F(\mathbf{x}) \tag{2.14}$$

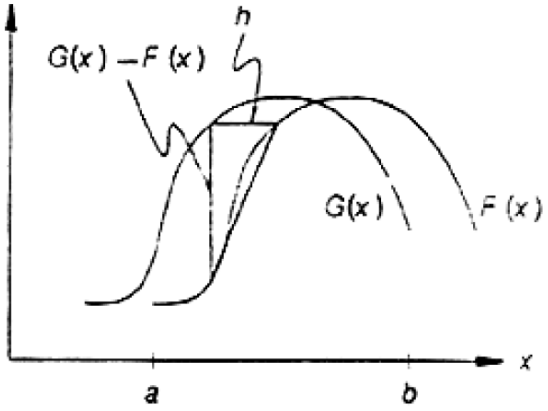


Figure 2.3: Disparity h between the one-dimensional brightness functions F(x) and G(x) of a sequence of two images. Retrieved from Lucas and Kanade (1981).

Substituting Equation 2.14 into Equation 2.13, an expression for h can be found which minimizes the difference between the pixel values $F(\mathbf{x}+\mathbf{h})$ and $G(\mathbf{x})$ according to $\frac{\partial}{\partial \mathbf{h}} E = 0$:

$$h = \left[\sum_x \left(\frac{\partial F}{\partial \mathbf{x}}\right)^T [G(\mathbf{x}) - F(\mathbf{x})]\right] \left[\sum_x \left(\frac{\partial F}{\partial \mathbf{x}}\right)^T \left(\frac{\partial F}{\partial \mathbf{x}}\right)\right]^{-1} \tag{2.15}$$

After a first estimate of h is obtained, the function F(**x**) is shifted by the corresponding value and the procedure is repeated until the approximations of the disparity converges. In order to improve the convergence of the algorithm Lucas and Kanade (1981) assumed the brightness to be constant over a small window of pixels to smoothen the image.

2.4.2. Horn-Schunk

The method presented by Horn and Schunck (1981) relies on the assumption that the illumination of the scene is constant. This implies that changes in brightness at specific image locations only occur due to motion. Consequently, if the brightness at the image location (x, y) at time t is denoted by E(x,y,t), Equation 2.16 holds.

$$\frac{dE}{dt} = 0 \tag{2.16}$$

By applying the chain rule for differentiation and introducing the variables $u = \frac{dx}{dt}, v = \frac{dy}{dt}, E_x = \frac{\partial E}{\partial x}$ and $E_y = \frac{\partial E}{\partial y}$ the expression in Equation 2.17 can be obtained. However, this one equation is not sufficient to obtain estimates for the two unknown OF components u and v. This problem is also commonly referred to as the aperture problem.

$$\epsilon_b = E_x u + E_y v + E_t = 0 \tag{2.17}$$

Horn and Schunck (1981) approached this problem by assuming that neighboring points move at similar velocities. In particular, they introduced a smoothness constraint ϵ_c whose minimization penalizes changes of the velocities u and v with x and y. The OF components can then be determined by finding the values u and v that globally minimize Equation 2.18, where α represents a weighting factor that accounts for noise in the image.

$$\begin{aligned} \epsilon_c^2 = \frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2 + \frac{\partial v}{\partial x}^2 + \frac{\partial v}{\partial x}^2 \\ \iint (\alpha^2 \epsilon_c^2 + \epsilon_b^2) dx dy \end{aligned} \tag{2.18}$$

2.4.3. Limitations

Frame-based cameras provide brightness information about all pixels at every time step. While this rich source of information can be favorable for some applications that do not necessarily require instantaneous results such as object recognition or image segmentation, it poses a problem for real-time applications with limited computational resources available.

The reason for that becomes clear when considering the two methods for OF computation presented in the previous section. To arrive at an estimate both techniques need to perform computations on every single pixel. However, for OA only dynamic parts of the visual scene are of importance. This means that every time step redundant calculations are performed on static parts of the visual scene increasing the computational load and consequently the power consumption. At the same time, the fixed frame rate limits the accuracy of the OF estimates at highly dynamic part of the visual scene. Furthermore, these approaches rely on the fact that the brightness in the visual field does not change which makes them less robust to rapid changes in the environment. Looking once again at Equation 2.15 it can be seen that frame-based OF estimation relies on comparing the properties (in this case the pixel-wise difference in brightness) of two subsequent frames. However, fast-moving texture elements can travel large distances within a short period of time which means that information about their movement in between two frames is lost.

This phenomenon is also called motion blur and is illustrated on the right side of Figure 2.4 (retrieved from Chen (2017)). It depicts a fast-moving scene captured by a grayscale frame-based camera. It can be seen that the slow sampling (compared to the dynamics of the visual scene) of frame-based cameras leads to blurred edges in the image. This represents a problem for MAVs which often operate at high velocities and need to be able to react quickly to obstacles in real-time.

While other, more efficient methods exist (e.g. (Hu et al., 2016; Krähenbühl and Koltun, 2012)), they still perform computations on the entire image to identify relevant regions containing fast movements. Furthermore, these methods also rely on comparing brightness values in two subsequent images and are thus also affected by the problems described in this section. With OF being one of the most commonly used methods for insect-inspired OA it consequently makes sense to consider alternative techniques for OF estimation.

Inspired by biological systems more and more engineering applications have been proposed in recent years which utilize asynchronous processing (e.g. (Schoepe et al., 2019; Milde et al., 2017)). This means that data is transmitted in a continuous stream rather than following an external clock which can prevent the processing of redundant data. An example of such an application are event-based cameras which will be introduced in

Figure 2.4: Fast moving scene captured with an event-based camera (left side) and with a grayscale frame-based camera (right side). Retrieved from Chen (2017).

section 4.1. Instead of considering the brightness of every pixel at every time step, they only provide information about pixels at which the brightness has changed. This helps to tackle some of the issues introduced in this section. On the left-hand side of Figure 2.4 e.g. the same scene as on the right side of the figure was captured but this time using an event-based camera. It can be seen that no more motion blurring occurs and that the edges are clearly defined despite the fast motion. In the remainder of this work, additional alternatives to frame-based OF estimation will be explored. This will be done by considering the asynchronous working principle of motion detection in insects and by introducing existing methods to translate these principles to engineering applications.

3

Motion Detection and Obstacle Avoidance in Flying Insects

In the previous section it was shown that a number of problems arise when trying to use frame-based approaches for OF estimation onboard MAVs. Flying insects on the other hand successfully use OF to avoid obstacles in unpredictable surroundings at high velocities while also having very limited computational resources available (about one million for honeybees e.g. (Srinivasan, 2011)). For this reason, many researchers have turned towards flying insects for inspiration. Milde et al. (2018) e.g. proposed a spiking elementary motion detector (EMD) (see subsection 3.3.2) which unlike conventional EMDs mimics the asynchronous, spike-like nature of OF computation in flying insects. Reversely, creating and implementing models explaining insect behavior can shed light on the biological processes involved. This research aims to draw inspiration from insects by considering how they sense their environment and process the obtained information in order to compute OF and perform OA. The first step towards achieving this goal is to illustrate how flying insects perform these tasks. This will be done in this chapter.

In Figure 3.1 (adapted from Graham and Philippides (2014)) a schematic overview of a typical insect visual pathway is depicted. It consists of three major regions, namely the eye, the optic lobe comprising the lamina, medulla, lobula and lobula plate, and the central brain. In the remainder of this chapter, it is explained how OF is extracted along each region of the visual pathway. Since these computations are carried out on a neural basis, first the general architecture and working principle of a generic neuron are explained in section 3.1. Subsequently, the first region of the visual pathway, namely the compound eye is presented in section 3.2. In section 3.3 it is explained how motion information is extracted in the optic lobe. Finally, in section 3.4 it is shown how OF is used to perform OA in the central brain.

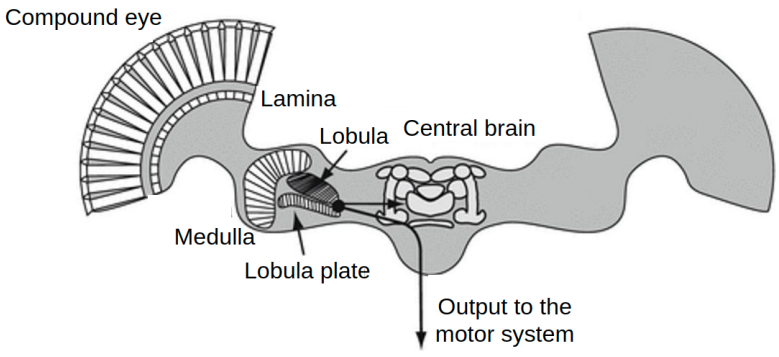


Figure 3.1: The insect visual pathway. Adapted from Graham and Philippides (2014).

3.1. The Generic Neuron

Neurons represent the building blocks of the nervous system and are able to communicate information quickly through a network of cells. They do this by generating electrical signals which are also called action potentials. While there are many different kinds of neurons that vary in their exact function and architecture, they all share the same underlying principles. This leads to the definition of a 'generic neuron' whose basic architecture and working principle will be explained in this section based on the elaborations in (Brown, 2001).

Generally speaking, a generic neuron fulfills three functions. These include receiving information or signals, integrating arriving signals, and communicating information to subsequent cells. The left side of Figure 3.2 (adapted from Wei et al. (2019)) provides a schematic view of a generic neuron. It consists of a cell body, the soma, which spreads out into several smaller branches, the dendrites, and the axon which connects the soma to the synapses at the output terminal. The dendrites are connected to preceding or presynaptic neurons and receive incoming information which can either be excitatory or inhibitory. The difference in electric potential between the soma and its surroundings is called the membrane potential. It increases with the arrival of excitatory signals and decreases for inhibitory signals. This process is called depolarization. In the absence of input spikes, the membrane potential returns to its resting state which is also referred to as repolarization. The electric input is integrated in the soma and if the membrane potential exceeds a certain threshold, the neuron spikes. Subsequently, its action potential is conducted down the axon to the synapses at the output terminals where it is transmitted to postsynaptic cells. Right after spiking the neuron enters a refractory period during which incoming spikes have no effect on its membrane potential. This ensures that individual spikes can be clearly separated in time.

The process of spike generation is summarized on the right side of Figure 3.2. The upper image depicts the depolarization of the membrane potential after an input stimulus and the subsequent return to the resting state. Furthermore, the membrane potentials of two signals failing to drive the action potential to the voltage threshold are indicated. The lower image provides an example for strong and weak input stimuli and the corresponding outputs of the cell.

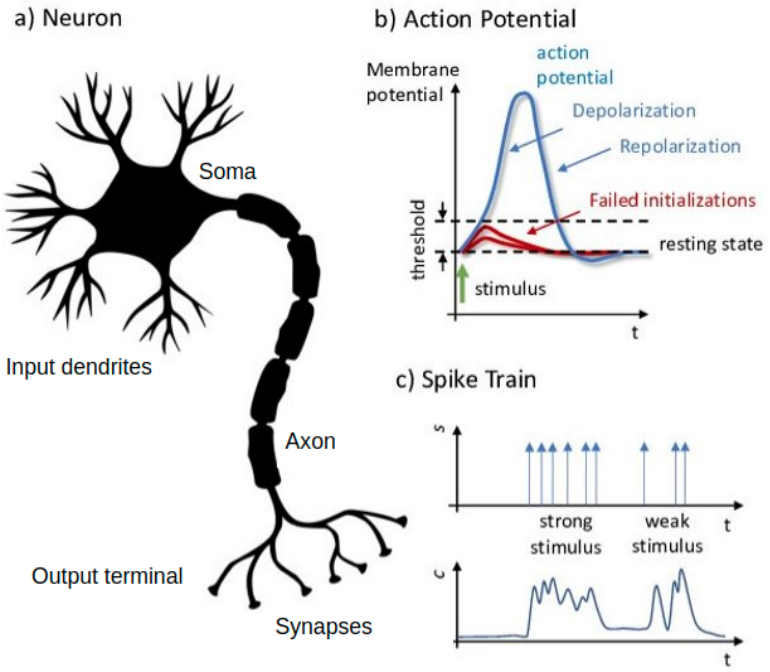


Figure 3.2: Illustration of: (a) the anatomy of a generic neuron, (b) the action potential and (c) input spikes and output for strong and weak stimuli. Adapted from Wei et al. (2019)

The strength of a synaptic connection between two neurons, i.e. to what extend an input spike from the presynaptic neuron increases the postsynaptic neuron's membrane potential, is called efficacy. It can be modified by a mechanism called synaptic plasticity (Baudry, 1998). Hebb (1949) postulated that the efficacy of a synapse increases if the presynaptic cell persistently activates a nearby postsynaptic cell, an idea that is commonly summarized as "neurons that fire together, wire together". Since spiking neurons only have a limited range of responses available, there is another mechanism called intrinsic plasticity (IP) which can adjust the neuron's response to the stimuli statistics in the environment. However, rather than changing the synaptic strength, the intrinsic parameters of the neuron itself (such as e.g. the voltage threshold) are modified to keep the average activity of individual neurons within a desired range (Lazar et al., 2007). In chapter 4 several engineering applications aiming to mimic these kinds of neural plasticity will be presented.

3.2. The Compound Eye

In order to sense their environment, most flying insects make use of compound eyes such as the one of the Drosophila fruit fly depicted in Figure 3.3 [1]. It consists of around 800 elongated cells arranged in a regular, hexagonal pattern. These cells are called ommatidia and they have a length of around 100 μm. Furthermore, each ommatidium is covered by a separate cornea which gives the compound eye its facet-like appearance (Song et al., 2009). In the upper left corner of Figure 3.1 a cross-section of the compound eye depicting the elongated ommatidia can be seen.

Each ommatidium covers a specific solid angle of the surroundings which enables it to provide information about the illumination of the corresponding region (Zhu, 2013). In Drosophilas the interommatidial angle is around 4.6 deg with a total visual field of almost 180 deg. In comparison, the human's fovea covering around 2 deg of the visual field contains about 60,000 cones. This leads to a spatial resolution of approximately 0.01 deg. This means that the fruit fly's visual acuity is roughly 500 times worse than that of a human being (Borst, 2009). Moreover, each ommatidium can be considered as a pixel in the visual field of the fly, resulting in a total resolution of only 800 pixels (Nériec and Desplan, 2016).

A detailed illustration of a single ommatidium is shown in Figure 3.4 (retrieved from Wolff and Ready (1993)). There are eight different kinds of photoreceptors present in each ommatidium which are referred to as R1 - R8. The outer receptors (R1 - R6) are arranged in a circle around the inner R7 and R8 photoreceptors which are stacked on top of each other. Similarly to the ommatidia themselves, the photoreceptors are also elongated cells and their plasma membranes consist of two parts, namely the photo-sensitive (rhabdomere) and the photo-insensitive (basal) membranes. The corneal lens and the pseudocone focus the light energy onto the rhabdomere. The rhabdomere then transforms the perceived light into current and the basal membrane converts this current into a voltage-response which is subsequently mapped onto the optic lobe for further processing (Mishra and Knust, 2013).

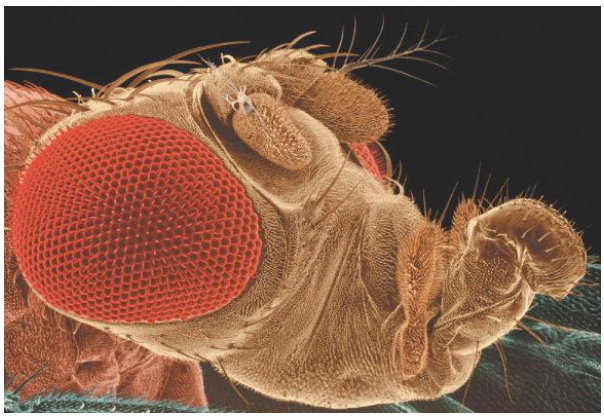


Figure 3.3: The Drosophila compound eye [1]

[1] Retrieved from https://neurophilosophy.wordpress.com/2006/10/02/loss-of-spam-de-evolves-the-fruit-flys-compound-eye/

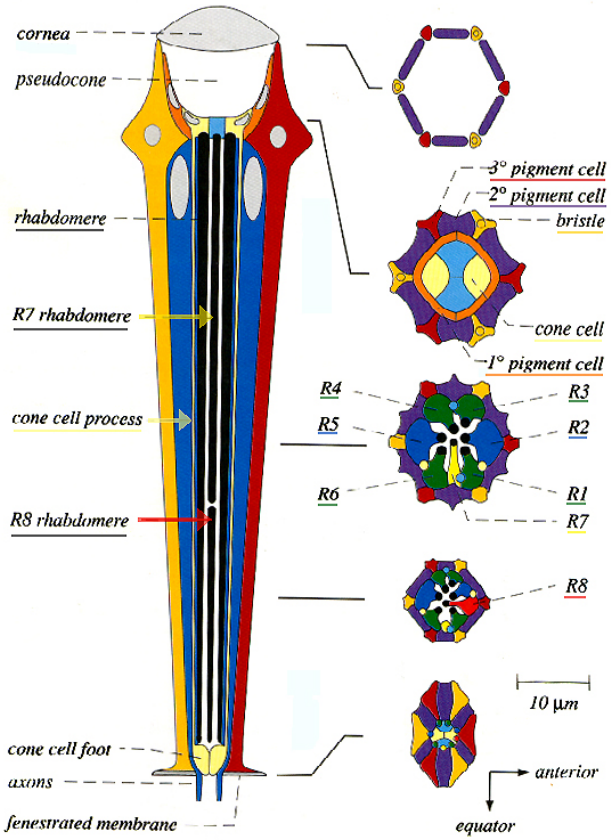


Figure 3.4: Detailed view of a single Ommatidium. Retrieved from Wolff and Ready (1993).

3.3. Mechanisms for Motion Detection in Flying Insects

The computation of OF is performed in the optic lobe which makes up for more than 60 % of the brain's neurons. As can be seen in Figure 3.1 it comprises four neuropils, namely the lamina, medulla, lobula, and lobula plate. The photoreceptors project retinotopically onto the optic lobe meaning that each photoreceptor transmits signals to one column of neurons that correspond to the same position in the two-dimensional visual array.

In this section, the mechanisms involved in obtaining motion information from the visual input in flying insects are explained. First, saccadic flight strategies are introduced in subsection 3.3.1. Subsequently, in subsection 3.3.2 and subsection 3.3.3 the two most-frequently researched motion detectors in flying insects are explained, namely the Elementary Motion Detector (EMD) and the Lobula Giant Movement Detector (LGMD).

3.3.1. Saccadic Flight Strategy

In section 2.2 it was shown that only the translational component of OF contains depth information about the environment and that it thus needs to be separated from the rotational part to use it for navigational purposes. While it is possible to perform this task computationally, many insects extract the translational component through behavioral action (Egelhaaf et al., 2012). In particular, they apply a saccadic flight strategy which consists of alternating phases of straight (almost) purely translational flight, the intersaccades, and flight including fast rotations, the saccades. Insects change between these two modes at a frequency of up to 10 Hz with the translational phases making up for more than 80% of the overall flight time (Egelhaaf et al., 2012). This way insects can retrieve depth information from the translational flow during intersaccades which is not 'contaminated' with rotational OF. The obtained ego-motion information can then be used to determine the parameters for the next saccade during which the flight direction is changed in order to e.g. avoid obstacles. This strategy reduces the computational load on the insect's brain and thus enables them to perform computationally more efficient obstacle avoidance.

While not directly involved in the computation of OF, it is still noteworthy that there are other behavioral actions utilized by insects to obtain depth estimates. Locusts, mantids, and dragonflies for example carry out lateral body and head movements before jumping since the resulting displacement of their eyes allows them to obtain a depth estimate of their surroundings (Collett, 1978; Krahl and Poteser, 1997; Olberg, 1981).

3.3.2. The Elementary Motion Detector

Neurons in the lobula plate of flying insects have been found to respond selectively to motion in the four cardinal directions. This selectiveness has been attributed to temporal and spatial cross-correlation between different parts of the visual input. The elementary motion detector (EMD) is a model that was created to explain how the motion direction of an agent is obtained from the activity of its photoreceptors using the minimum number of computations (Frye, 2015). In this section, the neural implementation of the EMD in flying insects as proposed by current research is shown using once again the Drosophilia as an example. Furthermore, computational models of the EMD are presented keeping in mind that the final goal is to draw inspiration for applications onboard MAVs.

Neural Implementation While a rather complete map of the neurons in the optic lobe has already been available for a long time, it was not clear how exactly they contribute to the determination of the motion direction. Only recently with the development of sophisticated neurogenetic methods for the Drosophila, it has become clear which role the various neurons take on in the extraction of OF. Borst et al. (2019) provide an extensive description of the neural implementation of the EMD in the Drosophilia in the optic lobe. Some of the key points are summarized hereafter.

Figure 3.5 (retrieved from (Borst et al., 2019)) depicts an overview of the neurons involved in elementary motion detection. Information about the brightness of the surroundings is provided by the photoreceptors R1-6 and split into two parallel motion circuits corresponding to luminance increments (ON) and decrements (OFF). In the lamina so-called L2 and L4 neurons and L1 and L5 neurons are specialized to detect OFF- and ON-increments, respectively. The L3 neuron is connected to both the OFF and the ON circuit. Each neuron column in the medulla comprises more than 60 different cell types which can be clustered into a number of different groups (for more detailed information about these groups, please refer to (Borst et al., 2019)). Analysis of the neurons in the medulla showed that they can be categorized into two classes, namely temporal low-pass filters and temporal band-pass filters. The neurons in the medulla synapse onto the T4 and the T5 cells which are the first motion-sensitive neurons along the visual pathway. Per channel there exist four subtypes for both T-neurons that are tuned to the cardinal directions (up, down, left, right). The direction to which the neurons are sensitive is called their preferred side while the opposite direction is referred to as their null side. The dendrites of the T4 and T5 cells span several columns in their respective direction of motion, unlike their presynaptic neurons whose dendrites are restricted to one column. This means that T4 and T5 cells sample inputs from neighboring points in space with their multi-columnar dendrite input. Their preferred and null sides are connected to cells in the medulla which represent low-pass filters while the neurons synaptic in the center show band-pass characteristics. This way direction enhancement on the preferred side is achieved. The four subtypes of the T4 and the T5 cells finally map onto the four layers in the lobula plate which correspond to the four cardinal directions. In addition to exciting the layers corresponding to their own preferred direction, the T4 and T5 cells also inhibit the layers corresponding to the opposite directions. This is indicated by the purple arrows in Figure 3.5. In the lobula plate, the well-known lobula plate tangential cells are located which integrate the excitatory and inhibitory signals of the elementary motion-sensing T4 and T5 neurons over a large part of the visual field. This finally leads to motion selectivity for neurons in the lobula plate.

In several experiments investigating the dynamic response properties of fly motion detectors (Borst and Egelhaaf, 1987; de Ruter van Steveninck et al., 1986), neurons in the lobula plate have been found to show adaptation. This means that their steady-state response to sustained motion decreases over time as illustrated in Figure 3.6 (retrieved from Clifford et al. (1997)). It depicts the normalized response of neurons in the fly lobula plate to sustained motion at two different temporal frequencies. This adaptation leads to a decrease in sensitivity to sustained motion while increasing the sensitivity towards changes in image velocity. Consequently, it is believed to help insects in keeping the neuron activity within a feasible operating range.

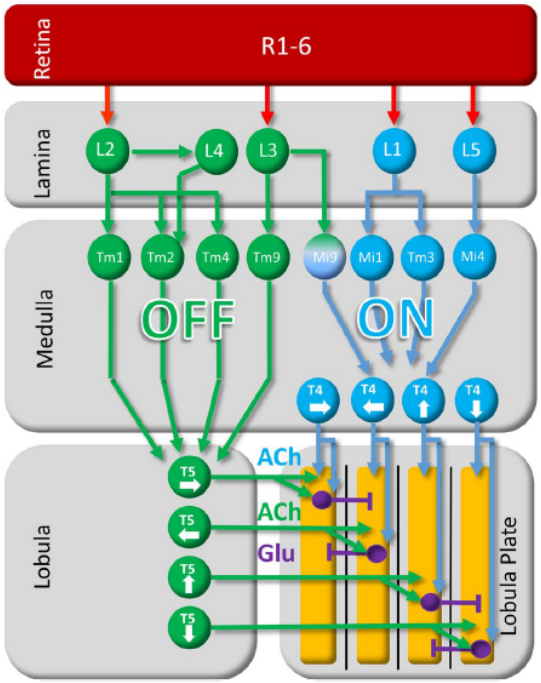


Figure 3.5: Neural implementation of the EMD in the Drosophila. Retrieved from Borst et al. (2019).

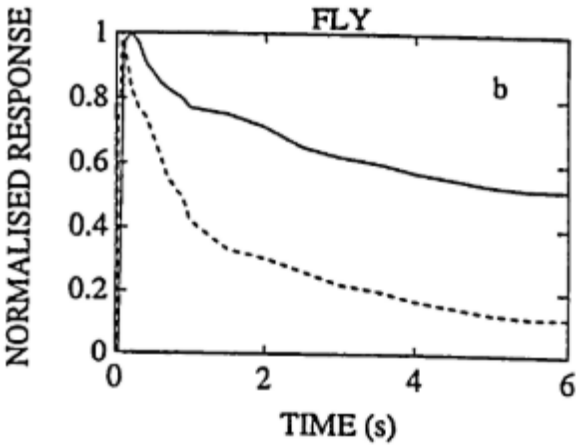


Figure 3.6: Neuronal response in the fly lobula plate to sustained motion at two temporal frequencies (solid line: 2.4 Hz, dashed line: 19.2 Hz). Retrieved from Clifford et al. (1997).

Computational Model A computational model of the EMD was first established by Reichardt and Rosenblith (1961). It is illustrated on the left side of Figure 3.7 and consists of two spatially separated photoreceptors that are activated by changes in illumination. A time delay is applied to the signals using a low pass filter (LP) and they are subsequently multiplied (M) with the non-delayed version of the other signal. By computing the difference between the two signals, the movement direction of a brightness element can be determined. Combining multiple EMDs oriented in different directions yields an estimate for the motion direction of edges in the visual field. The resulting OF field does however not only depend on the velocity of the motion stimuli but also on its textural properties (Schwegmann et al., 2014). This implies, that an array of EMDs is not able to distinguish between a pattern of narrow stripes moving at low velocity and a pattern of wide stripes moving at a faster velocity (Frye, 2015).

As explained in the previous section, the visual input has been shown to be split into ON and OFF channels in the Drosophila. Joesch et al. (2010); Eichner et al. (2011) investigated two models that account for this division of the visual input. The first one is the "4-Quadrant-Detector" (Figure 3.7B) which consists of four EMDs applied in parallel accounting for all four possible combinations of input signals (ON-ON, ON-OFF, OFF-ON, and OFF-OFF). The second model, comprises only two EMDs, namely an ON-ON and an OFF-OFF detector (Figure 3.7C). Experiments and the distinct nature of neurons postsynaptic to the visual input suggest that the existence of the second model only comprising two detectors is more likely (Eichner et al., 2011). This is consistent with the neural implementation of the EMD which was presented above.

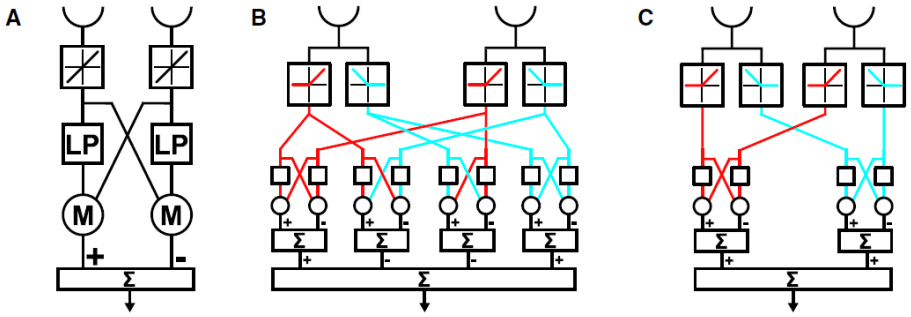


Figure 3.7: Illustration of A: the classical Reichhardt detector, B: a Reichhardt detector with four combinations of ON/OFF channels and C: a Reichhard detecor with two combinations of ON/OFF channels. Retrieved from Eichner et al. (2011)

While the proposed EMDs can accurately model most of the characteristics observed in insect motion detection, they do not account for the adaptation of the neurons in the lobula plate. In order to include this mechanism several authors suggested the use of an adaptive delay within the detector (Clifford et al., 1997; Clifford and Langley, 1996; Sarikaya and Ogmen, 1994). An example of such a modified EMD is shown in Figure 3.8 (retrieved from Clifford and Langley (1996)), where Δx, Δt, and R represent the spatial distance between two cells, the applied time delay, and the output of the detector, respectively.

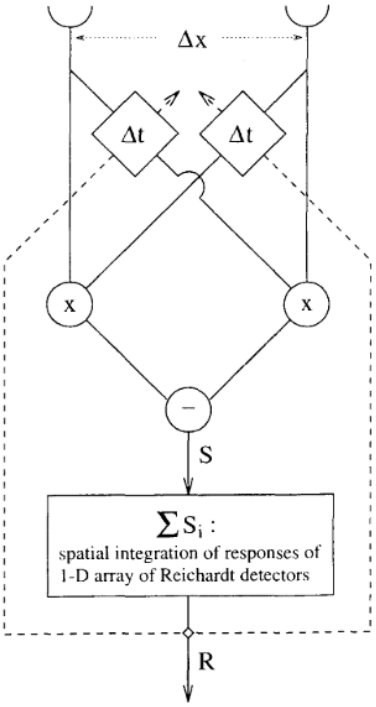


Figure 3.8: Schematic illustration of an EMD with adaptive delays. Retrieved from Clifford and Langley (1996)

For increasing values of the output R, the time delay is decreased such that the EMD becomes more sensitive to faster motion while decreasing its sensitivity for the current velocity. A leakage term ensures that the delay returns to its unadapted resting value in the absence of stimuli. This is illustrated in Equation 3.1 where the time delay is represented as τ for ease of notation. The variable η represents the adaptation gain and μ a parameter controlling the amount of leakage.

$$\frac{d\tau(t)}{dt} = -\eta\tau(t)|R(t)| + \mu(\tau_0 - \tau(t)) \tag{3.1}$$

In order to provide a better understanding of the relationship between the neural implementation and the computational models of motion detection, a mapping of the traditional EMD onto the various layers of the optic lobe is provided in Figure 3.9 (retrieved from (Tuthill et al., 2013)). The high-pass and low-pass filters are represented by τ_{hp} and τ_{lp}, respectively. It should be noted that this representation does not include the features of the modified EMDs such as e.g. ON and OFF channels or adaptive delays. It does however nicely illustrate which steps of the elementary motion detection are performed in which part of the optic lobe.

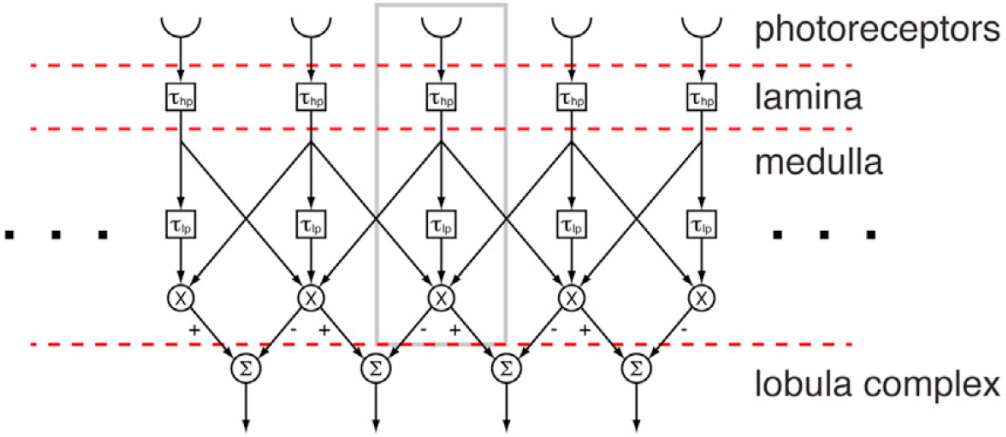


Figure 3.9: Mapping of the traditional Reichhardt detector onto the optic lobe. Retrieved from Tuthill et al. (2013)

3.3.3. The Lobula Giant Movement Detector

In addition to the EMD, there is another commonly researched form of motion detection in flying insects which involves the Lobula Giant Movement Detector (LGMD). The LGMD neuron can be found in the lobula of the locust. Rather than responding to cardinal motion directions at a fixed distance like the EMD, it has been found to react to looming stimuli. Using the same approach as in the previous section, first, the neural implementation of the LGMD is presented followed by a computational model aiming to mimic the LGMD's characteristic traits.

Neural Implementation The spiking response of the LGMD is a consequence of a critical race between neurons that are excited due to illumination changes in the visual field and delayed inhibition resulting from the same changes in illumination (Rind, 2002). In particular, an increase in the edge velocity of objects within the visual field and an increase in the total amount of visible edge have been found to be responsible for the reaction of the LGMD to looming objects (Rind and Bramwell, 1996). The preprocessing of the visual input is believed to take place in the medulla since this is the first layer in the optic lobe in which small-field neurons react specifically to movement instead of random changes in illumination (James and Osorio, 1996; Osorio, 1991). While the exact neural circuits resulting in the spiking response of the LGMD to looming stimuli are not fully understood, several computational implementations exist which are able to mimic the characteristics of this neuron.

Computational Model A computational model involving the LGMD neuron was first developed by Rind and Bramwell (1996). The model was realized using a neural network consisting of three layers with 250 units each. The first layer comprises photoreceptors or P-units, the second one excitatory (E) and inhibitory (I) units, and the third one summing (S) units. In addition, there is a single feed-forward inhibition cell and one final layer consisting of only one LGMD neuron. The architecture of this model is depicted in Figure 5.1 (retrieved from Rind and Bramwell (1996)).

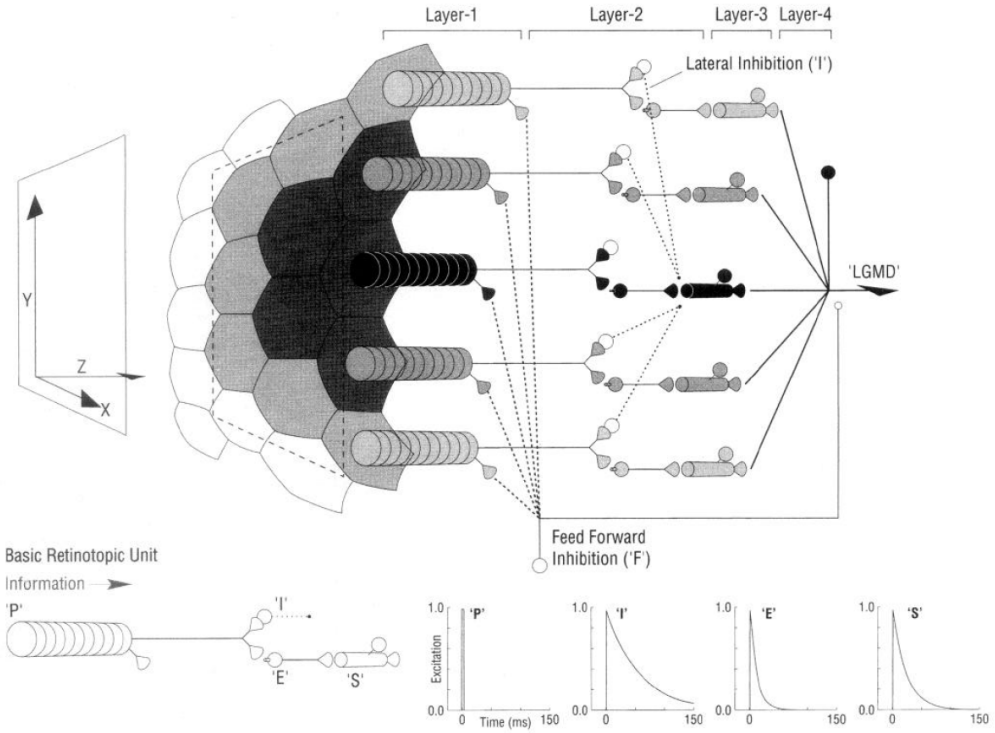


Figure 3.10: Computational model of a neural network involving the LGMD neuron. Retrieved from Rind and Bramwell (1996).

Incoming images are mapped onto the hexagonally arranged P-units such that each unit covers a small area of the view. The P-units in the first layer respond to changes in illumination with brief excitations as can be seen at the bottom of Figure 3.10. Their output is then passed to the excitatory E-cells and the inhibitory I-cells in the second layer and the single feed-forward inhibitory F-cell. While the E-units directly forward the signal to the corresponding summing S-units in the same retinotopic position in the third layer, inhibition is forwarded laterally to two rings of S-units surrounding the position of the transmitting I-unit (see Figure 3.10, the greyscale of the units represent the distance from the central photoreceptor). Furthermore, no delay is applied to excitatory signals but inhibition is delayed by 1 - 4 ms. The output of the S-cells depends on the total sum of the excitation and inhibition received. They spike once a certain threshold has been reached. The excitation of the I-, E-, and S-units is followed by a refractory phase during which they cannot be activated and the excitement decreases exponentially as indicated at the bottom of Figure 3.10. Finally, in the fourth layer, the input to the LGMD is computed as the difference between the total excitation of the S-cells and the inhibition of the F-cell. Again, if the excitation of the LGMD neuron reaches a certain threshold, it spikes which can be interpreted as the detection of a looming object.

The detection of looming objects can be seen as a race between the excitatory and inhibitory signals. When an object is approaching, the number of spikes from the E-neurons is increasing exponentially since the edges in the visual field are expanding. However, due to the delay applied to the signal transmitted by the I-neurons, it still shows the smaller magnitude from several ms before. Consequently, the excitatory signal is stronger such that the LGMD neuron is triggered. Conversely, the excitatory units will lose the race when an object is receding since the edges in the visual field will then retract. When an object is translating, the number of excited E-units stays approximately constant meaning that the delayed inhibitory signal is of similar magnitude as the excitatory one. This way the sum of the two signals cancels out and the LGMD neuron is not triggered. These mechanisms ensure that the LGMD-neuron does indeed only spike for looming stimuli.

3.4. Obstacle Avoidance in Flying Insects

The final step along the visual pathway of flying insects is the conversion of the computed OF into motor commands. While this research focuses on possible ways to improve OF computation, the ultimate goal is to use the obtained OF in order to perform OA onboard MAVs. Consequently, this section provides a brief overview of collision avoidance approaches in flying insects.

The LGMD neuron is postsynaptically connected to the descending contralateral movement detector (DCMD). This neuron is linked to leg and flight motor- and interneurons which are involved in escaping maneuvers of the locust (Rind, 2002). Since the LGMD neuron triggers a one-to-one spike response in the descending contralateral movement detector (DCMD) (Rind, 1984), the excitation of the LGMD neuron due ta a looming stimulus triggers a direct escape response in the locus. Unfortunately, the neural translation into motor commands is not as straightforward for the EMD. It is known that the information about elementary motion direction computed in the lobula plate is subsequently projected onto the ventrolateral neuropils (VLNP) in the central brain for further processing. However, the exact neural processes translating the optical flow into motor avoidance commands are unknown (Nériec and Desplan, 2016). Based on behavioral observations in insects, several models have been suggested aiming to explain how flying insects utilize OF to avoid obstacles. The most commonly researched ones are presented in the remainder of this section.

Many OF-based strategies do not directly consider the avoidance of obstacles but are more closely related to navigational tasks. It has for example been observed that bees tend to stay centered when flying through tunnels. Experiments suggest that they do this by keeping the OF perceived in the left and right parts of the visual field equal. Furthermore, it was shown that they slow down when the tunnel becomes more narrow and speed up when it becomes wider. This indicates that they keep the bilateral OF constant. It was also shown that bees land at a constant approach angle implying that they keep the magnitude of the OF constant (Serres and Ruffier, 2017). While these observations can be used to avoid collisions when performing tasks such as wall following or keeping a predefined distance to the ground, the aim of this research is to provide an OA method that can be applied to the operation in complex environments. Consequently, these OF techniques are not further considered and special attention will be drawn to OA methods that can deal with a larger variety of obstacles.

In (Tammero and Dickinson, 2002) experiments were performed to investigate insect's avoidance reactions to certain stimuli. These included expanding squares presented at varying azimuth angles and expansion rates. In order to measure the response of the flies, they were tethered and their wing-stroke amplitude and frequency were recorded. Furthermore, the position of their legs was optically tracked to detect stereotypical landing responses. The results of this experiment were then analyzed in light of four existing models for OA in insects in order to check their plausibility. The first model was the "TTC model" in which the fly triggers an OA response if the TTC (see subsection 2.3.4) falls below a certain threshold (e.g. (Wagner, 1982)). In the second model, the "temporal contrast model", avoidance reactions are set off if darkening occurs in the field of view (e.g. (Trimarchi and Schneiderman, 2009; Holmqvist and Srinivasan, 2004)). The third model suggests that a fly might generate a response if the image covering its field of view subtends a certain width or area (e.g. (Wittekind, 1998; Wicklein and Strausfeld, 2000)). In the fourth and final model image motion is integrated over space and time and an obstacle avoidance reaction is triggered when the integrals exceed a certain threshold. This is referred to as the "integration model" (e.g. (Borst, 1990)).

The results of the experiments showed that flies generate the strongest collision avoidance reaction in response to image expansion in the lateral parts of the visual field, whereas expansions in the frontal regions trigger landing responses (extensions of legs). Furthermore, it was found that the latency of the collision avoidance reaction remains more or less constant for varying expansion velocities while the latency for landing maneuvers changes with the rate of image expansion. Finally, it was revealed that by opening the feedback loop of the experiment, i.e. by eliminating the effect of the fly's response on the visual stimulus, the amplitude of the avoidance response increased. This implies that the collision avoidance response does indeed rely on visual feedback. These results show that the TTC model appears to be an unlikely candidate since the latency of the avoidance reaction did not vary with the expansion rate which should be the case if flies did indeed utilize the TTC to trigger avoidance responses. While the latency for the landing responses did vary with the expansion rate, for other species of flies it was shown to also depend on image contrast and size, which is inconsistent with the TTC model (Eckert and Hamdorf, 1980). The constant latency of avoidance reactions

also makes it unlikely that avoidance responses are triggered by an absolute stimulus size as suggested by the stimulus size model. Finally, large changes in temporal contrast induced by a sudden increase of the size of the square did not trigger any avoidance or landing responses in the flies indicating that the temporal contract model is also an unlikely candidate. Consequently, the integration model appeared to be the most promising one to explain how motion direction information is translated into motor commands.

Based on these findings, Tammero and Dickinson (2002) suggested an adapted integration model for OA. This model is depicted in Figure 3.11 (retrieved from Tammero and Dickinson (2002)). It can be seen that information about the image expansion is obtained from an array of EMDs as described in subsection 3.3.2. In order to ensure independent initiation of the two behaviors (landing and avoidance maneuvers), the expansion has to be calculated over at least three separate regions, namely the lateral left, the lateral right, and the frontal part of the field of view. The expansion signals are then integrated and when reaching a certain threshold they trigger a saccade and a landing response for the lateral and central part of the visual field, respectively. In order to prevent weak motion stimuli from triggering an avoidance response, the temporal integration must be "leaky" meaning that it decreases over time when no further inputs are received. Finally, the outputs of the lateral expansion detectors inhibit each other in order to prevent sudden changes in flight direction.

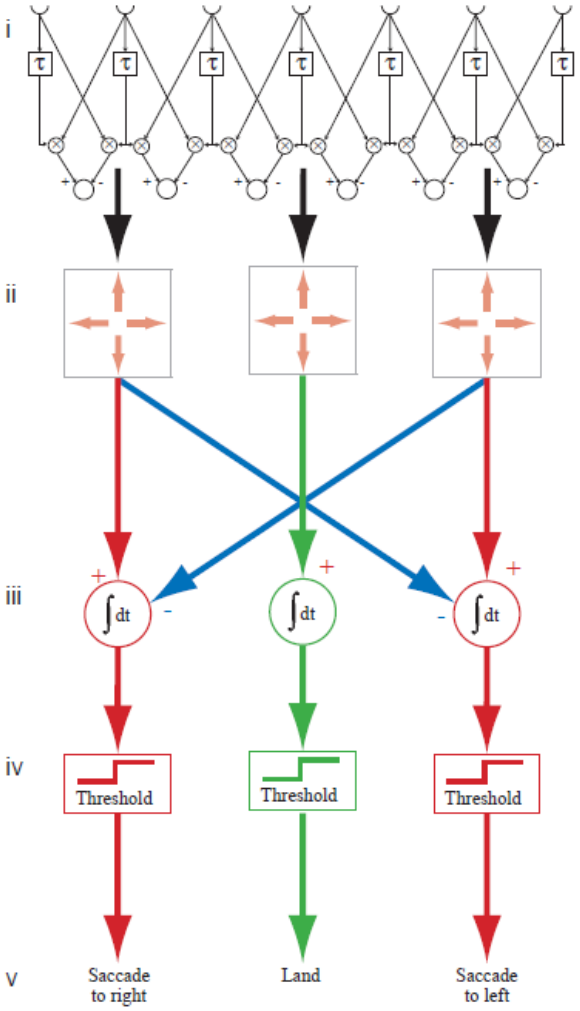


Figure 3.11: Illustration of the computational model explaining avoidance and landing reactions in the fly based on optic flow input. Retrieved from Tammero and Dickinson (2002).

4

Towards a Neuromorphic Approach for Optic Flow Estimation

In the previous section it was shown how insects perform OF estimation and OA despite their limited computational resources. It was established that they employ neural asynchronous computations for both perception and processing unlike conventional means of OF estimation which rely on synchronous algorithmic approaches. Furthermore, it was found that insect eyes have a very low resolution and acuity and show adaptation which cannot be observed in conventional methods for OF estimation. However, it still has to be established how these low-power, lightweight solutions can be translated into engineering applications. In this chapter recent technological advances which have the potential to close this gap will be presented

Since most biological systems share many of the desirable properties which can be found in insects, this chapter does not only focus on insect-inspired approaches but more generally speaking on so-called neuromorphic systems. The term neuromorphic was first introduced by Mead (1990) and describes artificial neural systems and design principles reminiscent of those in biological nervous systems (Indiveri et al., 2011). In section 4.1, section 4.2 and section 4.3 three neuromorphic technologies which are promising for the use of insect-inspired OF estimation and OA will be introduced. Furthermore, existing OF estimation methods utilizing these technological advancements will be explained in the respective sections.

4.1. Event Cameras

In recent years event cameras such as the Dynamic Vision Sensor (DVS) (Lichtsteiner et al., 2008) have been developed which mimic the working principle of biological retinas. Rather than sampling the visual scene at a fixed frame rate like conventional cameras, they asynchronously generate events in response to changes in the perceived brightness. This makes their working principle similar to vision in flying insects. Their output consists of an event stream that contains information about changes in the image intensity at a specific time and pixel. An example of this is shown in Figure 4.1 (retrieved from Barranco et al. (2014)) where the output of an event camera in response to a moving hand can be seen. The left side illustrates the output events buffered over a specific time window and the right side depicts the spatial locations of the events as a function of time. This section introduces the basic working principle of event cameras and their advantages over frame-based cameras. It furthermore provides an overview of the sensors currently available on the market and introduces existing methods for event-based OF estimation.

4.1.1. Working Principle

Event cameras generate an output whenever the logarithmic local change in brightness exceeds a predefined threshold. The changes in image intensity are measured with respect to a reference log-illumination saved during the last event at the concerned pixel. This information is communicated in form of an event stream containing the location (x, y), the time stamp t, and the polarity p of the event. The polarity contains information about the sign of the illumination change with ON- and OFF-events indicating increases and decreases in the log-intensity, respectively (Gallego et al., 2019).

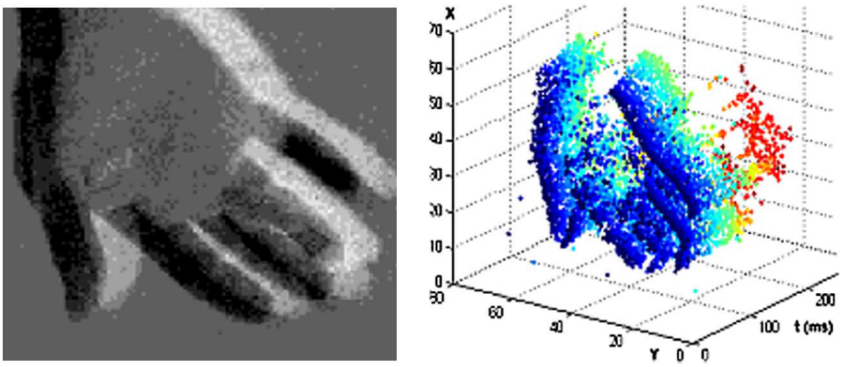


Figure 4.1: Illustration of event camera output in response to a moving hand. The left image shows the output events buffered over a time window of 250 ms. ON- and OFF-events are represented as light and dark gray-values. In the right image the corresponding spatiotemporal representation is depicted. Retrieved from Barranco et al. (2014).

The basic working principle of a DVS pixel is depicted in Figure 4.2 (retrieved from Lichtsteiner et al. (2008)). The upper image provides an example of the voltage output V_p, representing the log-illumination at a specific pixel, and the lower image shows the corresponding change in voltage V_{diff}. It can be seen that the change in voltage is reset whenever it reaches the specified threshold. This leads to the generation of an output event and updates the reference voltage level illustrated as 'reconstruction' in the upper image.

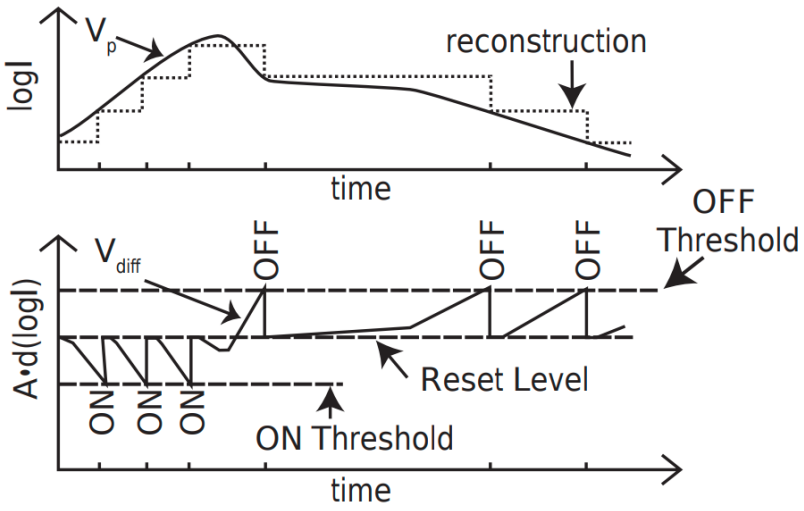


Figure 4.2: Illustration of the working principle of an event camera pixel. Retrieved from Lichtsteiner et al. (2008)

4.1.2. Available Event Cameras

The most commonly used event-cameras in scientific literature are the "DVS128" (Lichtsteiner et al., 2008), the Active-pixel Vision Sensor (DAVIS) (Brandli et al., 2014), and the Asynchronous Time-based Image Sensor (ATIS) (Posch et al., 2011). The DVS128 was the first commercially available event camera and could be purchased at iniLabs. However, this camera did not provide any information about absolute brightness levels which are often used as ground truth in machine learning applications. Furthermore, it makes use of a rather small pixel array. The DAVIS and ATIS both showed improvements with respect to these shortcomings. They offer increased resolutions of 240x180 and 304x240 pixels, respectively, and provide a grayscale image in addition to the event-based representation of the visual field. Moreover, the DAVIS240 also supplies IMU measurements. The specifications of these three event cameras are shown in Table 4.1 (adapted from Gallego et al. (2019)).

Active research is still being performed in the development of more advanced event cameras. Li et al. (2015); Moeys et al. (2017); Marcireau et al. (2018) e.g. propose event cameras that also respond to different colors by employing integrated color filter arrays (CFA) or color-splitter prisms. Furthermore, iniLabs is currently developing the Embedded Dynamic Vision Sensor (eDVS) and the Miniature Embedded Dynamic Vision Sensor (meDVS) in an effort to miniaturize event-based cameras. Their significantly reduced size and weight make them promising for the use onboard MAVs.

Table 4.1: Specifications of the DVS128, DAVIS240 and ATIS. Adapted from Gallego et al. (2019)

	DVS128	DAVIS240	ATIS
Supplier	iniVation	iniVation	Prophese
Reference	Lichtsteiner et al. (2008)	Brandli et al. (2014)	Posch et al. (2011)
Year	2008	2014	2011
Resolution [pixels]	128x128	240x180	304x240
Latency [μs]	12 @ 1 klux	12 @ 1 klux	3
Dynamic range [dB][1]	120	120	143
Contrast sensitivity [%]	17	11	13
Power consumption [mW]	23	5-14	50-175
Pixel size [μm^2]	40x40	18.5x18.5	30x30
Grayscale output	No	yes	yes
Max. bandwidth [Meps][2]	1	12	-
Interface	USB 2	USB 2	-
IMU output	no	yes	no

4.1.3. Advantages Over Frame-Based Cameras

Since the pixels in event-cameras operate asynchronously and only transmit information when changes in the corresponding part of the visual scene are occurring, the amount of transmitted data is highly reduced. Moreover, similarly to insects, event cameras typically have a lower resolution than frame-based cameras which further reduces the amount of data to be processed. This in return leads to a lower power consumption when compared to frame-based cameras (Posch et al., 2014). Furthermore, the sampling of the visual scene is adjusted to the scene dynamics and not driven by an external clock. This means that dynamic parts in the field of view are sampled at higher rates resulting in a high temporal resolution. The asynchronous operation of event-based cameras leads to a low latency since detected events can be communicated straight away without having to wait for the global exposure time of the frame. Finally, event cameras show a very high dynamic range of more than 120 dB as opposed to the roughly 60dB typically found in frame-based cameras. This is due to the logarithmic scale used by the photoreceptors and the fact that the pixels work independently (Gallego et al., 2019). Finally, the event-based nature of the output already contains information about the spatio-temporal dynamics of the scene which can be useful during the computation of OF. It can be concluded that event cameras possess many of the desired properties observed in the visual system of insects and that they are thus promising for the use in insect-inspired OA.

4.1.4. Event-Based Optic Flow Estimation

Since event-based cameras encode information about the spatiotemporal dynamics of the visual field at a high temporal resolution, they have recently gained a lot of attention in the field of OF estimation. However, as shown in subsection 4.1.1 their output varies greatly from that of frame-based cameras. This means that novel approaches are required to compute OF estimates from event-based input data. Based on the extensive review provided in (Gallego et al., 2019), this section presents existing methods to perform this task.

Gradient-Based Methods Gradient-based methods compute OF based on information about the spatio-temporal changes of pixel brightness and are mainly based on existing frame-based techniques. Benosman et al. (2011) e.g. adapted the well known gradient-based Horn-Schunk method (Horn and Schunck, 1981) in order to extract OF from an event-based camera. While the required temporal and spatial derivatives of the

[1] Dynamic range refers to the ratio of the brightest and darkest colors a camera can picture in a single image.
[2] Million events per second

pixel intensities are not directly available due to the lack of gray levels, they were approximated by comparing the activity levels of two adjacent active pixels. In particular, they were computed according to Equation 4.1 and Equation 4.2, where $e(x, y, t)$ represents an event at time t and pixel location $(x.y)$. The OF was then retrieved by assuming a constant pixel brightness over a window of pixels as proposed in (Lucas and Kanade, 1981) resulting in an additional constraint. This made it possible to solve for the OF. However, Brosch et al. (2015) showed that gradient-based methods suffer from the small number of events available per pixel and that they are consequently quite noisy and unstable.

$$\begin{cases} \frac{\partial e(x,y,t)}{\partial x} \sim \sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x-1,y,t) \\ \frac{\partial e(x,y,t)}{\partial y} \sim \sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x,y-1,t) \end{cases} \tag{4.1}$$

$$\frac{\partial e(x,y,t)}{\partial t} \sim \frac{\sum_{t-\Delta t}^{t_1} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x,y,t)}{t - t_1}, \quad \text{with} \quad t_1 < t \tag{4.2}$$

Plane-Fitting By plotting the time of events as a function of their pixel locations, a spatiotemporal surface can be obtained which displays the movement of image features in time. The gray surface in Figure 4.3 (retrieved from Aung et al. (2018)) e.g. illustrates the event representation of a bar moving in the x-y plane. Benosman et al. (2014) showed that the normal to this surface is the OF component perpendicular to the moving edge. This means that the normal OF can be obtained if the gradient of the surface is known. By only considering events within a small spatiotemporal box, Benosman et al. (2014) approximated the velocity at which events are propagating as constant resulting in a planar spatiotemporal surface. Using a least square algorithm they then estimated the surface and its gradients from the events within the box. The plane-fitting algorithm is illustrated in Figure 4.3 where the blue dots represent the measurements by the event camera and the red dots predictions performed by the plane-fitting method.

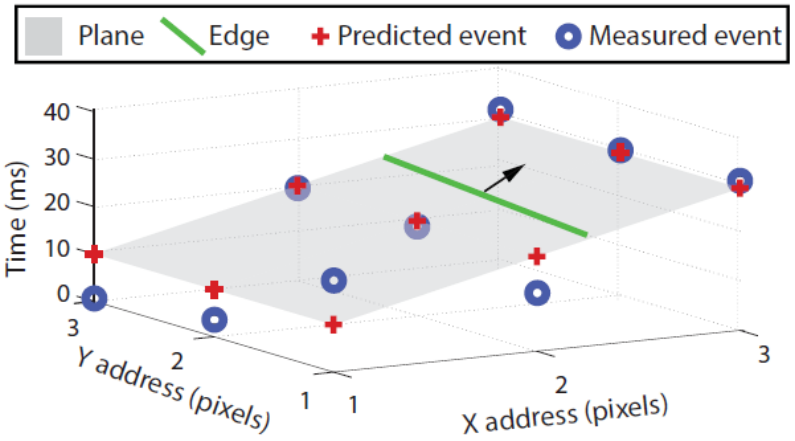


Figure 4.3: Illustration of the plane-fitting algorithm. Retrieved from Aung et al. (2018).

A problem with this method is that the normal OF is defined as the inverse of the plane gradients which leads to singularities for very small gradient values. This issue was addressed in (Brosch et al., 2015) by introducing a new expression for the OF and the surface normal. To improve the accuracy of the suggested approach Rueckauer and Delbruck (2016) presented an iterative method which was then implemented on an FPGA by Aung et al. (2018). In (Barranco et al., 2014) the method was adapted to also cope with more gradual intensity gradients. Hordijk et al. (2018) added efficiency improvements to the method proposed in (Benosman et al., 2014) by reducing the number of parameters of the fitted plane and assuming that incoming events coincide with the estimated plane. Furthermore, they introduced a "time-stamp based clustering of the event cloud" (Hordijk et al., 2018) which deals with the problems arising when using a fixed value for the time window Δt.

Direction-Selective Filters Direction-selective filters mimic the working principle of the frame-based frequency methods introduced in section 5.1. In particular, they apply a set of spatiotemporal filters to achieve selectivity to specific motion directions and speeds (Gallego et al., 2019). It can be differentiated between approaches in which these filters are handcrafted (e.g. (Orchard et al., 2013; Tschechne et al., 2014; Brosch et al., 2015)) and approaches in which the filters are learned from event data (e.g. (Paredes-Valles et al., 2020)). Orchard et al. (2013) extracted motion patterns by employing synaptic connections with delays and using neurons as coincidence detectors (Gallego et al., 2019). Tschechne et al. (2014) on the other hand proposed directional selectivity in motion selective neurons as a result of a combination of two spatial and two temporal filters whose parameters were tuned based on experimental results. This approach was further developed in (Brosch et al., 2015) where the filter outputs were integrated over larger areas of the visual field and fed back to perform response normalization. While these filter-based approaches have proven to provide accurate OF estimates, it should be noted that the resolution of their estimates is limited by the number of available filters.

4.2. Spiking Neural Networks

In section 3.4 it was shown that insects rely on neural computation to perform visual navigation. This stands in contrast to the algorithmic frame-based methods for OF computation presented in chapter 2. While this section also introduced frame-based methods which utilize artificial neural networks (ANN), these approaches do not accurately represent the working principles of neural networks in living creatures. All computations within the layers of ANNs are synchronized and have to be completed before the final output is provided. Furthermore, they have non-linear but continuous states, meaning that the advantages of event cameras such as their high temporal resolution cannot be fully utilized. In the majority of biological systems, however, spike-like data representations can be found in which information is communicated in discrete spikes which are often triggered by specific events (Pfeiffer and Pfeil, 2018).

In recent years attempts have been made to resolve this discrepancy by introducing spiking neural networks (SNN) which aim to mimic real neural structures more closely. This was done by implementing artificial neurons using mathematical models to describe the voltage output or the probability of spikes over time. This section provides an introduction to SNNs by describing their basic working principle and the most commonly used neuron models. Furthermore, models for synaptic and intrinsic plasticity are provided.

4.2.1. Working Principle

SNNs operate asynchronously and in a parallel fashion. This means that neurons work independently and immediately transmit occurring spikes to postsynaptic neurons without having to wait for the entire input sequence to end. This way important information can be quickly passed through multiple layers of the network resulting in an initial estimate of the output as soon as the first input spikes have arrived. Consequently, the processing of information boils down to the timing of the spikes and the "Identity of the synapses used" (Pfeiffer and Pfeil, 2018). More specifically, timing refers to the firing frequencies, the relative timing between pre- and postsynaptic spikes, and specific firing patterns. The identity of the synapses describes which neurons are connected, whether excitatory or inhibitory synapses are used, and their synaptic strengths. This is also referred to as pseudo-simultaneous information processing (Farabet et al., 2012; Camuñas-Mesa et al., 2014). This kind of processing is very efficient since computations only have to be performed in the active parts of the network.

In Figure 4.4 (retrieved from (Pfeiffer and Pfeil, 2018)) a comparison between a conventional deep neural network (DNN) and a deep SNN is performed. In figure A an example of an ANN with two hidden layers is provided. Figures B and C show the computational output for the several layers as a function of time with a time step of ΔT for the conventional DNN. The different grey values represent the activation values of the various neurons (figure C shows binarized activations). Figure D depicts the propagation of information in a deep SNN with the same architecture. The membrane potential of the neuron marked in green is shown in figure E as a function of time. Finally, the test accuracies of the DNN and the SNN are indicated in Figure F in blue and red, respectively. It can be seen that the DNN only provides outputs after all layers have been processed while the SNN supplies an output almost instantly at an only slightly lower accuracy. These properties make SNN highly attractive for the handling of DVS data since they can fully utilize the asynchronous, event-based, low-power, and low-latency operation of these sensors.

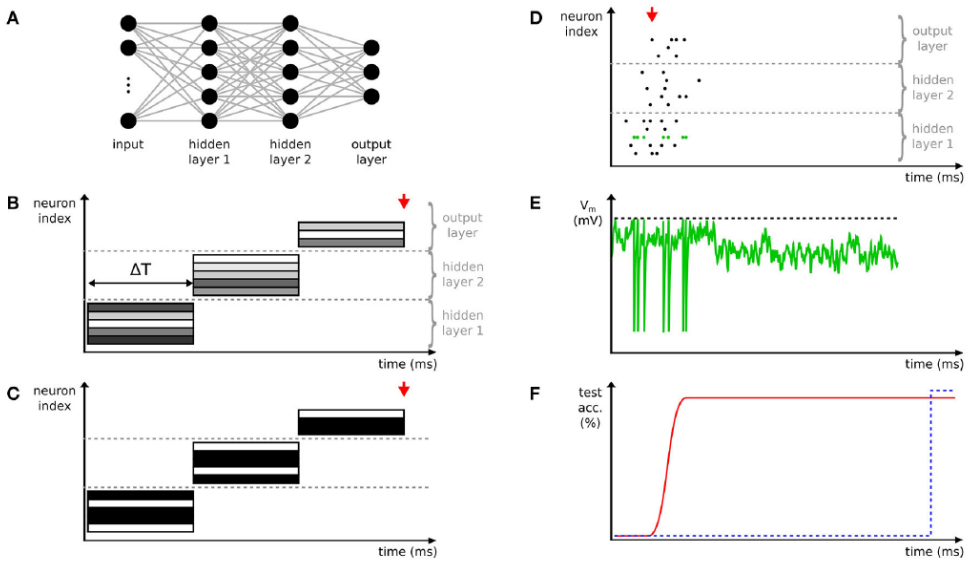


Figure 4.4: Comparison between working principle of deep ANN and SNN. More details can be found in the text. Retrieved from Pfeiffer and Pfeil (2018)

4.2.2. Spiking Neuron Models

Utilizing SNNs requires a mathematical description of the general neuron dynamics presented in section 3.1. For this purpose, a lot of different formulations have been suggested with varying levels of abstraction. These models perform a trade-off between the level of biological plausibility, i.e. how closely they can match the neural dynamics in real living beings, and the required computational complexity. In this section first, a general overview of existing neuron models is provided and subsequently, models which are of particular interest for this research are explained in more detail.

Overview of Existing Neuron Models Generally speaking, it can be differentiated between population and single neuron models. Population models utilize the fact that neurons are grouped into populations of units with similar characteristics in large areas of the brain. The neuron dynamics are then modeled as the mean activity of the neural population which is also called the population activity. Knight (1972); Brunel (2000); Eggert and van Hemmen (2001) e.g. computed the population activity by counting the number of spikes within a specified small time window and subsequently dividing it by the population size and the time window (Gerstner and Kistler, 2002).

Single neuron models on the other hand mimic the dynamics of individual neurons. Here it can be differentiated between conductance-based models that aim to model the actual electrophysiological processes of neurons and phenomenological models which simply aim to match the output of real neurons. The Hodgkin-Huxley Model (Hodgkin and Huxley, 1990) is considered one of the most realistic conductance-based models. It consists of a set of differential equations representing the neuron dynamics as electric circuits comprising capacitors and resistors. While this model is not commonly used in engineering applications due to its high complexity it has served as an inspiration for more efficient, practical approaches. The Morris-Lecar Model (Morris and Lecar, 1981) and FitzHugh-Nagumo Model (FitzHugh, 1961) e.g. simplified the Hodgkin-Huxley Model to two dimensions while still showing high levels of biological plausibility. The Spike-Response-Model (SRM) (Kistler et al., 1997), (Leaky) Integrate-and-Fire Model (LIF) (Stein, 1965), and Izhikevich Model (Izhikevich, 2003) represent examples of phenomenological models which show even lower computational cost which does however come at the price of lower biological feasibility. Since this research is not aiming to precisely model the working principle of neurons but to find an efficient means of OF estimation and OA, the phenomenological models are explained in more detail in the following sections. It should be noted that other more elaborate neuron models exist which include adaptive properties. These models will be presented in subsection 4.2.4.

Spike-Response-Model The Spike-Response-Model (SRM) was first suggested by Kistler et al. (1997) and characterizes the state of a neuron i by the membrane potential v_i. The value of this variable depends on the spiking history of the presynaptic neurons j which are connected to neuron i with synaptic weights ω_{ij}. Assuming that neuron i has fired its last spike at time $\hat{t}_i$, the evolution of v_i after firing can be expressed as:

$$\begin{aligned} v_i(t) = \eta(t-\hat{t}_i) + \sum_j \omega_{ij} \sum_f \epsilon_{ij}\left(t-\hat{t}_i, t-t_j^{(f)}\right) \\ + \int_0^\infty \kappa\left(t-\hat{t}_i, s\right) I^{ext}(t-s)ds \end{aligned} \tag{4.3}$$

If there are no input spikes, this value coincides with the resting value $v_{\text{rest}} = 0$. If the membrane potential reaches the spiking threshold v_{th} an output spike is triggered and the neuron enters into a refractory period. The function η defines the evolution of the membrane potential after the firing threshold has been reached, the kernel ϵ represents the time course of v_i upon receiving a presynaptic spike where $t_j^{(f)}$ is the firing time of the respective presynaptic neurons, and κ describes how the membrane potential varies when an external driving current I^{ext} is received (Gerstner and Kistler, 2002).

(Leaky) Integrate-and-Fire Model Stein (1965) proposed the Leaky Integrate-and-Fire model (LIF) which consists of a parallel combination of a leaky resistor (R) and a capacitor (C). The input from presynaptic neurons is modeled as a current source I(t) charging the capacitor which produces a potential v(t). Once the potential reaches a predefined threshold v_{th}, the capacitor discharges and returns to its resting potential v_{rest}. This process corresponds to a neuron producing an output spike. Defining the time constant of the neuron membrane as $\lambda = R \cdot C$, the change in membrane potential can be expressed as:

$$\lambda \frac{dv(t)}{dt} = v_{\text{rest}} - v(t) + RI(t) \tag{4.4}$$

Right after emitting a spike, the neuron enters a refractory period during which incoming currents no longer affect the membrane potential of the neuron. The LIF-model can be further simplified by neglecting the leaky term in Equation 4.4. In that case the membrane voltage is only influenced by the incoming current which simplifies the neuron dynamics to the differential shown in Equation 4.5. This model is often referred to as the Non-leaky Integrate-and-Fire or simply as the Integrate-and-Fire (IF) model.

$$C \frac{dv(t)}{dt} = I(t) \tag{4.5}$$

Izhikevich model In an effort to combine the biological plausibility of the Hodgkin-Huxley model with the computational efficiency of the LIF model, Izhikevich (2003) proposed the system of ordinary differential equations (ODE) shown in Equation 4.6. Here, v represents the membrane potential and u a membrane recovery variable. The parameters a, b, c, and d are all dimensionless and control the exact dynamics of the neuron. In particular, a determines the time scale of the recovery variable $u(t)$ while b controls the sensitivity of $u(t)$ to the sub-threshold fluctuations of the membrane potential. The reset values of the membrane potential $v(t)$ and the recovery variable $u(t)$, are described by the parameters c and d, respectively. The model was obtained from Hodkin-Huxley-type neuron models using bifurcation methodologies (Izhikevich and Moehlis, 2008).

$$\begin{aligned} \frac{dv(t)}{dt} &= 0.04v^2(t) + 5v(t) + 140 - u(t) + I(t) \\ \frac{du(t)}{dt} &= a(bv(t) - u(t)) \end{aligned} \tag{4.6}$$

4.2.3. Synaptic Plasticity

In section 3.1 the concept of synaptic plasticity was introduced. It represents the basis for learning and memory in biological neural networks and is defined as the ability to change the strengths of the synaptic connection, between pre- and postsynaptic neurons. Generally speaking, it can be differentiated between unsupervised, supervised, and reinforcement learning. Hereafter, an overview of these approaches is given.

Unsupervised Learning In unsupervised learning, no ground truth is available and there is no notion of specific adaptations being 'good' or 'bad' (Morrison et al., 2008). Consequently, changes in synaptic efficacy merely emerge locally from the spatiotemporal patterns of the neural input. Hebb's postulate of "neurons that fire together, wire together" as introduced in section 3.1 has inspired a variety of unsupervised learning approaches, the most common one being Spike-Timing-Dependent Plasticity (STDP) (e.g. (Gerstner et al., 1996; Kempter et al., 1999)).

The underlying idea of this approach is illustrated in Figure 4.5 (retrieved from Madadi Asl et al. (2018)). The change in synaptic strength between two neurons increases or decreases based on the relative timing Δt between their output spikes. If $\Delta t < 0$, i.e. the post-synaptic neuron fires first, the synaptic connection is weakened while it is increased if the post-synaptic neuron fires after the presynaptic spike occurs. These processes are also called long-term depression (LTP) and long-term potentiation (LTP), respectively. Furthermore, the magnitude of this change increases exponentially for decreasing values of the absolute time difference.

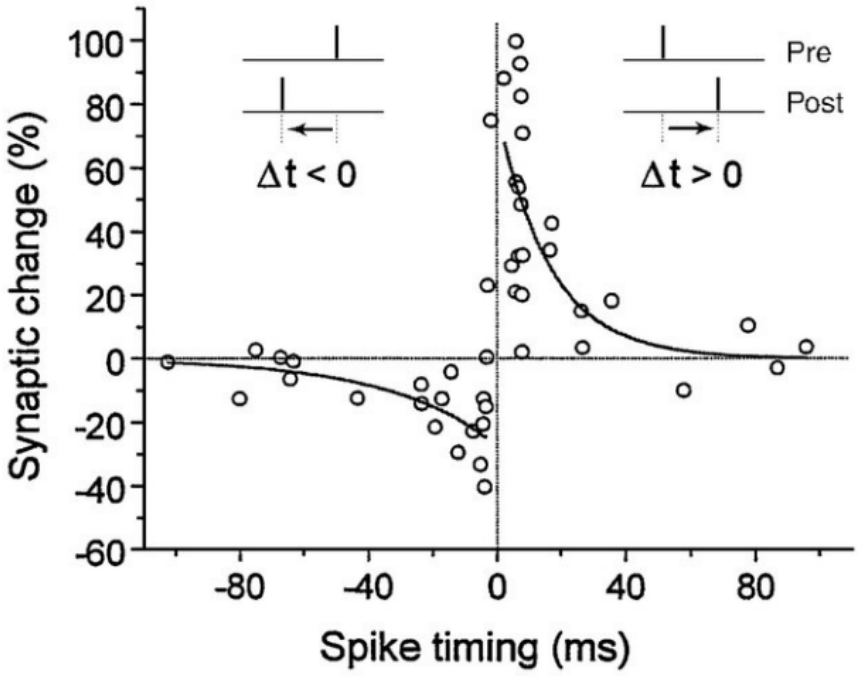


Figure 4.5: Illustration asymmetric STDP learning window. Retrieved from Madadi Asl et al. (2018).

Morrison et al. (2008) provided the general mathematical formulation for STDP shown in Equation 4.7. Here, Δw, τ, and $F(w)$ represent the changes in the synaptic strength, the time constant, and the dependence of the update on the current weight of the synapse, respectively. The superscripts + and − refer to the LTP and LTD, respectively. The time difference $\Delta t = t_j - t_i$ is defined as the temporal difference between the time of the postsynaptic spike t_j and of the presynaptic spike t_i.

$$\begin{cases} \Delta w^+ = F_+(w) \cdot \exp\left(\frac{-|\Delta t|}{\tau_+}\right), & \text{if} \quad \Delta t > 0 \\ \Delta w^- = -F_-(w) \cdot \exp\left(\frac{-|\Delta t|}{\tau_-}\right), & \text{if} \quad \Delta t \leq 0 \end{cases} \tag{4.7}$$

A simple example for the dependence of the update on the current weight of the synapse is given in Equation 4.8 (**?**), where w^{max} represents the maximum value of the weights and $a_{+/-}$, some scaling constants. This approach ensures that the weights do not exceed the previously specified maximum value w^{max}.

$$F_+(w) = (w^{max} - w)a_+, \quad F_-(w) = wa_- \tag{4.8}$$

While traditional STDP formulations have already shown great success in computer vision applications such as image classification (e.g. (Iakymchuk et al., 2015; Mozafari et al., 2019)), most of them still utilize static training approaches. This means they do no longer update their weights to adapt to new input statistics once the training phase has finished. As a consequence, current applications are mainly bound to a specific set of operating environments. In an effort to overcome this issue, several authors have proposed STDP rules that utilize controlled forgetting (Panda et al., 2018; Allred and Roy, 2020; Putra and Shafique, 2021). This means that the weights retain life-long plasticity without the risk of catastrophic forgetting[3]. This is achieved by introducing a leaky term in the formulation of the traditional STDP-formulation. Considering that MAVs typically operate in a wide range of operating conditions these approaches are especially promising for the use of navigational tasks on-board MAVs.

Supervised Learning During supervised learning the output of the network is compared to a previously established ground truth and the synaptic weights are modified to minimize the difference between the actual and the desired output. While a wide range of supervised training methods exists for ANNs, the discontinuous nature of spiking neuron models makes the translation of commonly used approaches, such as backpropagation, to SNNs difficult (Mostafa, 2018). In recent years four main strategies have emerged for the supervised training of SNNs (Pfeiffer and Pfeil, 2018). A brief review of these methods is provided hereafter.

The first approach does not train SNNs in the stricter sense but simply binarizes activations of ANNs for efficient inference (Hubara et al., 2016; Kim and Smaragdis, 2016; Rastegari et al., 2016). While binarized networks resemble the spike-like nature of SNNs, they maintain synchronous layer-by-layer information processing. However, binarization still leads to a more energy-efficient computation on neuromorphic systems due to the sparse activations. There are two major approaches to training these kinds of networks consisting of deterministic and stochastic methods. Deterministic methods commonly make use of so-called straight-through estimators (Bengio et al., 2013) in order to estimate non-differentiable activation functions during backpropagation (Courbariaux et al., 2015). In stochastic methods neuron activations and synaptic weights are often modeled using probability distributions which is also referred to as expectation backpropagation (Soudry et al., 2014). Networks with binary activations usually perform worse than conventional ANN which can be compensated for by increasing the size of the network. Furthermore, it usually takes longer to train binarized ANN due to the increased complexity of the training methods.

In the second common approach the problem of gradient descent in SNN is avoided altogether. This is done by first performing the training on a conventional ANN and subsequently converting it into an SNN. In (Pérez-Carrasco et al., 2013) a first methodical approach to perform this mapping was presented which consists of a conversion from activations of analog neurons to the firing rate of spiking neurons. Furthermore, the weights of the neurons are translated into SNN parameters such as leak rates and refractory time. Alternatively, the Neural Engineering Framework (Eliasmith and Anderson, 2004), can be used to convert ANNs into SNNs. The SNNs obtained this way show similar performances as their underlying conventional neural networks in benchmark tests (Diehl et al., 2015; Sengupta et al., 2018). However, one drawback is the fact that ANN activations can take on both positive and negative values while firing rates in SNNs can only be positive. A possible solution for this issue was proposed by (Pérez-Carrasco et al., 2013) stating that each ANN neuron can be represented by two spiking neurons with each one executing either positive or negative activations exclusively. Doing this does however contribute to another problem introduced by the conversion of activations into spikes, namely the large number of spikes required to represent the dynamics of the network. While spike operations are computationally cheap, SNNs are especially appreciated for their sparse representations meaning that the large number of spikes characteristic of this approach is an undesirable property.

Esser et al. (2015) introduced the term constrain-then-train for methods that account for constraints arising from the characteristics of spiking neurons or the hardware on which the network will be implemented, during training. Similarly to the method presented above, this approach converts the SNNs in question into ANNs for training purposes. However, constrain-then-train approaches train the network for one specific set of spiking neuron model parameters. While this enables them to achieve higher levels of accuracy when compared to converted models without constraints, they have to be retrained for different choices of parameters.

[3]Catastrophic forgetting refers to the tendency of artificial neural networks to forget previously learned patterns when confronted with new input for learning.

In the fourth and final approach training is directly performed on the SNN on the level of spikes. Consequently, these learning techniques are not constrained to mean rate codes but can incorporate the spatiotemporal patterns in spikes during training. These methods are not aiming to mimic learning as it occurs in biological systems but mostly rely on some sort of backpropagation variant to perform training. One possible way to deal with the discontinuities of SNN is to perform stochastic gradient descent on real-valued membrane potentials. The discontinuities introduced by the firing of a spike can then be coped with by applying low-pass filtering (Lee et al., 2016). While this direct approach reduces the number of spikes needed to represent the system, it comes at the cost of larger training times. There are various options for the nature of the target signal. While there are methods that require the output to match the temporal pattern of a specific spike train for a certain input (Bohté et al., 2002), for most models it is enough if a target label is provided. This label can then be represented by having the correct output neuron fire first (e.g. Mostafa (2018)), the highest number of times, or the largest total number of times (e.g. (Lee et al., 2016)).

Reinforcement Learning In reinforcement learning (RL) no explicit ground truth is needed for the training of the network. Instead, some sort of evaluation function is employed which judges the quality of the output by assigning a score to it. Different kinds of RL differ in how they use the score in order to update the network parameters. A popular approach for RL in SNNs is the use of evolutionary algorithms (EA) in which the performance of different sets of network parameters is evaluated. The most successful ones are then selected and recombined to produce 'offspring'. This process is repeated until a suitable set of network parameters is found. Using this approach Belatreche et al. (2003) e.g. were able to successfully update both synaptic efficacies and delays in an SNN. By manually evaluating the behavior of a mobile robot utilizing an SNN Stewart et al. (2016) were able to make it avoid a frontal mirror. However, EAs still show very large computational costs (or require manual supervision as in Stewart et al. (2016)) which makes it difficult to apply them to large SNNs.

4.2.4. Intrinsic Plasticity

In section 3.1 it was explained that neurons have been observed to change their intrinsic properties such as the voltage threshold in addition to adjusting their synaptic weights. This concept is also called intrinsic plasticity (IP) and is useful in keeping the neuron dynamics within a desired operating range.

Lazar et al. (2007) proposed a voltage threshold update rule which is designed to make neurons spike an average k out of N times. Similarly, Li et al. (2018) presented an update rule for the neuron's excitability in the Izhikevich model. This update rule is aiming to adjust the output firing activity to match a specific goal. While these approaches do successfully limit the operating range of the neurons in question, they do not optimize the output distribution to maximize the information content. Baddeley et al. (1998) showed that the output response of neurons in the visual cortical follows an approximately exponential distribution. Furthermore, they argued that this exponential distribution yields maximal neural information transmission or in other words maximal information entropy[4].

One of the first attempts to achieve an exponential output spikes distribution was shown in Stemmler and Koch (1999). Using the Hodgkin-Huxley neuron model, an update rule for the conductance was proposed which aimed to maximize the overlap between the stimulus and firing rate entropy. Triesch (2005) matched the output spike distribution of Sigmoid neurons to an exponential distribution by using the Kullback Leibler (KL) divergence. This approach served as an inspiration for several spike-based applications. Li and Li (2013) e.g. applied the proposed method to LIF-neurons by simply substituting LIF-parameters into the suggested formulation. Zhang and Li (2019); Zhang et al. (2020) on the other hand derived an update rule for LIF-neuron parameters from scratch. To deal with the discontinuous nature of spiking neurons Zhang and Li (2019) assumed a constant input firing rate which allowed them to derive a continuous formulation for the output firing rate. This could then be matched to an exponential distribution using the KL-divergence. Zhang et al. (2020) tackled the issue of discontinuity by introducing a soft-reset spiking neuron which resets the neuron membrane potential in a mathematically continuous manner after spikes occur. This allowed them to match the membrane potential to an exponential distribution using once again the KL divergence.

[4] Information entropy describes the average level of information or uncertainty contained in a random variable's possible outcomes.

While these models showed improvements in accuracy of up to 17% in classification tasks (Zhang and Li, 2019), they all implement IP only in the neurons of hidden layers within the network while utilizing standard neuron dynamics in the output layers. It is thus not clear if the same improvements could be achieved in networks without hidden layers.

4.2.5. Spike-Based Optic Flow Estimation

Since SNNs allow to fully utilize the sparse output of event cameras they are promising for insect-inspired OF estimation and OA. However, SNNs are still in their infancy and as of now, they are not many spike-based approaches to OF estimation. The few that exist can be divided into methods that simulate SNNs on conventional synchronous processors and those that are implemented on real neuromorphic hardware (see section 4.3). In this section approaches using simulated SNNs are presented while the latter ones are introduced in section 4.3.

In (Orchard et al., 2013) a first attempt was presented to compute OF with the help of an SNN using events created by an ATIS. The proposed network consists of an array of LIF neurons which each cover an area of 5x5 pixels in the visual field. All connections had the same weights but varying delays are applied, such that the neural connections act as spatiotemporal filters. This is illustrated in Figure 4.6 (retrieved from Orchard et al. (2013)). For each pixel location 128 different filters were defined, one for each of 8 different speeds and orientations and for the 2 polarities. Furthermore, a second layer of neurons with a larger receptive field was implemented to overcome issues related to the aperture problem. While the network was able to provide accurate OF estimates, it required a very large number of neurons which makes it unsuitable for the use onboard MAVs.

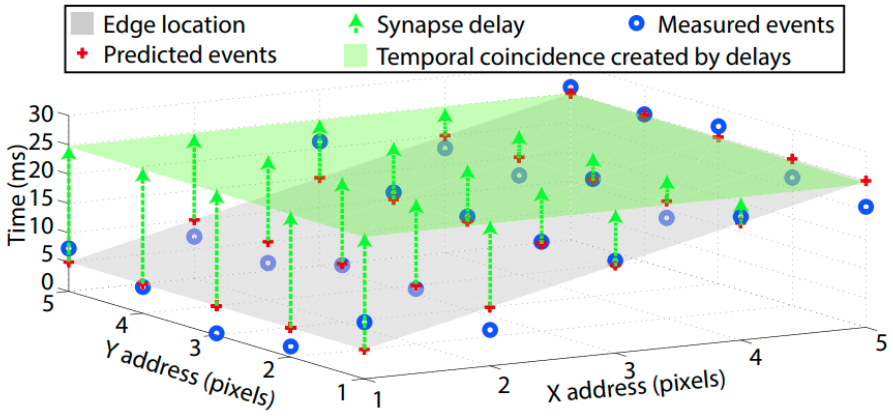


Figure 4.6: Illustration of spatio-temporal filters for optic flow estimation. Retrieved from Orchard et al. (2013).

Paredes-Valles et al. (2020) also proposed an approach that utilizes spatiotemporal filters with delays. However, rather than manually defining the filters, they were learned from a set of event-based video sequences using a novel STDP learning rule. This rule naturally constrains the values of the weights to be in a range between zero and one. A hierarchical network architecture was proposed consisting of five layers, in which the input is downsampled, features are extracted and pooled, and sparse local and global OF estimates are provided. For this purpose, a modified LIF neuron was proposed which decreases the current of neurons based on their spiking history. This work is especially promising since it utilizes a biologically plausible unsupervised learning rule and mimics some of the adaptive properties of neurons that have been introduced in subsection 3.3.2. However, despite using an adaptive formulation for the LIF neuron, this approach still requires the use of different parameters for varying operating environments. Furthermore, this approach did not utilize life-long learning which means that the weights can no longer change once they have converged. This makes it difficult to employ it on MAVs which typically operate in a wide range of environments.

4.3. Neuromorphic Processors

In neuromorphic processors neural networks are directly implemented on hardware enabling asynchronous and parallel processing similar to the neural computations performed in biological brains. When using conventional synchronous architectures to implement event-driven vision systems, the events need to be converted to frames first, such that advantages gained by the asynchronous nature of event cameras are compromised. Furthermore, their asynchronous working principle makes neuromorphic processors more power-efficient. These properties make them highly desirable for insect-inspired OF estimation and OA. This section provides a brief introduction to neuromorphic processors. The explanation of their exact working principle is beyond the scope of this research. However, commercially available processors are introduced in subsection 4.3.1 and approaches utilizing neuromorphic processors for OF estimation are shown in subsection 4.3.2.

4.3.1. Available Neuromorphic Processors

In Table 6.1 an overview of currently available neuromorphic processors is depicted (adapted from Gallego et al. (2019)). It can be differentiated between analog, digital, or mixed analog/digital processors depending on the method used to implement the neuron model. In analog implementations the desired behavior is achieved by utilizing inherent properties of electronic devices while digital processors use Boolean logic-based gates to achieve this (Schuman et al., 2017). In mixed analog/ digital systems a combination of the two approaches is used. Furthermore, the various processors differ in the number of neurons available per chip and in whether or not they support on-chip learning.

Table 4.2: Overview of currently available neuromorphic processors. Adapted from Gallego et al. (2019).

	SpiNNaker	TrueNorth	Loihi	DYNAP	Braindrop
Supplier	U. Manchester	IBM	Intel	aiCTX	Stanford U.
Reference	Furber et al. (2013)	Akopyan et al. (2015)	Davies et al. (2018)	Moradi et al. (2018)	Neckar et al. (2019)
Year	2011	2014	2018	2017	2018
Neuron model	Software	Digital	Digital	Analog	Analog
Neurons/chip	4k	1024k	131k	1k	4k
On-chip learning	Yes	No	Yes	No	No

4.3.2. Optic Flow Estimation with Neuromorphic Processors

To the best of the author's knowledge all existing methods for OF estimation utilizing end-to-end spiking processing make use of some sort of elementary motion detector. Richter et al. (2014) e.g. implemented the classical Reichardt detector (Reichardt and Rosenblith, 1961) on the SpiNNaker platform (Furber et al., 2013). Similarly, a motion detector based on the direction-selective ganglion cells in the rabbit's retina was implemented on a custom Very-Large-Scale-Integration (VLSI) chip in (Giulioni et al., 2016) and on the TrueNorth chip (Akopyan et al., 2015) in (Haessig et al., 2018). Finally, Milde et al. (2018) proposed the spiking EMD (sEMD) which encodes the time for an object to move across an observer's retina into a burst of spikes. For this purpose, they used the adaptive exponential LIF neuron circuit presented in (Indiveri et al., 2006) utilizing an STDP rule to update the synaptic efficacies. However, since the model was implemented on a custom chip only comprising 8 circuits it was tested in simulation.

To the best of this author's knowledge, only few works have attempted to implement the LGMD neuron onto neuromorphic hardware. One of these attempts was presented in (Salt et al., 2017) where the traditional LGMD model was implemented on the mixed-signal analog-digital neuromorphic device presented in (Indiveri et al., 2015). This was done by introducing two additional layers, the IP and the IS layer which helped to reduce the overall number of neurons required (see subsection 5.2.2). Salt et al. (2020) further built on this approach by proposing differential evolution and Bayesian optimization techniques to ease the search for suitable parameters.

While all of these methods were able to successfully create motion selectivity in the respective SNNs, they are still subject to the general limitations of EMD/LGMD models introduced in subsection 3.3.2. These mainly include sensitivity to changes in lighting and structural properties of the visual field. While these are accounted for in real biological systems by adaptation, this is not/only partially the case in the proposed approaches. This means that they would require some sort of gain-scheduling in order to increase their robustness and be useful for real-world applications involving MAVs.

5

Insect-Inspired Obstacle Avoidance

The previous chapter has provided an extensive discussion on insect-inspired/neuromorphic means of OF estimation. However, while the ultimate goal of this research is to use the improved means of OF estimation to perform OA on-board MAVs, it has not been established yet how OF can actually be used to perform collision avoidance. For this purpose, an overview of existing insect-inspired/neuromorphic approaches is presented in this chapter. In section 5.1 and section 5.2 frame-based and event-based/neuromorphic approaches are presented, respectively.

5.1. Frame-Based Obstacle Avoidance

Since the vast majority of neuromorphic OA methods are translated from frame-based approaches, an overview of the latter ones is presented in this section. This shall allow drawing inspiration for future work in this field. It can be distinguished between algorithmic and neural approaches which will be presented in subsection 5.1.1 and subsection 5.1.2, respectively.

5.1.1. Algorithmic

Algorithmic approaches to OA utilize some sort of prior knowledge to translate the OF input into collision avoidance commands. In fact, many methods mimic the OA techniques observed in flying insects as explained in section 3.4. They do this by utilizing OF observables (see section 2.3) to determine an avoidance direction. In the remainder of this section, the most influential algorithmic OA methods using frame-based cameras will be presented.

Time-to-Contact Inspired by the working principle of OA in insects, many collision avoidance models utilize the time-to-contact (TTC) presented in section 3.4. An early attempt of this was shown in (Camus, 1995a), where the TTC was used to detect a chair positioned in front of a mobile robot. This was done with the help of Equation 2.10, meaning that the FoE had to be computed in addition to the OF. Assuming forward translation and only a single object in the field of view (FoV), this was done by computing the intersection of all OF vectors. While this approach was able to successfully predict the TTC, it relied on several assumptions including a constant velocity, only one visible obstacle in the FoV, and unidirectional motion. Since these assumptions are most likely not fulfilled for OA on autonomous MAVs, this approach cannot directly be applied.

To overcome the assumption of a single obstacle in the FoV several authors (e.g. (Low and Wyeth, 2012; Souhila and Achour, 2007)) also performed velocity measurements which allowed them to create a depth map from the TTC which could then be used to avoid obstacles. However, this method requires the agent to be equipped with a velocity sensor which might not be possible for autonomous MAVs with strict weight and power limitations.

Nearness Maps To circumvent the issue of requiring a velocity sensor to create depth maps, the concept of relative nearness has been utilized by several authors. Assuming a spherical eye Bertrand et al. (2015) e.g. derived the expression for the relative nearness shown in Equation 5.1 where μ and ν represent the

nearness to an obstacle and the velocity of the agent, respectively and the variables ϕ and ϵ the azimuth and the elevation within a spherical coordinate system, respectively.

$$(v\mu(\epsilon,\phi))^2 = \mathrm{OF}(\epsilon,\phi)^2_{\hat{\phi}} + \frac{\mathrm{OF}(\epsilon,\phi)^2_{\hat{\epsilon}}}{\sin^2(\epsilon)} \tag{5.1}$$

The translational OF was separated from the rotational OF by copying insect's saccadic flight strategy (see subsection 3.3.1) and singularities at the FoE (see subsection 2.3.3) were avoided by slightly changing the flight direction after each saccade. This way the agent could obtain several similar relative nearness maps but with slightly differently positioned FOEs. By integrating the obtained nearness maps, a map with no singularities was obtained.

To determine the obstacle avoidance direction from the relative nearness map, the Center-Of-Mass Average Nearness Vector (COMANV) was computed. It points in the average direction of close-by obstacles and is obtained by averaging the nearness vectors along the elevation and subsequently taking the vector sum of the averaged nearness maps. The opposite direction is then used to avoid collisions. This is illustrated in Figure 5.1 (retrieved from (Bertrand et al., 2015)) where the first image depicts the top view of two cylindrical objects and an agent performing the saccadic flight strategy. The second and third images depict the OF field and the time-integrated OF field obtained by averaging the data from several saccades, respectively. The resulting nearness map is shown in the fourth image. In images 5 and 6, the result of integrating the nearness vectors along the elevation and the resulting COMANV are shown, respectively.

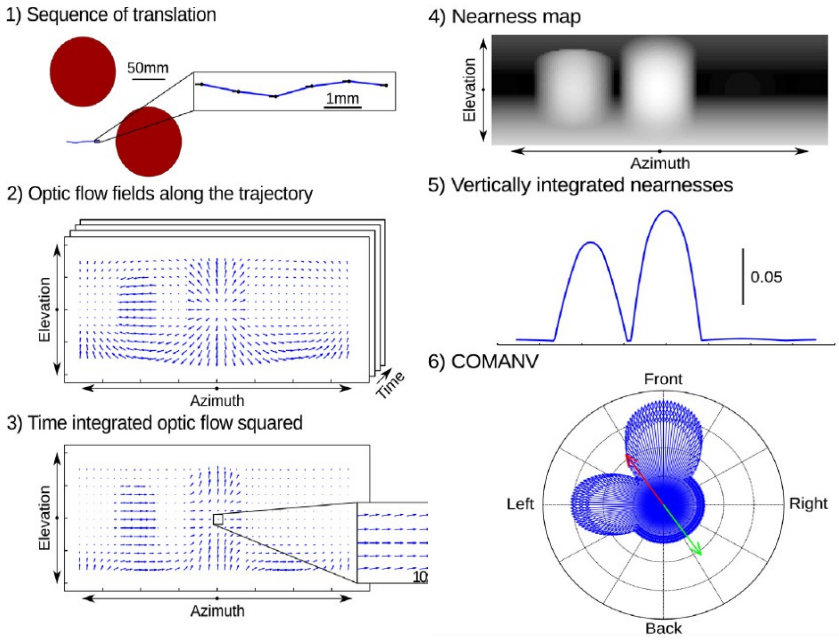


Figure 5.1: Illustration of collision avoidance method proposed in (Bertrand et al., 2015). Details can be found in the text. Retrieved from Bertrand et al. (2015).

Simulation results showed that this method could successfully avoid obstacles using geometrically computed OF and computed with EMDs. However, these tests restricted the movement of the agent to a 2D plane which is undesirable for applications involving MAVs.

The model presented above was also implemented on a walking hexapod robot in (Meyer et al., 2016). To counteract rotations introduced by the walking motion, the head of the robot was kept stable such that the recorded images only included translational motion. The OF was obtained using EMDs and the model was tested in two different simulation environments. The first one included a set of pillars in a virtual box and the second one the reconstruction of a natural environment including trees. The robot was able to avoid all obstacles in both simulations.

Divergence To obtain an estimate for the collision avoidance direction the OF divergence is also commonly considered since it does not require the computation of a nearness map. Zufferey and Floreano (2005) e.g. presented an attempt to use OF for OA on a very light-weight indoor aircraft (30 g). Due to the strict weight limitations, only two 1D cameras using 28 pixels each were used to determine the OF. All computations were performed on-board using a microchip.

To determine an avoidance direction, the OF divergence was estimated as the difference between the signed OF on the right and the left side of the FoE. Since the side with the higher OF magnitude is generally located closer to the detected obstacle, a maneuver pointing towards the opposite direction was triggered if a certain OF divergence threshold had been reached. This is illustrated in Figure 5.2 (retrieved from Zufferey and Floreano (2005)), where the indoor aircraft is shown in the left column approaching a wall frontally (first row) and at an angle of 30 deg (second row). The second column shows the respective OF fields and the third column the divergence as a function of the distance to the wall. The FoV of the 1D-cameras is illustrated by the rectangle ABCD.

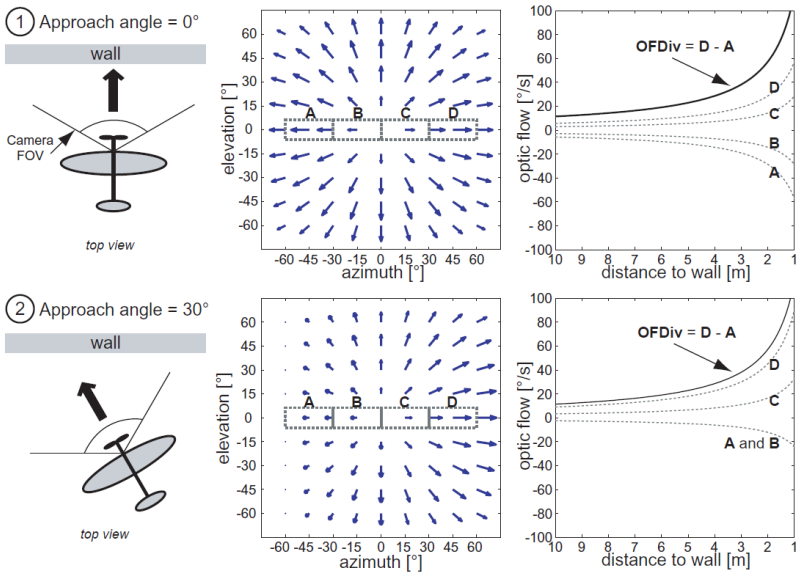


Figure 5.2: Optical flow as seen by a 120 deg FoV 2D camera when approaching a wall. The first column shows the 30 g indoor aircraft when approaching a wall frontally (first row) and when approaching at an angle of 30 deg (second row). In the second column, the OF fields for the corresponding cases are shown. Column three depicts the OF divergences for the two different cases. Retrieved from Zufferey and Floreano (2005).

While this approach utilizes a very lightweight flying robot, it was tested in a very favorable environment consisting of an empty flight arena constrained by clearly structured curtains. Consequently, no judgments about the robustness of this method can be made.

The OF divergence approach was extended to three dimensions in Beyeler et al. (2009) by introducing the polar coordinates θ and ψ in the visual field as illustrated in Figure 5.3 (retrieved from Beyeler et al. (2009)). The FoV was then divided into equal sections along a circle of a specific polar angle $\hat{\theta}$ resulting in a constant inter-azimuth angle $\hat{\psi}$. This is shown on the right side of Figure 5.4 (retrieved from Beyeler et al. (2009)). The polar angle for the section definition was set to $\hat{\theta} = 45$ deg considering that obstacles located at high elevations are non-dangerous and that there is little OF information available at the FoE at $\theta = 0$ deg (see subsection 2.3.3). This is illustrated on the left side of Figure 5.4.

The approach was implemented on a flying wing equipped with several optic mouse sensors that each were able to determine a single OF estimation. By assuming a constant aircraft velocity and that the vision direction was aligned with the body axis of the flying wing a control law was established. This control law directly determined the control signals for pitch and roll by summing the OF estimates at each section scaled by suitable weights. This way the flying wing always steered away from the direction with the largest OF magnitude.

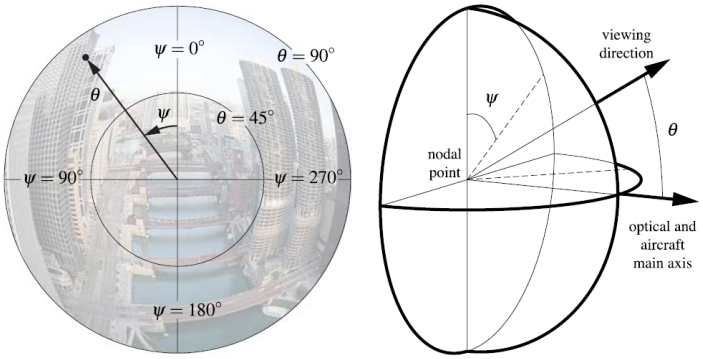


Figure 5.3: Definition of the polar angle θ and the azimuth angle ψ. Retrieved from Beyeler et al. (2009).

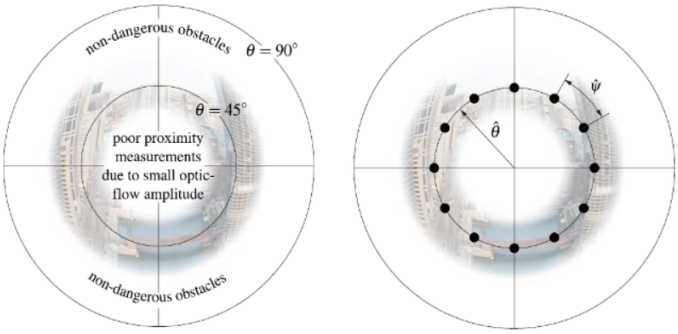


Figure 5.4: Illustration of chosen polar angle $\hat{\theta}$ (left) and division of the FoV (right). Retrieved from Beyeler et al. (2009).

The model was tested in both simulation and a real-world environment. It was found that the flying was indeed able to avoid obstacles most of the time. However, several instances were observed in which the wing crashed into obstacles symmetrically positioned in front of it. The reason for this was that the OF on the left and the right side of the (FoV) canceled each other out leading to an overall small magnitude for the control inputs. Furthermore, only small OF vectors were present at the FoE making it hard to avoid small frontal obstacles whose edges do not extend to other sections of the FoV.

Other Approaches As shown in the previous section many OF-based approaches suffer from the lack of information at the FoE. As a consequence, several authors have proposed insect-inspired OA methods which do not (purely) rely on OF. These are presented hereafter.

Mori and Scherer (2013) tackled this problem by employing a relative size detector that can predict collisions utilizing the fact that approaching obstacles appear to become bigger in the visual field. In order to do this, SURF features (Bay et al., 2008) were used, which are useful because they are relatively fast to compute and invariant to large-scale changes. This way, detected features in two subsequent frames could be matched to determine whether the area around the key point had increased and the drone was approaching an obstacle. The algorithm used a reactive guidance law, meaning that the drone simply flew sideways when an obstacle was detected and then continued to fly towards its goal. The proposed method was implemented on a Parrot AR Drone and was tested in an obstacle course resembling a forest. It was found that it avoided 97% of all obstacles successfully. While this method was able to avoid frontal obstacles that are difficult to detect with OF, the obstacles needed to have sufficient texture such that the SURF key points could be extracted.

Another method trying to overcome the difficulty of avoiding frontal obstacles was developed by Croon et al. (2011). It introduced a new visual cue, the appearance variation, which describes the variation in texture that can be found in a single image. The algorithm was based on two fundamental ideas. The first idea was that the size of an object increases in an image as the camera gets closer to it while other objects move out of view. The second idea was that the detailed texture of objects ahead becomes more and more visible upon

approach. Furthermore, it was assumed that the appearance variation is decreased more by the first effect than it is increased by the second one, leading to an overall reduction in appearance variation. During experiments, it was found that this assumption does not hold for highly textured obstacles. However, for highly textured surfaces OF-based methods perform well, implying that the two methods complement each other. In fact, it was found that a combination of both methods performed better than each one individually.

In (Klaptocz et al., 2010) it was taken into account that even insects are not always able to detect and avoid obstacles on time and sometimes do crash into objects. Consequently, a flying robot was designed that can recover from collisions autonomously without the intervention of a human being. The designed MAV is a hybrid utilizing both a wing and propellers to achieve efficient flight while still being able to hover. Robustness to collisions was achieved by reinforcing the perimeter of the wing with a carbon rod and applying a protective ring around the propellers as illustrated in Figure 5.5. To ensure autonomous self-recovery, a passive system was applied which returns the aircraft to a favorable position after a crash. This was achieved with the geometry and the right positioning of the center of gravity (CoG) as shown in Figure 5.6.

Figure 5.5: Design of a MAV robust to collisions. Retrieved from Klaptocz et al. (2010).

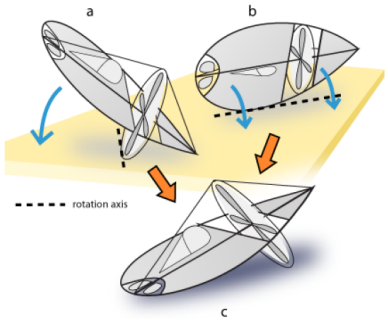


Figure 5.6: Illustration of passive recentering mechanism. Retrieved from Klaptocz et al. (2010).

Flight tests were performed which showed that light contact with walls did not always make the aircraft crash. Instead, it could fly along the wall with its front tip touching the surface. This is a behavior that can also be observed in insects looking for an exit after flying into a windowpane. Furthermore, the MAV always settled for one of the stable positions after falling to the ground and was in most cases able to subsequently take off again. While this approach is of course not suitable for environments with human beings present where a collision could cause injury, it is a promising backup for applications in cluttered environments with no humans present in case the primary means of collision avoidance is not able to detect an obstacle.

5.1.2. Neural

While the algorithmic approaches introduced in the previous section already represent a step towards insect-inspired obstacle avoidance, insect behavior can be mimicked more accurately when making use of neural architectures. In these architectures, computations are performed by a set of interconnected artificial neurons resembling the neural networks that can be found in living beings. These approaches will be presented in this section. It will be differentiated between models aiming to copy the working principle of specific neurons present in insects and those imitating the general neural architecture.

Specific Neuron Models Due to the simple conversion of the LGMD neuron's output to motor commands as explained in section 3.4 it has been a popular foundation for OA approaches (e.g. (Hu et al., 2017; Blanchard et al., 2000; Cizek et al., 2017)). In these models, simple OA reactions, such as a turn to the left, were triggered when the LGMD model detected an obstacle. While all of these models still showed the desired behavior of outputting a larger number of spikes when approaching an obstacle, some obstacles could not be avoided after rotations had been performed. Furthermore, the network's parameters had to be tuned according to the robot's velocity. Consequently, some sort of gain scheduling would be required to use these approaches on-board MAVs

In previous implementations of the LGMD-model very simple avoidance strategies were applied that did not take into account the direction from which an obstacle is approaching. Inspired by research on escape direction control in cockroaches, Yue and Rind (2009) made use of two LGMD neurons, corresponding to the left and the right eye of the locust, to determine an OA direction. This direction was then either defined as the opposite direction of the neuron spiking more frequently or the scaled difference of the spiking activity of the two neurons. The model was tested on a Khepera mobile robot for which the computations were performed in real-time on a PC connected with a cable. It was found that the system was able to avoid approaching obstacles of different colors, shapes, and materials. However, occasionally an escape was triggered by translating rather than looming motion which had previously been shown to be prevented by lateral and feed-forward inhibition.

The determination of the OA direction was even further refined in (He et al., 2020) where a deep neural network was used in order to map the LGMD neuron's response onto a collision-avoidance direction. Since this approach showed higher robustness and used a lightweight implementation, it represents a promising approach for OA on-board MAVs.

Generic Neural Networks As opposed to the approaches presented above, hereafter methods are introduced which do not aim to model specific neurons in the insect-brain but utilize generic architectures in which the synaptic efficacies emerge during training. In (Low and Wyeth, 2007) e.g. a neural network was introduced that learns OA techniques from the structural properties of OF. The neural network was trained by correlating OF fields obtained from the robot's camera with heading directions computed with the help of a laser range finder. A three-layer feed-forward fully interconnected neural network was used to train the robot. Using the method presented in (Bouguet et al., 2001) the OF was computed and subsequently, the OF magnitude and direction were used as input to the neural network. The second layer of the neural network consisted of 8 hidden neurons and the third and final layer comprised 31 neurons with each one representing a heading direction in the range from -15 to 15 deg/s. The network structure is illustrated in Figure 5.7 (retrieved from Low and Wyeth (2007)). The weights of the neural network were determined using an R-Prop training algorithm. After training it, the presented neural network was tested in an office environment and it was found that 79.6% of the chosen heading velocities were within a 5 deg/s interval from the ones suggested by the laser range finder. Furthermore, some frontal obstacles were not avoided.

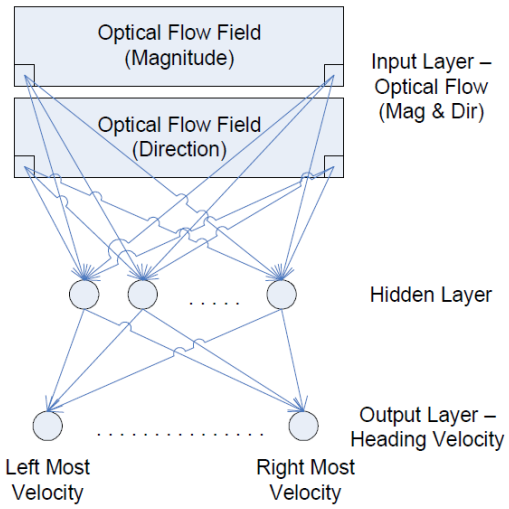


Figure 5.7: Architecture of SNN presented in (Low and Wyeth, 2007). Retrieved from Low and Wyeth (2007)

Considering that fast-moving vehicles need to detect obstacles from larger distances for safe operation, Mancini et al. (2016) implemented a convolutional neural network (CNN) that can detect obstacles at a large range and very high frequencies by creating a depth map of the surroundings. They achieved this by trading some depth accuracy for increased computational speed. In particular, they used a Fully Convolutional Network (FCN)

which does not make use of a fully connected layer and thus reduces the number of parameters that need to be determined. This in return increases the training speed and allows the network to operate at very high frame rates. Two possible network input options were presented. The first one consisted of only RGB images and the second of both RGB images and the corresponding OF fields. The OF was computed offline using the method presented in (Brox et al., 2004). The network was structured in an encoder-decoder architecture as shown in Figure 5.8 (retrieved from Mancini et al. (2016)). The proposed network was tested on the KITTI data set (Geiger et al., 2012). It was found that it performed better when using both the RGB image and OF as input. Furthermore, when tested on the KITTI data set it showed similar performance as state-of-the-art models. However, it showed a comparatively weak performance for close-range estimations meaning that it should be used in conjunction with a model that performs well for short-range obstacle detection. Furthermore, the proposed model has not yet been implemented in an obstacle avoidance control loop and no control strategy has been suggested.

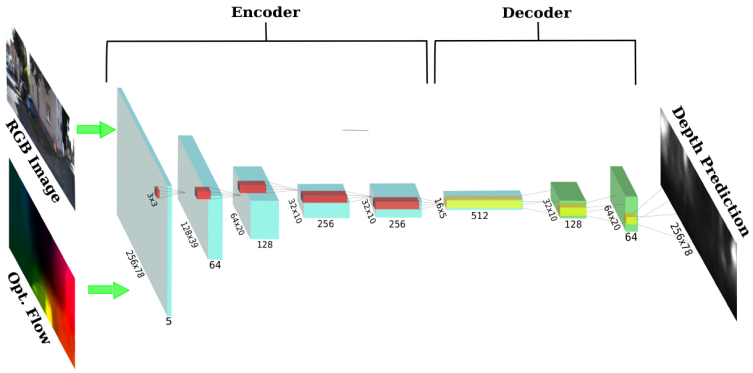


Figure 5.8: Architecture of the CNN presented in (Mancini et al., 2016). Retrieved from Mancini et al. (2016)

This gap was closed by Ponce et al. (2018) who also utilized a CNN to create a depth estimate for OA. However, rather than outputting an entire depth map, the network provided only one distance estimate for the entire image. Furthermore, the CNN only consisted of the input, a convolutional layer, a fully connected layer, and the output layer as shown in Figure 5.9 (retrieved from Ponce et al. (2018)). The ground truth data was collected in the simulation robot-environment V-REP (Rohmer et al., 2013), where a Pioneer robot equipped with a camera was moving straight until the distance measured with an ultrasonic sensor dropped below 0.2 m. For obstacle avoidance, a simple fuzzy controller was implemented on a differential wheeled robot which mapped the distance estimates to the velocities for the left and the right wheel. In particular, large and medium distances were mapped to a clockwise (CW) rotation of both wheels, while small distances were mapped to a CW rotation of only the right wheel. It was found that the CNN performed significantly better when using OF as input and was able to estimate distances with high accuracy. Furthermore, it was able to successfully avoid obstacles. However, since the CNN only output one distance estimate at a time, the model was not able to cope with more than one obstacle in the FoV. In addition, a very simple testing environment was chosen which showed very little clutter and clear textures.

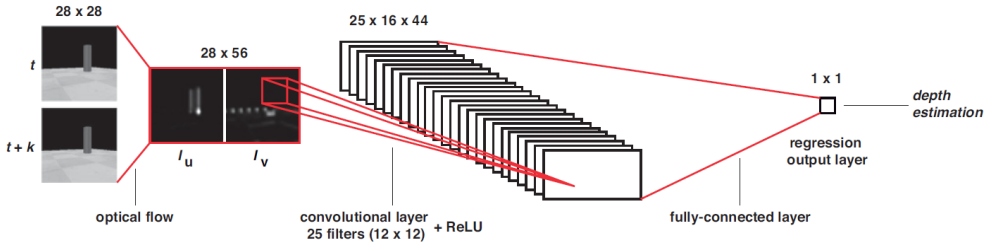


Figure 5.9: Architecture of the CNN presented in (Ponce et al., 2018). Retrieved from Ponce et al. (2018)

5.2. Event-Based Obstacle Avoidance

Similarly to event-based OF estimation, event-based OA is still in its infancy, and accordingly the number of proposed methods is limited. For this reason, a detailed overview of the existing methods is provided in this section. Following the same approach as in the previous section, first algorithmic methods are illustrated in subsection 5.2.1 and subsequently, models that utilize neural architectures are introduced in subsection 5.2.2.

5.2.1. Algorithmic

As explained in subsection 5.1.1 algorithmic approaches utilize some sort of prior knowledge to determine an OA direction from the OF. To the best of this author's knowledge, all event-based model-based methods are direct translations from existing frame-based approaches. As will be shown hereafter these include models utilizing the TTC and nearness maps.

Time-to-Contact Inspired by algorithms that have been realized with conventional cameras, Clady et al. (2014) implemented an event-based model that utilizes the TTC to perform OA. To obtain the OF, they used the plane-fitting method presented in (Benosman et al., 2014). Since this method only provides the normal OF, Clady et al. (2014) derived an expression for the TTC that merely depends on the FoE and the spatial gradient of the events. They did this by realizing that the FoE is located in the "negative semi-plane defined by the normal motion flow vector" (Clady et al., 2014). This is illustrated on the left side of Figure 5.10 (retrieved from Clady et al. (2014)) where several OF vectors and their corresponding negative semi-planes are depicted. The gray area indicates the region of the expected location of the FoE. This was translated into a probability map as shown on the left side of Figure 5.10.

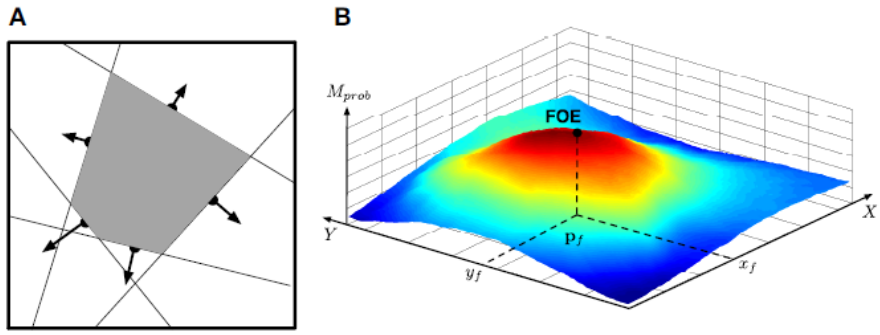


Figure 5.10: Demonstration of the determination of the FoE. Retrieved from Clady et al. (2014).

In order to detect obstacles, the FoV was divided into four regions of interest (ROI). If the TTC dropped below a certain threshold in any of the ROI an OA reaction steering away from this region was triggered. It was found that the TTC algorithm performed as well as a laser range finder but operated at a much higher frequency. However, this method also showed some drawbacks. First of all, it assumed a constant velocity for the robot which is not necessarily true. Furthermore, the obstacles were assumed to be located on the ground which especially for MAV applications is not always the case. Finally, it was reported that pitch motions could result in unstable estimations of the FoE location. Taking into account that many MAVs are equipped with propellers and pitch to accelerate, this is an unfavorable property.

Some of these issues were addressed in (Colonnier et al., 2018) where a similar method was used to perform OA. The proposed model was implemented on a quadrotor and the OF was determined using the algorithm described in (Aung et al., 2018). To determine the FoE, the method in (Clady et al., 2014) was used but the algorithm was only applied to selected ROIs. For each ROI, a Kalman filter was applied to obtain an overall TTC estimate from the noisy individual measurements. This made the algorithm more robust against small pitch movements occurring during flight. To compute the OA direction, the frame was divided into a left, middle, and right region for which the respective TTC averages were computed such that the drone could fly towards the region with the largest TTC. This model showed improvements when compared to (Clady et al., 2014) as it could handle the pitch movements naturally occurring during the flight of a quadcopter. Furthermore, it did not assume that obstacles are located on the ground. However, it was only validated in a 2D scenario and also assumed a constant flight velocity.

Dinaux et al. (2021) aimed to further increase the computational efficiency of the negative semi-half plane approach with their proposed FAst ITerative Half- plane method (FAITH). This method did not consider all OF vectors for the FoE estimate but instead, the estimate was initialized with two randomly selected vectors. Subsequently, the FoE area resulting from iteratively choosing a third random vector was computed. The algorithm was stopped once the updated area was no longer smaller than the previous one. Using this method an obstacle avoidance maneuver was triggered if the computed FoE was located within the highest priority obstacle region determined based on clustered TTC estimates. The proposed method was tested in both simulation and onboard a MAV where it was programmed to fly straight-forward at a constant speed encountering a pillar roughly halfway through the sequence. It was found that the MAV could successfully avoid 80% of all obstacles (Dinaux et al., 2021) which makes it promising for the use on MAVs with limited computational resources.

Nearness Maps Inspired by Bertrand et al. (2015), another algorithm that utilizes relative nearness estimates was implemented in (Milde et al., 2015). However, the OF was retrieved using the event-based method proposed in (Benosman et al., 2014). Similarly to the algorithm presented by Bertrand et al. (2015), the relative nearness was obtained by computing the retinotopic norm of the OF. In order to retrieve an OA direction from the relative nearness estimates, each pixel was mapped onto a virtual angle ranging from $-\pi$ at the very left of the pixel array to π at the very right of the array. By taking the vector sum of the relative nearness estimates computed within a certain time window, the vector pointing towards close objects was determined. The obstacle avoidance direction was then defined as pointing in the opposite direction. It was shown that the algorithm originally proposed by Bertrand et al. (2015) also works when the OF is determined using an event-based camera despite the much more sparse information provided. However, the approach was evaluated offline in an open-loop meaning that it cannot be guaranteed that it will also work in real-time in a closed-loop system. Furthermore, a robotic platform rather than a MAV was used, restricting the problem to a 2D plane.

5.2.2. Neural

In order to fully utilize the asynchronous nature of event-based cameras and close the gap to truly bio-inspired OA, several authors have proposed neural networks to perform collision avoidance. Similarly to the frame-based approaches, these can once again be divided into methods aiming to mimic specific neurons and those simply utilizing general neural architectures.

Specific Neuron Models In subsection 5.1.2 several methods that make use of the LGMD model to detect obstacles have been presented. However, all of these methods used conventional frame-based data as input meaning that they did not take full advantage of the spiking nature of the LGMD. In (Salt et al., 2017) a spiking implementation of the LGMD and DCMD neurons was presented. While this approach was based on the same neuron model, additional constraints arose from the fact that a neuromorphic processor was used. Consequently, the original LGMD model had to be adjusted to adhere to these constraints. This was done by introducing two additional layers, the IP and the IS layer which helped to reduce the overall number of neurons required. The architecture of the adjusted neural network is shown on the left side of Figure 5.11 (retrieved from (Salt et al., 2017)). The black and the red lines represent excitatory and inhibitory connections, respectively and the solid and the dashed lines correspond to fast and slow connections, respectively.

Similarly to the method in (Yue and Rind, 2009), the determination of the OA direction was achieved by splitting the FoV and assigning one LGMD neuron to each section. However, Salt et al. (2017) used four (top, bottom, left, right), rather than two (left, right) sections. To determine the escape direction, the four LGMD neurons were linked through excitatory connections to their DGMD counterparts (left to right, right to left, etc.) and through inhibitory connections to their corresponding DGMD neurons. This is illustrated on the left side of Figure 5.11 where solid lines represent excitatory connections and dashed lines inhibitory connections. The proposed model showed increased performance when compared to previous LGMD implementations except for cases with particularly sparse inputs (at e.g. very low velocities). Furthermore, it was shown that the spiking behavior of the four DCMD neurons did indeed lead to the desired escape headings. Since the parameters of the proposed model were very sensitive to the operating environment, Salt et al. (2020) further improved it by utilizing a self-adaptive LIF neuron model and differential evolution. It was found that this approach was indeed able to autonomously find suitable parameters, which makes it very promising for adaptive OA.

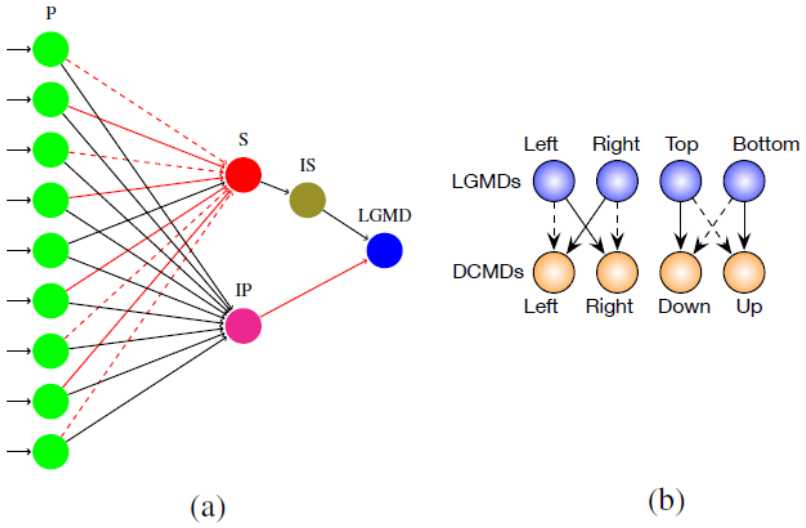


Figure 5.11: Illustration of network presented in (Salt et al., 2017). a) Overview of the neural architecture. b) Illustration of the control law. Retrieved from Salt et al. (2017).

Generic Neural Networks In this section, neural networks for OA with generic neural architectures are presented. Unlike frame-based camera output, event-based data already holds rich information about the spatio-temporal dynamics of the scene. For this reason, the majority of event-based neural networks for OA omit the OF extraction step and directly feed the output of the event camera into the neural networks.

Milde et al. (2017) e.g. proposed an SNN comprising seven populations of neurons, including two speed populations (speed and excitatory speed), two obstacle populations (left, right), two turn populations (left, right), and a gyro population. This is depicted in Figure 5.12 (retrieved from Milde et al. (2017)), where the numbers represent the weights of neural connections. The speed (sp) population receives constant input from the excitatory speed (exc) population driving the robot forward. The obstacle populations (OL and OR) consist of 32 obstacle neurons divided equally over the left and the right FoV of the image, respectively. The 128 x 128 pixels of the DVS are partitioned into 32 vertical columns, and events occurring within those sections trigger the corresponding neurons in the left (OL) or right (OR) OA populations. After a specified number of events have arrived, the obstacle neurons spike and emit an excitatory signal to the connected turn population (DR and DL) of the opposite direction (e.g. left to right) and an inhibitory signal to the turn population of the same direction (e.g. left to left). Finally, the number of spikes is counted, and by multiplying the firing rates of the turn populations with a scaling factor, velocity commands for the left and right wheels of the robot are obtained.

It was found that the robot was indeed able to avoid obstacles and that it showed variable speeds and turning rates depending on the location of the obstacle. At low speeds, some obstacles were not avoided due to the lack of generated events. Furthermore, some objects of lighter colors could not be detected since their contrast was too low to be spotted by the DVS. However, this method was once again restricted to a 2D-environment meaning that it is not clear how it would perform on a flying platform.

Inspired by birds as e.g. the female budgerigar Schoepe et al. (2019) developed a mobile agent that can avoid obstacles while following a sound source. For this purpose, an embedded Dynamic Vision Sensor (eDVS) (Lichtsteiner et al., 2008) and a Neuropmorhic Auditory Sensor (NAS) (Jiménez-Fernández et al., 2017) were used as sensory input. The neural architecture consisted of four groups of neural populations, including an optical flow encoder (OFE), a sound source direction (SSD), a lateral sound transmitter (LST), and a decision-making winner take all (DMWTA) population. The output maps of the OFE and the SSD networks were arranged topographically and projected onto the DMWTA population. The OFE population comprised 32 x 32 LIF neurons which downsampled and filtered the raw eDVS data to reduce noise and 32 x 32 sEMD (Haes-

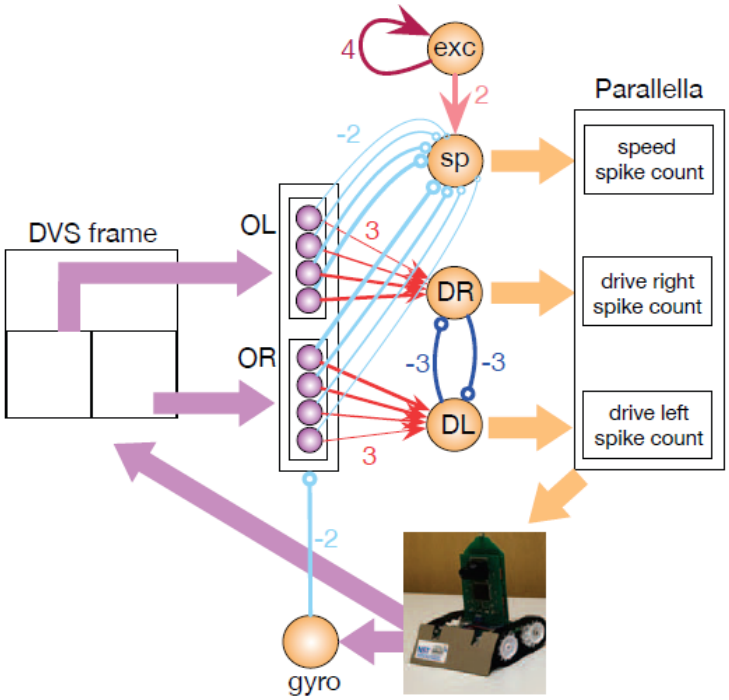


Figure 5.12: Overview of the neural architecture presented in (Milde et al., 2017). Retrieved from Milde et al. (2017).

sig et al., 2018) which provided topographically arranged relative distance information in form of OF. The DMWTA also contained 32 LIF neurons and followed a "winner take all" policy. To prevent the agent from colliding with an obstacle while following the sound source, the OFE population strongly inhibited the corresponding neurons in the DMWTA population such that a neuron that corresponds to the direction of the sound does not win if an obstacle is located in the same direction. In the SSD population the direction from which the sound is coming is determined by correlating the left and the right side of the auditory input. Subsequently, the corresponding neuron in the LST population is excited which in return excites its two neighboring neurons in the same layer. This process is repeated until a neuron corresponding to zero optical flow is found. At the same position the DMWTA population releases a spike and inhibits all other decision-making neurons. This way, the winning heading direction always turns out to be at the position as close as possible to the sound source while still showing zero OF. This architecture is illustrated in Figure 5.13 (retrieved from Schoepe et al. (2019)). During experiments it was found that the headings determined by the neural network did indeed point towards the sound source but away from obstacles. This model is useful in the sense that it enables the integration of navigation with OA. While this comprised following a sound source, in this case, it could also be adjusted to follow another kind of signal.

The methods for OA presented so far showed very simple behavior with the agents usually just moving towards the direction containing the fewest obstacles. In (Stewart et al., 2016) a model that allows for more complex behavior was presented and tested on a wheeled robot. The method relied on both explicit programming and autonomous learning to achieve collision avoidance. Inspired by living creatures who are born with a set of reflexive behaviors and then learn more complex behaviors during their lifetime, the idea was to implement a set of automatic behaviors using explicit programming and then let the agent learn more complex behaviors autonomously. The basic set of skills included going forward when no obstacle is in sight, backing up when being too close to an obstacle, and avoiding left or right when approaching one. To achieve this, a neural network consisting of five layers was proposed. The first layer takes the sensor measurements as an input and is connected through sensory neurons to the action strength layer. This layer consists of four neurons each representing one of the basic actions (moving forward, stopping, turning left, turning right). Through a layer of action neurons, this layer is finally connected to two output neurons representing the velocity of the left and right wheels of the robot. The neural architecture of this model is shown in Figure 5.14 (retrieved from (Stewart et al., 2016)).

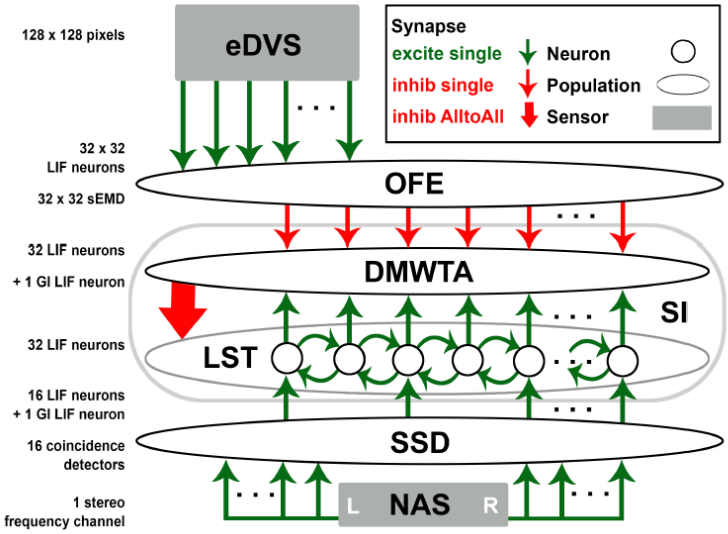


Figure 5.13: Overview of neural architecture presented in (Schoepe et al., 2019). Retrieved from Schoepe et al. (2019).

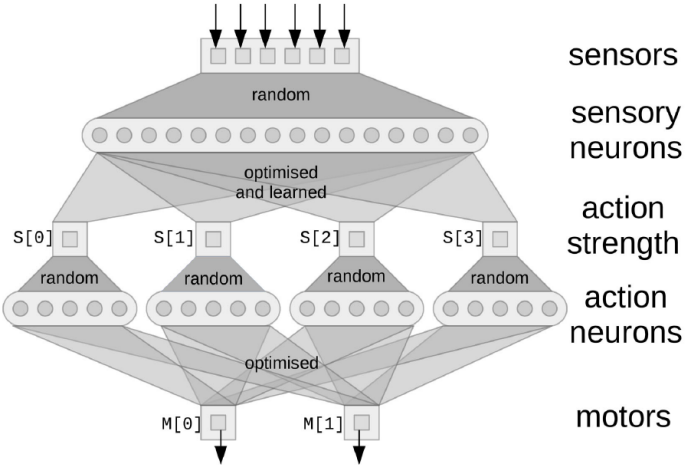


Figure 5.14: Illustration of the network architecture presented in (Stewart et al., 2016). Retrieved from Stewart et al. (2016).

After training the network to perform the basic skills, the robot was able to explore its surroundings autonomously and all input data, as well as internal states, were recorded during that phase. If the robots performed a desirable action by chance, the corresponding data was saved and used to update the weights to ensure similar behavior in the future. Using this model the network was indeed able to learn complex behavior such as avoiding a frontal mirror. However, the supervised training was performed by manually observing the robot. It would thus be difficult to generalize this approach to more extensive training.

6

Synthesis and Conclusion

In the previous chapters a literature study about insect-inspired optic flow (OF) estimation and obstacle avoidance (OA) was presented. This was done in light of the ever-increasing use of autonomous Micro Aerial Vehicles (MAVs) which have very strict weight and power limitations and typically operate in complex and cluttered environments. Insects were used as a source of inspiration since they are known to utilize OF to perform complex navigational tasks despite their limited neural resources. However, existing methods for insect-inspired OF estimation are often fragile and require different sets of parameters for varying environments. Consequently, this literature review has aimed to investigate how to make OF estimation more robust and how to then use OF to perform insect-inspired OA on-board MAVs. The purpose of this chapter is to synthesize the literature presented in this report to conclude these two questions.

6.1. Robust Event-Based Optic Flow Estimation

The first step towards identifying more robust event-based means of OF estimation was to understand the shortcoming of traditional frame-based methods in the context of insect-inspired OA on-board MAVs. For this purpose, a brief overview of frame-based OF estimation methods was provided and their drawbacks were illustrated by explaining the two most influential approaches (Lucas and Kanade, 1981; Horn and Schunck, 1981). It was found that frame-based methods perform redundant computations since they have to consider the brightness values of all pixels. Furthermore, it was argued that the synchronous frame-based nature of these approaches leads to motion blur and high latencies. Finally, the majority of frame-based methods assume a constant brightness of the environment which limits the flexibility of these approaches.

To overcome some of these shortcomings, it was subsequently explained how insects perform motion detection. It was established that all computations in the insect brain are performed on a neural basis which allows for low-latency asynchronous processing. Following the elaborations in (Nériec and Desplan, 2016) it was shown that most insects perceive their environment with the help of compound eyes which have a very low resolution and visual acuity and also function in an asynchronous manner. The obtained brightness information is then converted into ego-motion information with the help of motion detectors such as the elementary motion detector (EMD) and the Lobula Giant Movement Detector (LGMD). Several computational models for the EMD were presented (Reichardt and Rosenblith, 1961; Joesch et al., 2010; Eichner et al., 2011) utilizing different combinations of ON and OFF events. Based on the neural implementation of the EMD described in (Borst et al., 2019), it was argued that the model presented in (Eichner et al., 2011) is the most biologically plausible. However, it was also shown that none of these models accounted for the neural adaptations observed in the lobula plate (Borst and Egelhaaf, 1987; de Ruter van Steveninck et al., 1986). Consequently several computational models which do account for this were presented afterwards (Clifford et al., 1997; Clifford and Langley, 1996; Sarikaya and Ogmen, 1994).
When comparing conventional frame-based methods for OF estimation to motion detection in insects, it was found that the main discrepancies lie in the fact that insects employ neural asynchronous computation while frame-based methods rely on synchronous algorithmic approaches. Furthermore, it was established that the insect eye shows a lower resolution and visual acuity than typical frame-based cameras. Finally, the adaptation to changes in the environment cannot be found in frame-based approaches.

The next step was to investigate which technological advancements have already been made to address these discrepancies. On a more general level event-based cameras (Lichtsteiner et al., 2008; Brandli et al., 2014; Posch et al., 2011) were introduced which mimic the asynchronous spike-like nature of visual perception in living beings. Furthermore, it was shown that spiking neural networks (SNN) in combination with neuromorphic processors can imitate the asynchronous processing of information in insects very closely. In order to analyze to what degree existing engineering applications already address these discrepancies, an overview of all event-based approaches introduced in this literature review is given in Table 6.1

Table 6.1: Overview of presented event-based methods for optic flow estimation. "Simulation" in the Implementation tab refers to models in which spiking neurons are simulated on conventional synchronous Von-Neumann architectures.

	Input	Model	Implementation	Adaptive	Bio-inspired	Trained
Benosman et al. (2011)	DVS	Gradient/ Algorithmic	Simulation	No	No	No
Benosman et al. (2014)	DVS	Plane-fitting/Algorithmic	Simulation	No	No	No
Rueckauer and Delbruck (2016)	DVS/DAVIS	Plane-fitting/Algorithmic	Simulation	No	No	No
Aung et al. (2018)	ATIS	Plane-fitting/Algorithmic	FPGA	No	No	No
Barranco et al. (2014)	DVS/DAVIS	Plane-fitting/Algorithmic	Simulation	No	No	No
Hordijk et al. (2018)	DVS	Plane-fitting/Algorithmic	Simulation	No	No	No
Brosch et al. (2015)	DVS	Direction-selective filters/Algorithmic	Simulation	No	Yes	No
Orchard et al. (2013)	ATIS	Direction-selective filters/SNN	Simulation	No	Yes	No
Tschechne et al. (2014)	DVS	Direction-selective filters/Algorithmic	Simulation	No	Yes	No
Paredes-Valles et al. (2020)	DVS/DAVIS	Direction-selective filters/SNN	Simulation	Yes	Yes	Yes
Richter et al. (2014)	DVS	EMD/SNN	SpiNNaker	No	Yes	No
Giulioni et al. (2016)	ATIS	EMD/SNN	Custom VLSI	No	Yes	No
Haessig et al. (2018)	ATIS	EMD/SNN	TrueNorth	No	Yes	No
Milde et al. (2018)	DVS	EMD/SNN	Custom chip/Simulation	Yes	Yes	Yes
Salt et al. (2017)	DVS	LGMD/SNN	VLSI	No	Yes	No
Salt et al. (2020)	DVS	LGMD/SNN	VLSI	Yes	Yes	Yes

Since all of the presented approaches utilize event cameras they all address the issues of motion blur and high latency arising from the use of frame-based cameras. However, the different methods use models with varying levels of biological plausibility. All of the presented gradient and plane-fitting methods e.g. do not make use of SNNs but perform algorithmic computations. This means that they cannot fully benefit from the asynchronous nature of the employed event-based vision sensors. Furthermore, the gradient and plane-fitting models are not bio-inspired which makes a translation onto a spiking architecture more difficult. While the direction-selective filters draw direct inspiration from nature, Tschechne et al. (2014); Brosch et al. (2015) still utilized algorithmic models for their implementation. This means that these approaches were also not able to benefit from all of the advantages resulting from the use of event-based cameras. Orchard et al. (2013) and Paredes-Valles et al. (2020) on the other hand realized the direction-selective filters with the help of SNNs which means that these approaches are highly biologically plausible. However, neither of them implemented the SNNs on neuromorphic hardware but merely simulated the corresponding neuron models on synchronous architectures. The approaches in (Richter et al., 2014; Giulioni et al., 2016; Haessig et al., 2018; Milde et al., 2018; Salt et al., 2017; Salt et al., 2020) closed this final gap towards an end-to-end spiking implementation of OF estimation by employing neuromorphic processors. However, all of these approaches are implementations of EMD/LGMD models which are known to be particularly sensitive to changes in ego-motion velocity and patterns in the visual field. While several neuron models for adaptation or intrinsic plasticity (IP) were introduced (e.g. (Lazar et al., 2007; Li et al., 2018; Zhang et al., 2020; Li and Li, 2013)), only three of the presented OF estimation methods utilize some sort of adaptive mechanism (Milde et al., 2018; Paredes-Valles et al., 2020; Salt et al., 2020). These three approaches are also the only ones utilizing synaptic plasticity. In particular, they all make use of spike-timing-dependent plasticity (STDP) which is a biologically plausible way of learning. The method in (Salt et al., 2020) is particularly interesting as it uses adaptation to address the issue of finding suitable parameters. This adaptation is however only applied during training and can thus not be used to increase the operating range of an existing network.

Since the exact neural processes involved in insect motion detection are not fully understood yet, approaches utilizing learning are especially interesting as they require little prior knowledge. Furthermore, they might even shed light on the working principle of motion detection in insects. It can be concluded that the approaches presented by Milde et al. (2018); Paredes-Valles et al. (2020); Salt et al. (2020) are as of now the most promising ones for the use of OA on-board MAVs.

However, there are still some discrepancies that these approaches do not address. While they utilize some sort of adaptation as observed in insects, the approach in (Paredes-Valles et al., 2020) has so far only been implemented in simulation and not on neuromorphic hardware. Furthermore, it still requires the use of varying parameters for different operating environments and makes a distinction between a training and an inference phase. In natural systems, however, there is no such distinction. In fact, life-long learning can be observed in living beings which means that synapses remain plastic and keep changing throughout a lifetime. Several models for life-long learning with STDP were presented using forgetting weights (Putra and Shafique, 2021; Panda et al., 2018; Allred and Roy, 2020). However, to the best of this author's knowledge, there exist no approaches to event-based OF estimation which utilize this life-long synaptic plasticity.

It can be concluded that all the building blocks for robust event-based OF estimation already exist but have not been combined yet. This analysis showed that the approach presented in (Paredes-Valles et al., 2020) appears to be the most promising one and can be further improved by employing some of the adaptive neuron models and life-long learning approaches presented in this literature study. Since (Paredes-Valles et al., 2020) proposed a novel neuron model, an extensive analysis will have to be performed to evaluate how the existing approaches for adaptation and life-long learning can be translated to this model. Finally, for a robust insect-inspired end-to-end spiking means of OF estimation, it would also have to be investigated how the approach could be translated onto neuromorphic hardware.

6.2. Insect-Inspired Obstacle Avoidance

To answer the second research question asking how the obtained OF can be used to perform insect-inspired OA, once again the individual SRQs proposed in chapter 1 will be addressed and subsequently synthesized to arrive at an answer to the main question.

The first step was to establish how insects perform this task. It was explained that the output of the Locust Giant Movement Detector (LGMD) in the locust brain is linked one-to-one to a motor neuron triggering escape responses. It is however still largely unknown how motion estimates from elementary motion detectors (EMD) are converted to OA motor commands. It was shown that several behavioral models exist proposing that insects mainly rely on visual cues such as the time-to-contact (TTC), the temporal contrast in the visual scene, the size of objects in the visual field, and motion integrated over space and time to perform OA. Based on experiments performed in (Tammero and Dickinson, 2002), it was argued that the integration model appears to be the most biologically plausible.

Since most event-based OA approaches are directly derived from frame-based methods and since event-based OA is still in its infancy, both frame-based and event-based methods were considered. An overview of all presented approaches is shown in Table 6.2. It can be seen that there is a large overlap between the models of OA in insects and the presented frame-based approaches. Rind and Bramwell (1996); Blanchard et al. (2000); Yue and Rind (2009) e.g. implemented models of the LGMD-neuron, Camus (1995a); Low and Wyeth (2012) utilized the TTC, Bertrand et al. (2015); Meyer et al. (2016) made use of nearness maps, Beyeler et al. (2009); Zufferey and Floreano (2005) computed the divergence by spatially integrating the OF field and Mori and Scherer (2013) considered the relative size of objects in the visual field. While Croon et al. (2011) did not directly consider the size of objects, the proposed appearance variation is also directly linked to the proximity of objects in the visual field. In addition, several approaches were presented in which OA commands are learned in artificial neural networks (ANN) (Low and Wyeth, 2007; Mancini et al., 2016; Ponce et al., 2018). Again, these approaches are particularly interesting since they require no prior knowledge which allows compensating for the uncertainty about the neural implementation of OA in insects.

However, since all of these approaches are frame-based they naturally do not show the asynchronous spike-based processing of OA in insects. This gap was closed by the event-based OA approaches presented in (Clady et al., 2014; Colonnier et al., 2018; Dinaux et al., 2021; Milde et al., 2015) which provided spike-based implementations of the TTC and nearness maps. The sparse output of event cameras has however made it difficult to translate non-OF-based approaches dealing with frontal obstacles (Mori and Scherer, 2013; Croon et al., 2011) to event-based applications. Furthermore, the approaches utilizing nearness maps and the TTC do not make use of SNNs which makes them less biologically plausible. While several SNN-based approaches were presented (Milde et al., 2017; Schoepe et al., 2019; Stewart et al., 2016) they do not make use of synaptic plasticity, unlike their frame-based counter-parts. Furthermore, they do not use OF as an input as it is believed to happen in insects. Instead, they directly utilize the output of the event-based vision sensors. It is interesting to note that all of the proposed SNNs for OA integrate input spikes in space and time which was presented as the most biologically plausible model for OA in insects (Tammero and Dickinson, 2002).

Based on this synthesis it can once again be concluded that the building blocks for insect-inspired OA already exist but still need to be put together. In particular, it should be investigated whether spike-based OA still profits from using OF as an input or if the raw event data already contains sufficient information about the spatio-temporal dynamics of the scene. Considering that the majority of OA methods struggle with frontal obstacles it should furthermore be explored whether some of the approaches addressing this issue can be translated to event-based applications. Finally, taking into account the uncertainty about the exact neural processes involved in OA in insects it would be interesting to implement an SNN with synaptic plasticity which can learn OA from OF or raw events using bio-inspired rules such as STDP. Ideally, this could also lead to behavioral actions such as a saccadic flight strategy to decompose OF and deal with frontal obstacles in a similar fashion as presented in (Bertrand et al., 2015).

Table 6.2: Overview of presented obstacle avoidance models.

		Method	Platform	2D/3D
Frame-based	Algorithmic	TTC (Camus, 1995a)	Ground robot	2D
		TTC with velocity measurement (Low and Wyeth, 2012)	Ground robot in simulation	3D
		Divergence (Zufferey and Floreano, 2005)	30g aircraft	2D
		Divergence (Beyeler et al., 2009)	Flying wing	3D
		Frontal: relative size detector (Mori and Scherer, 2013)	Parrot AR Drone	3D
		Frontal: appearance variation (Croon et al., 2011)	DelFly II	3D
		Frontal: merged nearness maps (Bertrand et al., 2015)	Simulation	3D
		Frontal: merged nearness maps (Meyer et al., 2016)	Hexapod robot	3D
		Crash resilient (Klaptocz et al., 2010)	Light flying robot	3D
	Neural	LGMD: first computer model (Rind and Bramwell, 1996)	Simulation	3D
		LGMD: implementation on robot (Blanchard et al., 2000)	Khepera mobile robot	3D
		LGMD: multidirectional (Yue et al., 2009)	Khepera mobile robot	3D
		ANN: laser ground truth, velocity command output(Low and Wyeth, 2007)	Pioneer robot	3D
		CNN: artificial images ground truth, depth map output (Mancini et al., 2016)	Only provides depth map	3D
		CNN: sonar ground truth, single depth output (Ponce et al., 2018)	Pioneer robot in simulation (V-REO)	3D
Event-based	Algorithmic	TTC (Clady et al., 2014)	Pioneer 2 robot	2D
		TTC with Kalman filter (Colonnier et al., 2018)	Parrot Bebop 2	3D
		TTC FAITH (Dinaux et al., 2021)	MAV	2D
		Merged nearness maps (Milde et al., 2015)	Pioneer 2-DX robot	3D
	Neural	LGMD (Salt et al., 2017)	Quadrotor UAV	3D
		SNN: dividing FOV (Milde et al., 2017)	Pushbot mobile robot	3D
		SNN: follow sound source (Schoepe et al., 2019)	Ground robot	3D
		SNN: reinforcement learning (Stewart et al., 2016)	PushBot robot	3D

Bibliography

Adelson, E. H. and Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America. A, Optics and image science*, 2 2:284–99.

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G., Taba, B., Beakes, M., Brezzo, B., Kuang, J. B., Manohar, R., Risk, W. P., Jackson, B., and Modha, D. S. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557.

Allred, J. and Roy, K. (2020). Controlled forgetting: Targeted stimulation and dopaminergic plasticity modulation for unsupervised lifelong learning in spiking neural networks. *Frontiers in Neuroscience*, 14.

Arnold, R., Yamaguchi, H., and Tanaka, T. (2018). Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. *Journal of International Humanitarian Action*, 3.

Aung, M. T., Teo, R., and Orchard, G. (2018). Event-based plane-fitting optical flow for dynamic vision sensors in fpga. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5.

Baddeley, R., Abbott, L., Booth, M., Sengpiel, F., Freeman, T., Wakeman, E., and Rolls, E. (1998). Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proceedings. Biological sciences / The Royal Society*, 264:1775–83.

Barranco, F., Fermüller, C., and Aloimonos, Y. (2014). Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, 102:1537–1556.

Barron, J., Fleet, D., and Beauchemin, S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77.

Baudry, M. (1998). Synaptic plasticity and learning and memory: 15 years of progress. *Neurobiology of Learning and Memory*, 70:113–118.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359. Similarity Matching in Computer Vision and Multimedia.

Belatreche, A., Maguire, L. P., McGinnity, M., and Wu, Q. (2003). An evolutionary strategy for supervised training of biologically plausible neural networks. In *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*, pages 1524–1527.

Bengio, Y., Léonard, N., and Courville, A. C. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, abs/1308.3432.

Benosman, R., Clercq, C., Lagorce, X., Ieng, S., and Bartolozzi, C. (2014). Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417.

Benosman, R., Ieng, S.-H., Clercq, C., Bartolozzi, C., and Srinivasan, M. (2011). Asynchronous frameless event-based optical flow. *Neural networks : the official journal of the International Neural Network Society*, 27:32–7.

Bertrand, O. J. N., Lindemann, J. P., and Egelhaaf, M. (2015). A bio-inspired collision avoidance model based on spatial information derived from motion detectors leads to common routes. *PLOS Computational Biology*, 11(11):1–28.

Beyeler, A., Zufferey, J.-C., and Floreano, D. (2009). Vision-based control of near-obstacle flight. *Autonomous Robots*, 27(3):201–219.

Blanchard, M., Rind, F., and Verschure, P. (2000). Collision avoidance using a model of the locust lgmd neuron. *Robotics and Autonomous Systems*, 30:17–38.

Bohté, S., Kok, J., and Poutré, la, J. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 1-4:17–37.

Borst, A. (1990). How do flies land? from behavior to neuronal circuits. *BioScience*, 40(4):292+.

Borst, A. (2009). Drosophila's view on insect vision. *Current biology : CB*, 19:R36–47.

Borst, A. and Egelhaaf, M. (1987). Temporal modulation of luminance adapts time constant of fly movement detectors. *Biological Cybernetics*, 56(4):209–215.

Borst, A., Haag, J., and Mauss, A. S. (2019). How fly neurons compute the direction of visual motion. *Journal of Comparative Physiology. A, Neuroethology, Sensory, Neural, and Behavioral Physiology*, 206:109 – 124.

Bouguet, J.-Y. et al. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4.

Brandli, C., Berner, R., Yang, M., Liu, S.-C., and Delbruck, T. (2014). A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49:2333–2341.

Brosch, T., Tschechne, S., and Neumann, H. (2015). On event-based optical flow detection. *Frontiers in neuroscience*, 9:137.

Brown, S. (2001). General properties of intercellular communication in the nervous system. In *Nerve Cells and Nervous Systems*. Springer, London.

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3024, pages 25–36.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, 8:183–208.

Camus, T. (1995a). Calculating time-to-contact using real-time quantized optical flow. *NIST Interagency/Internal Report (NISTIR)*.

Camus, T. (1995b). Real-time quantized optical flow. In *Proceedings of Conference on Computer Architectures for Machine Perception*, pages 126–131.

Camuñas-Mesa, L. A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2014). Event-driven sensing and processing for high-speed robotic vision. In *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, pages 516–519.

Chen, A., Frangopol, D., Ruan, X., Hallermann, N., and Morgenthal, G. (2014). Visual inspection strategies for large bridges using unmanned aerial vehicles (uav). In *Bridge Maintenance, Safety, Management and Life Extension*, pages 661–667.

Chen, N. F. Y. (2017). Pseudo-labels for supervised learning on event-based data. *CoRR*, abs/1709.09323.

Christiansen, M. P., Laursen, M., Jørgensen, R., Skovsen, S., and Gislum, R. (2017). Designing and testing a uav mapping system for agricultural field surveying. *Sensors*, 17:2703.

Cizek, P., Milicka, P., and Faigl, J. (2017). Neural based obstacle avoidance with cpg controlled hexapod walking robot. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2017-May, pages 650–656.

Clady, X., Clercq, C., Ieng, S.-H., Houseini, F., Randazzo, M., Natale, L., Bartolozzi, C., and Benosman, R. (2014). Asynchronous visual event-based time-to-contact. *Frontiers in neuroscience*, 8:9.

Clifford, C. W. G., Ibbotson, M. R., and Langley, K. (1997). An adaptive reichardt detector model of motion adaptation in insects and mammals. *Visual neuroscience*, 14(4):741–749.

Clifford, C. W. G. and Langley, K. (1996). A model of temporal adaptation in fly motion vision. *Vision research*, 36(16):2595–2608.

Collett, T. (1978). Peering - a locust behaviour pattern for obtaining motion parallax information. *The Journal of Experimental Biology*, 76.

Colonnier, F., Vedova, L. D., Teo, R. S. H., and Orchard, G. (2018). Obstacle avoidance using event-based visual sensor and time-to-contact processing. In *Australasian Conference on Robotics and Automation (ACRA)*, volume 2018-December, pages 1–10.

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3123–3131, Cambridge, MA, USA. MIT Press.

Croon, G., Ho, H. W., De Wagter, C., Van Kampen, E.-J., Remes, B., and Chu, Q. (2013). Optic-flow based slope estimation for autonomous landing. *International Journal of Micro Air Vehicles*, 5:287 – 297.

Croon, G., van de Weerdt, E., De Wagter, C., and Remes, B. (2011). The appearance variation cue for obstacle avoidance. pages 1606 – 1611.

Davies, M., Srinivasa, N., Lin, T., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y., Wild, A., Yang, Y., and Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99.

de Ruter van Steveninck, R. R., Zaagman, W. H., and Mastebroek, H. A. K. (1986). Adaptation of transient responses of a movement-sensitive neuron in the visual system of the blowfly calliphora erythrocephala. *Biological Cybernetics*, 54(4–5):223–236.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S., and Pfeiffer, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Dinaux, R., Wessendorp, N., Dupeyroux, J., and Croon, G. C. H. E. d. (2021). Faith: Fast iterative half-plane focus of expansion estimation using optic flow. *IEEE Robotics and Automation Letters*, 6(4):7627–7634.

Duisterhof, B. P., Li, S., Burgués, J., Reddi, V. J., and de Croon, G. C. H. E. (2021). Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments.

Eckert, H. E. and Hamdorf, K. (1980). Excitatory and inhibitory response components in the landing response of the blowfly,calliphora erythrocephala. *Journal of comparative physiology*, 138:253–264.

Egelhaaf, M., Boeddeker, N., Kern, R., Kurtz, R., and Lindemann, J. (2012). Spatial vision in insects is facilitated by shaping the dynamics of visual input through behavioral action. front. *Frontiers in neural circuits*, 6:108.

Eggert, J. and van Hemmen, L. (2001). Modeling neuronal assemblies: Theory and implementation. *Neural computation*, 13:1923–74.

Eichner, H., Joesch, M., Schnell, B., Reiff, D., and Borst, A. (2011). Internal structure of the fly elementary motion detector. *Neuron*, 70:1155–64.

Eliasmith, C. and Anderson, C. H. (2004). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Esser, S. K., Appuswamy, R., Merolla, P. A., Arthur, J. V., and Modha, D. S. (2015). Backpropagation for energy-efficient neuromorphic computing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1117–1125, Cambridge, MA, USA. MIT Press.

Faisal, M. and Barron, J. (2007). High accuracy optical flow method based on a theory for warping: Implementation and qualitative/quantitative evaluation. In *Proceedings of the 4th International Conference on Image Analysis and Recognition*, ICIAR'07, page 513–525, Berlin, Heidelberg. Springer-Verlag.

Farabet, C., Paz-Vicente, R., Pérez-Carrasco, J., Zamarreño-Ramos, C., Linares-Barranco, A., Lecun, Y., Culurciello, E., Serrano-Gotarredona, T., and Linares-Barranco, B. (2012). Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing. *Frontiers in Neuroscience*, 6:1–12.

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1 6:445–66.

Frye, M. W. (2015). Elementary motion detectors. *Current Biology*, 25:R215–R217.

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467.

Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., et al. (2019). Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*.

Gautama, T. and Van Hulle, M. (2002). A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 13(5):1127–1136.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361.

Gerstner, W., Kempter, R., van Hemmen, L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76–81.

Gerstner, W. and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Houghton Mifflin.

Giulioni, M., Lagorce, X., Galluppi, F., and Benosman, R. (2016). Event-based computation of motion flow on a neuromorphic analog neural platform. *Frontiers in Neuroscience*, 10.

Graham, P. and Philippides, A. (2014). *Insect-Inspired Visual Systems and Visually Guided Behavior*, pages 1–9. Springer Netherlands, Dordrecht.

Haessig, G., Cassidy, A. S., Alvarez, R., Benosman, R. B., and Orchard, G. (2018). Spiking optical flow for event-based sensors using ibm's truenorth neurosynaptic system. *IEEE Transactions on Biomedical Circuits and Systems*, 12:860–870.

He, L., Aouf, N., Whidborne, J. F., and Song, B. (2020). Integrated moment-based lgmd and deep reinforcement learning for uav obstacle avoidance. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7491–7497.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. Wiley, New York.

Heeger, D. (1988). Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1(4):279–302.

Hodgkin, A. and Huxley, A. (1990). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52(1):25 – 71.

Holmqvist, M. H. and Srinivasan, M. V. (2004). A visually evoked escape response of the housefly. *Journal of Comparative Physiology A*, 169:451–459.

Hordijk, B., Scheper, K., and Croon, G. (2018). Vertical landing for micro air vehicles using event-based optical flow. *Journal of Field Robotics*, 35:69–90.

Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203.

Hu, C., Arvin, F., Xiong, C., and Yue, S. (2017). Bio-inspired embedded vision system for autonomous micro-robots: The lgmd case. *IEEE Transactions on Cognitive and Developmental Systems*, 9(3):241–254.

Hu, Y., Song, R., and Li, Y. (2016). Efficient coarse-to-fine patch match for large displacement optical flow. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5704–5712.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Iakymchuk, T., Rosado-Muñoz, A., Guerrero-Martínez, J., Bataller-Mompeán, M., and Francés-Víllora, J. (2015). Simplified spiking neural network architecture and stdp learning algorithm applied to image classification. *Eurasip Journal on Image and Video Processing*, 2015(1).

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Indiveri, G., Chicca, E., and Douglas, R. (2006). A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Transactions on Neural Networks*, 17(1):211–221.

Indiveri, G., Corradi, F., and Qiao, N. (2015). Neuromorphic architectures for spiking deep neural networks. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 4.2.1–4.2.4.

Indiveri, G., Linares-Barranco, B., Hamilton, T., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saïghi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., and Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73.

Izhikevich, E. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572.

Izhikevich, E. M. and Moehlis, J. (2008). Dynamical systems in neuroscience: The geometry of excitability and bursting. *SIAM review*, 50(2):397.

James, A. C. and Osorio, D. (1996). Characterisation of columnar neurons and visual signal processing in the medulla of the locust optic lobe by system identification techniques. *Journal of Comparative Physiology A*, 178:183–199.

Jiménez-Fernández, A., Cerezuela-Escudero, E., Miró-Amarante, L., Domínguez-Morales, M. J., de Asís Gómez-Rodríguez, F., Linares-Barranco, A., and Jiménez-Moreno, G. (2017). A binaural neuromorphic auditory sensor for fpga: A spike signal processing approach. *IEEE Transactions on Neural Networks and Learning Systems*, 28(4):804–818.

Joesch, M., Schnell, B., Varija Raghu, S., Reiff, D., and Borst, A. (2010). On and off pathways in drosophila motion vision. *Nature*, 468:300–4.

Kempter, R., Gerstner, W., and van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Physical Review E*, 59:4498–4514.

Kim, M. and Smaragdis, P. (2016). Bitwise neural networks. *ArXiv*, abs/1601.06071.

Kistler, W. M., Gerstner, W., and Hemmen, J. L. v. (1997). Reduction of the hodgkin-huxley equations to a single-variable threshold model. *Neural Computation*, 9(5):1015–1045.

Klaptocz, A., Boutinard-Rouelle, G., Briod, A., Zufferey, J., and Floreano, D. (2010). An indoor flying platform with collision robustness and self-recovery. In *2010 IEEE International Conference on Robotics and Automation*, pages 3349–3354.

Knight, B. W. (1972). Dynamics of Encoding in a Population of Neurons . *Journal of General Physiology*, 59(6):734–766.

Krähenbühl, P. and Koltun, V. (2012). Efficient nonlocal regularization for optical flow. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision – ECCV 2012*, pages 356–369, Berlin, Heidelberg. Springer Berlin Heidelberg.

Krahl, K. and Poteser, M. (1997). Motion parallax as a source of distance information in locusts and mantids. *Journal of insect behavior.*, 10(1):145—163.

Lazar, A., Pipa, G., and Triesch, J. (2007). Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks*, 20(3):312–322.

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508.

Li, C., Brandli, C., Berner, R., Liu, H., Yang, M., Liu, S.-C., and Delbruck, T. (2015). Design of an rgbw color vga rolling and global shutter dynamic and active-pixel vision sensor. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 718–721.

Li, C. and Li, Y. (2013). A spike-based model of neuronal intrinsic plasticity. *IEEE Transactions on Autonomous Mental Development*, 5(1):62–73.

Li, X., Wang, W., Xue, F., and Song, Y. (2018). Computational modeling of spiking neural network with learning rules from stdp and intrinsic plasticity. *Physica A: Statistical Mechanics and its Applications*, 491:716–728.

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576.

Longuet-Higgins, H. C. and Prazdny, K. (1980). The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 208:385 – 397.

Low, T. and Wyeth, G. (2007). Learning to avoid indoor obstacles from optical flow. In Dunbabin, M. and Srinivasan, M., editors, *Proceedings of the 2007 Australasian Conference on Robotics and Automation:*, pages 1–10. Australian Robotics and Automation Association, Australia.

Low, T. and Wyeth, G. (2012). Obstacle detection using optical flow. *Proceedings of the 2005 Australasian Conference on Robotics and Automation, ACRA 2005*.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, page 674–679, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Madadi Asl, M., Valizadeh, A., and Tass, P. (2018). *Propagation Delays Determine the Effects of Synaptic Plasticity on the Structure and Dynamics of Neuronal Networks*. PhD thesis.

Mancini, M., Costante, G., Valigi, P., and Ciarfuglia, T. A. (2016). Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4296–4303.

Marcireau, A., Ieng, S.-H., Simon-Chane, C., and Benosman, R. B. (2018). Event-based color segmentation with a high dynamic range sensor. *Frontiers in Neuroscience*, 12.

Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636.

Menze, M., Heipke, C., and Geiger, A. (2015). Discrete optimization for optical flow. In *German Conference on Pattern Recognition (GCPR)*, volume 9358, pages 16–28. Springer International Publishing.

Meyer, H., Bertrand, O., Paskarbeit, J., Lindemann, J., Schneider, A., and Egelhaaf, M. (2016). A bio-inspired model for visual collision avoidance on a hexapod walking robot. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9793:167–178.

Milde, M. B., Bertrand, O. J. N., Benosmanz, R., Egelhaaf, M., and Chicca, E. (2015). Bioinspired event-driven collision avoidance algorithm based on optic flow. In *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–7.

Milde, M. B., Bertrand, O. J. N., Ramachandran, H., Egelhaaf, M., and Chicca, E. (2018). Spiking Elementary Motion Detector in Neuromorphic Systems. *Neural Computation*, 30(9):2384–2417.

Milde, M. B., Dietmüller, A., Blum, H., Indiveri, G., and Sandamirskaya, Y. (2017). Obstacle avoidance and target acquisition in mobile robots equipped with neuromorphic sensory-processing systems. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4.

Mishra, M. and Knust, E. (2013). Analysis of the drosophila compound eye with light and electron microscopy. *Methods in molecular biology (Clifton, N.J.)*, 935:161–182.

Moeys, D. P., Li, C., Martel, J. N., Bamford, S., Longinotti, L., Motsnyi, V., San Segundo Bello, D., and Delbruck, T. (2017). Color temporal contrast sensitivity in dynamic vision sensors. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4.

Moradi, S., Qiao, N., Stefanini, F., and Indiveri, G. (2018). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Transactions on Biomedical Circuits and Systems*, 12(1):106–122.

Mori, T. and Scherer, S. (2013). First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *2013 IEEE International Conference on Robotics and Automation*, pages 1750–1757.

Morris, C. E. and Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophysical journal*, 35 1:193–213.

Morrison, A., Diesmann, M., and Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological cybernetics*, 98:459–78.

Mostafa, H. (2018). Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):3227–3235.

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S., and Masquelier, T. (2019). Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, 94:87–95.

Neckar, A., Fok, S., Benjamin, B. V., Stewart, T. C., Oza, N. N., Voelker, A. R., Eliasmith, C., Manohar, R., and Boahen, K. (2019). Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proceedings of the IEEE*, 107(1):144–164.

Nériec, N. and Desplan, C. (2016). Chapter fourteen - from the eye to the brain: Development of the drosophila visual system. In Wassarman, P. M., editor, *Essays on Developmental Biology, Part A*, volume 116 of *Current Topics in Developmental Biology*, pages 247 – 271. Academic Press.

Olberg, R. M. (1981). Object- and self-movement detectors in the ventral nerve cord of the dragonfly. *Journal of comparative physiology*, 141:9.

Orchard, G., Benosman, R., Etienne-Cummings, R., and Thakor, N. V. (2013). A spiking neural network architecture for visual motion estimation. In *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 298–301.

Oron, S., Bar-Hille, A., and Avidan, S. (2014). Extended lucas-kanade tracking. In *Computer Vision – ECCV 2014*, pages 142–156. Springer International Publishing.

Osorio, D. (1991). Mechanisms of early visual processing in the medulla of the locust optic lobe: How self-inhibition, spatial-pooling, and signal rectification contribute to the properties of transient cells. *Visual Neuroscience*, 7(4):345–355.

Panda, P., Allred, J. M., Ramanathan, S., and Roy, K. (2018). Asp: Learning to forget with adaptive synaptic plasticity in spiking neural networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8:51–64.

Paredes-Vallés, F., Hagenaars, J. J., and de Croon, G. C. (2021). Self-supervised learning of event-based optical flow with spiking neural networks. In *NeurIPS*.

Paredes-Valles, F., Scheper, K. Y. W., and de Croon, G. C. H. E. (2020). 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*, 42(8):2051–2064.

Pfeiffer, M. and Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774.

Ponce, H., Brieva, J., and Moya-Albor, E. (2018). Distance estimation using a bio-inspired optical flow strategy applied to neuro-robotics. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7.

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

Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., and Delbruck, T. (2014). Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484.

Putra, R. V. W. and Shafique, M. (2021). Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments. *CoRR*, abs/2103.00424.

Pérez-Carrasco, J., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., and Linares-Barranco, B. (2013). Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate-coding and coincidence processing. application to feed forward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:2706 – 2719.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham. Springer International Publishing.

Reichardt, W. and Rosenblith, W. A. (1961). Autocorrelation, a principle for evaluation of sensory information by the central nervous system.

Richter, C., Röhrbein, F., and Conradt, J. (2014). Bio-inspired optic flow detection using neuromorphic hardware. *Bernstein Conference 2014*.

Rind, F. (1984). A chemical synapse between two motion detecting neurons in the locust brain. *The Journal of experimental biology*, 110:143–67.

Rind, F. (2002). Motion detectors in the locust visual system: From biology to robot sensors. *Microscopy research and technique*, 56:256–69.

Rind, F. and Bramwell, D. (1996). Neural network based on the input organization of an identified neuron signaling impending collision. *Journal of neurophysiology*, 75:967–85.

Rohmer, E., Singh, S. P. N., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326.

Rueckauer, B. and Delbruck, T. (2016). Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in Neuroscience*, 10:176.

Salt, L., Howard, D., Indiveri, G., and Sandamirskaya, Y. (2020). Parameter optimization and learning in a spiking neural network for uav obstacle avoidance targeting neuromorphic processors. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3305–3318.

Salt, L., Indiveri, G., and Sandamirskaya, Y. (2017). Obstacle avoidance with lgmd neuron: Towards a neuromorphic uav implementation. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4.

Sarikaya, M. and Ogmen, H. (1994). Nonlinear and adaptive properties of visual information processing in the fly nervous system. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 296–301 vol.1.

Schoepe, T., Gutierrez-Galan, D., Dominguez-Morales, J. P., Jimenez-Fernandez, A., Linares-Barranco, A., and Chicca, E. (2019). Neuromorphic sensory integration for combining sound source localization and collision avoidance. In *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4.

Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., and Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*.

Schwegmann, A., Lindemann, J. P., and Egelhaaf, M. (2014). Depth information in natural environments derived from optic flow by insect motion detection system: a model analysis. *Frontiers in Computational Neuroscience*, 8:83.

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2018). Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13.

Serres, J. R. and Ruffier, F. (2017). Optic flow-based collision-free strategies: From insects to robots. *Arthropod Structure Development*, 46(5):703 – 717. From Insects to Robots.

Shafiee, M., Zhou, Z., Mei, L., Dinmohammadi, F., Karama, J., and Flynn, D. (2021). Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis. *Robotics*, 10:26.

Song, Z., Coca, D., Billings, S., Postma, M., Hardie, R. C., and Juusola, M. (2009). Biophysical modeling of a drosophila photoreceptor. In Leung, C. S., Lee, M., and Chan, J. H., editors, *Neural Information Processing*, pages 57–71, Berlin, Heidelberg. Springer Berlin Heidelberg.

Soudry, D., Hubara, I., and Meir, R. (2014). Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in Neural Information Processing Systems*, volume 2.

Souhila, K. and Achour, K. (2007). Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4.

Srinivasan, M. V. (2011). Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics. *Physiological Reviews*, 91(2):413–460. PMID: 21527730.

Stein, R. B. (1965). A theoretical analysis of neuronal variability. *Biophysical journal*, 5:173–94.

Stemmler, M. and Koch, C. (1999). How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. *Nature neuroscience*, 2(6):521–527.

Stewart, T., Kleinhans, A., Mundy, A., and Conradt, J. (2016). Serendipitous offline learning in a neuromorphic robot. *Frontiers in Neurorobotics*, 10.

Tammero, L. and Dickinson, M. (2002). Collision-avoidance and landing responses are mediated by separate pathways in the fruit fly, drosophila melanogaster. *The Journal of experimental biology*, 205:2785–98.

Triesch, J. (2005). A gradient rule for the plasticity of a neuron's intrinsic excitability. In *Proceedings of the International Conference on Artificial Neural Networks, 2005*, pages 65–70. Springer.

Trimarchi, J. and Schneiderman, A. (2009). Initiation of flight in the unrestrained fly, drosophila melanogaster. *Journal of Zoology*, 235:211 – 222.

Tschechne, S., Sailer, R., and Neumann, H. (2014). Bio-inspired optic flow from event-based neuromorphic sensor input. In *ANNPR*.

Tu, Z., Xie, W., Zhang, D., Poppe, R., Veltkamp, R. C., Li, B., and Yuan, J. (2019). A survey of variational and cnn-based optical flow techniques. *Signal Processing: Image Communication*, 72:9–24.

Tuthill, J., Nern, A., Holtz, S., Rubin, G., and Reiser, M. (2013). Contributions of the 12 neuron classes in the fly lamina to motion vision. *Neuron*, 79:128–140.

Wagner, H. (1982). Flow-field variables trigger landing in flies. *Nature*, 297:147–148.

Weber, J. and Malik, J. (1992). Robust computation of optical flow in a multi-scale differential framework. Technical Report UCB/CSD-92-709, EECS Department, University of California, Berkeley.

Wei, Z., Li, B., Guo, W., Hu, W., and Zhao, C. (2019). Sequential bayesian detection of spike activities from fluorescence observations. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 5(1):3–18.

Wicklein, M. and Strausfeld, N. (2000). Organization and significance of neurons that detect change of visual depth in the hawk moth manduca sexta. *Journal of Comparative Neurology*, 424:356–376.

Wills, J. and Belongie, S. J. (2004). A feature-based approach for determining dense long range correspondences. In *ECCV*.

Wittekind, W. C. (1998). Short Communication: The Landing Response of Tethered Flying Drosophila is Induced at a Critical Object Angle. *Journal of Experimental Biology*, 135(1):491–493.

Wolff, T. and Ready, D. (1993). Pattern formation in the drosophila retina. *The Development of Drosophila melanogaster*, 2:1277–1325.

Xiang, X., Zhai, M., Zhang, R., Qiao, Y., and El Saddik, A. (2018). Deep optical flow supervised learning with prior assumptions. *IEEE Access*, 6:43222–43232.

Xiao, F., Zheng, P., Tria, J. d., Kocer, B. B., and Kovac, M. (2021). Optic flow-based reactive collision prevention for mavs using the fictitious obstacle hypothesis. *IEEE Robotics and Automation Letters*, 6(2):3144–3151.

Yue, S. and Rind, F. (2009). Near range path navigation using lgmd visual neural networks. *Proceedings - 2009 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009*.

Zhai, M., Xiang, X., Lv, N., and Kong, X. (2021). Optical flow and scene flow estimation: A survey. *Pattern Recognition*, 114:107861.

Zhang, A., Gao, Y., Niu, Y., Li, X., and Chen, Q. (2020). Intrinsic plasticity for online unsupervised learning based on soft-reset spiking neuron model. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1.

Zhang, W. and Li, P. (2019). Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks. *Frontiers in Neuroscience*, 13(FEB).

Zhao, S., Li, X., and Bourahla, O. E. F. (2017). Deep optical flow estimation via multi-scale correspondence structure learning. *CoRR*, abs/1707.07301.

Zhu, A., Yuan, L., Chaney, K., and Daniilidis, K. (2018). Ev-flownet: Self-supervised optical flow estimation for event-based cameras.

Zhu, Y. (2013). The drosophila visual system. *Cell Adhesion & Migration*, 7(4):333–344. PMID: 23880926.

Zingg, S., Scaramuzza, D., Weiss, S., and Siegwart, R. (2010). Mav navigation through indoor corridors using optical flow. In *2010 IEEE International Conference on Robotics and Automation*, pages 3361–3368.

Zufferey, J. . and Floreano, D. (2005). Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2594–2599.