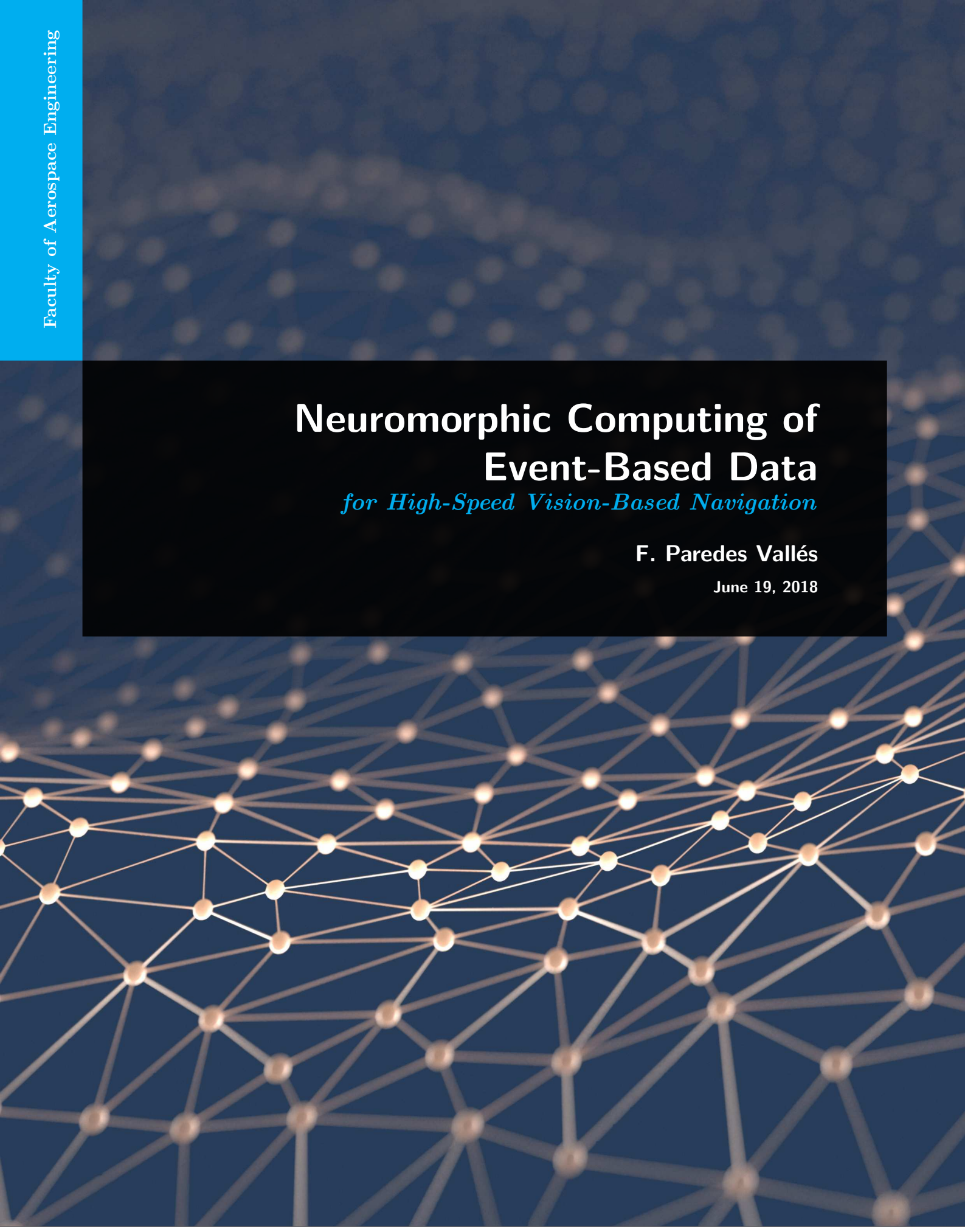
Faculty of Aerospace Engineering

Neuromorphic Computing of Event-Based Data

for High-Speed Vision-Based Navigation

F. Paredes Vallés

June 19, 2018



TU Delft Delft University of Technology

Challenge the future

Neuromorphic Computing of Event-Based Data

for High-Speed Vision-Based Navigation

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

F. Paredes Vallés

June 19, 2018

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Neuromorphic Computing of Event-Based Data"** by **F. Paredes Vallés** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: June 19, 2018

Readers:

Dr. G.C.H.E. de Croon

Dr. ir. E. van Kampen

Prof. Dr. S. M. Bohte

ir. K.Y.W. Scheper

Acknowledgments

First and foremost, I would like to thank my supervisors Guido de Croon and Kirk Scheper for their valuable guidance and support throughout the journey that has been this thesis. Working with you has been a very enriching experience, and I will be glad to continue doing so in the upcoming years.

A word of appreciation also goes out to my friends—particularly Marta Camarero and Suresh Sharma—who have been a fundamental part of my life in the past years.

Lastly, and most importantly, I would like to thank my family for their emotional support. With the reader's permission, the following words of thanks are in Spanish.

Papá, Mamá, Jesús, Clara, gracias por vuestra paciencia y apoyo incondicional a lo largo de todos estos años. Este trabajo no hubiera sido posible sin vosotros.

F. Paredes Vallés
Delft, June 19, 2018

Abstract

The combination of Spiking Neural Networks and event-based vision sensors holds the potential of highly efficient and high-bandwidth optical flow estimation. This thesis presents, to the best of the author's knowledge, the first hierarchical spiking architecture in which motion (direction and speed) selectivity emerges in a biologically plausible unsupervised fashion from the stimuli generated with an event-based camera. A novel adaptive neuron model and Spike-Timing-Dependent Plasticity formulation are at the core of this neural network governing its spike-based processing and learning, respectively. After convergence, the neural architecture exhibits the main properties of biological visual motion systems: feature extraction and local and global motion perception. To assess the outcome of the learning, a shallow conventional Artificial Neural Network is trained to map the activation traces of the penultimate layer to the optical flow visual observables of ventral flows. The proposed solution is validated for simulated event sequences with ground truth measurements. Experimental results show that accurate estimates of these parameters can be obtained over a wide range of speeds.

Besides the research on Spiking Neural Networks, this thesis includes an in-depth review of the relevant literate on the main topics of this work, and an extensive preliminary evaluation of conventional deep Artificial Neural Networks in the estimation of optical flow visual observables from event-based data.

Acronyms

Adam	Adaptive Moment Estimation
AER	Address-Event Representation
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ATIS	Asynchronous Time-based Image Sensor
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DAVIS	Dynamic and Active-pixel Vision Sensor
DCSB	Different-Channel-Same-Brightness
DL	Deep Learning
DNN	Deep Neural Network
DoF	Degrees-of-Freedom
DoG	Difference of Gaussians
DVS	Dynamic Vision Sensor
eDVS	Embedded Dynamic Vision Sensor
EKF	Extended Kalman Filter
EMD	Elementary Motion Detector
EVO	Event-based Visual Odometry
FoC	Focus of Contraction
FoE	Focus of Expansion
FPGA	Field-Programmable Gate Array
FPS	Frames Per Second
GPS	Global Positioning System
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit

jAER	Java Address-Event Representation
LIF	Leaky Integrate-and-Fire
LSTM	Long Short-Term Memory
MAV	Micro Air Vehicle
MAVLab	Micro Air Vehicle Laboratory
MDN	Mixture Density Network
meDVS	Miniature Embedded Dynamic Vision Sensor
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PTAM	Parallel Tracking And Mapping
RCNN	Recurrent Convolutional Neural Network
ReLU	Rectified Linear Unit
ReSuMe	Remote Supervised Method
RNN	Recurrent Neural Network
ROS	Robot Operating System
SCDB	Same-Channel-Different-Brightness
SCSB	Same-Channel-Same-Brightness
SfM	Structure-from-Motion
SGD	Stochastic Gradient Descent
SHL	Supervised Hebbian Learning
SLAM	Simultaneous Localization And Mapping
SNN	Spiking Neural Network
SRM	Spike Response Model
SSD	Sum of Squared Difference
STDP	Spike-Timing-Dependent Plasticity
SVO	Semi-direct Visual Odometry
V-SLAM	Visual Simultaneous Localization And Mapping
VLSI	Very-Large-Scale Integration
VTOL	Vertical Take-Off and Landing
WTA	Winner-Take-All

List of Symbols

Greek Symbols

α	Homeostasic scaling factor
β	Impact of inhibitory synaptic connections on neural dynamics
Δt_{ref}	Length of the neural refractory period
Δt_{sim}	Simulation time step
$\Delta \boldsymbol{W}$	Synaptic weight update matrix
$\varepsilon_{\omega_x}, \varepsilon_{\omega_y}$	Absolute error of ventral flow components
η	STDP learning rate
$\vartheta_x, \vartheta_y, \vartheta_z$	Normalized translational velocities of the vision sensor
$\lambda_v, \lambda_X, \lambda_y$	Time constants of the membrane potential and synaptic traces
τ	Synaptic transmission delay
$\boldsymbol{\tau}$	Synaptic transmission delay vector
ϕ, θ, ψ	Euler angles representing sensor's attitude
ω_x, ω_y	Ventral flow components in the X- and Y-axis of the image plane

Roman Symbols

a	Scalar controlling the stability of the STDP rule
C	Event-triggering threshold of the vision sensor
f	Number of feature maps in a neural layer
$\boldsymbol{i}$	Forcing functions of the connections between two neural layers
I	Brightness level perceived by a pixel
$\mathcal{L}$	Mean square error criterion to track convergence of STDP
m	Number of multisynaptic connections between two neurons

n	Number of neurons in a feature map
N	Map-specific direct neural neighborhood of a neuron, including itself
p, q, r	Sensor's angular rates corresponding to ϕ, θ, ψ
s	Neural spike train
t	Time
u, v	Optical flow components
u_T, v_T	Translational optical flow components
U, V, W	Sensor's translational velocities corresponding to X, Y, Z
v	Neural membrane potential
v_{reset}	Reset membrane potential
v_{rest}	Resting membrane potential
v_{th}	Firing threshold
w_{init}	Initialization synaptic weight
$\boldsymbol{W}$	Weight matrix of the synaptic connections between two neural layers
x, y	Pixel coordinates on a pixel array
X, Y, Z	Metric position in Cartesian coordinates
$\boldsymbol{X}$	Synaptic trace matrix of the connections between two neural layers
Z_0	Distance to the surface along the optical axis of the sensor
Z_X, Z_Y	Slopes of a planar surface with respect to the image plane

Subscripts

b	Presynaptic neuron in N
$\mathcal{C}$	Camera reference frame
ch	Presynaptic feature map
d	Synaptic delay
i	Postsynaptic neuron
j	Presynaptic neuron
k	Postsynaptic feature map
$\mathcal{W}$	World reference frame

Superscripts

l	Neural layer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Whenever an animal endowed with a visual system navigates through an environment, turns its gaze, or simply observes a moving object from a resting state, motion patterns are perceivable at the retina level as spatiotemporal variations of brightness (Feng, 2003; Borst et al., 2010; Borst & Helmstaedter, 2015). These patterns of apparent motion, formally referred to as *optical flow* (Gibson, 1950), are a crucial source of information for these animals to estimate their ego-motion and to have a better understanding of the visual scene. For instance, expanding flow fields are normally induced by forward motion, and nearby obstacles can be discerned as regions of high flow surrounded by patterns of much lower motion. In essence, the information that can be derived from optical flow is the ratio of velocity to distance (Longuet-Higgins & Prazdny, 1980), which can be represented as a set of visual observables (de Croon et al., 2013), such as ventral flows or divergence. Flying insects are seen to rely on these parameters to perform high-speed maneuvers such as horizontal translation or landing (Srinivasan et al., 1996; Chahl et al., 2004; Baird et al., 2013).

Considering their size and weight limitations, insects are a clear indicator of the efficiency, robustness, and low latency of the optical flow estimation conducted by biological systems. The ability to reliably mimic this procedure would have a significant impact on the field of micro-robotics due to the limited computational capacity of onboard processors. As an example, Micro Air Vehicles (MAVs), such as the 20-gram DelFly Explorer (De Wagter et al., 2014), could benefit from a bio-realistic visual motion estimation for high-speed autonomous navigation in cluttered environments. Accordingly, the design of optical flow techniques is currently an appealing research topic, and numerous solutions have been proposed to date at various levels of abstraction (Fortun et al., 2015). Nevertheless, there are still remarkable differences with their biological counterparts regarding data acquisition and processing.

In biological visual systems, the process of seeing starts at the photoreceptors in the retina, which are light-sensitive neurons that absorb and convert incoming light into electrical signals. These signals serve as input to the so-called ganglion cells. There are two major types of these neurons: ON cells, which react to an increase in the brightness perceived at a specific location in the retina; and OFF cells, which do so to a decrease (Posch et al., 2014). The activity of these neurons consists in temporal sequences of discrete *spikes* (voltage pulses) that are sent

to large networks of interconnected cells for motion estimation, among other tasks. Since it is spike-driven, these biological architectures are characterized by a sparse, asynchronous, and massively parallelized computation. Further, they are seen to adapt their topology, i.e. connectivity pattern, in response to visual experience (Wiesel, 1982; Kirkwood & Bear, 1994). This adaptation, or *learning* mechanism, allows them to operate robustly in different environments under a wide range of lighting conditions.

In contrast, the working principle of the majority of input sensors employed for artificial visual perception is categorized as *frame-based*: static images are obtained by measuring the brightness levels of a pixel array at fixed time intervals. Although convenient for some computer vision applications, these sensors are highly inefficient for the task of motion estimation since the output rate at which frames are acquired is independent of the dynamics of the visual scene. In addition, due to the limited temporal resolution of these sensors, rapidly moving objects may introduce motion blur, thus limiting the accuracy of optical flow measurements.

However, not all artificial systems rely on conventional frame-based cameras for visual motion estimation. Inspired by biological retinas, several *event-based* sensors have recently been presented (Lichtsteiner et al., 2008; Posch et al., 2011; Brandli et al., 2014) in an attempt of pushing the state-of-the-art of artificial vision systems towards more efficient processing. Similarly to ganglion cells, each of the elements of the pixel array reacts asynchronously to brightness changes in its corresponding receptive field (Posch et al., 2014). Whenever the change exceeds a predefined threshold, the sensor registers an *event* that contains information about the sign of the change, the location in the array where it was detected, and a timestamp. A microsecond resolution, latencies in this order of magnitude, a very high dynamic range, and a low power consumption make these sensors an ideal choice for visual perception (Conradt et al., 2009; Delbruck & Lang, 2013; Mueggler et al., 2014; Hordijk et al., 2018).

Regardless of the vision sensor, the estimation of optical flow by artificial systems is normally performed algorithmically, with solutions that are built on simplifying assumptions that make this problem tractable (Horn & Schunck, 1981; Sutton et al., 1983; Adelson & Bergen, 1985; Heeger, 1987; Benosman et al., 2014; Brosch & Neumann, 2014). In spite of this, the recent progress in parallel computing hardware has enabled artificial visual perception to be addressed from a more bio-inspired perspective: Artificial Neural Networks (ANNs). Similarly to biological architectures, ANNs consist of large sets of artificial neurons whose interconnections can be optimized for the task at hand. However, despite the high accuracy reported (Dosovitskiy et al., 2015; Ilg et al., 2017; Ranjan & Black, 2017; Ren et al., 2017; Zhao et al., 2017; Lai et al., 2017), there is still a fundamental difference: the underlying data transfer mechanism of ANNs is based on a continuous stream of information (Goodfellow et al., 2016) rather than on trains of discrete spikes. As a consequence, these architectures are in general computationally very expensive.

Taking further inspiration from nature, Spiking Neural Networks (SNNs) have been proposed as a new generation of ANNs (Maass, 1997). As the name suggests, the computation carried out by these bio-realistic neural models is asynchronous and spike-based, which makes them a suitable processing framework for the sparse data generated by event-based vision sensors (Orchard & Etienne-Cummings, 2014). Moreover, SNNs can benefit from an efficient real-time implementation in *neuromorphic hardware*, such as IBM's TrueNorth chip (Merolla et al., 2014) or Intel's Loihi processor (Davies et al., 2018). Despite these promising characteristics, the spiking nature of these networks limits the application of the successful gradient-based

optimization algorithms normally employed in ANNs. Instead, learning in SNNs is dominated by Spike-Timing-Dependent Plasticity (STDP) (Bi & Poo, 1998; Markram et al., 1997), a biologically plausible protocol that adapts the strength of a connection between two neurons based on their correlated activity. STDP has been successfully applied to simple image classification tasks (Masquelier & Thorpe, 2007; Iakymchuk et al., 2015; Diehl & Cook, 2015; Tavanaei & Maida, 2017). However, until now, no study has discussed the use of this learning rule for the estimation of event-based optical flow.

1-1 Motivation and research question

Among other projects, the Micro Air Vehicle Laboratory (MAVLab) of Delft University of Technology conducts research on the application of optical flow to the development of vision-based navigation solutions for MAVs. For instance, experiments have recently been performed for the characterization of the constant-divergence landing maneuver using rotorcraft MAVs equipped with downward-looking cameras (e.g., de Croon et al., 2013; Ho & de Croon, 2016; de Croon, 2016). Moreover, McGuire et al. (2017) presented an efficient optical flow estimation method that, using a lightweight forward-looking stereo camera, is suitable for performing autonomous navigation on-board very small MAVs.

The combination of event-based cameras and SNNs is of great interest in this research field. Firstly, due to the high temporal resolution of these sensors and the efficiency of SNNs, this combination has the potential to significantly increase the update rate at which optical flow visual observables are obtained, thus enabling high-speed vision-based navigation. Secondly, regarding the inspiration from biology (Chahl et al., 2004; Baird et al., 2013), learning to estimate these parameters adds a new level of realism to the algorithm. Instead of relying on mathematical models describing the relation between optical flow and ego-motion states (e.g., Longuet-Higgins & Prazdny, 1980), these neural architectures are potentially capable of learning this input-output mapping from data samples. Further, as previously mentioned, they can be implemented in neuromorphic hardware (Merolla et al., 2014; Davies et al., 2018) due to their event-based communication. These devices are characterized by a real-time performance that, in conjunction with a low power consumption, makes them a very suitable computational platform for MAVs.

According to this motivation, the main research question of this thesis is formulated as follows:

Can a spiking neural network learn the main functionalities of biological visual motion systems, namely feature extraction and local and global motion perception, and hence be used for ego-motion estimation?

1-2 Structure of this work

The main contributions of this thesis are presented in the scientific paper in Part I. This paper can be read as a standalone document and consists primarily of, first, a concise introduction to the main concepts and previous contributions of relevance to the work; second, a description and evaluation of the proposed spiking architecture, neuron model, and learning rule;

and third, concluding remarks and recommendations for future research on the topic. The remainder of this thesis provides supporting material for this paper. Readers unfamiliar with the topics of optical flow, event-based vision, and neuromorphic computing are encouraged to read this documentation beforehand.

Part II presents an in-depth review of relevant literature on the topics of optical flow estimation, its use for vision-based navigation, and neuromorphic computing. In Chapter 2, the concept of optical flow is introduced, together with its most-common frame-based estimation algorithms and applications in MAVs with navigation purposes. Next, Chapter 3 introduces the field of event-based vision. Here, the working principle of these sensors is detailed, and the existing approaches for estimating event-based optical flow are covered. Chapter 4 reviews the field of neuromorphic computing by describing the main aspects of spiking and artificial neural networks, together with their corresponding applications of optical flow and motion estimation. Lastly, this literature review is synthesized in Chapter 5.

Next, Part III documents a preliminary evaluation of conventional deep neural architectures in the task of estimating optical flow visual observables from event-based data. This analysis serves as a good practical introduction to the field of Deep Learning (DL), and to several relevant aspects that have been taken into account for the development of the final spiking architecture proposed in this thesis (e.g., network topology, polarity encoding). In Chapter 6, the methodology and datasets used throughout the analysis are introduced. Next, Chapter 7 presents the neural networks employed, their training details, and an evaluation of their performance when estimating ventral flows. These preliminary results are further discussed in Chapter 8.

Finally, Part IV contains a set of individual appendices with supplementary material that support the scientific paper included in Part I. Appendix A provides an overview of the *cuda-snn* simulator: the computational framework that has been developed throughout this thesis for the efficient simulation of the communication and learning mechanisms of SNNs. On the other hand, Appendix B provides recommendations for future research on the main topics of this work.

Part I

Scientific Paper

Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception

F. Paredes Vallés[*‡], K.Y.W. Scheper[†‡], G.C.H.E. de Croon[†‡]
[*]MSc student, [†]Supervisor
[‡]Control & Simulation, Control & Operations Department
Delft University of Technology, Delft, The Netherlands

***Abstract*—The combination of Spiking Neural Networks and event-based vision sensors holds the potential of highly efficient and high-bandwidth optical flow estimation. This paper presents the first hierarchical spiking architecture in which motion (direction and speed) selectivity emerges in a biologically plausible unsupervised fashion from the stimuli generated with an event-based camera. A novel adaptive neuron model and Spike-Timing-Dependent Plasticity formulation are at the core of this neural network governing its spike-based processing and learning, respectively. After convergence, the neural architecture exhibits the main properties of biological visual motion systems, namely feature extraction and local and global motion perception. To assess the outcome of the learning, a shallow conventional Artificial Neural Network is trained to map the activation traces of the penultimate layer to the optical flow visual observables of ventral flows. The proposed solution is validated for simulated event sequences with ground truth measurements. Experimental results show that accurate estimates of these parameters can be obtained over a wide range of speeds**

***Index Terms*—Event-based vision, neuromorphic, optical flow, spiking neural networks, spike-timing-dependent plasticity.**

I. Introduction

WHENEVER an animal endowed with a visual system navigates through an environment, turns its gaze, or simply observes a moving object from a resting state, motion patterns are perceivable at the retina level as spatiotemporal variations of brightness [1]–[3]. These patterns of apparent motion, formally referred to as *optical flow* [4], are a crucial source of information for these animals to estimate their ego-motion and to have a better understanding of the visual scene. For instance, expanding flow fields are normally induced by forward motion, and nearby obstacles can be discerned as regions of high flow surrounded by patterns of much lower motion. In essence, the information that can be derived from optical flow is the ratio of velocity to distance [5], which can be represented as a set of *visual observables* [6], such as ventral flows or divergence. Flying insects are seen to rely on these parameters to perform high-speed maneuvers such as horizontal translation or landing [7]–[9].

Considering their size and weight limitations, insects are a clear indicator of the efficiency, robustness, and low latency of the optical flow estimation conducted by biological systems. The ability to reliably mimic this procedure would have a significant impact on the field of micro-robotics due to the limited computational capacity of onboard processors. As an example, Micro Air Vehicles (MAVs), such as the 20-gram DelFly Explorer [10], could benefit from a bio-realistic visual motion estimation for high-speed autonomous navigation in cluttered environments. Accordingly, the design of optical flow techniques is currently an appealing research topic, and numerous solutions have been proposed to date at various levels of abstraction [11]. Nevertheless, there are still remarkable differences with their biological counterparts regarding data acquisition and processing.

In biological visual systems, the process of seeing starts at the photoreceptors in the retina, which are light-sensitive neurons that absorb and convert incoming light into electrical signals. These signals serve as input to the so-called ganglion cells. There are two major types of these neurons: ON cells, which react to an increase in the brightness perceived at a specific location in the retina; and OFF cells, which do so to a decrease [12]. The activity of these neurons consists in temporal sequences of discrete *spikes* (voltage pulses) that are sent to large networks of interconnected cells for motion estimation, among other tasks. Since it is spike-driven, these biological architectures are characterized by a sparse, asynchronous, and massively parallelized computation. Further, they are seen to adapt their topology, i.e. connectivity pattern, in response to visual experience [13], [14]. This adaptation, or *learning* mechanism, allows them to operate robustly in different environments under a wide range of lighting conditions.

In contrast, the working principle of the majority of input sensors employed for artificial visual perception is categorized as *frame-based*: static images are obtained by measuring the brightness levels of a pixel array at fixed time intervals. Although convenient for some computer vision applications, these sensors are highly inefficient for the task of motion estimation since the output rate at which frames are acquired is independent of the dynamics of the visual scene. In addition, due to the limited temporal resolution of these sensors, rapidly moving objects may introduce motion blur, thus limiting the accuracy of optical flow measurements.

However, not all artificial systems rely on conventional frame-based cameras for visual motion estimation. Inspired by biological retinas, several *event-based* sensors have recently been presented [15]–[17] in an attempt of pushing the state-

of-the-art of artificial vision systems towards more efficient processing. Similarly to ganglion cells, each of the elements of the pixel array reacts asynchronously to brightness changes in its corresponding receptive field [12]. Whenever the change exceeds a predefined threshold, the sensor registers an *event* that contains information about the sign of the change, the location in the array where it was detected, and a timestamp. A microsecond resolution, latencies in this order of magnitude, a very high dynamic range, and a low power consumption make these sensors an ideal choice for visual perception [18]–[21].

Regardless of the vision sensor, the estimation of optical flow by artificial systems is normally performed algorithmically, with solutions that are built on simplifying assumptions that make this problem tractable [22]–[27]. In spite of this, the recent progress in parallel computing hardware has enabled artificial visual perception to be addressed from a more bio-inspired perspective: Artificial Neural Networks (ANNs). Similarly to biological architectures, ANNs consist of large sets of artificial neurons whose interconnections can be optimized for the task at hand. However, despite the high accuracy reported [28]–[33], there is still a fundamental difference: the underlying data transfer mechanism of ANNs is based on a continuous stream of information [34] rather than on trains of asynchronous discrete spikes. As a consequence, these architectures are in general computationally very expensive.

Taking further inspiration from nature, Spiking Neural Networks (SNNs) have been proposed as a new generation of ANNs [35]. As the name suggests, the computation carried out by these bio-realistic neural models is asynchronous and spike-based, which makes them a suitable processing framework for the sparse data generated by event-based vision sensors [36]. Moreover, SNNs can benefit from an efficient real-time implementation in *neuromorphic hardware*, such as IBM's TrueNorth chip [37] or Intel's Loihi processor [38]. Despite these promising characteristics, the spiking nature of these networks limits the application of the successful gradient-based optimization algorithms normally employed in ANNs. Instead, learning in SNNs is dominated by Spike-Timing-Dependent Plasticity (STDP) [39], [40], a biologically plausible protocol that adapts the strength of a connection between two neurons based on their correlated activity. STDP has been successfully applied to simple image classification tasks [41]–[44]. However, until now, no study has discussed the use of this learning rule for the estimation of event-based optical flow.

This paper contains *three main contributions*. First, a novel adaptive mechanism for the Leaky Integrate-and-Fire (LIF) spiking neuron model [45] is introduced. This adaptation extends the applicability of this model to the rapidly varying input statistics of a moving event-based vision sensor. Second, a novel inherently-stable STDP implementation is proposed. With this learning rule, the strength of neural connections is naturally constrained without the need for the ad-hoc mechanisms used by most of the existing formulations. Third, the proposed neuron model and STDP rule are combined in a bio-inspired hierarchical SNN architecture that, after learning, resembles the main functionalities of biological visual systems: feature extraction and local and global motion perception. To the best of the authors' knowledge, this paper shows, for the first time, that neural selectivity to the local and global motion of input stimuli can emerge from visual experience in a biologically plausible unsupervised fashion.

The rest of the paper is structured as follows. Firstly, Section II provides background information concerning event-based vision, SNNs, and optical flow estimation. Then, Section III introduces the mathematical model relating sensor ego-motion to the optical flow observables used in this work. In Section IV, the foundations of the spike-based processing and learning of the proposed SNN are detailed. Afterwards, network architecture and an ANN-based readout mechanism are described and evaluated in Sections V and VI, respectively. Lastly, concluding remarks and recommendations for future work are given in Section VII.

II. Related Work

This section serves as an introduction to the main concepts and previous contributions of relevance to the present work. Firstly, the theory behind event-based vision sensors is described in Section II-A. Secondly, Section II-B covers the field of SNNs, placing emphasis on existing spiking neuron models and unsupervised learning mechanisms. Lastly, an overview of the available optical flow estimation algorithms is provided in Section II-C, for both frame- and event-based sensors.

A. Event-Based Vision Sensors

Inspired by biological retinas, the working principle of event-based vision sensors differs significantly from that of their frame-based counterparts. Conventional frame-based cameras perceive the world as a series of consecutive images, which are acquired by measuring the brightness levels of the pixel array at a fixed rate. In contrast, each of the pixels of an event-based vision sensor reacts asynchronously to local changes in brightness by generating discrete temporal events. Specifically, the generation of an event is triggered whenever the logarithmic change of the image intensity $I(x, y, t)$ exceeds

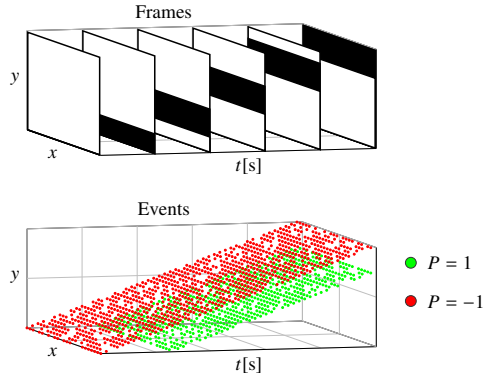


Fig. 1: Comparison of the output of frame- and event-based vision sensors under the stimulus of a black horizontal bar moving upward over a homogeneous white background. While frames are basically two-dimensional snapshots of the visual scene, events are spatiotemporal sparse points tracking the leading and trailing edges of the bar.

a predefined threshold C, as shown in Eq. (1). This variation is computed with respect to a reference brightness level set by the last occurring event at that pixel [15].

$$|\Delta \log(I(x, y, t))| > C \tag{1}$$

Each event encodes information about the timestamp t in which it was generated, the corresponding (x, y) location in the pixel array, and the polarity $P \in \{-1, 1\}$ of the intensity change. This communication protocol is formally referred to as Address-Event Representation (AER) [46]. A visual comparison of the output of frame- and event-based sensors under the same stimulus is illustrated in Fig. 1.

The event-based camera employed in this work is the Dynamic Vision Sensor (DVS), more specifically, the DVS128. This device features a 128×128 pixel array characterized by a temporal resolution of 1 µs, a latency of 12 µs, an intrascene dynamic range of 120 dB, and an average power consumption of 23 mW [15]. Due to the large amount of data required for training neural networks, in this work, the streams of DVS events are generated with the DVS simulator [47], rather than with the actual sensor. This simulator renders intensity images from a three-dimensional virtual scene at a high rate (1000 Hz), and estimates events according to Eq. (1) by linearly interpolating logarithmic brightness levels between frames. The use of virtual scenes facilitates the acquisition of ground truth odometry measurements, which are essential for the performance assessment of motion estimation approaches.

B. Spiking Neural Networks

SNNs offer a novel neuromorphic framework with potential for overcoming the considerable computational effort and energy required by conventional ANNs [35]. In the following, the main aspects and prevalent models of spiking neurons are presented, along with an introduction to the learning paradigms normally employed for optimizing these architectures.

1) Models of spiking neurons: In biological networks, neural communication consists in the exchange of voltage pulses, whose precise timing encodes the information transmitted [35]. For the reproduction of this asynchronous and spike-based mechanism in SNNs, multiple models of spiking neurons have been presented at various levels of abstraction. Biophysical formulations lead to accurate representations of neural dynamics [48]. However, their complexity limits their use in large-scale networks. On the other hand, phenomenological formulations, which offer a compromise between computational load and biological realism, have proven to be effective in numerous applications [41]–[44]. Examples of these models are the aforementioned LIF [45], the Izhikevich [49], and the Spike Response Model [50].

From a conceptual perspective, the majority of these models share some fundamental principles and definitions [35]. The junction of two neurons is called *synapse*; and relative to these cells, the transmitting neuron is labeled as *presynaptic*, while the receiving as *postsynaptic*. Each spiking neuron, as processing unit, is characterized by an internal state variable,

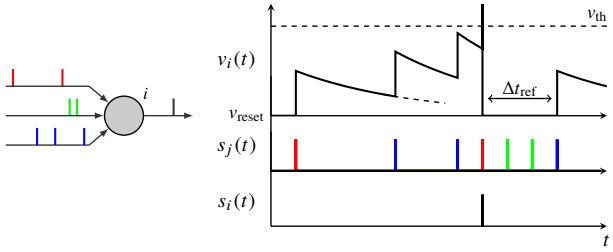


Fig. 2: A model of a LIF neuron. The graphic (right) shows the temporal course of the membrane potential $v_i(t)$ of the ith neuron (left), driven by a sample presynaptic spike train $s_j(t)$ from three input neurons $j = 1, 2, 3$. Spikes are depicted as vertical bars at the time at which they are received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

known as *membrane potential* $v(t)$, which temporally integrates presynaptic spikes over time. If the arrival of a spike leads to an increase (decrease) in $v(t)$, then the spike is said to have an *excitatory* (*inhibitory*) effect on the neuron. $v(t)$ decays to a resting potential v_{rest} in case no input is received. Lastly, a postsynaptic spike is triggered whenever $v(t)$ crosses the *firing threshold* v_{th}. Afterwards, the neuron resets its membrane potential to v_{reset}, and enters in a *refractory period* Δt_{ref} during which new incoming spikes have negligible effect on $v(t)$. Fig. 2 illustrates these concepts for the case of a LIF neuron [45].

2) Synaptic plasticity: Defined as the ability to modify the *efficacy* (weight) of neural connections, *synaptic plasticity* is the basic mechanism underlying learning in biological networks [51]. These architectures are seen to rely on different learning paradigms depending on their duty [52]. For instance, information encoding in biological visual systems is established in an unsupervised fashion, while reinforcement and supervised learning are employed for tasks such as decision making and motor control [52]. Accordingly, various forms of synaptic plasticity have been proposed for SNNs. A brief review of these learning models is presented hereunder.

In the context of SNNs, unsupervised learning is generally referred to as Hebbian learning, since plasticity rules from this paradigm are based on Hebb's postulate: "*cells that fire together, wire together*" [53]. In essence, these methods adapt the efficacy of a connection based on the correlated activity of pre- and postsynaptic cells. Among others, the biologically plausible STDP protocol [39], [40] is, by far, the most popular Hebbian rule for SNNs [54]. With STDP, the repeated arrival of presynaptic spikes to a neuron shortly before it fires leads to synaptic strengthening, also known as Long-Term Potentiation (LTP); whereas if the arrival occurs shortly after the postsynaptic spike, synapses are weakened through Long-Term Depression (LTD). Therefore, the change of efficacy ΔW is normally expressed as a function of the relative timing between these two events. STDP formulations exclusively dependent on this parameter are referred to as *additive* rules [55]. These models, despite their success in pattern recognition problems [41]–[44], are inherently unstable

and require the use of constraints for the synaptic weights, thus resulting in bimodal distributions [56]. On the other hand, *multiplicative* STDP rules incorporate the current efficacy value in the computation of ΔW in an inversely proportional manner. As shown in [57], [58], this additional dependency leads to stable unimodal weight distributions. However, the stability of these approaches results from a complex temporal LTP-LTD balance, and it is not theoretically guaranteed.

Several lines of research can be distinguished regarding the use of supervised learning in SNNs, being the most promising based on the well-known error backpropagation algorithm [59]. Firstly, numerous adaptations to the discontinuous dynamics of SNNs have been proposed for learning temporally precise spike patterns [60]–[63]; but no vision-based applications have been reported. Alternatively, due to the popularity of this method in ANNs, SNNs normally rely on transferring optimization results from their non-spiking counterparts [64]–[66]. Even though similar accuracy levels are obtained more efficiently, information is no longer encoded in the precise spike timing but rather in the neural firing rate, which differs from biological visual systems [67], [68].

With respect to reinforcement learning in SNNs, various models have been presented, the majority of which consist in the modulation of STDP with a reward function [69]–[71]. However, applications of this paradigm are mainly focused on neuroscience research [72], [73], besides several goal-directed navigation problems [74], [75] and the digit-recognition application recently presented in [76].

C. Optical Flow Estimation

In the following, the state-of-the-art of optical flow estimation techniques is discussed. Depending on the sensor type, two categories are distinguished: frame- and event-based methods.

1) Frame-based methods: Most of the solutions presented to date for optical flow estimation from sequences of static images derive from the *brightness constancy constraint* [77]. This assumption states that the intensity structure of a local region in the image plane remains approximately constant under motion for a short period of time. Hence, this constraint is formulated as follows:

$$I_x(x, y)u + I_y(x, y)v + I_t(x, y) = 0 \tag{2}$$

where (u, v) denote the optical flow components to be estimated, and (I_x, I_y, I_t) are the spatial and temporal partial derivatives of the image intensity function at the (x, y) coordinates. Consequently, local optical flow methods can only estimate *normal flow*, which is the motion component normal to the direction of the local intensity gradient $\nabla I = (I_x, I_y)$. This limitation is formally referred to as the *aperture problem* [78]. Depending on the interpretation of this constraint, three major classes of algorithmic optical flow estimation techniques can be distinguished [77]: gradient-based, correlation-based, and frequency-based methods.

Firstly, gradient-based approaches address the estimation problem from the definition of the constraint. However, since Eq. (2) provides two unknown components, a secondary assumption is required. For instance, the popular Lucas-Kanade algorithm [79] estimates the optical flow of sparse image features by assuming that (u, v) are constant within their direct neighborhood. On the other hand, global gradient-based methods combine the aforementioned constraint with a spatial [22], [77] or spatiotemporal [80] cost function to be optimized for the computation of dense image flow.

Correlation-based approaches also formulate the estimation of optical flow as an optimization problem. In this case, the cost functions are based on the comparison of the local intensity structure of an image point in two different frames [81]. As a result, these methods monitor the location of sparse features over successive images, and optical flow is computed from their relative displacements between frames.

Lastly, frequency-based methods are based on the use of velocity-tuned filters designed in the Fourier domain. Inspired by the working principle of direction-selective cells at early stages of the visual cortex [82]–[85], these methods rely on a class of spatiotemporally-oriented filters that exploit the fact that motion can be estimated by extracting orientation in this domain [24]. The spatial convolution of these filters results in a dense optical flow estimation. However, unlike previous approaches, each filter computes a location-specific confidence value that the stimulus is moving in a specific direction with certain speed, rather than the actual flow components. Fig. 3 shows a spatiotemporal illustration of the motion of a simple stimulus and the appearance of the best fitting filter.

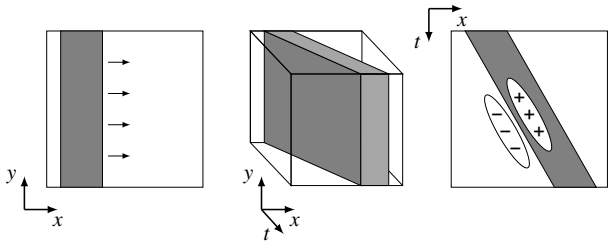


Fig. 3: Illustration of the right horizontal motion of a vertical bar in the spatial (left) and spatiotemporal domains (center, right). An example of the detection of this motion with a spatiotemporally-oriented filter is shown as well (right). Adapted from [24].

The algorithmic perspective is not the only angle from which the visual motion estimation problem has been addressed. The Reichardt model [86], also known as the Elementary Motion Detector (EMD), determines the minimal computations required by biological neural circuits to perceive motion. In its most basic form, the EMD is based on the correlation of the input signals from two neighboring photoreceptors, after one them has been temporally delayed. Despite its accuracy in reproducing the activity of direction-selective neurons from the visual system of flying insects, its cellular implementation is still undiscovered [87].

Alternatively, several ANNs have recently been proposed for dense optical flow estimation from pairs of consecutive

images [28]–[33]. These architectures, which are optimized via supervised learning from large sets of labeled data, report higher accuracy than conventional approaches at the expense of being computationally more expensive [30]. Convolutional [88] and deconvolutional [89] neural layers are at the core of these networks for feature extraction and dense inference, respectively.

2) Event-based methods: The recent introduction of the DVS [15] and other retinomorphic vision sensors [16], [17] has precipitated the development of several novel approaches to event-based optical flow estimation. As before, depending on their working principle, these solutions are divided into algorithmic and neural methods.

Regarding the former, a gradient-based, a plane-fitting, and various frequency-based approaches set the basis of the algorithmic state-of-the-art. These techniques compute sparse optical flow estimates for each newly detected event based on its polarity-specific spatiotemporal neighborhood. Firstly, an adaptation of the Lucas-Kanade algorithm [79] was presented in [90], and reformulated in [91]. Contrary to the frame-based implementation, this solution relies on the definition of the brightness constancy constraint in terms of the second-order partial derivatives of the image intensity function (I_{tx}, I_{ty}, I_{tt}), since events are considered local representations of I_t. Secondly, the method proposed in [26] extracts optical flow information by computing the gradients of a local plane fitted to represent a small spatiotemporal surface of events. This plane-fitting algorithm is further explored in [21], where its applicability is extended to a wider range of velocities through an adaptive temporal window. Lastly, multiple adaptations of the bio-inspired frequency-based methods have been introduced [91]–[93]. Although relying on the same principle as their frame-based counterparts, the event-driven computations allow the implementation in neuromorphic hardware [94].

The estimation of event-based optical flow with neural models is dominated, almost in its entirety, by networks of spiking neurons, i.e. SNNs. In [95], the authors propose an architecture in which motion selectivity results from the synaptic connections of a bursting neuron to two neighboring photoreceptors, being one excitatory and the other inhibitory. If the edge is detected first by the excitatory cell, then spikes are emitted at a fixed rate until the inhibitory pulse is received. Otherwise, the neuron remains inactive. Optical flow is consequently encoded in the burst length and in the relative orientation of the photoreceptors.

In contrast, the SNNs presented in [96], [97] extract motion information through synaptic delays and spiking neurons acting as coincidence detectors. A simple spike-based adaptation of the Reichardt model is introduced in [96] to shown the potential of this approach. This idea is explored in more detail in [97], in which the authors propose the convolution of event sequences with a bank of spatiotemporally-oriented filters, each of which is comprised of non-plastic synapses with equal efficacies, but with delays tuned to capture a particular direction and speed. Similarly to frequency-based methods [24], these filters compute a confidence measure, encoded in the neural activity, rather than the optical flow components. Additionally, this solution employs a second spike-based pooling layer for mitigating the effect of the aperture problem [78].

Whether, and how, direction and speed selectivity emerge in biological networks from visual experience still remains an open question. To date, and to the best of the authors' knowledge, only the development of the former has been addressed. In [98]–[100], the authors show that robust local direction selectivity arises in neural maps through STDP if, apart from presynaptic feedforward connections, neurons also receive input spikes from cells in their spatial neighborhood through plastic synapses with distance-dependent transmission delays. However, despite their success, optical flow estimation does not only involve direction but also speed selectivity.

The motion-sensitive SNN proposed in this work takes inspiration from several of the concepts introduced in the studies mentioned above, but various additions and modifications are necessary. As shown in Section V, local direction and speed selectivity emerge in a bank of excitatory and inhibitory spatiotemporal filters through an STDP-based competitive learning process.

III. Relations between Optical Flow, Ego-Motion, and Visual Observables

The optical flow formulation employed throughout this study is introduced in the present section. This model relates the ego-motion of a downward-looking camera over a static planar scene to the perceived optical flow and its corresponding visual observables. We use this setting as it corresponds to the widely studied problem of optical-flow-based landing [6], [21], [101], [102]. These relations set the basis of the training and evaluation of the readout mechanism presented in Section VI, for which ground truth measurements are required.

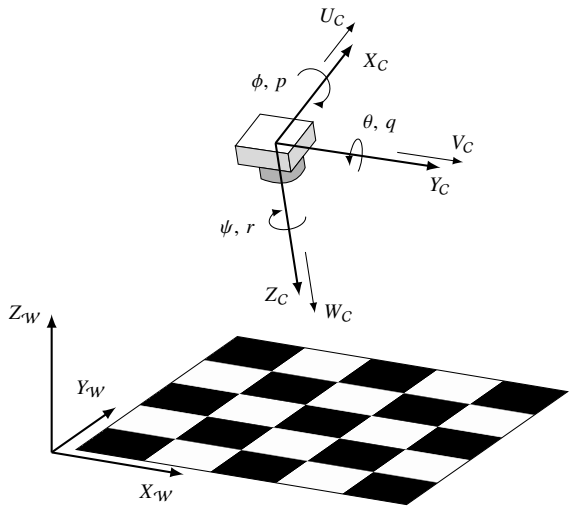


Fig. 4: Definitions of the world ($\mathcal{W}$) and camera (C) references frames. The Euler angles, rotational rates, and translational velocities that describe the motion of C are shown as well.

The derivation of this optical flow model relies on the two reference frames illustrated in Fig. 4. The inertial world frame is denoted by $\mathcal{W}$, whilst C describes the camera frame centered at the focal point of the event-based vision sensor. In each of these frames, position is defined through the coordinates (X, Y, Z), with (U, V, W) as the corresponding velocity components. The orientation of C with respect to $\mathcal{W}$ is described by the Euler angles ϕ, θ, and ψ, denoting roll, pitch, and yaw, respectively. Similarly, p, q, and r denote the corresponding rotational rates.

The relations between sensor ego-motion, optical flow, and visual observables are based on the pinhole camera model [5]. In this formulation, pixel coordinates in the sensor's pixel array are denoted by (x, y), while (u, v) represent optical flow components, measured in pixels per second. Note that this model assumes an undistorted vision sensor, as it is the case when using the DVS simulator [47]. The actual DVS can be corrected from lens distortion as indicated in [21].

Consider the situation depicted in Fig. 4, in which the sensor C moves arbitrarily through a static environment subjected to translational (U_C, V_C, W_C) and rotational (p, q, r) velocities. Due to this ego-motion, the projection of a world point onto the image plane describes a translational motion, i.e. its optical flow, whose components are obtained as follows:

$$\begin{aligned} u &= -\frac{U_C}{Z_C} + \frac{W_C}{Z_C}x - q + ry + pxy - qx^2 \\ v &= -\frac{V_C}{Z_C} + \frac{W_C}{Z_C}y + p - rx - qxy + py^2 \end{aligned} \tag{3}$$

From Eq. (3), the optical flow of a point can be resolved into translational and rotational components [5]. Since the latter is independent of the three-dimensional structure of the visual scene, these expressions can be derotated if information on the rotational rates of the sensor is available. This derotation leads to pure translational optical flow components, denoted by (u_T, v_T). Moreover, if the scene is a planar surface, the depth Z_C of all visible world points is interrelated through:

$$Z_C = Z_0 + Z_X X_C + Z_Y Y_C \tag{4}$$

where Z_0 is defined as the distance to the surface along the optical axis of the sensor, and Z_X and Z_Y represent the slopes of the planar scene with respect to the X- and Y-axis of C [6]. If information on the attitude of the sensor is available, these slopes can be computed from the pitch and roll angles as:

$$Z_X = \tan\theta, \quad Z_Y = -\tan\phi \tag{5}$$

In [5], the relation between the position of an arbitrary point in C and its projection onto the image plane is given by $(x, y) = (X_C/Z_C, Y_C/Z_C)$. Consequently, Eq. (4) may also be written in the form:

$$\frac{Z_C - Z_0}{Z_C} = Z_X x + Z_Y y \tag{6}$$

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

$$\vartheta_x = \frac{U_C}{Z_0}, \quad \vartheta_y = \frac{V_C}{Z_0}, \quad \vartheta_z = \frac{W_C}{Z_0} \tag{7}$$

Then, according to the derivations in [6], substituting Eqs. (6) and (7) into Eq. (3) leads to the following expressions for translational optical flow:

$$\begin{aligned} u_T &= (-\vartheta_x + \vartheta_z x)(1 - Z_X x - Z_Y y) \\ v_T &= (-\vartheta_y + \vartheta_z y)(1 - Z_X x - Z_Y y) \end{aligned} \tag{8}$$

From Eq. (8), and under the aforementioned assumptions, the scaled velocities, which provide non-metric information on sensor ego-motion, can be derived from the translational optical flow of multiple image points. ϑ_x and ϑ_y are the opposites of the so-called *ventral flows*, a quantification of the average flows in the X- and Y-axis of C respectively [21]. Hence, $\omega_x = -\vartheta_x$ and $\omega_y = -\vartheta_y$. On the other hand, ϑ_z is proportional to the *divergence* of the optical flow field, $D = 2\vartheta_z$ [21]. Throughout this work, these optical flow visual observables, more specifically the ventral flow components, are employed to refer to the stimulus speed in the image plane. Besides, these are the parameters to be estimated with the readout mechanism presented in Section VI.

IV. Adaptive Spiking Neuron Model and Stable STDP Learning Rule

This section describes the foundations of the spike-based processing and learning of our SNN proposal. Firstly, the neuron model of the architecture is introduced in Section IV-A as a novel variation of the LIF model [45] which adapts its response to varying input statistics. Secondly, Section IV-B presents the stable multiplicative STDP rule that forms the basis of the competitive Hebbian learning framework used for synaptic plasticity.

A. Adaptive Spiking Neuron Model

Let $j = 1, 2, \ldots, n^{l-1}$ denote a group of presynaptic neurons, from layer $l-1$, fully connected in a feedforward fashion to a set of postsynaptic cells $i = 1, 2, \ldots, n^l$, from layer l. As depicted in Fig. 5, these neural connections can be considered as *multisynaptic*, i.e. the link between two cells is not restricted to a single synapse, but several can coexist. In this implementation, the number of multisynaptic connections, denoted by m^l, is layer-specific; and thus, it applies to all pairs of linked

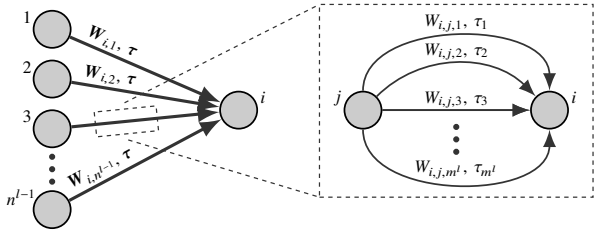


Fig. 5: Schematic of the feedforward connectivity between neurons from two adjacent layers (left). These connections can be considered as being multisynaptic (right), each one having its own efficacy, transmission delay, and trace.

neurons whose postsynaptic cell belongs l. Based on this, layer connectivity is characterized by two main elements: a weight matrix $\boldsymbol{W}^l \in \mathbb{R}^{n^l \times n^{l-1} \times m^l}$ and a delay vector $\boldsymbol{\tau}^l \in \mathbb{R}^{m^l}$. From the definition of $\boldsymbol{W}^l$, each connection has its own independent synaptic efficacy value. On the other hand, $\boldsymbol{\tau}^l$ determines the different transmission delays of a multisynaptic group of connections. Similarly to m^l, $\boldsymbol{\tau}^l$ is also layer-specific.

Apart from $\boldsymbol{W}^l$ and $\boldsymbol{\tau}^l$, each synapse keeps track of an additional parameter that models the recent history of spikes transmitted. Formally referred to as the *synaptic trace* [103], and defined as $\boldsymbol{X}^l \in \mathbb{R}^{n^l \times n^{l-1} \times m^l}$, its dynamics is given by the following differential equation:

$$\lambda_X \frac{dX^l_{i,j,d}(t)}{dt} = -X^l_{i,j,d}(t) + \alpha s^{l-1}_j(t - \tau^l_d) \tag{9}$$

where λ_X is the time constant of this first-order system, α is a scaling factor, and $\boldsymbol{s}^l(t) \in \mathbb{R}^{n^l}$ denotes the (binary) record of neural activity, or *spike train*, of cells from layer l. Note that $d = 1, 2, \ldots, m^l$ serves to refer both to connections within a multisynaptic group and their corresponding delays.

From Eq. (9), whenever a spike arrives at a postsynaptic neuron i via a synapse with transmission delay τ_d, the corresponding presynaptic trace $X^l_{i,j,d}(t)$ increases by a factor of α. In case no spike is received, the trace decays exponentially towards zero according to λ_X.

Once these parameters have been described, the proposed spiking neuron model is introduced. The LIF neuron [45], whose internal dynamics is illustrated in Fig. 2, is probably the most popular and simplest phenomenological spiking model presented to date [55]. Its main assumption is that, in SNNs, no information is encoded in the spike amplitude, but rather on the firing time. Consequently, neural activity is reduced to discrete and binary temporal events, thus ensuring computational tractability. Based on this principle, we propose the augmentation of the LIF neuron model with an adaptive mechanism that regulates its response using the presynaptic trace $\boldsymbol{X}^l$ of input connections. The dynamics of this adaptive neuron evolve according to the following expressions:

$$\lambda_v \frac{dv^l_i(t)}{dt} = -(v^l_i(t) - v_{\text{rest}}) + i^l_i(t) \tag{10}$$

$$i^l_i(t) = \sum_{j=1}^{n^{l-1}} \sum_{d=1}^{m^l} (W^l_{i,j,d} s^{l-1}_j(t - \tau^l_d) - X^l_{i,j,d}(t)) \tag{11}$$

where λ_v denotes the time constant of the membrane potential, and $\boldsymbol{i}^l(t)$ is the so-called *forcing function* of the system.

From Eqs. (10) and (11), the membrane potential $v^l_i(t)$ of a neuron evolves over time by integrating scaled presynaptic spikes from its input synapses, similarly to the conventional LIF model [45]. Whenever $v^l_i(t)$ reaches (or surpasses) the firing threshold v_{th}, a postsynaptic spike is generated, i.e. $s^l_i(t) = 1$, and $v^l_i(t)$ is reset to v_{reset}. In addition, the neuron enters in a refractory period Δt_{ref} during which presynaptic spikes have no effect on $v^l_i(t)$ to ensure the temporal separation of postsynaptic pulses. In case no spike is fired at time t, this is reflected in the neuron's spike train as $s^l_i(t) = 0$.

Unlike the traditional LIF implementation [45], the forcing function $\boldsymbol{i}^l(t)$ of our neuron model includes an additional term, further referred to as the *homeostasis* parameter, that adapts the neural response to the varying input statistics—in particular, to the input firing rate—using the presynaptic trace $\boldsymbol{X}^l$ as an *excitability* indicator. This term is named after the internal regulatory mechanisms of biological organisms [104]. Inferring from Eq. (11), this parameter acts, in essence, as an (inhibitory) penalty in the update rule of $v^l_i(t)$. A postsynaptic neuron connected to a group of highly-active presynaptic cells is said to have low excitability due to its relatively high $\boldsymbol{X}^l$. For this neuron to fire, it needs to receive a large number of presynaptic spikes shortly separated in time. On the contrary, the same cell connected to poorly-active neurons is highly excitable; and thus, the firing threshold v_{th} can still be reached despite the considerably larger time difference between input spikes. Note that, to get the desired neural adaptation, the scaling factor α, from Eq. (9), needs to be selected in accordance with the neural parameters, mainly v_{th} and the range of possible $\boldsymbol{W}^l$ values.

When dealing with an event-based camera as source of input spikes, the firing rate of the sensor is not only correlated to the appearance of features from the visual scene, but also to their optical flow. Slow apparent motion leads to successive events being more distant in time than those captured from fast motion. Consequently, if these events are to be processed with a network of spiking neurons, a homeostasis mechanism is required to ensure that similar features are detected regardless of the encoding spike rate. Apart from our proposal, other approaches to homeostasis have been presented to date, such as threshold balancing [105] or weight scaling [57]. However, instead of relying on presynaptic information, these methods need multiple postsynaptic spikes until the adaptive mechanism is adjusted for proper operation. This working principle makes them unsuitable for the rapidly varying statistics of the data generated by a moving event-based vision sensor.

The performance of our homeostasis method and its effect on the learning process are explored in Section V.

B. Stable STDP Learning Rule

As introduced in Section II-B, STDP [39], [40] is a biologically plausible local unsupervised learning rule for SNNs. Based on Hebb's postulate [53], this form of synaptic plasticity modifies the strength of a connection between neurons according to their correlated activity. In this work, we propose a novel multiplicative STDP implementation that, by combining the weight-dependent exponential rule from [58] with presynaptic trace information, becomes inherently stable. Whenever a postsynaptic neuron i fires a spike, the efficacy of its presynaptic connections is updated as follows:

$$\begin{aligned} W^l_{i,j,d} &= W^l_{i,j,d} + \Delta W^l_{i,j,d} \\ \Delta W^l_{i,j,d} &= \eta(\text{LTP} + \text{LTD}) \end{aligned} \tag{12}$$

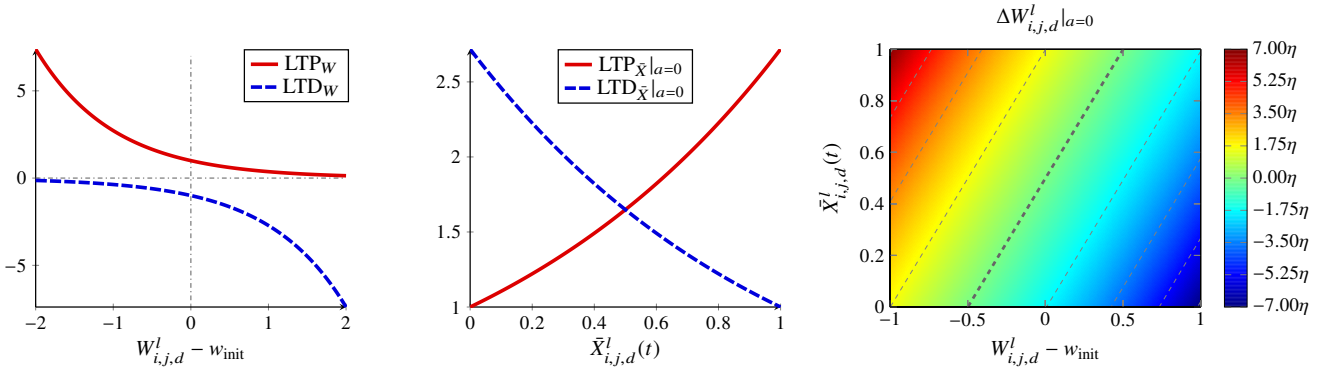


Fig. 6: Illustration of the novel multiplicative STDP rule proposed in this work. The weight update (right) results from the linear combination of the non-exclusive LTP and LTD processes. These, in turn, are characterized by symmetrical dependencies on the synaptic weights (left) and normalized presynaptic traces (center). Note that, in the schematic of the weight update (right), the weight axis is limited to the $[-1, 1]$ range only for the purpose of a better visualization of the equilibrium weights (dashed thick line) for $a = 0$.

$$\begin{aligned} \text{LTP} &= \text{LTP}_W \cdot \text{LTP}_{\bar{X}}, & \text{LTD} &= \text{LTD}_W \cdot \text{LTD}_{\bar{X}} \\ \text{LTP}_W &= e^{-(W^l_{i,j,d} - w_{\text{init}})}, & \text{LTD}_W &= -e^{(W^l_{i,j,d} - w_{\text{init}})} \\ \text{LTP}_{\bar{X}} &= e^{\bar{X}^l_{i,j,d}(t)} - a, & \text{LTD}_{\bar{X}} &= e^{(1-\bar{X}^l_{i,j,d}(t))} - a \end{aligned} \tag{13}$$

where η is the learning rate of the rule, w_{init} refers to the initialization weight of all synapses at the beginning of the learning process, and $\bar{\boldsymbol{X}}_i \in [0, 1]$ denotes the normalized presynaptic traces of neuron i at the moment of firing. Further, for stability, $a < 1$.

From Eqs. (12) and (13), the weight update $\Delta \boldsymbol{W}^l_i$ results from the linear combination of the output of two non-mutually exclusive processes: LTP, for strengthening, and LTD, for weakening synaptic connections. Both of these processes are dependent on the weights (LTP$_W$, LTD$_W$) and normalized traces (LTP$_{\bar{X}}$, LTD$_{\bar{X}}$) of the synapses under analysis. On the one hand, as mentioned before, the weight dependency of our learning rule takes inspiration from the STDP formulation presented in [58]. LTP$_W$ and LTD$_W$ are inversely proportional to $\boldsymbol{W}^l_i$ in an exponential fashion, and are centered around w_{init} (see Fig. 6, left). Consequently, the effect of LTP$_W$ decreases (increases) the larger (smaller) a synaptic weight is in comparison to w_{init}. The opposite relation holds true for LTD$_W$. On the other hand, rather than relying on the precise spike timing [58], our rule employs normalized presynaptic trace information as a measure of the relevance of a particular connection to the postsynaptic spike triggering the update. The higher (lower) the value of $\bar{X}_{i,j,d}(t)$, the larger (smaller) the effect of LTP$_{\bar{X}}$, and vice versa for LTD$_{\bar{X}}$ (see Fig. 6, center).

By definition, our STDP implementation is inherently stable for $a < 1$. With this condition, the formulation itself establishes an equilibrium weight for each value of $\bar{X}_{i,j,d}(t)$ by balancing the contributions of LTP and LTD on $\Delta \boldsymbol{W}^l_i$ (see Fig. 6, right). The parameter a has control over this mapping through the steepness of LTP$_{\bar{X}}$ and LTD$_{\bar{X}}$ in $\bar{\boldsymbol{X}}_i \in [0, 1]$. As a result of the stability, no additional mechanism is required for preventing weights from vanishing or exploding. Synapses characterized by weights that are higher (lower) than their corresponding equilibrium state are consistently depressed (potentiated) until synapse-specific stability is achieved.

To track the convergence of the learning process, we propose the use of the following mean square error criterion:

$$\mathcal{L}_i = \frac{1}{n^{l-1} m^l} \sum_{j=1}^{n^{l-1}} \sum_{d=1}^{m^l} (\bar{X}^l_{i,j,d}(t) - \bar{W}^l_{i,j,d})^2 \tag{14}$$

where $\bar{\boldsymbol{W}}_i \in [0, 1]$ denotes the presynaptic weights of neuron i after an update, normalized to the current maximum value. As the learning progresses, the moving average of $\mathcal{L}_i$ converges to a (close-to-zero) equilibrium state. In this work, we stop synaptic plasticity using a fixed threshold on this parameter.

Similarly to the adaptive neuron model, the performance of this learning rule is explored in Section V.

1) Local inter-lateral competition: For neurons to learn distinct features from the input data through STDP, this learning rule needs to be combined with what is known as a Winner-Take-All (WTA) mechanism [106]. This form of competition implies that, when a neuron fires a spike and updates its presynaptic weights according to Eqs. (12) and (13), the rest of postsynaptic cells (from the same layer) locally connected to the same input neurons get inhibited. As a result, these cells are prevented from triggering STDP while the neuron that fired first, i.e. the *winner*, remains in the refractory period.

Instead of relying on non-plastic synapses transmitting inhibitory spikes with a certain delay, our implementation assumes that the internal dynamics of these neurons are intercorrelated. Whenever the winner cells resets its membrane potential and enters in the refractory period, neurons affected by the WTA mechanism do the same immediately afterwards. In case multiple of these neurons fire simultaneously, the cell with the highest membrane potential has preference for triggering the weight update. Further, the postsynaptic spikes

from the other firing neurons are not considered. Lastly, to ensure coherence between the training and inference phases of our proposed SNN, layers trained with STDP maintain the WTA mechanism after the learning process.

V. Spiking Neural Network Architecture for Motion Perception

This section describes the different neural layers that comprise the feedforward architecture of our hierarchical SNN for the task of motion perception. An illustration of this network is shown in Fig. 7. Section V-A introduces the Input layer; a first stage that encodes event-based sensor data in a compatible format for the rest of the architecture. Secondly, input feature extraction is conducted with the single-synaptic convolutional layer (SS-Conv) described in Section V-B. Once these features are detected, Section V-C covers how their local motion is identified through a multi-synaptic convolutional layer (MS-Conv). Lastly, a Pooling (Section V-D) and a Dense, i.e. fully-connected, layer (Section V-E) lead to the emergence of sensitivity to the global motion of the visual scene in individual neurons. For a better understanding of their capabilities, these subsections include layer-specific evaluations.

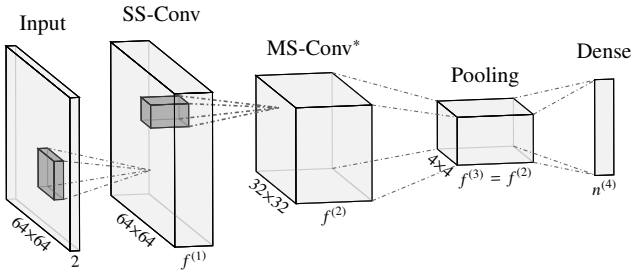
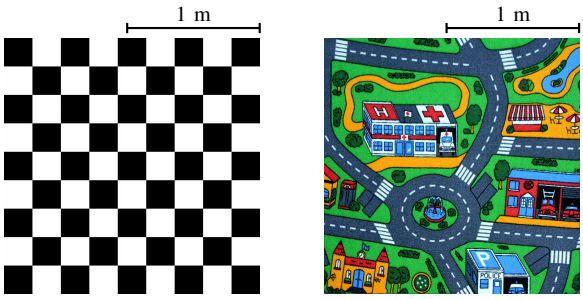


Fig. 7: Overview of the feedforward SNN architecture. Note that the depicted MS-Conv layer corresponds to the feature-invariant postsynaptic maps introduced in Section V-C and further shown in Fig. 12.

The spike-based computation carried out by the present network is based on the adaptive neuron model introduced in Section IV-A. The membrane potential of all non-input neurons evolves according to Eq. (10), and the definition of the model forcing function $\boldsymbol{i}^l(t)$ varies depending on the connectivity scheme of each layer. In this work, the resting potential v_{rest} is considered null, and the refractory period is set to $\Delta_{\text{ref}} = 3.0$ ms. Concerning learning, this SNN is trained in a *layer-by-layer* fashion using the unsupervised STDP rule presented in Section IV-B. Regardless of the layer type, the parameter a from Eq. (13) is set to 0, the initialization weight w_{init} to 0.5, and the learning rate η to 2.5×10^{-4}. As shown in Fig. 6 (right), this leads to weight distributions that, after convergence, are naturally constrained in the range $\boldsymbol{W}^l \in [0, 1]$. Throughout the learning phase, 150-ms event sequences from a training dataset are presented sequentially at random following a uniform distribution.

For the generation of the visual stimuli employed in this section, the simulated DVS [47] is moved in straight lines at different constant speeds, and at an altitude of $Z_{\mathcal{W}} = 0.5$ m with respect to a (virtual) textured planar surface, towards which it is facing. During motion, we force the Euler angles ϕ and θ to remain null so as to ensure the perpendicularity between sensor and surface. Fig. 8 shows the two different textures used in this work. The main experiments are based on the checkerboard pattern, shown in Fig. 8a. This texture provides high contrast and clear edges; hence facilitating optical flow estimation. On the other hand, the roadmap in Fig. 8b is largely characterized by unstructured and low-contrast visual features leading to noisy event sequences. It is used in the evaluation of the SS-Conv layer (Section V-B) to show the generalizability of this approach to more realistic scenarios.



(a) Checkerboard (b) Roadmap

Fig. 8: Texture patterns of the planar surface towards which the vision sensor is facing during motion. The scale is shown for reference purposes.

The simulation of the proposed SNN architecture is conducted using the *cuda-snn*[1] framework. This simulator, built in its entirety for this research, is written in the C++ programming language, and benefits from the CUDA libraries [107] for efficient parallel computation.

A. Input Layer

As mentioned before, the role of this first stage in the neural architecture is to encode the arriving sequences of DVS events into a format that is compatible with subsequent layers of the network. This layer can be understood as to be comprised of spiking neurons with no internal dynamics, whose neural activity is determined by event arrival. As shown in Fig. 7, neurons are arranged in two-dimensional *feature maps*, one per polarity, resembling the grid-like topology of the vision sensor. Depending on the spatial resolution of these maps, each neuron is assigned with the polarity-specific events of one or multiple DVS pixels (with no overlap). Additionally, for the simulation of the SNN, time is discretized in temporal units of length Δt_{sim}, in which neurons are only allowed to fire once. Consequently, all input cells receiving at least one event during a particular Δt_{sim} are treated equally.

In the experiments presented in this section, the 128×128 DVS pixel array is downsampled to a spatial resolution of 64×64 input neurons for computational efficiency purposes. On the other hand, contrary to the microsecond resolution of the DVS, Δt_{sim} is set to 1.0 ms, which is the timescale of the current state-of-the-art of neuromorphic devices [37], [38].

[1] Source code to be released in https://github.com/tudelft/

B. SS-Conv Layer: Feature Extraction

SS-Conv is a spiking adaptation of the well-known convolutional layers [88], whose role is the extraction of the visual features from which motion is perceived at a later stage.

As shown in Fig. 7, neurons in the SS-Conv layer are retinotopically arranged in $k = 1, 2, \ldots, f^l$ two-dimensional feature maps. Each of these neurons receives spikes from presynaptic cells within a specific spatial receptive field, of size r^{l-1}, in all maps of the previous layer. This sparse connectivity is characterized by a set of excitatory synaptic weights, formally referred to as a *convolutional kernel* $\boldsymbol{W}_k^l$, that is equal for all neurons belonging to the same map. Consequently, this form of *weight sharing* ensures that, within a map, neurons are selective to the same input feature but at different spatial locations. This layer maintains the input (spatial) dimensionality through a convolutional stride of one pixel and a zero-padding mechanism.

Let a postsynaptic neuron i from the feature map k be characterized by the convolutional kernel $\boldsymbol{W}_k^l \in \mathbb{R}^{r^{l-1} \times f^{l-1}}$, the presynaptic trace $\boldsymbol{X}_i^l \in \mathbb{R}^{r^{l-1} \times f^{-1}}$, and the spike train $s_{i,k}(t)$. Further, let $\boldsymbol{N}_{i,k}$ refer to the map-specific direct neural neighborhood of the cell, including itself. Then, considering neural connections as single-synaptic with transmission delay τ, the forcing function driving the internal dynamics of neurons in this layer is defined as follows:

$$i_{i,k}^l(t) = \sum_{ch=1}^{f^{l-1}} \sum_{j=1}^{r^{l-1}} W_{j,ch,k}^l s_{j,ch}^{l-1}(t-\tau) - \max_{\forall b \in \boldsymbol{N}_{i,k}} \sum_{ch=1}^{f^{l-1}} \sum_{j=1}^{r^{l-1}} X_{b,j,ch}^l(t) \tag{15}$$

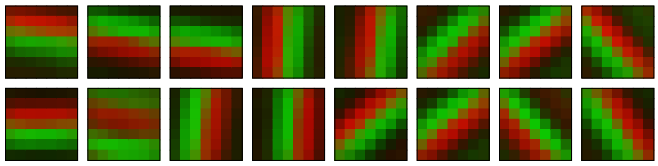
Apart from the sparse connectivity, the only difference between this expression and the fully-connected formulation, i.e. Eq. (11), is in the homeostasis parameter. When arranged retinotopically, the neurons' dynamics do not only depend on their own presynaptic trace $\boldsymbol{X}_i^l$, but also on the synaptic traces characterizing their direct spatial neural neighborhood $\boldsymbol{N}_{i,k}$. By using the maximum trace in the membrane potential update, neurons are prevented from specializing to the leading edge of moving visual features (high excitability, low robustness), rather than to the features themselves.

The implementation of STDP is also affected by the spatial arrangement of the neurons. Multiple cells from the same feature map can fire simultaneously at different locations, and thus, to different input data. Since these neurons share convolutional kernel, $\Delta \boldsymbol{W}_k^l$ is computed through synapse-specific averages of the local contributions. In addition, due to the high overlap of presynaptic receptive fields, the WTA inhibitory mechanism described in Section IV-B1 is expanded to cells within a small neighborhood of the firing neurons, regardless of the feature map. Note that, after learning, only the neuron-specific competition is maintained.

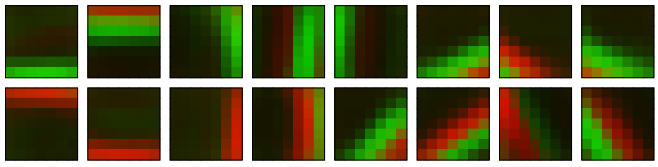
1) Evaluation: To qualitatively evaluate the performance of the SS-Conv layer in the learning of the strongest and most frequent visual features from the input dataset, we used the roadmap texture due to its complexity. In this experiment, the (constant) ventral flow components of the stimuli, ω_x and ω_y, ranged between 0.5 and 3.0 s^{-1}, in absolute terms. Further, neural parameters were empirically set to $v_{\text{th}} = 1.0$, $\lambda_v = \lambda_X = 5.0$ ms, $\tau = 1.0$ ms, and $\alpha = 0.4$. Fig. 9a shows the appearance of sixteen 7×7 convolutional kernels learned through the proposed STDP rule under these conditions.

With this kernel scale, our STDP implementation led to the successful identification of edges at different orientations within the receptive field, and with the two combinations of event polarity. A point to remark is the fact that, despite the WTA mechanism, several features were learned by multiple kernels instead of being specific to just one. This is an indicator that, first, the unsupervised nature of STDP prioritizes frequent features over other, more complex, that can still be useful for motion perception; and second and consequently, sixteen is an unnecessarily large number of kernels for this spatial scale and roadmap dataset.

On the other hand, Fig. 9b illustrates the need for the homeostasis parameter as detailed in Eq. (15), when dealing with retinotopically-arranged neurons. As shown, when the formulation in Eq. (11) is employed instead, convolutional kernels specialize to the leading edge of moving features, and hence most of these kernels are characterized by more ambiguous synaptic configurations in which the strong synapses are mainly located on the receptive field borders. Because of this selectivity, a larger number of kernels is required for the extraction of the same features as with Eq. (15).



(a) Homeostasis: Maximum presynaptic trace in $\boldsymbol{N}_{i,k}$.



(b) Homeostasis: Neuron-specific presynaptic trace.

Fig. 9: 7×7 SS-Conv kernels learned from the roadmap texture. Synaptic strength is encoded in color brightness: green for input neurons with positive (event) polarity, and red for negative. Results are shown for the homeostasis formulations in Eq. (15) (top) and Eq. (11) (bottom).

A more simplistic dataset was used to assess the effect of the homeostasis scaling factor α on the neural response. The DVS orientation was fixed with respect to the checkerboard pattern so that only vertical and horizontal edges of both (event) polarities were perceived, and the stimuli were characterized by pure horizontal and vertical motion of $\{|\omega_x|, |\omega_y|\} \in [0.2, 4.0]$ s^{-1}. Fig. 10 shows the four 7×7 convolutional kernels learned using this input data, further referred to as the checkerboard dataset, and the aforementioned neural settings.

Fig. 11 shows the neural response of this SS-Conv layer for the case of pure horizontal rightward motion. As discussed

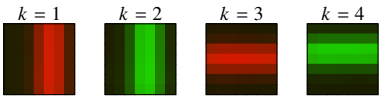


Fig. 10: 7×7 SS-Conv kernels learned from the checkerboard texture.

in Section IV-A, the lack of adaptation in the neuron model, i.e. $\alpha = 0$, entails a high correlation between neural activity and the speed of the stimulus. Results in this figure confirm that the homeostasis parameter, through the correct value of α, mitigates the effect of this correlation for a wide range of speeds. Additionally, due to the penalty nature of this term, α also affects the net response of the layer. The larger the value of this parameter, the lower the number of postsynaptic spikes fired for the same stimulus, but the more accurate these spikes are. As shown, neural maps associated with horizontal kernels, i.e. $k = 3$ and $k = 4$ in Fig. 10, reacted erroneously to input comprised only by vertical edges for $\alpha < 0.4$, but remained silent otherwise. Consequently, as a compromise between accuracy and net response, we set this factor to $\alpha = 0.4$ for this layer and neural parameters.

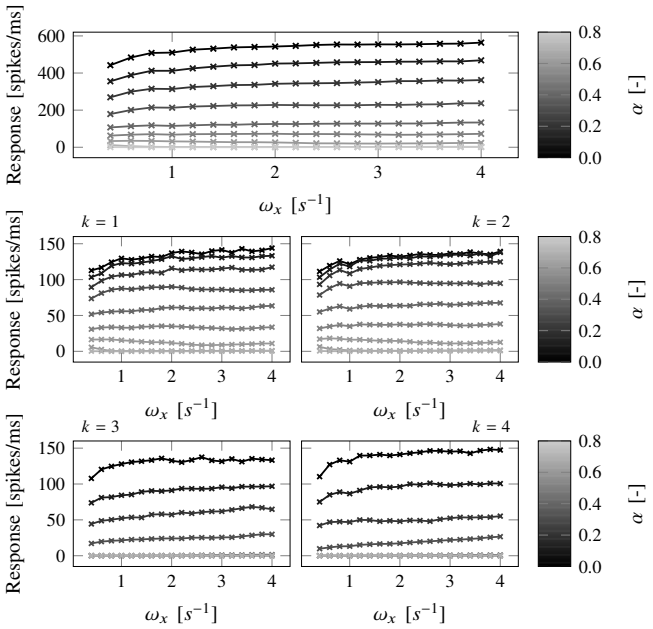


Fig. 11: SS-Conv layer response (top) and kernel-specific responses (bottom two rows) as a function of the homeostasis scaling factor α and the stimulus speed ω_x. Measurements obtained from the checkerboard dataset. The notation for the kernel-specific plots is introduced in Fig. 10.

C. MS-Conv Layer: Local Motion Perception

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

As the basis of this formulation, the information required to perceive the motion of a particular feature is assumed to be exclusively contained in its corresponding presynaptic

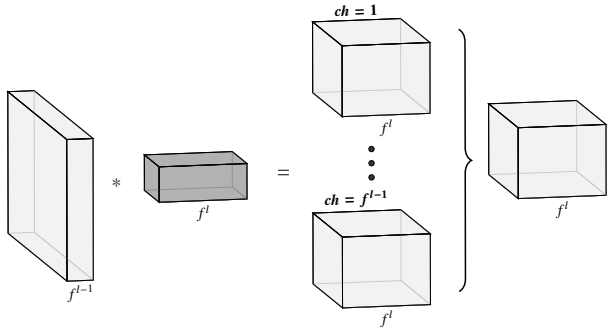


Fig. 12: Illustration of the convolutional operator of the MS-Conv layer. From left to right: Presynaptic neural maps, set of MS-Conv kernels, feature-specific postsynaptic maps, and feature-invariant postsynaptic maps.

neural map. Hence, contrary to those from the SS-Conv layer, convolutional kernels in MS-Conv are not connected to all presynaptic maps simultaneously. Instead, they are defined as $\boldsymbol{W}_k^l \in \mathbb{R}^{r^{l-1} \times m^l}$, and convolution is performed for each of these maps individually. As shown in Fig. 12, this operation leads to kernels assigned to f^{l-1} feature-specific postsynaptic maps, among which the presynaptic weights (and weight updates) are shared. Since feature-specific activity is of minor importance for later stages of the architecture, the neural responses of these maps are merged into a single feature-invariant map per convolutional kernel. Per spatial location, postsynaptic spike trains are combined by prioritizing the firing of spikes over silent neurons. In this layer, the spatial dimensions of postsynaptic maps are halved with respect to the previous through zero-padding and a convolutional stride of two pixels.

For the perception of local motion, we propose an augmentation of Eq. (15) based on the foundations of frequency-based optical flow methods [24] and bio-inspired motion detectors [86], [108]. Firstly, motion is to be extracted as orientation in the spatiotemporal domain. Therefore, neural connections in the MS-Conv layer are considered multisynaptic with different constant transmission delays as given by $\boldsymbol{\tau}^l \in \mathbb{R}^{m^l}$. Secondly, since these delays (and the rest of neural parameters) are equal for all (spatiotemporal) convolutional kernels, inhibitory synapses are required to prevent the firing of erroneous postsynaptic spikes when the input trace only fits part of the excitatory component of the kernels. To account for this, each MS-Conv kernel is defined by a pair of excitatory and inhibitory plastic weight matrices, denoted by $\boldsymbol{W}_k^{l,\text{exc}}$ and $\boldsymbol{W}_k^{l,\text{inh}}$, respectively. According to these additions, the forcing function of cells in this layer is expressed as:

$$i^l_{i,k,ch}(t) = \sum_{j=1}^{r^{l-1}} \sum_{d=1}^{m^l} (W^{l,\text{exc}}_{j,d,k} + \beta W^{l,\text{inh}}_{j,d,k}) s^{l-1}_{j,ch}(t - \tau^l_d) - \max_{\forall b \in \boldsymbol{N}_{i,k,ch}} \sum_{j=1}^{r^{l-1}} \sum_{d=1}^{m^l} X^l_{b,j,d,ch}(t) \tag{16}$$

where β scales the impact of inhibitory synapses, and the presynaptic trace of a neuron is defined as $\boldsymbol{X}^l_i \in \mathbb{R}^{r^{l-1} \times m^l \times f^{l-1}}$.

Due to the neural spatial disposition, the implementation of STDP in this layer is, in essence, identical to the one employed for SS-Conv. With respect to the learning of inhibitory kernels, the parameter a from Eq. (13) remains at 0, w_{init} is set to -0.5, and the weights are initialized at 0. This discrepancy between w_{init} and the initialization weight enables postsynaptic neurons to be reactive to different input features, i.e. to *explore* the input space, until kernel specialization. With this configuration, after convergence, inhibitory weight distributions are naturally constrained in the range $\boldsymbol{W}_k^{l,\text{inh}} \in [-1, 0]$.

Lastly, remark that due to the limited size of presynaptic receptive fields, the neural response of the layer is subjected to the aperture problem [78], and hence only normal flow can be perceived. The effect of this limitation is mitigated in subsequent layers via the pooling mechanism in Section V-D.

1) Evaluation: To evaluate the performance of the MS-Conv layer, sixteen (excitatory and inhibitory) 7×7 kernels were simultaneously trained using the pure horizontal and vertical stimuli from the checkerboard dataset. Neural parameters were empirically set to $v_{\text{th}} = 1.0$, $\lambda_v = \lambda_X = 5.0$ ms, $\alpha = 0.25$, and $\beta = 0.75$; and the SS-Conv kernels in Fig. 10 were employed for feature extraction. With respect to the spatiotemporal characteristics of MS-Conv kernels, each multisynaptic connection was comprised of ten sub-synapses with constant transmission delays linearly spaced between 1.0 and 50.0 ms. Figs. 13 and 14 show kernel appearance after convergence, and the response of the corresponding neural maps as a function of ω_x and ω_y.

These figures confirm that, with the MS-Conv connectivity pattern, STDP leads to the successful identification of the spatiotemporally-oriented traces of input features, and hence of their local motion. Out of the sixteen kernels trained, seven specialized to pure horizontal motion, and the remaining nine to pure vertical. Each direction of motion (up, down, left, and right) was captured by at least four kernels, which in turn were selective to a particular stimulus speed. For instance, upward motion was identified by kernels $k = 13$–16, from slow to fast tuning speed. Therefore, convolutional kernels in this layer can be understood as local velocity-tuned filters that resemble those employed in frequency-based optical flow methods (see Fig. 3) [24], [91]–[93], [97]. However, instead of being manually designed, these filters emerge from visual experience in a biologically plausible unsupervised fashion.

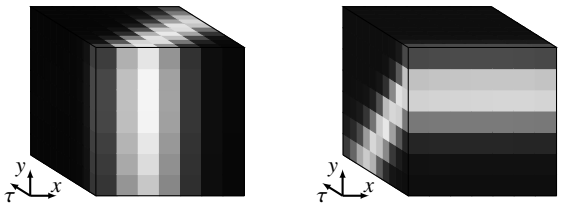
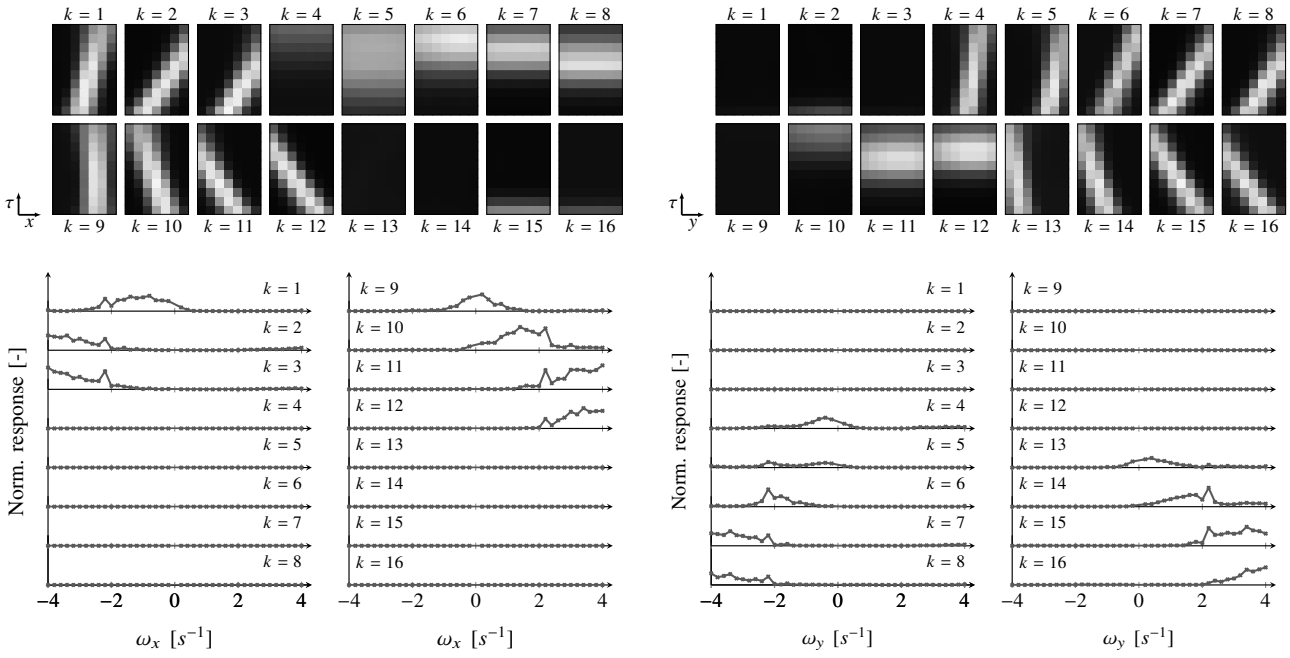


Fig. 14: Illustration of two of the sixteen $7 \times 7 \times 10$ MS-Conv kernels learned from the checkerboard dataset. According to the notation in Fig. 13, $k = 2$ (left) and $k = 15$ (right) are selective to leftward and upward image motion, respectively. Synaptic strength is encoded in color brightness: white for strong excitatory and weak inhibitory connections, and vice versa for black.



(a) x-τ representation and pure horizontal motion.

(b) y-τ representation and pure vertical motion.

Fig. 13: Appearance (top) and neural response (bottom) of the sixteen $7 \times 7 \times 10$ MS-Conv kernels learned from the checkerboard dataset. Response plots are normalized by the maximum kernel response on the stimuli evaluated: 8.2763 spikes/ms by $k = 11$ for $\omega_x = 4.0\ s^{-1}$.

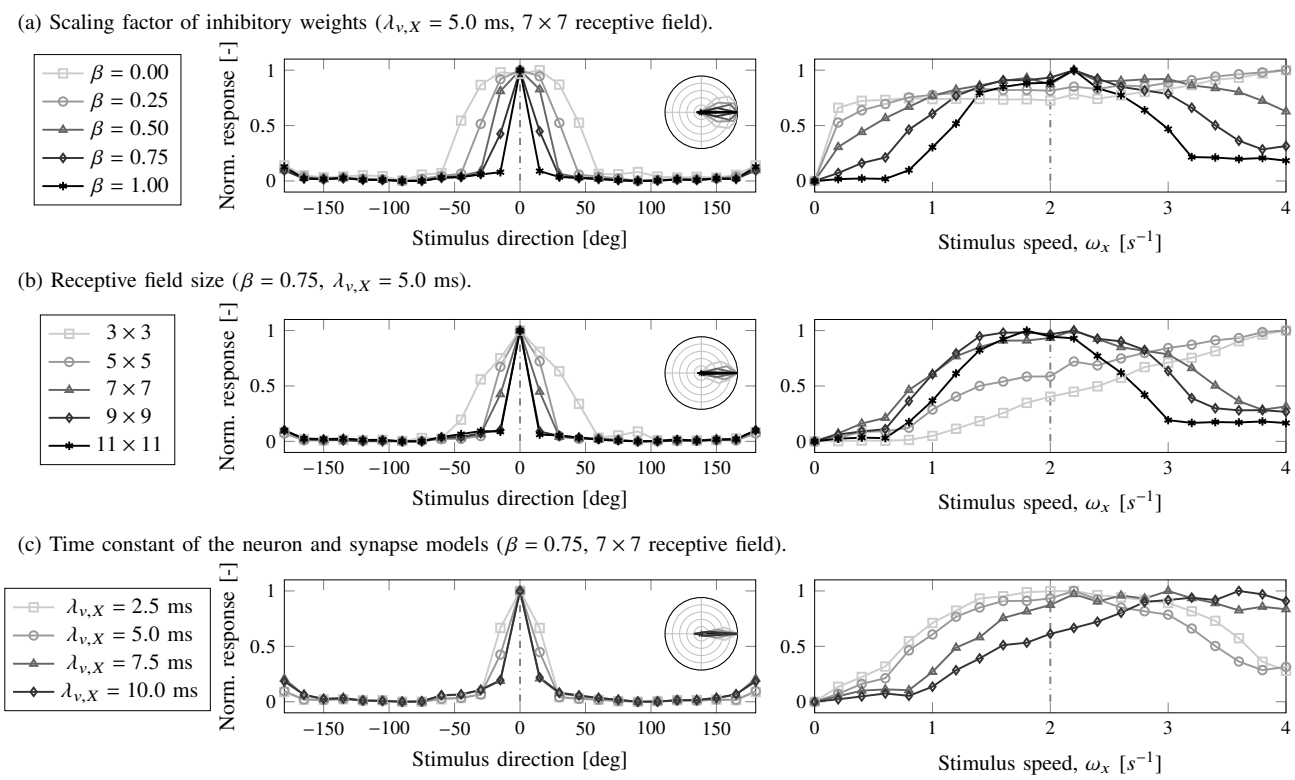


Fig. 15: Sensitivity analysis of the direction and speed selectivity of neurons in the MS-Conv layer using β, the receptive field size, and the time constant of the neuron and synapse models $\lambda_{v,X}$ as dependent variables. In both cases, strong selectivity is understood as the neural response being maximum for the training configuration, and close to zero for others. The dashed lines indicate the direction and speed of the visual stimulus used for training the spatiotemporal kernel under evaluation. The checkerboard pattern was used as input texture. Each response plot is normalized by its maximum value.

From Fig. 13, remarkable is the fact that two of the (generally) four kernels specialized to each of the aforementioned motion directions have overlapping neural responses despite the WTA mechanism. This is indicative of, first, a larger than sufficient number of kernels for this input dataset; and second and consequently, a speed selectivity that can be preliminarily assessed as relatively weak. To confirm this, Fig. 15 shows a sensitivity analysis of the direction and speed selectivity of these kernels as a function of β, their spatial dimensions, and the time constant of the neuron and synapse models, assumed equal and denoted by $\lambda_{v,X}$. The firing threshold v_{th} was set to 1.0, and the homeostasis scaling factor to $\alpha = 0.4$. For this evaluation, a single spatiotemporal kernel was trained at $\omega_x = 2.0$ s^{-1} in the checkerboard pattern. Further, we rescinded the use of the SS-Conv layer, by directly attaching MS-Conv to the input, to avoid the need for learning edges at multiple orientations. Rightward stimulus motion in the range $\omega_x \in [0.0, 4.0]$ s^{-1} was employed for the assessment of speed selectivity, while the training speed but at different motion directions was used for the evaluation of direction selectivity. Note that, for the latter, the DVS orientation with respect to the checkerboard texture is adjusted so that the edges perceived are perpendicular with the direction of motion.

With respect to the effect of β, Fig. 15a confirms the need for inhibition in the convolutional kernels. As shown, both direction and speed selectivities increase with this parameter since, as mentioned before, it prevents neurons from firing to spatiotemporal input traces that does not fit completely $\boldsymbol{W}_k^{l,\text{exc}}$. If not employed, one would have to rely on sufficiently large firing thresholds specifically designed for each kernel [97], which is inconsistent with the learning process proposed in the present work. Closely related to this, Fig. 15b shows that the receptive field size also has a profound impact on neural selectivity. The input feature whose motion is to be tracked needs to be discernible in the receptive field; and moreover, the size of the field has to be sufficiently large so that inhibitory weights contribute to the response. Therefore, for a given set of neural parameters, the larger the receptive field, the stronger the direction and speed selectivities of these kernels. Lastly, Fig. 15c shows that, despite a negligible effect on direction selectivity, speed selectivity diminishes with the time constant of the neuron and synapse models. The higher the value of this parameter, the wider the features learned through STDP; and thus, the lower the robustness of the convolutional kernels for a given receptive field size. Overall, these spatiotemporal kernels are characterized by a strong direction and a considerably weaker speed selectivities, as derived from Fig. 13.

D. Pooling Layer: From Local to Global

As an intermediate stage between the MS-Conv and Dense layers, the Pooling layer is employed in the SNN architecture as a means to reduce the spatial dimensionality of the former, and hence to facilitate the learning of the latter. The intuition of this layer is that, by pooling local motion estimates over large portions of the visual scene, a more accurate measure of the global motion in each of these regions can be obtained, thus mitigating the effect of the aperture problem [78].

Similarly to SS-Conv, the Pooling layer is convolutional and single-synaptic. The internal dynamics of its neurons is driven by the forcing function described in Eq. (15), but without the need for $\boldsymbol{N}_{i,k}$ since presynaptic connections are not plastic. Instead, this layer is characterized by the same number of neural maps as the MS-Conv, each one assigned with an excitatory kernel $\boldsymbol{W}_k^l$ that has unitary weights with its presynaptic counterpart and null with the rest. In addition, the application of the WTA inhibitory mechanism is neglected, and there is no overlap between receptive fields. As shown in Fig. 7, we used 8×8 convolutional kernels in this layer, thus resulting in postsynaptic neural maps with 4×4 cells each. Further, neural parameters were empirically set to $v_{\text{th}} = 0.5$, $\lambda_v = \lambda_X = 5.0$ ms, $\alpha = 0.15$, and $\tau = 1.0$ ms.

E. Dense Layer: Global Motion Perception

The Dense layer, as the final stage of the SNN architecture, is comprised of individual neurons fully connected to cells in the Pooling layer via single-synaptic plastic connections. Similarly to final regions of biological visual motion systems [3], neurons in this layer develop selectivity to the global motion of the scene from visual experience through STDP.

With respect to implementation details, synaptic plasticity is conducted as described in Section IV-B, and the forcing function of Dense neurons resembles Eq. (11), but referring to the convolutional presynaptic layer to which these cells are connected. This expression is then defined as:

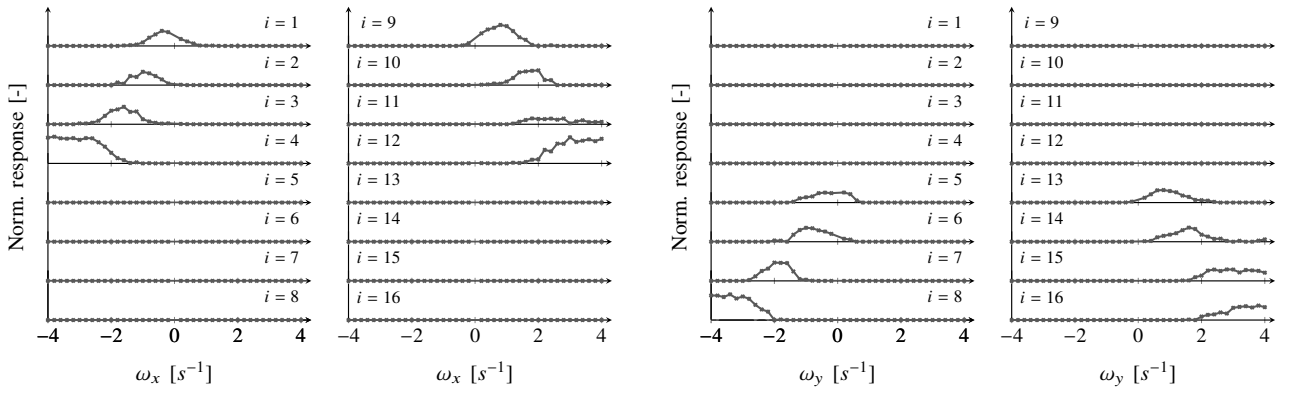
$$i_i^l(t) = \sum_{ch=1}^{f^{l-1}} \sum_{j=1}^{n^{l-1}} \left(W_{i,j,ch}^l s_{j,ch}^{l-1}(t-\tau) - X_{i,j,ch}^l(t) \right) \tag{17}$$

where the efficacy and trace of presynaptic connections are defined as $\boldsymbol{W}_i^l \in \mathbb{R}^{n^{l-1} \times f^{l-1}}$ and $\boldsymbol{X}_i^l \in \mathbb{R}^{n^{l-1} \times f^{l-1}}$, respectively.

1) Evaluation: To evaluate the performance, a Dense layer with sixteen individual neurons was trained as the final stage of a neural architecture comprised of the SS-Conv in Fig. 10 for feature extraction, the MS-Conv in Fig. 13 for local motion perception, and a Pooling layer for spatial dimensionality reduction. Consistently, the checkerboard dataset was employed as source of input stimuli; and neural parameters were empirically set to $v_{\text{th}} = 0.5$, $\lambda_v = \lambda_X = 5.0$ ms, $\alpha = 0.1$, and $\tau = 1.0$ ms. Fig. 16 shows the neural response (after convergence) of cells in this layer as a function of ω_x and ω_y.

From this figure, neurons in the Dense layer are successful at learning the spatial distribution of local motion estimates that corresponds to each of the different motion types extracted by the MS-Conv, and further encoded in the neural activity of the Pooling layer. Consequently, these cells are said to be selective to the global motion of the stimuli. Out of the sixteen neurons trained, groups of four specialized to each motion direction, with different tuning speeds so as to cover the ventral flow range of the dataset, i.e. $\{|\omega_x|, |\omega_y|\} \in [0.2, 4.0]\ s^{-1}$. Note that the direction and speed selectivity of these neurons is exclusively dependent on those of the spatiotemporal kernels from the MS-Conv layer, as shown in Fig. 15.

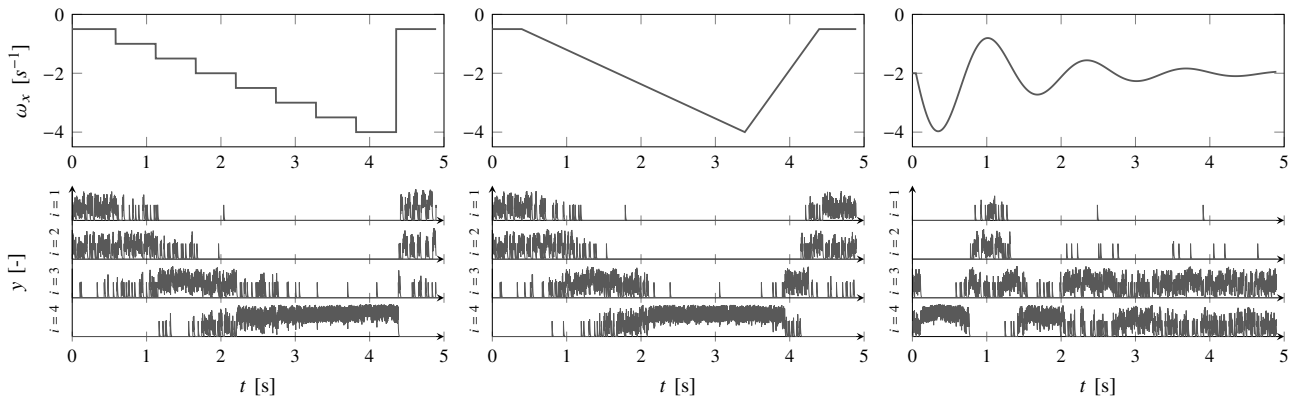
In addition to this evaluation, Fig. 17 is shown to assess the activity of neurons from this layer in response to speed profiles that differ from the constant-speed sequences employed during training. Specifically, three different profiles of leftward horizontal motion were used in combination with the checkerboard texture. These sequences were characterized respectively by, first, constant-speed segments and step changes (see Fig. 17a);



(a) Pure horizontal motion.

(b) Pure vertical motion.

Fig. 16: Neural response of the sixteen individual neurons from the Dense layer trained in the checkerboard dataset. Response plots are normalized by the maximum neural response on the stimuli evaluated: 0.3 spikes/ms by $i = 4$ for $\omega_x = -3.8\ s^{-1}$.



(a) Constant speed. (b) Constant acceleration. (c) Fluctuating acceleration.

Fig. 17: Temporal course of the postsynaptic trace of neurons i = 1–4 from the Dense layer (bottom, see Fig. 16) in response to leftward input stimuli with different speed profiles (top) and from the checkerboard texture. Plots are normalized by the maximum trace on the stimuli evaluated: 0.4098 by $i = 4$ at $t = 0.36$ s for the fluctuating acceleration case.

second, constant accelerations (see Fig. 17b); and third, a fluctuating acceleration (see Fig. 17c). Due to the pure leftward motion of the stimuli, only the activity of neurons specialized to this motion direction is shown, i.e. cells i = 1–4 from Fig. 16. Neural activity is measured through the postsynaptic trace $y_i(t)$ of these units, which, similarly to Eq. (9), keeps track of the recent history of postsynaptic spikes emitted by a particular neuron. The dynamics of this parameter is given by:

$$\lambda_y \frac{dy_i(t)}{dt} = -y_i(t) + s_i(t) \tag{18}$$

where $\lambda_y = 5.0$ ms to maintain consistency with the rest of the neural architecture.

Results in Fig. 17 confirm that, even though the network was trained using constant-speed stimuli, the spiking activity of neurons in the Dense layer can still be seen as indicative of the global motion of the visual scene in accelerated, or with abrupt speed variations, sequences. Response aspects, such as the overlap of neural activity for some ventral flow ranges, or the dominance of $i = 4$ for fast leftward motion, are ratified by the selectivity of these neurons, as shown in Fig. 16a.

VI. Estimation of Visual Observables from Spike-Based Optical Flow

As derived from the evaluation results in Section V-E, the global motion of the visual scene can be extracted from the spiking activity of the Pooling layer (see Section V-D) by training a set of postsynaptic individual neurons, from the so-called Dense layer, with STDP. As a result, after convergence, each of these cells is selective to a global motion pattern in a particular direction and speed. As a subsequent step, a reinforcement-based spiking learning mechanism [72], [73] could be used to exploit this information for biologically plausible optical-flow-based control, such as the landing of a flying robot [6], [21], [101], [102]. We consider such a setup beyond the scope of the current research, and instead investigate how, alternatively to the Dense layer, optical flow visual observables could be estimated through a readout mechanism based on a shallow ANN trained in a supervised fashion. Specifically, the visual observables to be estimated are the ventral flow components ω_x and ω_y, which, as described in Section III, are also a measure of the global motion of the stimuli. The neural architecture of the mechanism is defined in Section VI-A, after which Section VI-B includes training details and experimental results for circular maneuvers at constant altitude.

A. Readout Mechanism

The readout mechanism that we propose in this paper is a Multi-Layer Perceptron (MLP) network [34] that maps the neural activity of the Pooling layer into the aforementioned optical flow visual observables. Specifically, the architecture is comprised of three hidden layers of 256, 128, and 32 Rectified Linear Units (ReLUs) [109] respectively, and an output layer with two linear neurons for the ventral flow components ω_x and ω_y. With respect to the input data, individual snapshots of the postsynaptic trace $y_i(t)$ of neurons in the Pooling layer are employed. Note that trace information is normalized to the maximum value of each snapshot.

B. Evaluation

For the performance assessment of the proposed MLP in the task of ventral flows estimation, the SNN trained in the checkerboard dataset (see Figs. 10 and 13) was used as back-end network. Consistently, the visual stimuli employed in this evaluation was also generated by moving the simulated DVS [47] at a constant altitude of $Z_{\mathcal{W}} = 0.5$ m with respect to a (virtual) checkerboard surface, towards which it is facing with $\phi = \theta = 0.0$ deg. Further, we fixed sensor orientation with respect to the scene so that only vertical and horizontal edges were perceivable.

Two datasets were generated for this analysis. On the one hand, a training set comprised of approximately 22 000 snapshots was created by moving the DVS in straight trajectories of different motion directions and constant speeds, with $\{|\omega_x|, |\omega_y|\} \in [0.2, 4.5]$ s^{-1}. The MLP was trained on this data using the Adam optimization algorithm [110] with a constant learning rate of 1.0×10^{-4} and a batch size of 128 samples. On the other hand, the spiking activity of the Pooling layer was recorded throughout eight counterclockwise circular trajectories, of radii $r \in [0.25, 2.00]$ m and 6.0 s of duration, for the evaluation of the inference capabilities of the readout architecture. This dataset is further referred to as the test set, and its interests derives from a great number of unseen (ω_x, ω_y) combinations, the presence of sign changes, and a maximum ventral flow of $|\omega_x| = |\omega_y| = 4.2$ s^{-1}. Note that, for both datasets, ground truth measurements were obtained through the optical flow formulation introduced in Section III.

Fig. 18 shows the resulting ventral flow estimates from the evaluation of the readout mechanism on the test dataset after convergence. From this figure, the proposed MLP is successful at mapping the spiking normal flow estimates to the desired ventral flow components. Despite of this, its performance is characterized for being relatively noisy, due to the spiking nature of the data, and sensitive to the sign changes of ω_x and ω_y, since no MS-Conv kernel of the back-end SNN is selective to variations of motion direction. Additionally, from Fig. 18c, the performance of the mechanism is limited for high ventral flow values. Fig. 19 and Table I are included for the understanding of the error variations with the magnitude of these parameters. The absolute error and error variance were the metrics employed in this evaluation.

Apart from reflecting the aforementioned performance limitations, these results show that, first, the absolute error distribution is approximately uniform on $\{|\omega_x|, |\omega_y|\} < 2.5$ s^{-1}; and second, this error increases significantly for ventral flow components above this boundary value. Since the readout mechanism was trained on input stimuli with speeds of up to 4.5 s^{-1}, this poor performance is due to the SNN and the relatively weak speed selectivity of MS-Conv convolutional kernels. As shown in Fig. 13, for $\{|\omega_x|, |\omega_y|\} > 2.5$ s^{-1} (approximately), there is a consistent overlap between the neural responses of two kernels per perceivable (normal) flow direction, and consequently, the discernibility of different speeds is restricted. For a more accurate performance in this range, the checkerboard dataset used to train the SNN needs to be extended with higher ventral flow values, and the MS-Conv layer with additional kernels. Note that increasing the number of kernels without varying the training dataset would results in a further response overlap, and hence, no positive impact on the readout performance.

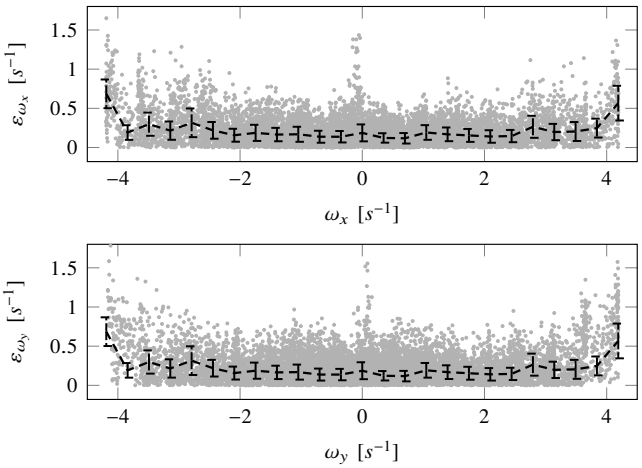
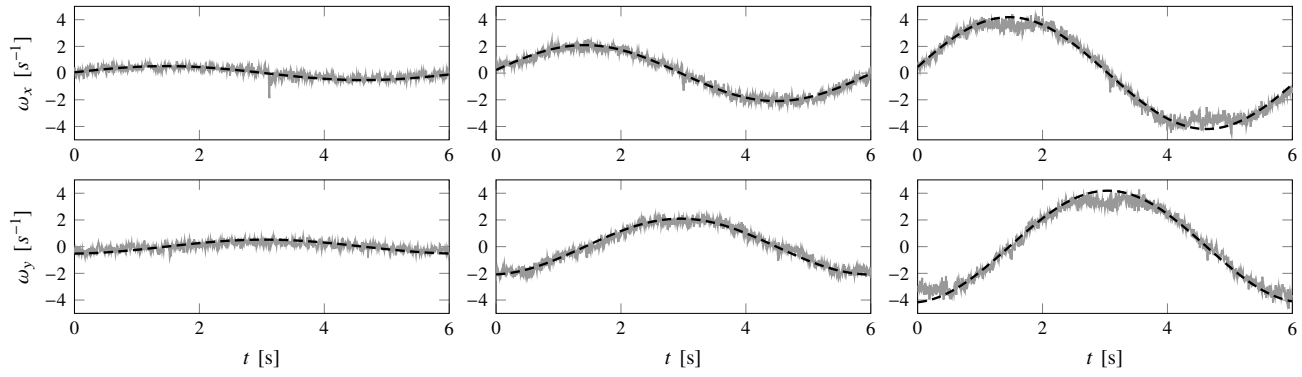


Fig. 19: Error distribution of the readout MLP architecture on the test set. From top to bottom: Distribution of the absolute errors of ω_x and ω_y, and distribution of the normalized error of the mean absolute ventral flow $\hat{\omega}_{x,y}$. Measurements are depicted as gray dots, while the dashed black lines show the 25%, 50%, and 75% percentiles of the data.



(a) $r = 0.25$ m (b) $r = 1.00$ m (c) $r = 2.00$ m

Fig. 18: Temporal evolution of the ventral flow components estimated with the readout MLP architecture on three of the circular trajectories comprising the test set. Ground truth ventral flow measurements are shown through dashed black lines.

TABLE I
MEAN ABSOLUTE ERROR AND ERROR VARIANCE OF THE READOUT ARCHITECTURE ON THE TEST SET. EACH CIRCULAR TRAJECTORY IS CHARACTERIZED BY ITS RADIUS AND MAXIMUM VENTRAL FLOW. VALUES HIGHLIGHTED IN BOLD CORRESPOND TO THE LOWEST OF EACH COLUMN.

r [m]	$\max(\omega)$ $[s^{-1}]$	ε_{ω_x} $[s^{-1}]$		ε_{ω_y} $[s^{-1}]$	
		Mean	Var	Mean	Var
0.25	0.5237	**0.1548**	**0.0190**	**0.1972**	**0.0182**
0.50	1.0474	0.1954	0.0244	0.2106	0.0239
0.75	1.5711	0.1890	0.0234	0.2081	0.0245
1.00	2.0947	0.1808	0.0209	0.2135	0.0273
1.25	2.6184	0.1940	0.0278	0.2238	0.0309
1.50	3.1421	0.2292	0.0339	0.2315	0.0338
1.75	3.6658	0.2646	0.0515	0.2767	0.0573
2.00	4.1895	0.3099	0.0722	0.3506	0.0902

VII. CONCLUSION

In this paper, we present a biologically plausible SNN architecture in which, similarly to biological visual systems, motion selectivity emerges through STDP from the stimuli generated by an event-based vision sensor. Three main contributions lead to these results.

First, a novel adaptive spiking neuron model is introduced as the formulation governing the internal dynamics of neurons in the SNN. The model is suitable for the rapidly varying input statistics of event-based sensors, since neural excitability is regulated using local presynaptic, rather than postsynaptic, information. Specifically, the adaptation is based on the synaptic trace of input connections. The evaluation conducted with the checkerboard texture confirms the model's ability to maintain a certain level of activation regardless of the sensor speed.

Second, the training of the SNN architecture is performed through a novel STDP implementation. Contrary to conventional formulations of this bio-inspired protocol, the learning rule proposed in this work is inherently stable as result of the linear combination of non-exclusive strengthening and weakening processes. In both cases, there is an exponential dependency on the weights and traces of presynaptic connections. Besides STDP, local inter-lateral competition is included through an inhibitory WTA mechanism that enables neurons to specialize to different input features.

Third, the proposed SNN consists mainly of four layers of distinct functionality. In hierarchical order, from shallow to deep, feature extraction is conducted in the SS-Conv, a single-synaptic convolutional layer. Subsequently, the local motion of these features is identified through a multi-synaptic convolutional layer with different transmission delays, and characterized by spatiotemporal kernels with excitatory and inhibitory contributions. This layer is referred to as the MS-Conv, and its neural response is pooled over large receptive fields in the Pooling layer to mitigate the effect of the aperture problem. Lastly, neurons in the fully-connected Dense layer are sensitive to the global motion of the visual scene. The performance of these layers is confirmed from evaluations on a checkerboard dataset comprised exclusively of pure horizontal and vertical image motion.

In addition, using the SNN as a back-end architecture, a readout mechanism consisting of a shallow ANN trained in a supervised fashion is employed to estimate ventral flow information. Inference results from the evaluation of this mechanism on circular trajectories show that, despite the spiking nature of input data, accurate estimates of the aforementioned optical flow visual observables are obtained.

This work lays the foundations for the application of SNNs in robotics for an efficient, biologically-plausible ego-motion estimation. Future research in this field should, firstly, relax the assumption of non-rotational motion at a constant altitude employed throughout this paper. This would lead to the identification of a more diverse set of spatial and spatiotemporal features, and to the additional emergence of neural selectivity to vertical and rotational motion in the three-dimensional space. Secondly, instead of relying on different constant transmission delays in the MS-Conv layer, a learning rule should be developed for the adaptation of these parameters. The end goal of these temporal adjustments should be the same spatial resolution for all the spatiotemporal convolutional kernels of this layer regardless of their tuning speed. Thirdly, to fully exploit the benefits of the proposed STDP implementation, the assumption of distinct training and inference phases should be relaxed. A reformulation of the WTA competition mechanism for convolutional layers, and the addition of a forgetting factor in the learning rule are potentially useful lines of work towards this purpose. Lastly, for onboard applications, effort should be made to implement the SNN architecture in neuromorphic hardware.

REFERENCES

[1] J. Feng, *Computational neuroscience: A comprehensive approach*. CRC press, 2003.

[2] A. Borst, J. Haag, and D. F. Reiff, "Fly motion vision," *Annual Review of Neuroscience*, vol. 33, pp. 49–70, 2010.

[3] A. Borst and M. Helmstaedter, "Common circuit design in fly and mammalian motion vision," *Nature Neuroscience*, vol. 18, no. 8, pp. 1067–1076, 2015.

[4] J. J. Gibson, "The perception of the visual world." 1950.

[5] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 208, no. 1173, pp. 385–397, 1980.

[6] G. C. H. E. de Croon, H. Ho, C. De Wagter, E. Van Kampen, B. Remes, and Q. P. Chu, "Optic-flow based slope estimation for autonomous landing," *International Journal of Micro Air Vehicles*, vol. 5, no. 4, pp. 287–297, 2013.

[7] M. V. Srinivasan, S. Zhang, M. Lehrer, and T. Collett, "Honeybee navigation en route to the goal: Visual flight control and odometry," *Journal of Experimental Biology*, vol. 199, no. 1, pp. 237–244, 1996.

[8] J. S. Chahl, M. V. Srinivasan, and S. W. Zhang, "Landing strategies in honeybees and applications to uninhabited airborne vehicles," *The International Journal of Robotics Research*, vol. 23, no. 2, pp. 101–110, 2004.

[9] E. Baird, N. Boeddeker, M. R. Ibbotson, and M. V. Srinivasan, "A universal strategy for visually guided landing," *Proceedings of the National Academy of Sciences*, vol. 110, no. 46, pp. 18 686–18 691, 2013.

[10] C. De Wagter, S. Tijmons, B. D. W. Remes, and G. C. H. E. de Croon, "Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system," in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 4982–4987.

[11] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: A survey," *Computer Vision and Image Understanding*, vol. 134, pp. 1–21, 2015.

[12] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.

[13] T. N. Wiesel, "The postnatal development of the visual cortex and the influence of environment," *Bioscience Reports*, vol. 2, no. 6, pp. 351–377, 1982.

[14] A. Kirkwood and M. F. Bear, "Hebbian synapses in visual cortex," *Journal of Neuroscience*, vol. 14, no. 3, pp. 1634–1645, 1994.

[15] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 µs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[16] 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, 2011.

[17] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck, "A 240x180 130 dB 3 µs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[18] J. Conradt, R. Berner, M. Cook, and T. Delbruck, "An embedded AER dynamic vision sensor for low-latency pole balancing," in *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops*, 2009, pp. 780–785.

[19] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, pp. 1–7, 2013.

[20] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2761–2768.

[21] B. J. P. Hordijk, K. Y. W. Scheper, and G. C. H. E. de Croon, "Vertical landing for micro air vehicles using event-based optical flow," *Journal of Field Robotics*, vol. 35, no. 1, pp. 69–90, 2018.

[22] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[23] M. A. Sutton, W. J. Wolters, W. H. Peters, W. F. Ranson, and S. R. McNeill, "Determination of displacements using an improved digital correlation method," *Image and Vision Computing*, vol. 1, no. 3, pp. 133–139, 1983.

[24] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America*, vol. 2, no. 2, pp. 284–299, 1985.

[25] D. J. Heeger, "Model for the extraction of image flow," *Journal of the Optical Society of America*, vol. 4, no. 8, pp. 1455–1471, 1987.

[26] 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, 2014.

[27] T. Brosch and H. Neumann, "Computing with a canonical neural circuits model with pool normalization and modulating feedback," *Neural Computation*, vol. 26, pp. 2735–2789, 2014.

[28] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[29] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, pp. 1647–1655.

[30] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, pp. 2720–2729.

[31] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised deep learning for optical flow estimation," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 1495–1501.

[32] S. Zhao, X. Li, and O. E. F. Bourahla, "Deep optical flow estimation via multi-scale correspondence structure learning," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 3490–3496.

[33] W. S. Lai, J. B. Huang, and M. H. Yang, "Semi-supervised learning for optical flow with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 353–363.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[35] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[36] G. Orchard and R. Etienne-Cummings, "Bioinspired visual motion estimation," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1520–1536, 2014.

[37] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[38] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[39] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.

[40] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs," *Science*, vol. 275, no. 5297, pp. 213–215, 1997.

[41] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *Public Library of Science: Computational Biology*, vol. 3, no. 2, pp. 247–257, 2007.

[42] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora, "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, pp. 1–11, 2015.

[43] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, pp. 1–9, 2015.

[44] A. Tavanaei and A. S. Maida, "Multi-layer unsupervised learning in a spiking convolutional neural network," in *Proceedings of the 2017 IEEE International Joint Conference on Neural Networks*, 2017, pp. 2023–2030.

[45] R. B. Stein, "A theoretical analysis of neuronal variability," *Biophysical Journal*, vol. 5, no. 2, pp. 173–194, 1965.

[46] D. D. Cho and T. Lee, "A review of bioinspired vision sensors and their applications," *Sensors and Materials*, vol. 27, no. 6, pp. 447–463, 2015.

[47] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[48] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 25–71, 1952.

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

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

[51] M. Baudry, "Synaptic plasticity and learning and memory: 15 years of progress," *Neurobiology of Learning and Memory*, vol. 70, no. 1, pp. 113–118, 1998.

[52] K. Doya, "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?" *Neural Networks*, vol. 12, no. 7-8, pp. 961–974, 1999.

[53] D. O. Hebb, *The organisation of behaviour: A neuropsychological theory*. Wiley, 1952.

[54] H. Markram, W. Gerstner, and P. J. Sjöström, "Spike-timing-dependent plasticity: A comprehensive overview," *Frontiers in Synaptic Neuroscience*, vol. 4, pp. 1–3, 2012.

[55] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press, 2002.

[56] J. Sjöström and W. Gerstner, "Spike-timing dependent plasticity," *Frontiers in Neuroscience*, pp. 35–44, 2010.

[57] M. C. W. Van Rossum, G. Q. Bi, and G. G. Turrigiano, "Stable Hebbian learning from spike timing-dependent plasticity," *Journal of Neuroscience*, vol. 20, no. 23, pp. 8812–8821, 2000.

[58] A. Shrestha, K. Ahmed, Y. Wang, and Q. Qiu, "Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning," in *Proceedings of the 2017 International Joint Conference on Neural Networks*. IEEE, 2017, pp. 1999–2006.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, no. 3, pp. 533–536, 1988.

[60] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1, pp. 17–37, 2002.

[61] O. Booij and H. T. Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Information Processing Letters*, vol. 95, no. 6, pp. 552–558, 2005.

[62] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection," *Neural Networks*, vol. 22, no. 10, pp. 1419–1431, 2009.

[63] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2018.

[64] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2706–2719, 2013.

[65] D. Zambrano and S. M. Bohte, "Fast and efficient asynchronous neural computation with adapting spiking neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02053

[66] D. Zambrano, R. Nusselder, H. S. Scholte, and S. M. Bohte, "Efficient computation in adaptive artificial spiking neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1710.04838

[67] A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature Neuroscience*, vol. 2, no. 11, pp. 947–957, 1999.

[68] R. Van Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex," *Neural Computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[69] R. P. N. Rao and T. J. Sejnowski, "Spike-timing-dependent hebbian plasticity as temporal difference learning," *Neural Computation*, vol. 13, no. 10, pp. 2221–2237, 2001.

[70] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, 2007.

[71] E. M. Izhikevich, "Solving the distal reward problem through linkage of STDP and dopamine signaling," *Cerebral Cortex*, vol. 17, no. 10, pp. 2443–2452, 2007.

[72] J. O. Rombouts, P. R. Roelfsema, and S. M. Bohte, "Neurally plausible reinforcement learning of working memory tasks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1871–1879.

[73] J. O. Rombouts, A. van Ooyen, P. R. Roelfsema, and S. M. Bohte, "Biologically plausible multi-dimensional reinforcement learning in neural networks," in *International Conference on Artificial Neural Networks*, 2012, pp. 443–450.

[74] E. Vasilaki, N. Frémaux, R. Urbanczik, W. Senn, and W. Gerstner, "Spike-based reinforcement learning in continuous state and action space: When policy gradient methods fail," *Public Library of Science: Computational Biology*, vol. 5, no. 12, pp. 1–17, 2009.

[75] J. Friedrich and M. Lengyel, "Goal-directed decision making with spiking neurons," *Journal of Neuroscience*, vol. 36, no. 5, pp. 1529–1546, 2016.

[76] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Combining STDP and reward-modulated STDP in deep convolutional spiking neural networks for digit recognition," 2018. [Online]. Available: https://arxiv.org/abs/1804.00227

[77] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys (CSUR)*, vol. 27, no. 3, pp. 433–466, 1995.

[78] S. Ullman, *The interpretation of visual motion.* MIT Press, 1979.

[79] B. D. Lucas and T. Kanade, "An iterative technique of image registration and its application to stereo," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, vol. 2, 1981, pp. 674–679.

[80] H. H. Nagel, "Extending the oriented smoothness constraint into the temporal domain and the estimation of derivatives of optical flow," in *European Conference on Computer Vision*. Springer, 1990, pp. 139–148.

[81] T. Camus, "Real-time quantized optical flow," *Real-Time Imaging*, vol. 3, no. 2, pp. 71–86, 1997.

[82] N. M. Grzywacz and A. L. Yuille, "A model for the estimate of local image velocity by cells in the visual cortex," *Proceedings of the Royal Society of London: Biological Sciences*, vol. 239, no. 1295, pp. 129–161, 1990.

[83] E. P. Simoncelli and D. J. Heeger, "A model of neuronal responses in visual area MT," *Vision Research*, vol. 38, no. 5, pp. 743–761, 1998.

[84] N. C. Rust, V. Mante, E. P. Simoncelli, and J. A. Movshon, "How MT cells analyze the motion of visual patterns," *Nature Neuroscience*, vol. 9, no. 11, pp. 1421–1431, 2006.

[85] A. Borst and T. Euler, "Seeing things in motion: Models, circuits, and mechanisms," *Neuron*, vol. 71, no. 6, pp. 974–994, 2011.

[86] W. Reichardt, "Autocorrelation, a principle for the evaluation of sensory information by the central nervous system," *Sensory Communication*, pp. 303–317, 1961.

[87] A. Borst, "Fly vision: Moving into the motion detection circuit," *Current Biology*, vol. 21, no. 24, pp. R990–R992, 2011.

[88] Y. LeCun, "Generalization and network design strategies," *Connectionism in Perspective*, pp. 143–155, 1989.

[89] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2528–2535.

[90] R. Benosman, S. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Networks*, vol. 27, pp. 32–37, 2012.

[91] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Frontiers in Neuroscience*, vol. 9, pp. 1–15, 2015.

[92] S. Tschechne, R. Sailer, and H. Neumann, "Bio-inspired optic flow from event-based neuromorphic sensor input." in *Proceedings of the 6th IAPR Workshop on Artificial Neural Networks in Pattern Recogniftion*. Springer, 2014, pp. 171–182.

[93] L. I. Abdul-Kreem and H. Neumann, "Neural mechanisms of cortical motion computation based on a neuromorphic sensory system," *Public Library of Science*, vol. 10, no. 11, pp. 1–33, 2015.

[94] T. Brosch and H. Neumann, "Event-based optical flow on neuromorphic hardware," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, 2016, pp. 551–558.

[95] M. Giulioni, X. Lagorce, F. Galluppi, and R. B. Benosman, "Event-based computation of motion flow on a neuromorphic analog neural platform," *Frontiers in Neuroscience*, vol. 10, pp. 1–13, 2016.

[96] C. Richter, F. Röhrbein, and J. Conradt, "Bio-inspired optic flow detection using neuromorphic hardware," *Bernstein Conference on Computational Neuroscience*, 2014, poster.

[97] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor, "A spiking neural network architecture for visual motion estimation," in *Proceedings of the 2013 IEEE Biomedical Circuits and Systems Conference*, 2013, pp. 298–301.

[98] A. P. Shon, R. P. Rao, and T. J. Sejnowski, "Motion detection and prediction through spike-timing dependent plasticity," *Network: Computation in Neural Systems*, vol. 15, no. 3, pp. 179–198, 2004.

[99] O. G. Wenisch, J. Noll, and J. L. Van Hemmen, "Spontaneously emerging direction selectivity maps in visual cortex through STDP," *Biological Cybernetics*, vol. 93, no. 4, pp. 239–247, 2005.

[100] S. V. Adams and C. M. Harris, "A computational model of innate directional selectivity refined by visual experience," *Scientific Reports*, vol. 5, pp. 1–13, 2015.

[101] H. W. Ho and G. C. H. E. de Croon, "Characterization of flow field divergence for MAVs vertical control landing," in *AIAA Guidance, Navigation, and Control Conference*, 2016, pp. 1–13.

[102] H. Ho, C. De Wagter, B. D. W. Remes, and G. C. H. E. de Croon, "Optical-flow based self-supervised learning of obstacle appearance applied to MAV landing," *Robotics and Autonomous Systems*, vol. 100, pp. 78–94, 2018.

[103] A. Morrison, A. Aertsen, and M. Diesmann, "Spike-timing-dependent plasticity in balanced random networks," *Neural Computation*, vol. 19, no. 6, pp. 1437–1467, 2007.

[104] M. Abercrombie, C. J. Hickman, and M. L. Johnson, *A dictionary of biology*. Routledge, 2017.
[105] S. M. Bohte, "Efficient spike-coding with multiplicative adaptation in a spike response model," in *Advances in Neural Information Processing Systems*, 2012, pp. 1835–1843.
[106] S. J. Thorpe, "Spike arrival times: A highly efficient coding scheme for neural networks," *Parallel Processing in Neural Systems*, pp. 91–94, 1990.
[107] D. Luebke, "CUDA: Scalable parallel programming for high-performance scientific computing," in *Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2008, pp. 836–838.
[108] H. B. Barlow and W. R. Levick, "The mechanism of directionally selective units in rabbit's retina." *The Journal of Physiology*, vol. 178, no. 3, pp. 477–504, 1965.
[109] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.
[110] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

Part II

Literature Study

Chapter 2

Vision-based Navigation Strategies for MAVs using Optical Flow

The concept of optical flow refers to the projection of the apparent motion[1] of objects in a scene onto the image plane of a visual sensor. Inspired by biological navigation models, the perceived optical flow can be used for ego-motion estimation and state reconstruction of Micro Air Vehicles (MAVs) in environments where engineered estimators, such as the Global Positioning System (GPS), are not available. This chapter serves as an introduction to the field of optical flow and its applications for vision-based navigation. First, Section 2-1 presents the mathematical foundations of the optical flow concepts used for motion estimation. Second, Section 2-2 introduces the working principles of the different classes of optical flow estimation methods using conventional cameras. Third, biologically-inspired applications of these techniques in MAVs are discussed in Section 2-3; and finally, other vision-based navigation strategies are described in Section 2-4.

2-1 Modeling optical flow

For the description of the mathematical model of optical flow, this work employs the formulation of Longuet-Higgins & Prazdny (1980). This frequently-used formulation (e.g., Ho et al., 2016; de Croon, 2016; Ho & de Croon, 2016; Ho et al., 2017) is based on the utilization of the *pinhole camera model* for perspective projection, i.e., for the projection of world points on the image plane. This model is based on the assumptions that, first, the camera aperture[2] is characterized as a point (a pinhole); and second, the retina is treated as a planar surface (the image plane). Therefore, the equations describing the perspective projection become simple expressions when using this model (Scaramuzza & Fraundorfer, 2011). Note that these assumptions are not valid for applications using cameras with wide field of view (e.g., fish-eye lenses). In these cases, a more advanced projection model, such as the omnidirectional or

[1] Relative motion between the elements in a scene and the observer.
[2] The opening in a vision sensor that controls the passage of light.

spherical camera model, has to be applied instead (Geyer & Daniilidis, 2000; Scaramuzza & Fraundorfer, 2011).

2-1-1 Optical flow in the pinhole camera model

Adapted from the optical flow formulation of Longuet-Higgins & Prazdny (1980), Figure 2-1 presents the projection of the world point A, with coordinates (X, Y, Z) in the observer-fixed reference frame $OXYZ$, onto the plane described by the $\hat{x}$- and $\hat{y}$-axis. The origin O denotes the *nodal point* of the observer, which is the location where the aperture of the sensor is defined. The Z-axis is the *optical axis* or line-of-sight. The plane perpendicular to the optical axis is the aforementioned *focal* or *image plane*, defined as the surface in which the image is formed based on the projection of world points. Finally, the intersection between the optical axis and the image plane is the *principal point*, denoted by $o = (0, 0, f)^\top$, where f is the focal length of the sensor. For convenience, $f = 1$ in this derivation.

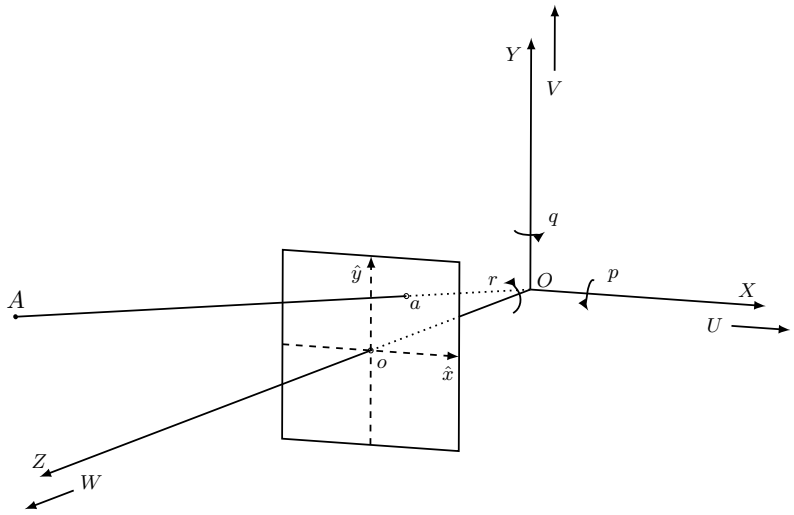


Figure 2-1: Projection of the world point A onto the image plane using the pinhole camera model. Adapted from (Longuet-Higgins & Prazdny, 1980).

Consider the situation in which a monocular observer arbitrarily moves through a static environment. In such a case, the observer-fixed reference frame is subjected to a set of translational velocities (U, V, W) along, and to a set of rotational velocities (p, q, r) around the X-, Y-, and Z-axis respectively. Then, the velocity components of point A with respect to the moving frame are given by:

$$\begin{aligned} \dot{X} &= -U - qZ + rY \\ \dot{Y} &= -V - rX + pZ \\ \dot{Z} &= -W - pY + qX \end{aligned} \tag{2-1}$$

Inferring from Figure 2-1, the projection of the point A on the image plane is denoted by a and defined by the coordinates $(\hat{x}, \hat{y})^\top = (X/Z, Y/Z)^\top$ at this plane. Due to the ego-motion of the observer, this point is characterized by an optical flow with velocity components

$(\hat{u}, \hat{v})^\top = (\dot{\hat{x}}, \dot{\hat{y}})^\top$. Accordingly, $\hat{u}$ and $\hat{v}$ can be related to the velocity of the observer and to the depth of the corresponding world point as:

$$\begin{aligned} \hat{u} &= \dot{X}/Z - X\dot{Z}/Z^2 = (-U/Z - q + r\hat{y}) - \hat{x}\,(-W/Z - p\hat{y} + q\hat{x}) \\ \hat{v} &= \dot{Y}/Z - Y\dot{Z}/Z^2 \;= (-V/Z - r\hat{x} + p) - \hat{y}\,(-W/Z - p\hat{y} + q\hat{x}) \end{aligned} \tag{2-2}$$

These equations may also be written in the form:

$$\begin{aligned} \hat{u} &= \hat{u}^T + \hat{u}^R, & \hat{v} &= \hat{v}^T + \hat{v}^R \\ \hat{u}^T &= (-U + \hat{x}W)/Z, & \hat{v}^T &= (-V + \hat{y}W)/Z \\ \hat{u}^R &= -q + r\hat{y} + p\hat{x}\hat{y} - q\hat{x}^2, & \hat{v}^R &= -r\hat{x} + p + p\hat{y}^2 - q\hat{x}\hat{y} \end{aligned} \tag{2-3}$$

Therefore, the optical flow of a point in the image plane can be resolved into a component due to the translational motion of the observer, and another component due to rotations.

2-1-2 Visual observables derived from optical flow

Once the optical flow components of an image point are related to the ego-motion states of an observer arbitrarily moving through a static environment, the next question to address is how to use this knowledge to retrieve information about this motion and about the structure of the scene. These problems form the basis for the fields of visual odometry (Nistér et al., 2004; Scaramuzza & Fraundorfer, 2011), Simultaneous Localization And Mapping (SLAM) (Davison et al., 2007), and Structure-from-Motion (SfM) (Adiv, 1985). Based on the assumption of a static scene, the ego-motion states (p, q, r, U, V, W) are equal for all world points, while the depth Z varies per each of these points. Hence, one could combine the perceived optical flow in a set of image coordinates to resolve for these unknown parameters. However, the complexity of the solutions to these problems, in combination with the limited computational capabilities of MAVs, result in low-rate navigation strategies when applied in this type of aerial vehicle (e.g., Kendoul et al., 2009; Fraundorfer et al., 2012).

Nevertheless, if a set of simplifying assumptions are used during the estimation of the optical flow field, a new set of parameters regarding the observer's ego-motion can be extracted. These quantities are the so-called *visual observables*. The rest of this section covers the assumptions and the derivation needed for the mathematical description of these parameters.

Derotation

If information on the rotational rates of the observer is available through external sensors (e.g., gyroscopes in the Inertial Measurement Unit, IMU), the rotational component of the perceived optical flow field can be corrected, i.e, the flow can be *derotated*. Since MAVs are frequently equipped with these rotational rate sensors, the derotation of the flow field is a common practice in applications like vision-based navigation (e.g., Izzo & de Croon, 2012; Herissé et al., 2012; Grabe et al., 2015; Ho & de Croon, 2016) or obstacle avoidance (e.g., McGuire et al., 2017).

As shown by Longuet-Higgins & Prazdny (1980), it is possible to extract some basic ego-motion information when the rotational component of the optical flow is negligible but the translational part is still significant (i.e., in pure translational motion or after flow derotation). For this, the intersection of the observer's line of motion with the image plane is defined by the coordinates $(\hat{x}_f, \hat{y}_f)$ as:

$$\hat{x}_f = U/W, \qquad \hat{y}_f = V/W \tag{2-4}$$

Then, assuming flow derotation and substituting Eq. (2-4) into (2-3), the velocity components of the optical flow field can be written in the form:

$$\hat{u} = (\hat{x} - \hat{x}_f)\, W/Z, \qquad \hat{v} = (\hat{y} - \hat{y}_f)\, W/Z \tag{2-5}$$

Inferring from Eq. (2-5), the point $(\hat{x}_f, \hat{y}_f)$ is a particular location of the perceived optical flow under the assumption of derotated flow. This point, acting as the vanishing point of the flow field, is the only location in the image plane where the flow is null independently of the depth of the corresponding world point. Moreover, the magnitude of the flow vectors increases further away from it. For these reasons, this point is referred to as the Focus of Expansion (FoE) in case of positive W, or Focus of Contraction (FoC) otherwise. This visual cue provides the observer with information about the direction of motion. Furthermore, if the depth of the FoE is known, the *time-to-contact* $\tau = Z_{FoE}/W$ is defined as the visual cue that provides information on how fast the observer is approaching this world point.

Planar flow

If the scene, apart from being static, is considered as a planar surface, then a simpler expression can be derived for the perceived optical flow field, as proposed by de Croon et al. (2013). For this derivation, let Z_0 be defined as the distance to the planar surface along the optical axis of the observer, and let Z_X and Z_Y represent the slopes of the scene with respect to the X- and Y-axis of the observer-fixed reference frame. Further, let the observer's normalized velocities be defined as $\vartheta_x = U/Z_0$, $\vartheta_y = V/Z_0$, and $\vartheta_z = W/Z_0$. Then, as shown by de Croon et al. (2013), the velocity components of this *planar flow field* are characterized by the following expressions:

$$\begin{aligned} \hat{u} &= -\vartheta_x + (\vartheta_x Z_X + \vartheta_z)\, \hat{x} + \vartheta_x Z_Y \hat{y} - Z_X \vartheta_z \hat{x}^2 - Z_Y \vartheta_z \hat{x}\hat{y} \\ \hat{v} &= -\vartheta_y + \vartheta_y Z_X \hat{x} + (\vartheta_y Z_Y + \vartheta_z)\, \hat{y} - Z_Y \vartheta_z \hat{y}^2 - Z_X \vartheta_z \hat{x}\hat{y} \end{aligned} \tag{2-6}$$

Note that, if the slopes of the planar scene are considered negligible (i.e., the scene is perpendicular to the optical axis), Eq. (2-6) reduces to a simple expression relating the planar flow field with the observer's normalized velocities:

$$\begin{aligned} \hat{u} &= -\vartheta_x + \vartheta_z \hat{x} \\ \hat{v} &= -\vartheta_y + \vartheta_z \hat{y} \end{aligned} \tag{2-7}$$

Based on Eqs. (2-6) and (2-7), these expressions reveal the importance of the normalized velocities $(\vartheta_x, \vartheta_y, \vartheta_z)$ in the estimation of the optical flow field under the assumption of planar flow. These velocities are the main visual cues, or *visual observables*, used for navigation and for the derivation of other observables, such as the previously introduced time-to-contact, divergence, and ventral flows. Firstly, τ can be obtained from ϑ_z as $\tau = Z_0/W = 1/\vartheta_z$. Secondly, following the formulation proposed by McCarthy et al. (2008), the *flow field divergence*, D, is defined as:

$$D\left(\hat{x}, \hat{y}\right) = \frac{\partial \hat{u}}{\partial \hat{x}}\left(\hat{x}, \hat{y}\right) + \frac{\partial \hat{v}}{\partial \hat{y}}\left(\hat{x}, \hat{y}\right) \tag{2-8}$$

Hence, under the assumption of translational motion towards a planar scene perpendicular to the optical axis of the observer, Eqs. (2-7) and (2-8) lead to a simple definition of D in terms of the normalized velocity ϑ_z:

$$D = 2\vartheta_z = \frac{2}{\tau} \tag{2-9}$$

Finally, the concept of *ventral flow* is introduced for the quantification of the components of the planar flow field that are generated by the observer's ego-motion in the X- and Y- axis. These ventral cues, one per axis, are defined as the opposites of the remaining normalized velocities $(\vartheta_x, \vartheta_y)$, i.e., $\omega_x = -\vartheta_x$ and $\omega_y = -\vartheta_y$.

2-2 Optical flow estimation

At the beginning of this chapter, the concept of optical flow was introduced as the projection of the apparent motion of objects in a scene onto the image plane of a visual sensor. For this motion to be perceptible, a sensor measuring brightness (intensity) at a certain rate is needed so any variation can be detected. These variations, which define the concept of *image motion*, are the basis for the optical flow estimation approaches presented in this section. For this purpose, the most-used visual sensors belong to the category of frame-based cameras, i.e., sensors that output a stream of consecutive images (e.g., Complementary Metal Oxide Semiconductor, CMOS). Besides for navigation with MAVs (e.g., Alkowatly et al., 2014, de Croon, 2016), these devices are widely used in the computer vision field for other tasks such as object recognition, classification, or scene reconstruction (Szeliski, 2010).

As introduced by Horn & Schunck (1981), the estimation of optical flow is based on the assumption that the intensity structure of a local region in the image plane remains approximately constant under motion for a short period of time. Beauchemin & Barron (1995) proved that this hypothesis mathematically leads to the *aperture problem* (Ullman, 1979) through:

$$\nabla I \cdot \hat{\boldsymbol{u}} + I_t = 0 \tag{2-10}$$

where the image intensity function is denoted by $I(\hat{x}, \hat{y}, t)$, its first-order derivatives by $\nabla I = (I_x, I_y)^\top$ and I_t, and the image velocity is defined as $\hat{\boldsymbol{u}} = (\hat{u}, \hat{v})^\top$.

This expression, which is commonly referred to as the *brightness constancy constraint*, implies that only the velocity component normal to the direction of the local gradient ∇I can be estimated. The consequences of this constraint are frequently seen in methods that estimate the optical flow based on a limited pixel neighborhood. In these cases, ambiguity may appear in the perceived flow if the gradient is not aligned with the direction of motion, as exemplified in Figure 2-2.

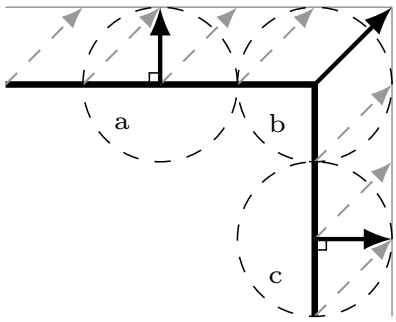


Figure 2-2: Illustration of the aperture problem occurring with an L-shaped object in motion. Black arrows indicate the optical flow component that can be estimated in each of the regions, while gray arrows represent the actual motion of the object.

From this figure, on the one hand, regions `a` and `c` are characterized by a normal flow component that does not accurately represent the motion of the object. Here, optical flow is estimated along an *edge*, whose gradient is dominant in one spatial direction. On the other hand, the motion can be fully estimated in region `b` since it is computed on a *corner*, i.e., a image feature where the gradient is observable in two linearly independent spatial directions. For this reason, some of the methods introduced in this section are preceded by a corner detector such as the popular Harris (Harris & Stephens, 1988) and FAST (Rosten et al., 2010) algorithms.

The rest of this section is structured according to the analysis of optical flow estimation techniques presented by Beauchemin & Barron (1995), in which three major classes are distinguished: gradient-based, correlation-based, and frequency-based methods. Additionally, examples of biologically-inspired motion detectors are also included.

2-2-1 Gradient-based methods

By definition, gradient-based methods make use of the spatiotemporal derivatives of the image intensity function (I_x, I_y, I_t), in conjunction with the brightness constancy constraint, to estimate optical flow. Depending on whether this estimation is performed within small local regions or using intensity information of the full image, these methods can be further divided into *local* and *global* approaches, respectively.

Regarding local techniques, the method proposed by Lucas & Kanade (1981) (further referred to as "Lucas-Kanade") is still the most-used approach for real-time optical flow applications despite its long existence, and it clearly outperforms other classical methods (McCarthy & Bames, 2004). Further, it is the basis from which most of the other techniques of this class are derived (e.g., Simoncelli et al., 1991; Weber & Malik, 1995). Based on Eq. (2-10) and its inability to resolve the optical flow components by itself, Lucas & Kanade (1981) proposed the

assumption that $\hat{u}$ and $\hat{v}$ are constant within the direct neighborhood of a point in the image. Therefore, an overdetermined system of equations can be obtained if $n > 2$ neighboring pixels are used, as shown in Eq. (2-11). Estimates for the optical flow components are obtained through ordinary least-squares.

$$\begin{bmatrix} (I_x)_1 & (I_y)_1 \\ (I_x)_2 & (I_y)_2 \\ \vdots & \vdots \\ (I_x)_n & (I_y)_n \end{bmatrix} \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \begin{bmatrix} (I_t)_1 \\ (I_t)_2 \\ \vdots \\ (I_t)_n \end{bmatrix} \tag{2-11}$$

Global gradient-based methods, instead of using the assumption that the image velocity remains constant within a small neighborhood, combine the brightness constancy constraint with a smoothness condition in order to compute dense image flow over a full image. This technique is referred to as *flow regularization*, and the conditions applied are normally spatial (e.g., Horn & Schunck, 1981; Beauchemin & Barron, 1995), or spatiotemporal (e.g., Nagel, 1990). For instance, Horn & Schunck (1981) proposed an iterative gradient-based method based on the following cost function:

$$\mathcal{L} = \underset{\hat{u},\hat{v}}{\arg\min} \int_{\mathcal{D}} (I_x\hat{u} + I_y\hat{v} + I_t)^2 + \lambda^2 \left(\|\Delta\hat{u}\|^2 + \|\Delta\hat{v}\|^2 \right) \mathrm{d}x \tag{2-12}$$

where λ denotes the influence of a spatial smoothness term, which is defined as the sum of the squared Laplacians of the velocity components, and D is the domain of interest.

Regarding the applications in MAVs, Lucas-Kanade is, by far, the most commonly used gradient-based method for real-time optical flow estimation in these platforms (e.g., Grabe et al., 2015; de Croon, 2016; Ho & de Croon, 2016). Global estimation approaches are rarely used in MAVs since, due to their iterative minimization process, they run significantly slower than local methods to achieve similar levels of accuracy. For this reason, and to the best knowledge of the author, global gradient-based estimation techniques have not been applied in MAVs for real-time optical flow estimation.

2-2-2 Correlation-based methods

For the estimation of optical flow components, correlation-based techniques identify the spatial variations, (d_x, d_y), that best describe the motion of a particular set of image features between two consecutive frames. In essence, the location of a feature in the most recent frame is estimated based on the maximization of a similarity measure, such as the normalized cross-correlation[3]; or based on the minimization of a distance measure, such as the Sum of Squared Difference (SSD) over a spatial window $\mathcal{D}$ centered at the feature. SSD is usually defined as:

$$\mathcal{L} = \underset{d_x,d_y}{\arg\min} \sum_{\hat{x}\in\mathcal{D}} \sum_{\hat{y}\in\mathcal{D}} \left(I\left(\hat{x}, \hat{y}, t-1\right) - I\left(\hat{x} + d_x, \hat{y} + d_y, t\right) \right)^2 \tag{2-13}$$

[3]Popular measure of similarity between the appearance of a feature point in two different views.

The main drawback of traditional correlation-based methods is that the search space $\mathcal{D}$ grows quadratically with the number of possible velocities of the feature being tracked, making the real-time implementation of these techniques impossible. Motivated by this issue, Camus (1997) proposed an alternative formulation that keeps $\mathcal{D}$ constant, and instead performs the region matching step with respect to a set of previous frames, i.e., using a spatiotemporal search space. As a consequence, $\mathcal{D}$ only grows linearly in time. For this reason, the approach proposed by Camus (1997) remains as the basis for the development of new correlation-based techniques for optical flow estimation.

A good example for an implementation of these techniques in MAVs is the vision-based navigation strategy proposed by Kendoul et al. (2009), and subsequently used by Kendoul et al. (2010). Here, instead of making use of the spatiotemporal search region defined by Camus (1997), the location and shape of $\mathcal{D}$ is defined according to inertial data from an IMU, thus allowing real-time implementation.

2-2-3 Frequency-based methods

According to Beauchemin & Barron (1995), this third class of optical flow estimators is based on the use of velocity-tuned filters designed in the Fourier domain, hence the name. Depending on the type of velocity estimated, these methods are further divided into energy-based and phase-based approaches.

Energy-based methods

Based on the working principle of direction-selective cells at early stages of the visual cortex of the human brain (Grzywacz & Yuille, 1990; Simoncelli & Heeger, 1998; Rust et al., 2006; Borst & Euler, 2011), Adelson & Bergen (1985) proposed a class of computational schemes, the energy-based models, that exploits the fact that motion can be estimated by extracting orientation in the spatiotemporal domain. As shown by Figure 2-3, velocity is inverse with the slope of the trace generated in this domain with respect to the temporal axis. Hence, motion can be estimated if a set of spatiotemporally oriented filters are tuned to cover a wide range of orientations and speeds.

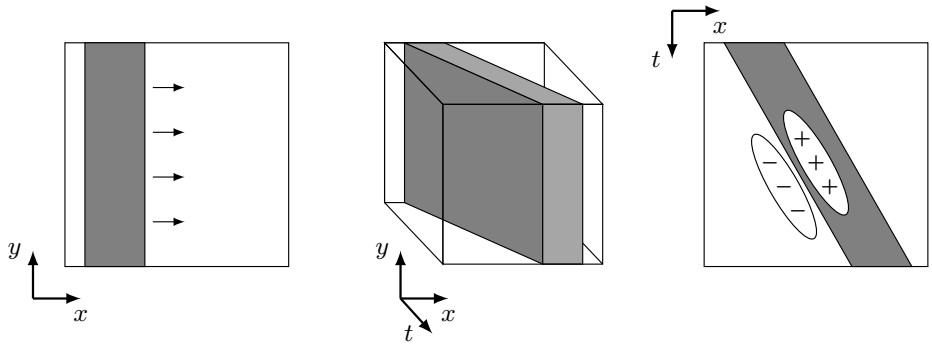


Figure 2-3: Illustration of the horizontal motion of a vertical bar in the spatial and spatiotemporal domains, and the oriented receptive fields used to detect this motion. Adapted from Adelson & Bergen (1985).

There are two main energy-based approaches that normally serve as a basis for the development of new techniques in this field: the works presented by Adelson & Bergen (1985), and by Heeger (1987, 1988). Firstly, in Adelson & Bergen (1985), the authors proposed the idea that a filter function is generated from the linear combination of two spatiotemporally separable filters, each built up as the product of a spatial (even- or odd-symmetric) and a temporal (mono- or bi-phasic) filter. For optical flow computation, each of these filters outputs a *confidence value* for a particular orientation and a specific speed. Secondly, Heeger (1987, 1988) formulated an approach that, based on the filters proposed by Adelson & Bergen (1985), fits spatiotemporal energy to a plane in the frequency space that characterizes the motion.

The major limitation of energy-based approaches is the large number of filters that are required for a robust estimation of motion. Moreover, gradient- and correlation-based methods outperform these techniques in terms of accuracy (Barron et al., 1992). Due to these limitations, and to the best knowledge of the author, there are no applications of energy-based methods in MAVs. However, an accurate and real-time implementation of the spatiotemporal energy model was recently presented by Orchard, Thakor, & Etienne-Cummings (2013). Here, the authors combined a conventional frame-based camera with a Field-Programmable Gate Array (FPGA)[4] for dense optical flow estimation at 60 Hz.

Phase-based methods

Fleet & Jepson (1990) proposed that, since phase is insensitive to changes in scene illumination, a better approximation to image velocity can be estimated if contours of constant phase, rather than contours of constant intensity, are tracked over time. Based on this idea, phase-based methods (e.g., Fleet & Jepson, 1990; Gautama & van Hulle, 2002; Fleet, 2012) compute the concept of *component velocity*, defined as the velocity normal to a constant-phase contour, from the outputs of band-pass velocity-tuned filters. Given the multiple component velocity estimates that result from the application of different filters at a single location, an affine model of optical flow is fitted to each local region of the image in order to estimate image motion.

Contrary to energy-based methods, the approach proposed by Fleet & Jepson (1990) is comparable to the well-known Lucas-Kanade algorithm in terms of accuracy, as reported by Barron et al. (1992). However, due to the number of filters required by this phase-based method, its ability to perform under real-time constraints is very limited. As a consequence, there are no applications of these methods in MAVs.

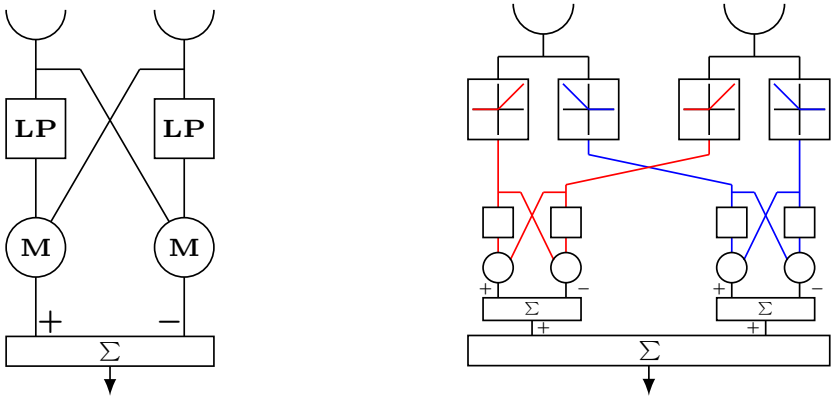
2-2-4 Bio-inspired motion detectors

Since the proposal of the Elementary Motion Detector (EMD) by Hassenstein & Reichardt (1956) and Reichardt (1961), the formulation of biologically-inspired methods for motion estimation has been an appealing research topic (Eichner et al., 2011). The reason for this interest is due to the robustness and efficiency of these alternative biological systems, which would improve the performance of the ego-motion estimation approaches available nowadays.

In an attempt to replicate the direction-selectivity found at early stages of the visual system of a fly, Hassenstein & Reichardt (1956) presented a correlation-based method in which two

[4]Integrated circuit whose logic block interconnection is programmable by the user after manufacturing.

input signals from two neighboring photoreceptors are multiplied after one of them has been temporally delayed by a low-pass filter. As illustrated in Figure 2-4a, this computation is symmetrically performed twice, and the outputs of both operations are subtracted to gain the desired direction selectivity. According to Borst et al. (2010), the Reichardt model accurately reproduces the neural responses to the motion of a stimulus, however, its cellular implementation is still undiscovered (Borst, 2011). Alternatives to this method have been published to more accurately reproduce the neural responses of *Drosophila* flies during motion estimation (Eichner et al., 2011). For example, the model proposed by Franceschini et al. (1989), shown in Figure 2-4b, implies that there is not interaction between signals of opposite sign. Furthermore, due to their simplicity, these motion detectors have inspired the development of optical flow sensors capable of efficiently estimating optical flow with decent levels of accuracy (Ruffier & Franceschini, 2005; Ruffier & Franceschini, 2015).



(a) Reichardt detector model. **(b)** Franceschini detector model.

Figure 2-4: Schematics of the EMD models investigated by Eichner et al. (2011). On top of the images there are two photoreceptor cells for each EMD, from which the signals flow downwards. The square blocks indicate time delays applied by a low-pass filter, while the circular cells indicate multiplication of two signals. Adapted from Eichner et al. (2011).

Progress is continuously being made towards the understanding of how motion is perceived in the mammalian retina and processed in the brain (Barlow & Levick, 1965; Albright, 1984; Grossman et al., 2000; Borst & Helmstaedter, 2015). Although there are some approaches that try to replicate the visual pathway to the brain using spiking neural architectures (Lappe & Rauschecker, 1993; Rao & Sejnowski, 2001), there is no agreement on a mammalian-inspired motion detector model yet.

2-3 Bio-inspired navigation using optical flow

Extensive research has been conducted on how different animals exploit optical flow information for ego-motion estimation and navigation. Several authors have succeeded in the formulation of mathematical models that relate the visual observables identified in Section 2-1-2 to vision-based maneuvers performed in biology (e.g., Srinivasan et al., 1996; Baird et al., 2013). Inspired by these models, this section introduces a set navigation strategies that have been successfully implemented in MAVs.

2-3-1 Navigation using ventral flow

To the best knowledge of the author, the first implementation of a vision-based navigation strategy in which ventral flow components were used in the control loop of a MAV corresponds to the work presented by Chahl et al. (2004). Based on biological findings, honeybees perform grazing landings maintaining the velocity of the ground plane as seen on the retina constant, i.e., keeping ventral flow constant (Srinivasan et al., 1996). According to Chahl et al. (2004), a linear relationship between forward and descend speed can be established, leading to the following evolution of the height above the surface over time, where c denotes a proportionality constant:

$$Z(t) = Z(t_0)\, e^{-c\omega_x (t-t_0)} \tag{2-14}$$

Chahl et al. (2004) analyzed this solution through an implementation on-board of a fixed-wing MAV equipped with a downward-looking frame-based camera. Optical flow was estimated using image interpolation (Srinivasan, 1994), a correlation-based approach. Further, results proving the performance of this method in achieving smooth landings were also included.

Besides this work, the application of the constant ventral flow strategy in MAVs was deeply investigated by Ruffier & Franceschini (2005, 2015) and Expert & Ruffier (2012, 2015). Here, the authors demonstrated the performance of this approach using a rotorcraft MAV equipped with an optical flow sensor that, being tethered, was moved in circular patterns above a textured surface. The two degrees-of-freedom (DoF) that described the flight condition of the MAV during these experiments were the pitch angle and the magnitude of the thrust vector. Landing was successfully performed by slowly pitching up, such that the height of the vehicle was decreased according to Eq. (2-14).

As it is possible to infer, these navigation strategies present the limitation that it is not possible to control vertical dynamics without forward motion. Although this situation is not problematic for fixed-wind MAVs, hovering and pure vertical motion, which are the main advantages of rotorcraft MAVs, become impossible.

2-3-2 Landing using divergence and time-to-contact

Constant divergence

Continuing with the research about the navigation of honeybees, Baird et al. (2013) showed that these insects land on vertical surfaces with constant image divergence, which is equivalent to keeping the time-to-contact, τ, also constant. This strategy results in a similar landing behavior as seen in grazing landings, but now the vertical and horizontal motion are independent of each other. Hence, this navigation solution can be applied in rotorcraft MAVs without limiting their flight envelope.

Herissé et al. (2012) applied the constant divergence strategy in a vertical take-off and landing (VTOL) MAV in order to hover and land on a moving platform. The vehicle was equipped with a downward-looking camera, and optical flow was estimated with the Lucas-Kanade algorithm (Lucas & Kanade, 1981). Based on the assumption that the platform was always in the field of view of the camera, hovering above target was achieved with a control law forcing

ventral flows to zero. Results demonstrating the performance of this navigation solution were also included.

In Ho & de Croon (2016), the authors focused on experimentally characterizing the concept of divergence when it is estimated from a monocular camera mounted on a MAV during a landing maneuver. This characterization was performed in terms of the delay and noise of the measurement, and based on it, the practicality of using this parameter in the control loops of MAVs was assessed. According to their results, noise and time delay generate significant oscillations in the estimated divergence, destabilizing the control system, when the MAV is close to the landing surface. Note that, since divergence is scaled by Z, the impact of disturbances on the estimated motion is much larger at low heights.

Constant rate-of-change in time-to-contact

For landing maneuvers, apart from the aforementioned constant divergence strategy, research was also conducted on τ-based laws. According to the results presented in D. N. Lee (1976) and D. N. Lee et al. (1993), the *rate-of-change* of the time-to-contact, $\dot{\tau}$, is normally kept constant by pigeons during landing, and by humans performing breaking maneuvers. This results in a different landing trajectory. Let $k = -\dot{\tau}$ be the rate-of-change setpoint. Then, as proved by Izzo & de Croon (2012), the height above the surface evolves over time following:

$$Z(t) = Z(t_0)\left(k\frac{t}{\tau(t_0)} + 1\right)^{1/k} \tag{2-15}$$

Apart from this proposal, Izzo & de Croon (2012) showed the versatility of this new approach performing experiments within the context of spacecraft landings. Based on this work, the constant $\dot{\tau}$ strategy was successfully applied in a rotorcraft MAV for landing maneuvers by Alkowatly et al. (2014). Here, the visual observables were estimated using planar optical flow from a downward-looking camera.

Although τ was not computed using vision but by fusing measurements from other sensors, Kendoul (2014) designed and implemented a τ-based autopilot in a rotorcraft MAV, in which τ estimates were used for navigation in three dimensions. As reported by Kendoul, the MAV was capable of following pre-computed τ-based trajectories with good performance.

2-4 Other vision-based navigation strategies

Apart from the bio-inspired approaches presented in this chapter for ego-motion estimation, extensive research has been conducted on the development of robust vision-based navigation solutions from a different perspective: the field of robot localization and mapping. Here, visual odometry (Nistér et al., 2004; Scaramuzza & Fraundorfer, 2011) and SLAM (Davison et al., 2007) are the two most frequently-used techniques for these purposes. The aim of this section is to give a brief overview of their working principles, and to present examples of their applications in MAVs.

2-4-1 Visual odometry

As defined by Scaramuzza & Fraundorfer (2011), visual odometry is "the process of estimating the ego-motion of an agent using only the input of a single or multiple cameras attached to it". For this, in monocular visual odometry, the camera pose is incrementally estimated from the changes in brightness induced by motion on the images generated by a single sensor. These methods can be divided into three main categories: feature-based, appearance-based, and hybrid methods.

Similarly to correlation-based optical flow estimation approaches, the first stage of feature-based visual odometry methods consists in tracking salient and repeatable image features over frames. Camera pose is then estimated by means of computing the optical flow components of these features and solving an overdetermined system of equations based on Eq. (2-3). For the first real-time implementation of this algorithm, Nistér et al. (2004) proposed the use of a five-point minimal solver (Nistér, 2004) to compute the solution of this system of equations, and RANSAC (Fischler & Bolles, 1981) for outlier rejection. To overcome the computational requirements of this solver, Weiss et al. (2013) proposed the combination of visual and IMU information. In this work, the authors used this visual-inertial solution for the navigation of a MAV in four Degrees-of-Freedom (DoF) and metric velocity.

Since feature tracking is not consistent in low-textured environments, appearance-based and hybrid methods use dense information of image intensity for motion estimation. Optical flow techniques, such as the Lucas-Kanade (Lucas & Kanade, 1981) or the Horn-Schunck (Horn & Schunck, 1981) algorithm, are used to find correspondences of intensity patches over a set of successive frames. Once optical flow components are obtained, these are normally fed into a camera-pose estimation stage based on the inverse camera projection matrix (Scaramuzza & Fraundorfer, 2011). Implementations of these methods are always combined with the generation of a map of the environment (Weiss et al., 2012; Forster et al., 2014). Therefore, a more detailed explanation of these works is presented in Section 2-4-2.

2-4-2 SLAM

SLAM is a technique for estimating agent ego-motion, and for obtaining the 3D structure of the unknown environment in which this agent is moving at the same time (Fuentes-Pacheco et al., 2015). When using images as the only source of external information, these techniques are categorized as Visual Simultaneous Localization And Mapping (V-SLAM). Similarly to visual odometry approaches, monocular V-SLAM methods are further divided in two groups: feature-based, and direct methods.

Regarding feature-based methods, there are two main approaches that set the basis for all the techniques in this group: MonoSLAM (Davison et al., 2007) and Parallel Tracking And Mapping (PTAM) (Klein & Murray, 2007). On the one hand, in MonoSLAM, camera motion and the 3D map of the environment are simultaneously estimated using an Extended Kalman Filter (EKF) framework. Here, the size of the state vector increases in proportion to the size of the map, which makes any real-time implementation impossible. On the other hand, PTAM solves this problem by splitting these two tasks into different threads for parallel computation. An adaptation of PTAM was implemented in MAVs by Weiss et al. (2012) through fusing visual information with IMU data to avoid the need for the five-point minimal solver (Nistér,

2004). Robust 6-DoF navigation was achieved by feeding this velocity estimate, and the 3D map, into an EKF.

In contrast to feature-based approaches, direct methods track the location of the agent by comparing input and synthetic images generated from the map. In this group, it is important to remark two main contributions to the field of vision-based navigation of MAVs: the works presented by Forster et al. (2014), and by Bloesch et al. (2015). Firstly, Forster et al. (2014) proposed a Semi-direct Visual Odometry (SVO) pipeline that, based on the PTAM algorithm, uses images alignment for relative pose estimation. Further, instead of using an EKF framework, this method includes iterative depth filters to incorporate newly detected 3D points to the mapping thread. Experiments conducted on a small rotorcraft MAV showed a decrease of one order-of-magnitude in the relative position error with respect to PTAM, while the rotation error is maintained. Secondly, instead of using image alignment, Bloesch et al. (2015) presented a method in which template matching is coupled to an EKF framework for relative pose estimation. This approach is successfully validated by using it in the control loop of a MAV for take-off and landing maneuvers.

Chapter 3

Dynamic Vision Sensor and Event-based Optical Flow Estimation

The optical flow estimation methods and applications discussed so far mainly involve the use of conventional frame-based cameras. This chapter provides an introduction to the field of event-based vision via the Dynamic Vision Sensor (DVS), which is the sensor employed in this work, in Section 3-1. Next, a detailed overview of the existing methods for computing optical flow from these sensors is included in Section 3-2. Finally, Section 3-3 describes the recent event-based approaches that have been proposed for ego-motion estimation.

3-1 The Dynamic Vision Sensor

Conventional cameras perceive the world as a series of consecutive images, which are generated by measuring the pixel values at fixed time intervals. Although these sensors are suitable for the great majority of computer vision applications (e.g., classification, object recognition), these frames generally contain enormous amount of redundant information.

Inspired by biological retinas, the pixels of event-based vision sensors react asynchronously to changes in perceived brightness by generating *events*. Accordingly, the outputs of these sensors are characterized by a stream of events encoding variations of image intensity at a particular time and location on the image plane. The Dynamic Vision Sensor (DVS) is an event-based camera with a 128×128 pixel array, designed by Lichtsteiner et al. (2008) and commercialized by iniLabs under the name of "DVS128". This device, shown in Figure 3-1, is the vision sensor used in this work. This section serves as an introduction to the working principle and the main characteristics of this sensor.

Figure 3-1: Picture of the DVS128. From `https://inilabs.com/`

3-1-1 Working principle

According to Lichtsteiner et al. (2008), DVS events are asynchronously generated whenever a pixel (i.e., a photoreceptor circuit) measures a *logarithmic local change in brightness* that exceeds a predefined threshold C, as shown in Eq. (3-1). This variation in image intensity is computed with respect to a reference brightness level set at the last occurring event at that particular pixel. Furthermore, depending on whether the change in brightness is positive or negative, the polarity of the generated event is ON (+1) or OFF (-1), respectively.

$$\left|\Delta \log\left(I\left(\hat{x},\hat{y},t\right)\right)\right| > C_{ON/OFF} \tag{3-1}$$

This principle of operation is illustrated in Figure 3-2. Firstly, the top graph shows an example of how the voltage output, V_p, of a DVS pixel looks like. As mentioned before, this voltage corresponds to a representation of the logarithmic level of brightness perceived. Secondly, the bottom graph shows the corresponding evolution of the local change in image intensity, denoted by V_{diff}. Note that the spiking behavior of this signal is due to the reconstruction of the reference brightness level when V_{diff} exceeds one of the thresholds. These spikes are the events generated by the DVS.

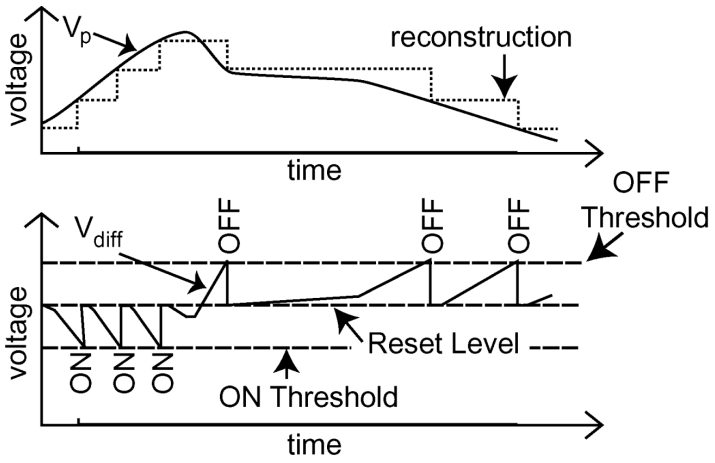


Figure 3-2: Working principle of a DVS pixel. From Lichtsteiner et al. (2008).

Each of these events encodes information about the timestamp (t) in which it was generated, the corresponding location on the pixel array ($\hat{x},\hat{y}$), and the polarity of the change in brightness (P). This method of transmitting information from asynchronous sensors is referred to as the Address-Event Representation (AER) protocol (Boahen, 2000; Lichtsteiner et al., 2008).

3-1-2 Sensor characteristics, advantages, and limitations

The DVS is the first event-based camera being commercialized (Posch et al., 2014; Cho & Lee, 2015), and it is available for purchase from iniLabs under the product name of "DVS128". Being specialized for general purpose and rapid application development, this sensor interfaces with other hardware through a USB 2.0 connection, which also powers the device. According

Table 3-1: DVS128 specifications (iniLabs, n.d.)

Array size	128×128
Pixel size	40×40 μm
Dimensions	5×5×2.5 cm (without lens)
Weight	120 g (including lens)
Connectivity	USB 2.0
Latency	12 μs (at 1 klux)
Temporal contrast sensitivity[1]	17%
Power consumption	23 mW
Dynamic range[2]	120 dB
Event bandwidth	1 million events per second

to Lichtsteiner et al. (2008), this connection has an event bandwidth of one million events per second using the AER protocol. An overview of the main specifications of this sensor is introduced in Table 3-1.

When compared to conventional frame-based cameras, the DVS offers distinct advantages mainly due to its asynchronous pixel readout. First of all, this sensor reacts only when changes in brightness are perceived. Hence, scene dynamics is correlated with the output events, avoiding the redundancy that characterizes the measurements of conventional cameras. Secondly, each DVS pixel asynchronously measures local changes at a very high temporal resolution (1 μs) and with small latency (12 μs). Finally, this low-power sensor is characterized by an average consumption of 23 mW, which is significantly lower than the power needed by other similar cameras (Lichtsteiner et al., 2008).

Regarding the main limitations of the DVS, it is important to remark the small pixel array, the lack of absolute brightness levels, and the limited temporal contrast sensitivity (see Table 3-1). To overcome these problems, several other event-based vision sensors have been presented since the release of the DVS in 2008. Among others, the most relevant designs are the Asynchronous Time-based Image Sensor (ATIS) (Posch et al., 2011) and the Dynamic and Active-pixel Vision Sensor (DAVIS) (Brandli et al., 2014). On the one hand, ATIS features a 304×240 pixel array that outputs a stream of events encoding relative changes as well as absolute brightness levels (Posch et al., 2011). On the other hand, DAVIS combines an improved DVS with a conventional global shutter frame-based sensor. Furthermore, the Embedded Dynamic Vision Sensor (eDVS) and the Miniature Embedded Dynamic Vision Sensor (meDVS) are the result of the miniaturization work that is currently being conducted at iniLabs. These sensors, being considerably smaller than the original DVS, are highly suitable for on-board MAV applications.

3-1-3 Processing software

The open-source tool Java Address-Event Representation (jAER) is the software developed by iniLabs (Delbruck, 2007) for visualization of real-time or recorded DVS events. As the name suggests, this program uses the same AER communication protocol for processing the

[1] Measure of the light level difference that a sensor needs to be able to discriminate a light source as flickering versus steady.

[2] The ratio between the maximum and minimum measurable light intensities.

events, thus it can be used for the rapid development of real-time event-based algorithms and applications, the so-called *filters*. The software uses a modular structure very suitable for the evaluation and visualization of these filters.

Apart from jAER, a version of this program has been developed based on the C programming language under the name cAER by iniLabs (Longinotti, 2014). This version was developed in order to improve the portability of event-based algorithms to embedded systems. This software has allowed the on-board utilization of the DVS in ground (Weikersdorfer & Conradt, 2012) and aerial vehicles (Hordijk et al., 2018) for real-time applications.

3-2 Event-based optical flow estimation

Recalling from Chapter 2, the concept of optical flow was introduced as a means to quantify motion from images perceived by a camera. For this image flow to be estimated, an algorithm comparing the brightness levels of successive frames is needed so any variation can be detected and interpreted as motion. Since DVS events already encode this dynamic information at a very high temporal resolution, the field of optical flow estimation from event-based vision sensors has recently gained much attention. This section provides an overview of the methods that are currently available, with some of their corresponding applications.

3-2-1 Event-based Lucas Kanade

As introduced in Section 2-2-1, gradient-based methods make use of the spatiotemporal derivatives of the image intensity function (I_x, I_y, I_t) in conjunction with the brightness constancy constraint (Horn & Schunck, 1981) for optical flow estimation. On this basis, Benosman et al. (2012), together with the reformulation made by Brosch et al. (2015), presented an adaptation of the well-known Lucas-Kanade algorithm (Lucas & Kanade, 1981) to event-based vision.

By definition, DVS events encode information about local changes in brightness over time. Hence, these events are regarded as a representation of the temporal derivative I_t. However, the computation of the spatial derivatives (I_x, I_y) seems to be not possible since absolute brightness levels are not available. To overcome this problem, Brosch et al. (2015) proposed a reformulation of the algorithm based on two main assumptions. Firstly, the brightness constancy constraint is redefined in terms of the *second-order partial derivatives*, as shown in Eq. (3-2). Secondly, the terms (I_{tx}, I_{ty}) are estimated by summing the polarities (P) of neighboring events within a time window $\mathcal{T}$.

$$I_{tx}\hat{u} + I_{ty}\hat{v} + I_{tt} = 0 \tag{3-2}$$

$$\begin{aligned}
I_{tx}\left(\hat{x},\hat{y},t\right) &= \sum_{t'\in\mathcal{T}} P\left(\hat{x}+1,\hat{y},t'\right) - \sum_{t'\in\mathcal{T}} P\left(\hat{x}-1,\hat{y},t'\right) \\
I_{ty}\left(\hat{x},\hat{y},t\right) &= \sum_{t'\in\mathcal{T}} P\left(\hat{x},\hat{y}+1,t'\right) - \sum_{t'\in\mathcal{T}} P\left(\hat{x},\hat{y}-1,t'\right) \\
I_{tt}\left(\hat{x},\hat{y},t\right) &= \frac{1}{\Delta t^2}\left(\sum_{t'\in\mathcal{T}_{-1}} P\left(\hat{x},\hat{y},t'\right) - \sum_{t'\in\mathcal{T}_{-2}} P\left(\hat{x},\hat{y},t'\right)\right)
\end{aligned} \tag{3-3}$$

Note that the temporal windows were defined as $\mathcal{T}_{-1} = (t - \Delta t, t]$ and $\mathcal{T}_{-2} = (t - 2\Delta t, t - \Delta t]$, with $\mathcal{T} = \mathcal{T}_{-2} \cup \mathcal{T}_{-1}$. As proposed by Lucas & Kanade (1981), an overdetermined system of equation needs to be created using Eq. (3-2) to resolve for the optical flow components.

The experiments presented by Benosman et al. (2012) showed the capabilities of this algorithm to accurately estimate direction of motion in real-time on a desktop computer. However, the magnitude of the optical flow vectors was often uncorrelated with the speed of motion. In Brosch et al. (2015), the authors argued that the limited number of events that are generated when an edge passes the receptive field of a DVS pixel leads to inaccuracies in the estimation of (I_{tx}, I_{ty}, I_{tt}). Therefore, the conclusion is that gradient-based approaches are not suited for the sparse encoding of event-based sensors.

3-2-2 Spatiotemporal plane fitting

Benosman et al. (2014) proposed an alternative to the event-based Lucas-Kanade algorithm that is based on the representation of events as part of a small volume in the x-y-t-space. This volume, also known as the *surface of active events* Σ_e and illustrated in Figure 3-3, is defined as a function that associates the location of an event to a third dimension that happens to be the time at which it was generated: $\Sigma(\hat{x}, \hat{y}) = t$. Based on the assumption that velocity is constant in Σ_e, Benosman et al. (2014) estimates optical flow components by computing the inverse gradients of a plane fitted to represent this surface, as shown in Eq. (3-4). In order to have an estimate for each newly detected event, a plane fitting procedure based on linear least-squares is applied to a spatiotemporal window centered on each of these events. Only events with the same polarity are considered for plane fitting. Experiments conducted by Benosman et al. (2014) demonstrated that this algorithm is capable of estimating optical flow with decent accuracy.

$$\nabla \Sigma_e (\hat{x}, \hat{y}) = \left[\Sigma_{e_x}, \Sigma_{e_y}\right]^T = \left[\frac{1}{\hat{u}\,(\hat{x}, \hat{y})}, \frac{1}{\hat{v}\,(\hat{x}, \hat{y})}\right]^T \tag{3-4}$$

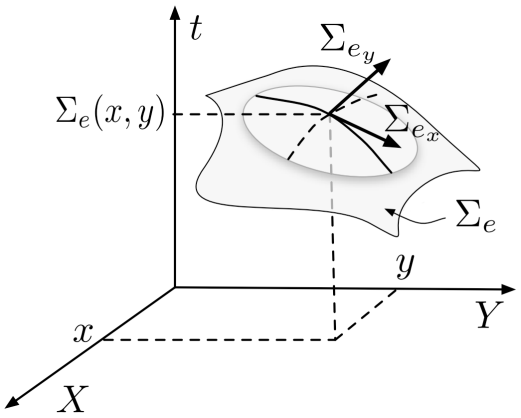


Figure 3-3: Illustration of the surface of active events and its gradients. From Benosman et al. (2014).

Using the spatiotemporal plane fitting technique as a basis, Hordijk et al. (2018) presented a novel optical flow estimation algorithm with two main differences with respect to the work proposed by Benosman et al. (2014). First, the computational efficiency of the method was improved by reducing the number of parameters needed for plane fitting. Second, a time window with adaptive length was used to extend the range of velocities that can be measured. Apart from this, the authors presented a method for estimating divergence out of the event-based optical flow components, and incorporated this estimation in a constant-divergence landing controller implemented on-board of a rotorcraft MAV. To the best knowledge of the author, the work presented by Hordijk et al. (2018) corresponds to the first time that an event-based camera is successfully integrated in the navigation loop of a MAV.

3-2-3 Direction-selective filters

In Section 2-2-3, energy-based methods were introduced as biologically-inspired approaches for optical flow estimation based on the extraction of orientation in the spatiotemporal domain. Extensive research has been conducted over recent years on the adaptation of these methods to event-based vision due to the similarities between the DVS and biological retinas. According to Medathati et al. (2016), this combination leads to the most similar approach to motion estimation at early stages of the visual cortex (De Valois et al., 2000).

Tschechne et al. (2014) presented, for the first time, an implementation of these methods using the DVS as input sensor. In this work, the authors proposed that directional selectivity in cortical cells is generated as a combination of two spatial (even- and odd-symmetric) and two temporal (mono- and bi-phasic) filters, as shown in Figure 3-4. Parameter tuning of these filters was performed according to experimental data obtained from V1 (primary visual cortex) cells by De Valois et al. (2000). For motion estimation at different orientations, a filter bank is created using different rotation settings for the spatial filters. Note that the output of these filters is a *confidence value* that the stimulus is moving in a specific direction with certain speed. Therefore, a considerably large filter bank needs to be generated to cover a wide range of speeds and directions.

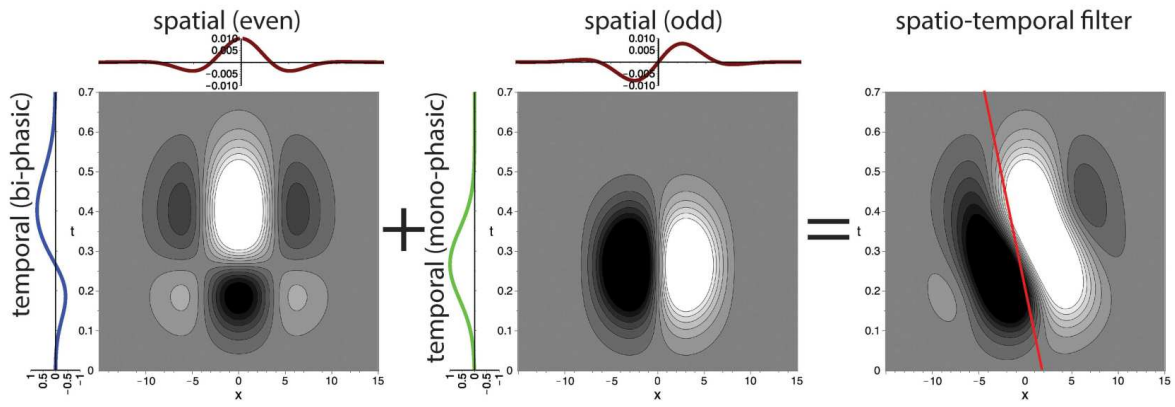


Figure 3-4: Generation of a direction-selective filter as a combination of two separable spatiotemporal responses in one spatial dimension and time. From Brosch et al. (2015).

Brosch et al. (2015) extended this approach by means of analyzing the relation between the filter parameters and the "preferred speed" through Fourier analysis. Further, to reduce the ambiguity caused by the aperture problem, the authors proposed a method of *response normalization* inspired by the feedback connections present in the visual pathway to the brain (Brosch & Neumann, 2014). Such normalization integrates the filter outputs over larger regions in the image plane, and the outcome is fed back to the filters.

This approach is further explored by Abdul-Kreem & Neumann (2015). In this work, instead of having just one layer of spatiotemporal filters, the authors proposed a two-layer method in order to model the two main visual areas of the brain, namely V1 and MT (middle temporal). Based on the response normalization stage from Brosch et al. (2015), V1 filters were modulated by feedback signals coming from MT.

Although these implementations of the filter-based approach are capable of robustly estimating direction of motion (Tschechne et al., 2014; Brosch et al., 2015; Abdul-Kreem & Neumann, 2015), they have accuracy problems when computing the magnitude of the optical flow vectors. As previously mentioned, this magnitude has to be calculated from confidence values, thus the precision of this parameter is limited to the number of velocity settings in the filter bank.

3-3 Visual odometry using event-based sensors

The application of event-based sensors in the fields of visual odometry and SLAM has gained interest since the availability of the DVS. Their key advantages (high temporal resolution, small latency, and high dynamic range, among others) enable the design of new camera pose estimation algorithms suitable for scenes characterized by high-speed motion (Mueggler et al., 2014) and high-dynamic range (Rebecq et al., 2017), where conventional frame-based cameras normally fail. This section presents an overview of the different methods that have recently been presented towards this goal.

Censi & Scaramuzza (2014) proposed a pose tracking algorithm based on the combination of the DVS with a conventional frame-based camera. The working principle of this approach was to accumulate DVS events over a temporal window, and then to compare this representation of the scene with the previous frame obtained from the normal camera, which is acting as a map of the environment. According to Censi & Scaramuzza (2014), rotational motion can be estimated with a small drift, but translation cannot be reliably computed since the estimates were noisy. Based on these results, the authors suggested that, for future projects, a promising idea would be to separate the use of the DVS for rotation estimation, while using frame-based cameras for translation.

Mueggler et al. (2014) presented an implementation of a visual odometry algorithm, using a quadrotor MAV as platform for the execution of high-speed maneuvers. In the experiments conducted for this work, the MAV performed high-speed lateral flips in front of a white wall with a black square maker on it. The working principle of this algorithm was to perform edge tracking for ego-motion estimation. Whenever a DVS event is generated by the sensor, this method checks whether it belongs to any of the lines, and if so, it updates the lines, and, subsequently, the pose estimate. Note that real-time implementation was achieved by streaming the DVS events to a laptop for off-board computation. Further, the authors argued

that this method was just a proof-of-concept to show the capabilities of event-based visual odometry. It cannot be used in arbitrary environments.

Gallego et al. (2017) continued working in this line of action, and presented a probabilistic pose-tracking method for event-based vision sensors in known environments. In this work, the authors characterized the scene using a sparse set of reference images accompanied by their corresponding poses and depth maps. By means of using this information, this Bayesian filter allows to estimate pose updates for each newly detected DVS event. The authors demonstrated the accuracy of this approach using natural indoor and outdoor scenes. Further, they showed the algorithm capability of tracking the DVS pose in spite of moving objects in the scene.

The first event-based visual odometry algorithm capable of robustly estimating camera pose in realistic and natural scenes was presented by Kueng et al. (2016). Being an event-based adaptation of the SVO method (Forster et al., 2014), the authors employed the DAVIS sensor since DVS events do not encode absolute brightness levels. Image features are detected using image intensity values and then tracked with the stream of asynchronous events. Results demonstrated successful feature tracking and visual odometry performance in unknown scenes with natural textures.

Finally, Rebecq et al. (2017) presented, for the first time, a real-time 6-DoF visual odometry approach based only on information retrieved from an event-based sensor, such as the DVS. This method, further referred to as Event-based Visual Odometry (EVO), uses the separation principle of SLAM systems (Klein & Murray, 2007) to perform pose tracking and mapping in different threads, thus allowing real-time implementation. In order to avoid the need of intensity information, the pose tracking stage uses geometric alignment error between two images of accumulated events. Experiments to evaluate the accuracy of this approach are based on the comparison with the SVO method (Forster et al., 2014). Rebecq et al. (2017) proved the high accuracy of EVO even during high-speed maneuvers or under poor lighting conditions.

Although these methods, especially the work presented by Rebecq et al. (2017), lay the groundwork for future implementations of event-based V-SLAM approaches in aerial vehicles, the first application of the DVS in the control loop of a quadrotor MAV corresponds to Hordijk et al. (2018).

Chapter 4

Neuromorphic Computing for Vision-based Navigation

Imagine an animal endowed with brain and visual cortex moving through a scene. When this animal perceives light (in the visible spectrum) reflected by the objects in this environment, neurons in different parts of the brain react and start communicating with each other via electrical and chemical pulses, also known as *spikes*. The outcome of this interaction being the ability to understand the surrounding environment, i.e., visual perception, which is crucial for tasks such as navigation, recognition, and classification. The concept of neuromorphic engineering, also referred to as *neuromorphic computing*, alludes to the progress that is recently being made in the field of Artificial Intelligence (AI) towards the establishment of computational frameworks capable of replicating how the brain processes, learns, and memorizes. These models are the so-called *neural networks*. If successfully implemented in robotics for vision purposes, neural networks would prevent the need for complex mathematical models describing the agent-environment visual relations (Chapters 2 and 3), since these networks can learn how to efficiently compute the desired output from the set of input images.

This chapter serves as an introduction to the two main computational models currently used in Artificial Intelligence (AI) for vision: Spiking Neural Networks (SNNs); and Artificial Neural Networks (ANNs). Due to the infancy stage of spiking networks, Section 4-1 places emphasis on the mathematical description, learning processes, and computer vision applications of these models. On the contrary, Section 4-2 exhaustively covers the common ANNs architectures for visual odometry applications.

4-1 Spiking Neural Networks

Starting from the computational model with highest level of realism, spiking neural networks attempt to replicate the spike-based communication protocol used in the brain. As mentioned before, neurons from biological neural systems exchange information by sending and receiving short electrical pulses, or spikes, whose amplitude remains approximately constant during

propagation. Since, in terms of appearance, all spikes of a given neuron look similar, the information is encoded in the precise timing at which these pulses are generated. According to numerous studies (Ferster & Spruston, 1995; Maass, 1997; Thorpe et al., 2001), this temporal coding is what enables many biological neural systems to perform high-speed computations. For this reason, spiking neurons from spiking neural networks are defined as asynchronous event-based processing units with temporal dynamics. The rest of this section delves into this powerful computational framework.

4-1-1 Biological background

For the description of the biological background of spiking neural networks, this work refers to the theoretical knowledge collected by Gerstner & Kistler (2002) about the structure and function of the brain.

As illustrated in Figure 4-1, four functionally distinct parts can be identified in a biological neuron, called dendrites, soma, axon, and synapse. First, the dendrites are the input ramifications of the neuron, and their role is to collect signals and transmit them to the soma. The soma, acting as the processing unit of the neuron, is characterized by a state variable known as *membrane potential*. If a sequence of input spikes raises this internal state above a certain threshold, the soma is in charge of generating an output signal, which is delivered to other neurons via the axon. The junction between two neurons is called synapse, so sending cells are commonly referred to as *presynaptic neurons*, while receiving cells are *postsynaptic neurons*.

The membrane potential, $v(t)$, of a postsynaptic biological neuron varies according to the incoming spikes (spike train) from a set of presynaptic neurons. If the neuron does not receive any input, its membrane potential remains constant at a particular value v_{rest}. If the change in the membrane potential is positive after the arrival of a spike, that synapse is said to be *excitatory*. If negative, the synapse is *inhibitory*. After this variation, the potential decays back to the resting potential. Further, after the emission of a spike, the neuron enters in a *refractory period* in which the membrane potential hardly changes due to new incoming spikes. This period, which only lasts a few milliseconds, ensures the event-based representation of these spikes as pulses clearly distributed over time. Figure 4-2 exemplifies this mechanism of spike generation using a postsynaptic neuron integrating the presynaptic spike train from two input cells.

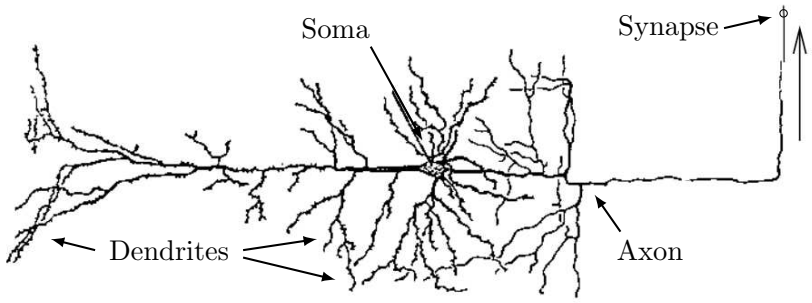


Figure 4-1: Illustration of the functionally distinct parts that compose a single neuron. Adapted from Ramón y Cajal & Azoulay (1955).

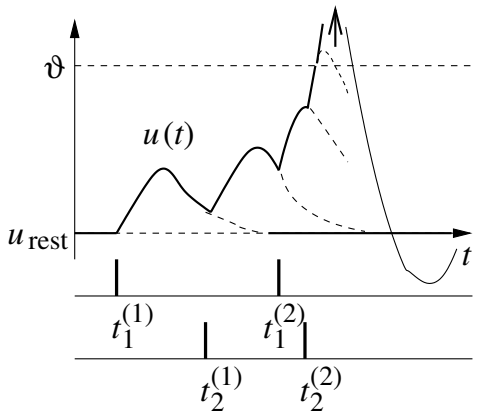


Figure 4-2: Mechanism of spike generation in biological neurons. Adapted from Gerstner & Kistler (2002). Note that, in this publication, the authors denoted the membrane potential as $u(t)$. Instead, this parameter is represented by $v(t)$ throughout this document.

4-1-2 Models of spiking neurons

When using spiking neural networks, the first step consists in deciding which model should be used for the mathematical representation of the spike generation mechanism illustrated in Figure 4-2. Since the beginning of scientific research in the field of neuroscience, different neuron models have been presented with this purpose, each of them with a different level of abstraction.

The spiking neuron model with highest level of realism was presented by Hodgkin & Huxley (1952), and referred to as the "Hodgkin-Huxley model". After experimenting with the axon of a squid, the authors succeeded in the identification of a set of differential equations capable of describing the dynamics of a neuron. In this work, Hodgkin & Huxley (1952) proposed, for the first time, the idea that this dynamics can be modeled using electric circuits made of capacitors and resistors. Although this model is not usable in large simulations of spiking neurons due to its computational complexity, it has served as a basis for the development of the more-efficient approaches now presented (Stein, 1965; Kistler et al., 1997; Izhikevich, 2003).

Leaky Integrate-and-Fire model

The Leaky Integrate-and-Fire (LIF) neuron model, presented by Stein (1965), simplified the Hodgkin-Huxley model by means of assuming that the input channels to the neuron (i.e., the dendrites) are static. Hence, the shape of the presynaptic spike train can be neglected, and all the information is encoded in the time of appearance of each presynaptic pulse. Based on this, Stein proposed a basic electric circuit comprised of a resistor, R, that apart from being driven by an input current, $I(t)$, is in parallel with a capacitor, C. Defining $\lambda = R \cdot C$ as the time constant of the neuron membrane, the dynamics of the membrane potential $v(t)$ is described by the following first-order linear differential equation:

$$\lambda \frac{dv(t)}{dt} = v_{\text{rest}} - v(t) + RI(t) \tag{4-1}$$

Additionally, the firing time $t^{(f)}$ is defined by the time instant at which the membrane potential of the neuron crosses a predetermined threshold, $v(t^{(f)}) = v_{\text{th}}$. Immediately after $t^{(f)}$, the membrane potential is reset to v_{rest}. Note that, after a spike emission, this model may also incorporate an absolute refractory period in which the dynamics described by Eq. (4-1) are interrupted a few milliseconds.

Due to its simplicity, the LIF neuron model is frequently used in large simulations of spiking neurons (e.g., Yang et al., 2006; Bichler et al., 2012; Diehl & Cook, 2015).

Spike Response Model

A mathematically simpler model was proposed by Kistler et al. (1997) under the name of Spike Response Model (SRM). This model, instead of using differential equations, describes the state of the membrane potential $v(t)$ as an integral over the presynaptic spikes received in the past. Let the subindex i refer to the postsynaptic neuron under analysis, and j to the presynaptic cells connected to this neuron via synapses of weights $w_{i,j}$. Further, suppose that neuron i has fired its last spike at time $\hat{t}_i$. Then, after firing, the evolution of the membrane potential is described by:

$$v_i(t) = \eta\left(t - \hat{t}_i\right) + \sum_j w_{i,j} \sum_f \epsilon_{i,j}\left(t - \hat{t}_i, t - t_j^{(f)}\right) + \int_0^\infty \kappa\left(t - \hat{t}_i, s\right) I^{\text{ext}}(t - s)\,\mathrm{d}s \quad (4\text{-}2)$$

According to Eq. (4-2), the behavior of this neuron is mainly determined by three kernels: η, $\epsilon_{i,j}$, and κ. First of all, η describes the evolution of the membrane potential once it reaches the firing threshold v_{th}. Second, the kernel $\epsilon_{i,j}$ can be interpreted as the time course of the membrane potential once it receives a presynaptic spike. Depending on whether the synapse is excitatory or inhibitory, the sign of $w_{i,j}$ indicates if this response produces an increase or a decrease in $v(t)$. Finally, κ describes the variation in membrane potential caused by an external input current. If there is no such current, this final component of Eq. (4-2) is null. For a better understanding of these kernels, and for their mathematical formulation, please refer to Kistler et al. (1997).

Using this model, the membrane potential crosses the firing threshold immediately before and after emitting a spike. Therefore, for the firing time $t^{(f)}$ to be properly defined, an additional condition regarding the derivative of $v(t)$ is needed. This condition is defined as: $dv_i(t)/dt > 0$.

Note that, despite the simplicity of Eq. (4-2), this neuron model is more general than the Leaky Integrate-and-Fire formulation. In fact, the first-order differential equation presented in Eq. (4-1) can be derived from the SRM, as shown by Kistler et al. (1997).

Izhikevich model

Presented as an efficient and biologically plausible neuron model, the "Izhikevich model" (Izhikevich, 2003) reduces the Hodgkin-Huxley proposal to the following two-dimensional system of ordinary differential equations using bifurcation methodologies (Izhikevich, 2007):

$$\begin{aligned}\frac{dv(t)}{dt} &= 0.04v^2(t) + 5v(t) + 140 - u(t) + I(t) \\ \frac{du(t)}{dt} &= a\left(bv(t) - u(t)\right)\end{aligned} \tag{4-3}$$

According to Izhikevich (2003), $u(t)$ denotes a membrane recovery variable, representing the dynamics of input synaptic connections. Further, the constant parameters a, b, c, and d control the firing patterns that can be generated using this neuron model. Briefly explained, a describes the time scale of $u(t)$; b varies the sensitivity of $u(t)$ to sub-threshold fluctuations of the membrane potential; the parameter c is equivalent to the after-spike reset value v_{rest}; and d is the equivalent to c but for the recovery variable.

4-1-3 Synaptic plasticity: Learning with spiking neurons

Defined as the ability to modify the strength of the synaptic connections of a neural architecture, *synaptic plasticity* is "the basic mechanism underlying learning and memory in biological neural networks" (Baudry, 1998). The strength of a synapse, also referred to as *efficacy*, can be represented by the weights $w_{i,j}$ used in the Spike Response Model, Eq. (4-2). Depending on whether the desired outcome of the network is available and used for guiding synaptic plasticity or not, the learning methods used in spiking neural networks can be divided in supervised and unsupervised approaches, respectively.

Unsupervised learning

In the context of spiking neural networks, unsupervised learning is also referred to as "Hebbian learning" since all the methods inducing changes in synaptic efficacies are based on the following postulate:

> "*When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*" Hebb (1952).

Inferring from this postulate, Hebbian methods use the correlation in the firing activity of pre- and postsynaptic neurons for the reorganization of connections within a neural network. Among other techniques, Spike-Timing-Dependent Plasticity (STDP) (Markram et al., 1997) is, by far, the most popular learning rule for spiking neural networks. Its main principle, illustrated in Figure 4-3, consists in increasing the efficacy of a synaptic connection if the presynaptic spike arrives to the postsynaptic neuron short time before it fires ($\Delta t > 0$). In this case, the presynaptic neuron is said to have an influence over the response of the postsynaptic cell, so this connection is strengthened. However, if the spike arrives after the postsynaptic spike ($\Delta t < 0$), then the connection is weakened. Moreover, if there is no correlation between the two spikes (i.e., they are distant in time), the synaptic efficacy remains unchanged.

The most commonly-used temporal windows in STDP correspond to the biological findings reported by Bi & Poo (1998, 2001), shown in Figure 4-3. However, for a better understanding, the temporal axis used in these works is often inverted so that the region $\Delta t < 0$ represents

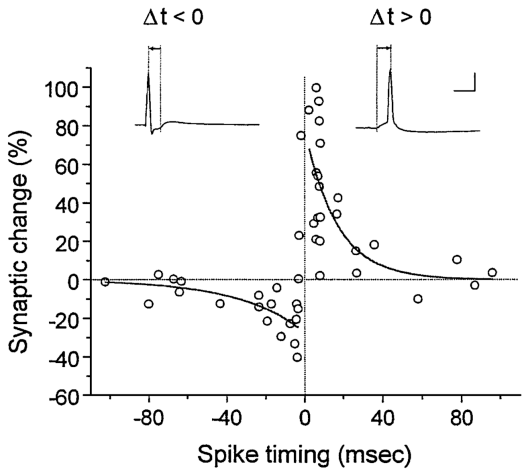


Figure 4-3: Plasticity window characteristic of the STDP rule. The change on synaptic efficacy is a function of the temporal difference between the emissions of pre- and postsynaptic spikes. From Bi & Poo (2001).

a presynaptic spike arriving before the postsynaptic cell fires. Gerstner & Kistler (2002) proposed a simple mathematical model to describe this behavior. Using the notation introduced for the Spike Response Model, and letting $s = t^{(f)} - \hat{t}_i$ denote the temporal difference between spikes, this model is given by:

$$\Delta w_{i,j}(s) = \begin{cases} A_+ \ \exp(s/\lambda_+) & \text{if } s < 0 \\ A_- \ \exp(s/\lambda_-) & \text{if } s > 0 \end{cases} \tag{4-4}$$

$$A_- = w_{i,j} a_-, \qquad A_+ = (w_{i,j}^{max} - w_{i,j}) a_+ \tag{4-5}$$

where the parameters a_+, a_-, τ_+, and τ_- are normally defined as constants. Note that Eq. (4-5) is needed to make sure that synaptic efficacies stay within certain bounds, so a realistic description of synaptic plasticity can be obtained. Furthermore, STDP is normally inverted for inhibitory synapses since these connections are characterized by negative efficacies.

STDP has been applied in numerous experiments trying to replicate the processing speed in the human visual system for tasks such as object recognition (Kheradpisheh et al., 2016), classification (Rousselet et al., 2002), or the extraction of temporally correlated features (Bichler et al., 2012).

Supervised learning

Regarding the use of supervised learning for synaptic plasticity in networks of spiking neurons, several lines of research can be distinguished. Firstly, rules exploiting the Hebbian postulate, such as Supervised Hebbian Learning (SHL), have been successfully validated in single-layer networks. Secondly, for training larger networks, extensive research has been conducted on the adaptation of the popular error backpropagation to the event-based representation of spiking neurons. Finally, evolutionary approaches have been employed for tuning not only synaptic efficacies, but also internal neuron parameters such as the firing threshold.

Extensively analyzed by Legenstein et al. (2008), the Supervised Hebbian Learning (SHL) approach is presented as the most biologically plausible method of performing supervised learning in spiking networks. Using the Hebbian postulate as a basis, this learning rule is based on the injection of external synaptic current to reinforce a postsynaptic neuron to fire at target times and not to do it at any other instant. Legenstein et al. (2008) demonstrated the ability of this learning algorithm to precisely approximate arbitrary mappings from input to output spike trains. However, the main drawback of this approach is that, since undesired spikes are avoided by these external currents, synaptic connections are never weakened. To overcome this problem, a reformulation of SHL was presented by Ponulak & Kasinski (2006) under the name of Remote Supervised Method (ReSuMe).

Due to the success of error backpropagation as supervised learning rule for large networks of artificial non-spiking neurons (for more details see Section 4-2), several adaptations have recently been proposed for spiking networks. Given a mapping between a set of inputs and outputs, the main principle of this technique is to compute the gradients of a specific loss function with respect to the synaptic weights of a network (Rumelhart et al., 1988). Briefly speaking, these gradients indicate the direction in the parameter weight space in which the error decreases. In networks comprised by artificial neurons, these error gradients can be easily computed since the relations between network's internal parameters are described by differentiable functions. However, due to the discontinuous dynamics that characterizes spiking cells, the derivation of these gradients requires a new set of assumptions.

The majority of the existing gradient-descent methods for these architectures rely on transferring the weights and biases from a pre-trained artificial network to its spiking counterpart. This is accomplished by replacing the activation functions (see Section 4-2-1) with spiking neurons, and performing parameter optimization (e.g., O'Connor et al., 2013; Diehl et al., 2015). Normally employed for image classification, these artificial networks are converted to spiking simply for efficiency improvements. Their inputs consist of static images that are encoded as discrete events using techniques such as Difference of Gaussians (DoG) filters. According to Wu et al. (2017), since these networks are trained in the spatial domain, the temporal aspects of the input event sequence are ignored.

The commercial availability of neuromorphic sensors, such as the DVS, has increased the interest on the adaptation of error backpropagation to spiking networks. Depending on the type of variables used in the loss function, two main categories can be discerned among recent proposals. First of all, the works of Bohte et al. (2002), Mostafa (2016), and Zenke & Ganguli (2017), demonstrate the possibility of training spiking architectures for reproducing precisely timed output spike trains. These authors argue that the time at which each spike is fired carries significant information, which is referred to as "temporal coding". Secondly, both J. H. Lee et al. (2016) and Wu et al. (2017) presented adaptations of backpropagation based on the spike rate of output neurons. Classification results in the N-MNIST[1] dataset (Orchard et al., 2015) confirm the validity of these approaches, since they perform similarly to conventional artificial networks in terms of accuracy. Note that all these learning rules are based on simplifying assumptions for gradient computation. The non-differentiable spike activity is either considered as noise or its derivative is mathematically approximated.

[1]Event-based version of the MNIST dataset, which consists of static images traditionally used for digit recognition with machine learning algorithms.

Apart from Hebbian- and gradient-based learning methods, different forms of evolutionary strategies, such as genetic algorithms, have also been applied for the supervised adaptation of synaptic weights. Remarkable is the work presented by Belatreche et al. (2003). Here, the authors used an evolutionary algorithm to adapt both the synaptic efficacies and delays, reporting classification accuracy comparable to SpikeProp on a set of pre-defined experiments. Despite these promising results, Belatreche et al. argued that evolutionary strategies are not suitable for training large networks due to their prohibitively large computational costs.

4-1-4 Event-based vision applications

The application of spiking neural networks in the field of computer vision has gained considerable interest since the availability of the DVS as neuromorphic input sensor. The reason is that both, sensor and computational framework, are based on the AER protocol for the transmission of information. Due to the infancy stage of spiking neural networks, a solution for the visual odometry problem has not been proposed yet. However, this section covers the recent implementations that may lay the foundation for future successful vision-based navigation solutions.

Firstly, Orchard, Benosman, et al. (2013) presented an architecture, connected to the ATIS sensor, to replicate energy-based methods for optical flow estimation (Section 2-2-3). In order to add a temporal aspect to the spatial receptive fields, input neurons were connected to a second layer of LIF cells using synapses with different time delays. As shown in Figure 4-4, these synapses are configured in such a way that they act as velocity-tuned filters. Despite the capability of this network to accurately estimate optical flow even in unstructured environments, the number of filters needed precludes its real-time implementation.

Inspired by the Elementary Motion Detector (EMD) (Hassenstein & Reichardt, 1956), Giulioni et al. (2016) presented an event-based adaptation using a combination of LIF spiking neurons with excitatory and inhibitory synapses. To assess the validity of this approach, the authors implemented these motion detectors in a custom Very-Large-Scale Integration (VLSI) chip called FLANN (Giulioni et al., 2008). However, the large number of motion detectors needed for an accurate estimation of optical flow in natural scenes again hinders real-time performance.

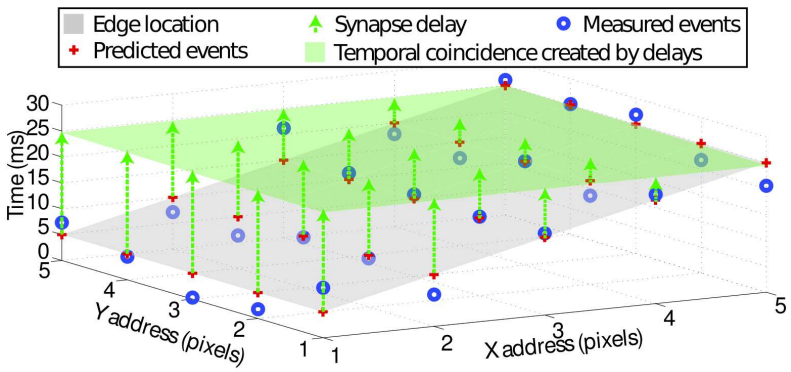


Figure 4-4: Illustration of the synaptic delays needed to capture the velocity of an edge moving according to the gray surface. From Orchard, Benosman, et al. (2013).

Finally, motivated by the event-based direction-selective filters covered in Section 3-2-3 (Brosch et al., 2015), Brosch & Neumann (2016) mapped the key computational elements of these filters onto the recently developed IBM TrueNorth neuromorphic chip (Merolla et al., 2014). Although this approach needs a large number of filters for robustness, the architecture of the TrueNorth chip allows real-time optical flow estimation with accuracy levels similar to the ones reported by (Brosch et al., 2015).

4-1-5 Simulators and hardware implementations

For the development of complex multi-layer SNNs, the availability of a fast running simulation framework supporting various neuron models and learning rules is of crucial importance. Biologically realistic simulators, such as the well known GENESIS (Bower & Beeman, 2012) and NEURON (Hines & Carnevale, 2006), are convenient for the behavior analysis of accurate biophysical neuron models. However, they are not suitable for simulations of large networks where the important aspect is not the model, but the interaction between neurons. For this purpose, event-driven simulators, such as SpikeNET (Delorme et al., 1999), DAMNED (Mouraud et al., 2005), NEST (Gewaltig & Diesmann, 2007), and Brian (Goodman & Brette, 2008), were successfully developed with the benefits of parallel computing.

Defining neuromorphic hardware as the electronic devices used for replicating computation in neural networks, this hardware can be divided in two main different groups. The first group is comprised of devices that can be adapted for replicating the structure of a particular neural network, for instance, FPGA boards and VLSI microchips. According to Paugam-Moisy (2006), spiking networks have successfully been implemented in these devices. However, apart from the design complexity, real-time performance is not ensured for large multi-layer architectures. The second group consists of hardware specifically designed for the efficient emulation of large-scale spiking neural networks, where TrueNorth (Merolla et al., 2014) and SpiNNaker (Khan et al., 2008) are the greatest exponents. On the one hand, SpiNNaker uses six interconnected boards for the real-time simulation of one million spiking neurons. On the other hand, TrueNorth potentially achieves a similar level of performance but just using one single chip with much lower power consumption (in the mW range). For these reasons, the development of the TrueNorth board is considered as a major achievement in the field of event-based neuromorphic systems, and in AI in general.

4-2 Artificial Neural Networks

Contrary to spiking architectures, ANNs do not encode information on the temporal aspects of presynaptic signals. Instead, the computations performed in neurons from these networks involve a weighted sum of the corresponding input values and a non-linear operation with an *activation function*. This computation leads, in theory, to the transformation of input signals into an output value that approximates the average firing rate of biological neurons (Ghosh-Dastidar & Adeli, 2009). For this reason, this coding scheme is referred to as *rate-based coding*. Although not biologically realistic (Section 4-1-1), this computational framework has successfully been used for a wide range of applications, especially since error backpropagation was proposed as an efficient learning algorithm (Rumelhart et al., 1988).

Due to the extensive literature available on artificial neural networks, this section focuses on the relevant aspects and common architectures needed for motion estimation and visual odometry applications. For more general information on artificial neural networks, please refer to the works presented by Demuth et al. (2014), and Goodfellow et al. (2016).

4-2-1 Basic components of artificial neural networks

For the introduction to the basics of artificial networks, this work refers to the theoretical knowledge collected by Sze et al. (2017) and Liu et al. (2017).

Instead of having multiple mathematical models as in the case of spiking neurons, the computation performed by artificial neurons is always divided in two main stages: the weighted sum of their inputs, and a non-linear functional operation that causes a neuron to generate an output only if the inputs exceed a certain threshold. Using Figure 4-5 as a reference, the output activation of the neuron under analysis is given by:

$$y_j = f\left(\sum_{i=1}^{p} w_{i,j} \cdot x_i\right) \tag{4-6}$$

where p denotes the number of presynaptic neurons, x_i their activation values, and $w_{i,j}$ the synaptic efficacies or weights. Note that, depending on the sign of $w_{i,j}$, synapses can also be categorized as excitatory or inhibitory. Further, $f(\cdot)$ represents the activation function that is commonly applied after the weighted sum in order to introduce non-linearities in the network. Examples of activation functions are the conventional sigmoid or hyperbolic tangent as well as the Rectified Linear Unit (ReLU) (Nair & Hinton, 2010), which has recently become popular due to its simplicity and advantages for learning.

Based on this computational principle, Figure 4-5 shows an illustration of the normal structure of an artificial neural network, the Multilayer Perceptron (MLP). For simplicity, this architecture is clearly divided in three distinct parts according to role played by each of the neurons. Starting from the left, the *input layer* receives the values that have to be processed for the generation of the desired output. For this purpose, these values are propagated to

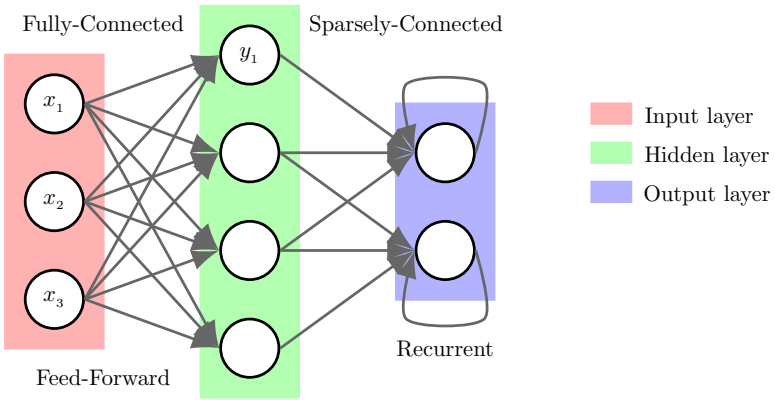


Figure 4-5: Schematic of a simple artificial neural network. Adapted from Sze et al. (2017).

neurons in the middle or *hidden layer*, and the output activations of these neurons are again propagated to the *output layer*, which presents the results to the user.

Additionally, from this figure it is also possible to infer that artificial neural networks process the data in two major forms: using feed-forward and recurrent connections. Networks with feed-forward connections are characterized by an output that has been generated as a sequence of operations on the activations of neurons from previous layers. According to Figure 4-5, two layers are *fully-connected* if all the presynaptic neurons have synapses with all the postsynaptic cells, and *sparsely-connected* otherwise. In contrast, recurrent connections allow the network to exhibit temporal behavior, since the output activation of one neuron is used as input to the same cell one timestep later. For these reasons, networks with recurrent connections are commonly employed with sequential input data, while feed-forward connections assume that this information is completely uncorrelated.

4-2-2 Deep Learning architectures

Neural architectures with only a few hidden layers are commonly referred to as *shallow* networks, being the one shown in Figure 4-5 a clear example of this category. These networks, although successfully trained with supervised learning rules such as error backpropagation, are limited due to the low-complexity of the input-output mappings that can be approximated. For this reason, the understanding and application of learning methods to neural architectures with more representational complexity has been an appealing research topic since Rumelhart et al. (1988). This line of work is known as Deep Learning (DL), and, as a consequence, the architectures employed are referred to as Deep Neural Networks (DNNs). The availability of large amounts of information, computational capacity, and open-source frameworks, has contributed to the recent success of DL techniques in numerous contests in pattern recognition and machine learning (Schmidhuber, 2015).

The two most frequently-used neural architectures in computer vision applications, such as image or video classification and visual odometry, are convolutional and Long Short-Term Memory (LSTM) layers. The computations, benefits, and some implementations of these schemes are now explained, together with a special mention to Mixture Density Networks (MDNs).

Convolutional layers

Presented by LeCun (1989), convolutional layers are especially designed for processing data that comes in the form of multiple arrays. For instance, time-series data can be thought of as a one-dimensional grid with samples taken at regular time intervals; and image data is usually comprised of two-dimensional arrays containing pixel intensities. Neural networks with at least one convolutional layer are categorized as Convolutional Neural Networks (CNNs).

Considering the solid red square in Figure 4-6 as an artificial neuron, it is possible to infer that, in a convolutional network, these cells are organized in feature maps. Each of these neurons is connected to local patches in the feature maps of the previous layer via a weight matrix called *convolutional kernel* (LeCun, 1989). The location of these patches strictly depends on the location of the neuron under analysis; and obviously, they are highly overlapped.

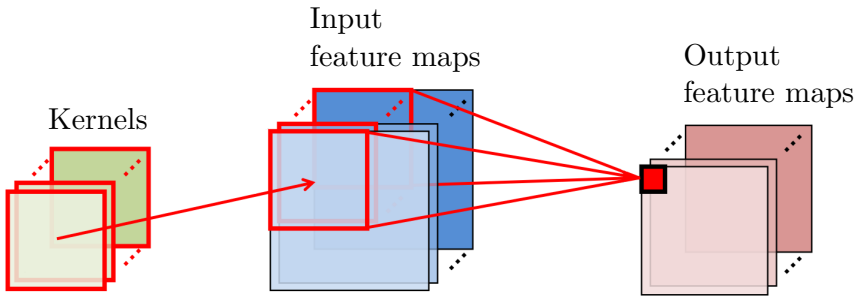


Figure 4-6: Illustration of the convolution operation performed in convolutional layers. Adapted from Sze et al. (2017).

Further, all the neurons in a particular feature map share the same kernel (*weight sharing*). This computation is known as *convolution* in the context of artificial networks. According to LeCun et al. (2015), there are two major reasons for the use of convolutional layers with this type of data. First, there is correlation within local groups of values, which implies that distinguishable "features" are commonly formed. Second, these features are invariant to location.

Similarly to shallow neural networks, an activation function is commonly applied after each convolutional layer to introduce non-linearities in the network. However, unlike these architectures, convolutional layers are also usually followed by a *pooling layer* in order to progressively reduce the dimensions of the input data. As reported by LeCun et al. (2015), units of these layers merge detected features into one single neuron by computing the maximum of a local patch in one of the output maps. This form of pooling is referred to as "max-pooling", and it is the most common form of performing this operation since it provides invariance to small shifts in the input image.

Note that these networks gained special attention from the computer vision and AI communities after the ImageNet competition (Russakovsky et al., 2015) in 2012. Here, a group of researchers from the University of Toronto managed to decrease the error rate from 25% to 10% using the "AlexNet" network (Krizhevsky et al., 2012), which was comprised of five convolutional layers fully-connected to three normal (dense) layers. This network set the basis for the development of more recent approaches such as "ResNet" (He et al., 2016), a very deep (152 layers) convolutional network which exceeded human-level accuracy in this competition.

Long Short-Term Memory

In contrast to conventional recurrent connections (Figure 4-5), the LSTM, presented by Hochreiter & Schmidhuber (1997), is a Recurrent Neural Network (RNN) architecture that is characterized by a special feedback unit called *memory cell*. For a better understanding of the working principle of this unit, note that a recurrent network can also be represented by feed-forward connections over the temporal axis, as illustrated in Figure 4-7.

According to the definition of recurrent network, feedback connections allow these architectures to maintain an internal state vector with information about the history of past samples of an input sequence. This internal state is illustrated in Figure 4-7 as the horizontal vector located at the top of each memory unit, represented as green surfaces. Based on Hochreiter

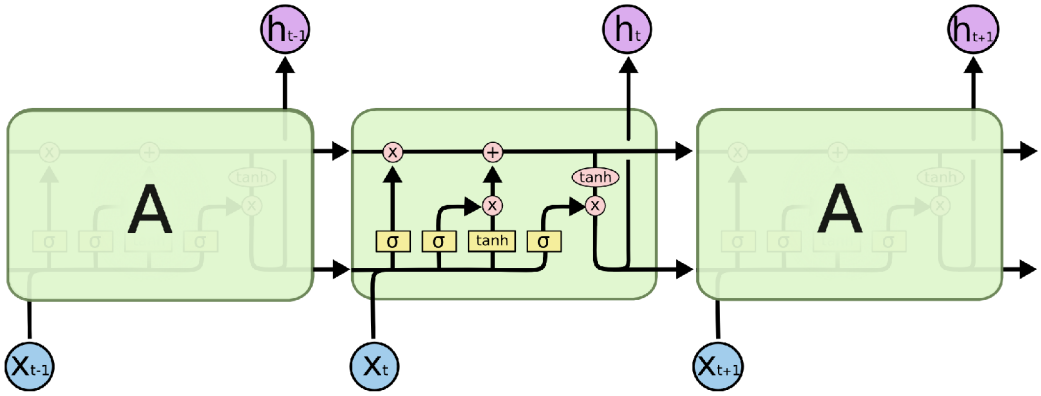


Figure 4-7: Representation of an LSTM layer by feed-forward connections over the temporal axis. The input activations (x_t) are propagated to the memory unit (A), which outputs an activation value (h_t) depending on the history of previous inputs. From Olah (2015).

& Schmidhuber (1997), these memory cells are characterized by four distinct computational units that interact with the state vector to update or retrieve information. Starting from the left, the first unit is the so-called "forget-gate layer", and its role is to decide whether the information in the state vector is valid or not. This decision is made by a sigmoid function with binary outputs. The second sigmoid unit, known as the "input-gate layer", decides which information from a vector of candidate values is used to update the internal state. These candidates are obtained as the output of an hyperbolic tangent function, the third of the computational units. Finally, the fourth unit is used to decide which components of the filtered internal state vector are the outputs of the memory block. Based on previous research works about the validity of LSTM, LeCun et al. (2015) argued that this architecture has proved to be more effective than conventional RNN for learning long-term dependencies.

One of the most successful applications of LSTM networks is speech recognition. Graves et al. (2013) proposed a recurrent architecture with several LSTM layers for each time step, which enabled the transcription of acoustic signals with an error rate considerably smaller that previous approaches. Apart from speech recognition, this type of recurrent network has robust performance in other tasks such as image captioning, hand-writing generation (Graves, 2013), and visual odometry (Wang et al., 2017); always working in cooperation with a set of convolutional layers.

Mixture Density Networks

When using error backpropagation as supervised learning rule in a DNN, the two most frequently-used cost functions to be minimized are the residual sum-of-squares and the cross-entropy error function (Goodfellow et al., 2016). According to Bishop (1994), the minimization of these functions leads to a neural network architecture that approximates the conditional averages of the data used for training. Although these averages can be regarded as optimal for classification problems with a suitable chosen training dataset; the description provided when used for regression of continuous variables is very limited. To overcome this problem, Bishop (1994) presented the Mixture Density Networks (MDNs) as a combination of conventional artificial neural networks and a mixture density model, as shown in Figure 4-8.

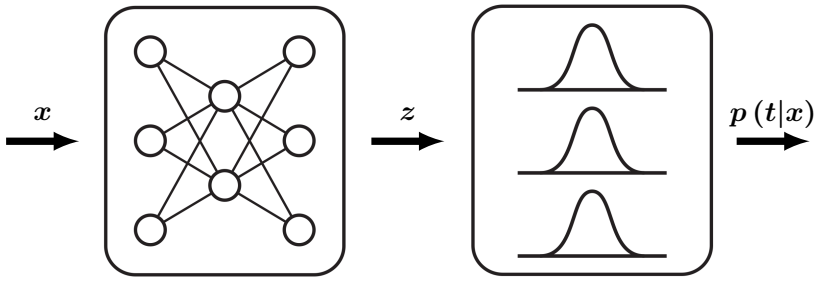


Figure 4-8: Schematics of the feed-forward architecture of Mixture Density Networks. Adapted from Bishop (1994).

These architectures enable the modeling of the conditional probability distributions needed to obtain a complete description of the target data.

The idea behind MDNs is that, first, from a set of input data $\boldsymbol{x}$, the neural network architecture is used for computing the parameters $\boldsymbol{z}$ in a mixture density model. Then, based on $\boldsymbol{z}$, the mixture model represents the probability density function of the target variables $\boldsymbol{t}$, conditioned on the input. Let this probability density be represented as the following convex combination of m Gaussian components:

$$p\left(\boldsymbol{t}|\boldsymbol{x}\right) = \sum_{i=1}^{m} \pi_i\left(\boldsymbol{x}\right) \mathcal{N}\left(\boldsymbol{t}|\mu_i\left(\boldsymbol{x}\right), \sigma_i^2\left(\boldsymbol{x}\right)\right) \tag{4-7}$$

where the parameters $\pi_i(\boldsymbol{x})$ are called *mixing coefficients*, and are a representation of the conditioned prior probability of $\boldsymbol{t}$ being generated from the i^{th} component of the mixture. Further, $\mu_i(\boldsymbol{x})$ and $\sigma_i^2(\boldsymbol{x})$ denote the means and variances of the Gaussian components, respectively. Since these parameters completely describe the mixture model, according to Figure 4-8, they compose the vector $\boldsymbol{z}$ that is estimated from the conventional neural network using $\boldsymbol{x}$ as inputs. Finally, $\boldsymbol{t}$ is estimated from the MDN architecture as a random value from the Gaussian component with highest mixing coefficient.

The error function to be minimized using error backpropagation is defined as the negative logarithmic of the likelihood over a finite set of n training examples $\{\boldsymbol{x}^q, \boldsymbol{x}^q\}$ (Bishop, 1994):

$$\mathcal{L} = -\sum_{q=1}^{n} \ln\left\{\sum_{i=1}^{m} \pi_i\left(\boldsymbol{x}^q\right) \mathcal{N}\left(\boldsymbol{t}^q|\mu_i\left(\boldsymbol{x}^q\right), \sigma_i^2\left(\boldsymbol{x}^q\right)\right)\right\} \tag{4-8}$$

4-2-3 Motion estimation and visual odometry

Due to the recent progress in the field of DL, extensive research on vision-based motion estimation has been conducted in the last couple of years. Instead of using the traditional methods (Chapters 2 and 3), the research trend is to approach these optimization problems from the perspective of DNNs. This section covers the main architectures that have recently been proposed for optical flow estimation and visual odometry. Note that, in general, these

networks gain spatial representation from a set of convolutional layers; and temporal characteristics by comparing consecutive images or by recurrent connections. Further, the learning process in all these solutions is performed using error backpropagation.

Optical flow estimation

The first application of deep networks to motion estimation was presented by Dosovitskiy et al. (2015), with their "FlowNetSimple" and "FlowNetCorr" neural architectures. According to this work, these end-to-end convolutional networks are capable of estimating dense optical flow fields in real-time from pairs of consecutive images. In order to have a dense output instead of a set of discrete values, these networks are clearly divided in two main sections. First, a convolutional part, comprised of eight layers, coarsely encodes motion information in a group of feature maps. Second, an *up-convolutional* section (Eigen et al., 2014) transfers this high-level information to a fine prediction using the kernels previously employed during convolution.

FlowNetSimple and FlowNetCorr differ in how the temporal dimension is introduced in the network. On the one hand, FlowNetSimple stacks the input colored pair of images into a single frame with six channels and applies normal convolution. On the other hand, FlowNetCorr separately performs convolution on these images; and then it combines the resulting feature maps with a *correlation layer*. According to Dosovitskiy et al. (2015), both networks achieved similar levels of accuracy, but they did not reach the performance reported by other state-of-the-art dense optical flow methods, such as EpicFlow (Revaud et al., 2015) or DeepFlow (Weinzaepfel et al., 2013). One year after the proposal of these architectures, Ilg et al. (2017) presented "FlowNet 2.0", a neural network generated by stacking FlowNetSimple and FlowNetCorr that improved the accuracy maintaining the real-time performance.

Based on the FlowNet architectures (Dosovitskiy et al., 2015), other authors have presented alternative approaches that slightly improved the accuracy levels of the original networks. Examples of these proposals are the works by Zweig & Wolf (2016), Ranjan & Black (2017), and Güney & Geiger (2016). Firstly, Zweig & Wolf (2016) used edge images (i.e., frames with information about sharp changes in brightness) as input to a convolutional network with a similar architecture to FlowNetSimple. The authors reported faster learning and lower error rate. Secondly, Ranjan & Black (2017) proposed a spatial-pyramid formulation to deal with large motions between frames. In this work, the authors rescale the images to

Figure 4-9: Comparison of the dense optical flow fields estimated with FlowNet and FlowNet 2.0. From Ilg et al. (2017).

create a pyramid representation of the input, and trained a FlowNetSimple network at each level. Finally, Güney & Geiger (2016) presented an innovative approach that consisted of *dilated convolutions* (Yu & Koltun, 2015), instead of using conventional pooling layers. These special architectures are characterized by the fact image resolution does not decrease, which is beneficial for the efficient estimation of dense information. In this work, dense fields of optical flow were obtained by connecting this network to the interpolation method employed in EpicFlow (Revaud et al., 2015).

Apart from these FlowNet adaptations, there is one extra approach that is worth mentioning: the work presented by Teney & Hebert (2016). Here, the authors proposed a neural architecture for motion estimation that differs from the methods already presented in two main aspects. Firstly, inspired by the biologically-realistic direction-selective filters introduced in Sections 2-2-3 and 3-2-3, spatiotemporal (three-dimensional) kernels are employed in the first convolutional layer of this network. Note that, since the temporal dimension of these kernels equals the number of images in the input sequence, the output consists of a set of two-dimensional feature maps. Hence, the rest of the network performs spatial convolution. Secondly, in order to boost performance and to ensure invariance to feature rotation, the gradients used for training are aligned with a specific canonical orientation. By doing so, the number of convolutional kernels significantly decreases, and motion can be robustly estimated by rotating them a set of different angles. As stated by Teney & Hebert (2016), this constraint removes the necessity of up-convolving to obtain dense information. Further, accuracy levels were improved with respect to the original FlowNet architectures.

Visual odometry applications

To the best knowledge of the author, the first end-to-end artificial neural network successfully applied to solve the visual odometry problem corresponds to Konda & Memisevic (2015). In this work, the authors proposed a convolutional network that, after estimating depth from a set of stereo images, is capable of predicting velocity and changes in the direction-of-motion. However, in order to do so, the network employs a *softmax layer*[2]. Hence, Konda & Memisevic (2015) formulated visual odometry as a classification problem rather than as pose regression.

Based on this work, but now using monocular configuration, Kendall et al. (2015) presented a CNN capable of estimating ego-motion as a regression problem. For this purpose, this architecture is based on the correspondence of images obtained from a conventional camera with a set of key-frames stored in memory, which are labeled with pose information. Since this solution highly depends on the ability of the network to recognize a particular scene but from different locations, GoogLeNet (Szegedy et al., 2015), a very deep open-source image classification network, is used as a basis for the development of this deep convolutional network. Although GoogLeNet has been pre-trained in very large classification datasets, the minor modifications included to perform pose regression imply that this architecture has to be re-trained, or at least fine-tuned, when applied in a new environment. According to Kendall et al. (2015), the accuracy of their work scales with the number of nearby images used for the characterization of a scene.

[2]Neural layer frequently used in neural networks with classification purposes. It results in the probability of the network's output to belong to one of the mutually exclusive classes of a particular problem.

In order to have a solution that could be used independently of the environment type, Costante et al. (2016) presented three different convolutional architectures that were provided with dense optical flow information instead of colored images. In this work, the authors evaluated whether convolution should be performed globally over the entire optical-flow frame, locally over small patches, or globally and locally in parallel. According to their results, all these architectures are able to learn appropriate features for the task of visual odometry, being the parallel architecture the one with highest accuracy.

Using optical flow information as input for a deep network has also been exploited by the approach presented by Pillai & Leonard (2017). However, their architecture differs from the previous method in the fact that it uses sparse flow vectors instead of dense images. Since a flow vector cannot be represented in the array format needed by convolutional layers, Pillai & Leonard (2017) proposed, for the first time, the utilization of Mixture Density Networks (MDNs) for the task of visual odometry. This particular architecture is used to compute the conditional probability of the pose for each flow vector. Probabilities that are further fused in order to get the camera pose at a particular instant in time. In order to ensure the coherence of the results, the authors employ a cost function that contains information not only about the estimation of the current pose, but also about the trajectory of the agent in the past. As shown by Pillai & Leonard (2017), this fully trainable end-to-end algorithm is sufficiently powerful to robustly estimate ego-motion. However, the main drawback of this approach and the network presented by Costante et al. (2016) is that they require pre-processed optical flow information as input.

To overcome this pre-processing requirement, the works presented by Wen (2016) and Wang et al. (2017) propose two similar recurrent convolutional neural networks that use a pair of colored images as input for each timestep. Starting with the former approach, Wen (2016) introduced "VINet", an architecture that combines the FlowNetCorr convolutional network (Dosovitskiy et al., 2015) with an LSTM unit fusing high-level image features with information from an on-board IMU. According to the author, the utilization of recurrent connections and inertial data allows this neural network to estimate ego-motion with levels of accuracy never achieved before by a DL approach. Further, Wen (2016) performed a result comparison between this network and the EKF-based PTAM solution proposed by Weiss et al. (2012). In these experiments, VINet outperformed its competitor in terms of translation error, but suffered in the estimation of orientation.

Using this approach as a basis, Wang et al. (2017) presented "DeepVO", the latest pure vision-based solution to the problem of visual odometry using conventional images as input. As shown in Figure 4-10, this deep network consists of a convolutional part directly connected to two LSTM recurrent layers. By combining these two architectures, this solution is capable of learning high-level features for pose estimation, while effectively modeling the implicit sequential dynamics. In order to reduce the training time and the amount of data needed for convergence, Wang et al. (2017) opted for the utilization of the pre-trained FlowNetSimple convolutional network (Dosovitskiy et al., 2015) as the basis for their architecture. For this reason, the input to this network is also generated by stacking two successive colored images together, as illustrated by Figure 4-10. This architecture was validated by comparing translation and rotation errors with the monocular and stereo versions of the solution proposed by Weiss et al. (2012), in the KITTI VO/SLAM benchmark (Geiger et al., 2012). Based on this comparison, the authors verified that this DL solution can be used for obtaining accurate visual odometry results with precise scale and in new environments.

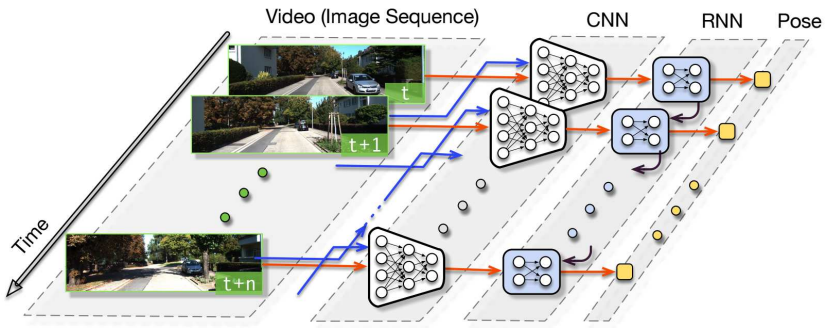


Figure 4-10: Architecture of the recurrent convolutional network proposed by Wang et al. (2017) for pose estimation. From Wang et al. (2017).

Finally, contrary to the approaches presented by Wen (2016) and Wang et al. (2017), Nguyen et al. (2017) recently published the first deep neural architecture that, firstly, estimates pose information from a single image using convolutional and recurrent layers; and secondly, makes use of the DAVIS event camera as neuromorphic input sensor. Instead of using FlowNetSimple as a basis for the convolutional segment, Nguyen et al. adapted the VGG16 model (Simonyan & Zisserman, 2014) by removing the softmax layer and stacking two LSTMs with 256 hidden units each. The event sequences chosen for experimenting with this architecture correspond to the event-based dataset presented by Mueggler et al. (2017); however, it is not specified how the six sequences comprising this dataset are divided into training, validation, and testing sets. Since overfitting is very common when dealing with recurrent networks (Goodfellow et al., 2016), the fact that no further information is provided about the training process demands a careful interpretation of the reported accuracy of this method. With respect to the pre-processing step of obtaining images from the DAVIS camera, Nguyen et al. (2017) opted for accumulating events over a temporal window of pre-defined length, assigning intensity levels depending on the polarity of each event.

4-2-4 Deep Learning frameworks

The recent progress in the field of DL has led to the emergence of different software libraries, the so-called *frameworks*, whose aim is to facilitate the utilization of neural networks to the general public. Among others, it is possible to remark TensorFlow (Abadi et al., 2016), Theano (Bastien et al., 2012), and Caffe (Jia et al., 2014). The positive aspect of these frameworks is that they have most of the frequently-used neural architectures (e.g., fully-connected, convolutional, and recurrent layers) and optimization techniques (e.g., error backpropagation) already implemented. Furthermore, they use Graphics Processing Units (GPUs) for parallel computing, ensuring fast training and low execution time. Note that these libraries are considered as low-level DL frameworks, i.e., a considerable amount of code needs to be written in order to train a simple shallow network, such as the one illustrated in Figure 4-5, for a small regression problem. High-level libraries such as Keras[3] and Lasagne[4] are commonly used as Python interfaces that work on top of TensorFlow and Theano.

[3]Keras documentation can be found in `https://keras.io/`
[4]Lasagne documentation can be found in `https://lasagne.readthedocs.io/`

Chapter 5

Synthesis of the Literature

A comprehensive literature study was conducted as the first part of the preliminary thesis presented in this document. The topic of this research was vision-based navigation for MAVs; however, three main areas were analyzed in detail, accentuating the role of optical flow in motion estimation. First of all, gaining inspiration from biology, valuable insights were obtained into the use of the visual observables derived from optical flow fields as basis for the development of efficient navigation controllers. Secondly, due to the asynchronous nature of the vision sensor used in this work, an overview of the existing approaches to event-based optical flow estimation was presented, together with some of the most relevant visual odometry solutions for this type of cameras. Finally, spiking and artificial neural architectures were presented as neuromorphic computational frameworks capable of learning the input-output mappings used for optical flow and motion estimation, without the need of simplifying assumptions. The vision-based navigation approaches developed with these architectures form a proper basis for the design of a network that, using the DVS as input sensor, is capable of efficiently and quickly estimating the desired visual observables. This chapter synthesizes and addresses the relevance of the main findings on each of these topics.

5-1 Optical-flow-based navigation strategies

Following the formulation proposed by Longuet-Higgins & Prazdny (1980), the concept of optical flow was defined as a means to quantify the relative motion between objects in a static scene and a moving observer. For the estimation of this flow from a set of successive images, a set of different approaches were discussed, where the gradient-based Lucas-Kanade method (Lucas & Kanade, 1981) stands out by its efficiency and accuracy. However, due to the computational requirements of estimating observer's ego-motion states from optical flow information (Nistér, 2004), the so-called visual observables were presented as alternative non-metric parameters for the development of vision-based navigation systems. As argued, the strong biological component of the majority of these strategies leads to computationally efficient and successful navigation solutions that can be implemented in MAVs and other

robotic platforms (e.g., Herissé et al., 2012; Kendoul, 2014). According to the discoveries presented by Baird et al. (2013), honeybees control pure vertical motion through the estimated values of time-to-contact or flow divergence. This strategy, together with horizontal control loops based on ventral flow components, enable maneuvers such as horizontal motion, hovering, and vertical landing; thus providing MAVs with their characteristic VTOL capabilities. Based on the experimental results reported by Ho & de Croon (2016), the main limitations of these strategies are due to the noise and latency of state-of-the-art methods for optical flow estimation from frame-based cameras.

5-2 Event-based vision sensors and optical flow estimation

Inspired by biological retinas, event-based vision sensors were presented as an alternative to conventional frame-based cameras that increases the temporal resolution and decreases the latency of these sensors. Instead of measuring brightness values at fixed time intervals, each of the pixels of these sensors operates asynchronously detecting local changes in brightness, and encoding this information in a set of sparse events transmitted using the AER communication protocol. Among other designs (e.g., ATIS, DAVIS), the main specifications, advantages, and limitations of the Dynamic Vision Sensor (DVS) (Lichtsteiner et al., 2008) were presented, since it corresponds to the vision sensor employed in this work.

As argued by several scientific researchers (e.g., Censi & Scaramuzza, 2014; Brosch et al., 2015), event-based sensors may lead to a gain in efficiency in the estimation of optical flow. Note that, besides being generated at a very high temporal resolution, the asynchronous events already contain dynamic information about the observer's ego-motion. For this reason, extensive research is being conducted towards the adaptation or development of new optical flow methods capable of benefiting from the use of these sensors. Examples of these approaches are the works presented by Benosman et al. (2012), Benosman et al. (2014), and Brosch et al. (2015). Regarding event-based navigation strategies, the work of Hordijk et al. (2018) consisted in the first application of the DVS in the control loop of a MAV. As reported by the authors, the event sensor enabled the vision-based execution of landing maneuvers at high speed.

5-3 Neuromorphic computing for vision-based navigation

Neuromorphic engineering was introduced as the field of AI that works towards the development of mathematical models capable of mimicking some of the most relevant functionalities of the brain: the possibility of learning input-output mappings, and its computational efficiency. Similarly to event-based vision sensors, neurons in a biological brains exchange information via spikes in an asynchronous fashion. Since SNNs attempt to replicate this communication protocol, they were presented as a powerful computational framework that, together with the DVS as input sensor, could give rise to a new generation of optical flow methods. Due to fact that these spiking computing systems remain in their early stages of development, learning to estimate motion has not be accomplished yet. However, several factors evidence the attainability of this goal.

First of all, extensive research is being conducted towards the adaptation of gradient-descent methods that, together with the error backpropagation technique, could serve as efficient supervised-learning rules for spiking networks. Although there is still no agreement on how to proceed due to the non-differentiable nature of these systems, promising results were presented by J. H. Lee et al. (2016) and Wu et al. (2017). The former method is based on the assumption that spikes are just noise in the measurement of the membrane potential, while the latter mathematically approximates the derivative of a pulse and back-propagates the error in the spatial and temporal dimensions. Both learning rules are validated in the task of image classification from event sequences.

Secondly, although learning was not involved, the networks proposed by Orchard, Benosman, et al. (2013) and Brosch & Neumann (2016) showed the possibility of estimating optical flow fields with these architectures. Orchard, Benosman, et al. (2013) proposed an spike-based adaptation of the Lucas-Kanade method (Lucas & Kanade, 1981) by manually tuning the synaptic time delays. Brosch & Neumann (2016) presented a real-time implementation of velocity-tuned filters in an IBM TrueNorth neuromorphic chip (Merolla et al., 2014). This latter approach is of relevant importance because it demonstrated that the biologically-realistic energy-based methods (see Section 2-2-3) are a real alternative to conventional optical flow estimation approaches; the main drawback being the required number of filters.

As a result of the current limitations of spiking computation, an overview of the basics of ANNs was presented, along with an introduction to common DL architectures, and some of their most relevant applications. Contrary to SNNs, these networks do not operate asynchronously, so their levels of efficiency and realism are not comparable. However, this computational framework has successfully been employed in a wide range of tasks, especially thanks to advances in the establishment of learning rules such as the aforementioned error backpropagation (Rumelhart et al., 1988). Among other research, Dosovitskiy et al. (2015) demonstrated the possibility of training a deep network to extract dense optical flow fields from a pair of successive images. Furthermore, Wang et al. (2017) outperformed traditional visual odometry approaches using a similar convolutional architecture stacked to a pair of recurrent LSTM layers. Based on these discoveries, this work includes a preliminary evaluation of the possibility of estimating ventral flow information from pre-processed events generated by a DVS sensor in a simulated environment. This analysis is included in Part III of this document.

Part III

Preliminary Evaluation of Motion Estimation using Deep Learning

Chapter 6

Methodology and Datasets

This analysis presents a preliminary evaluation of deep neural architectures in the task of estimating ventral flow information from events generated by the Dynamic Vision Sensor (DVS). For a better understanding of Deep Learning (DL), and based on the literature review included in Part II of this document, a preliminary implementation of the most common neural architectures was conducted. Further, due to the large amount of data needed for training and evaluating these networks –and any other DL design–, an event-camera simulator was adapted and employed, simplifying the time-consuming process of generating ground truth information with localization purposes. Finally, a pre-processing step to encode events as conventional frames was proposed, as a result of the incompatibility between this computational framework and the asynchronous spatiotemporal information obtained from the vision sensor.

The current chapter provides an introduction to this preliminary analysis. First of all, Section 6-1 briefly describes each of the steps involved in this process. The working principle of the DVS simulator is covered with details in Section 6-2, together with a description of the virtual environments used for data acquisition. Section 6-3 shows the mathematical derivation needed to retrieve ground truth ventral flow information from the evolution of the camera pose over time. Finally, Section 6-4 introduces the different approaches that have been employed for encoding events as images, along with a short explanation of the input datasets. Once this introductory part is done, the performance of Deep Neural Networks (DNNs) in the task of ventral flow estimation is evaluated in Chapter 7; and concluding remarks of this preliminary analysis are presented in Chapter 8.

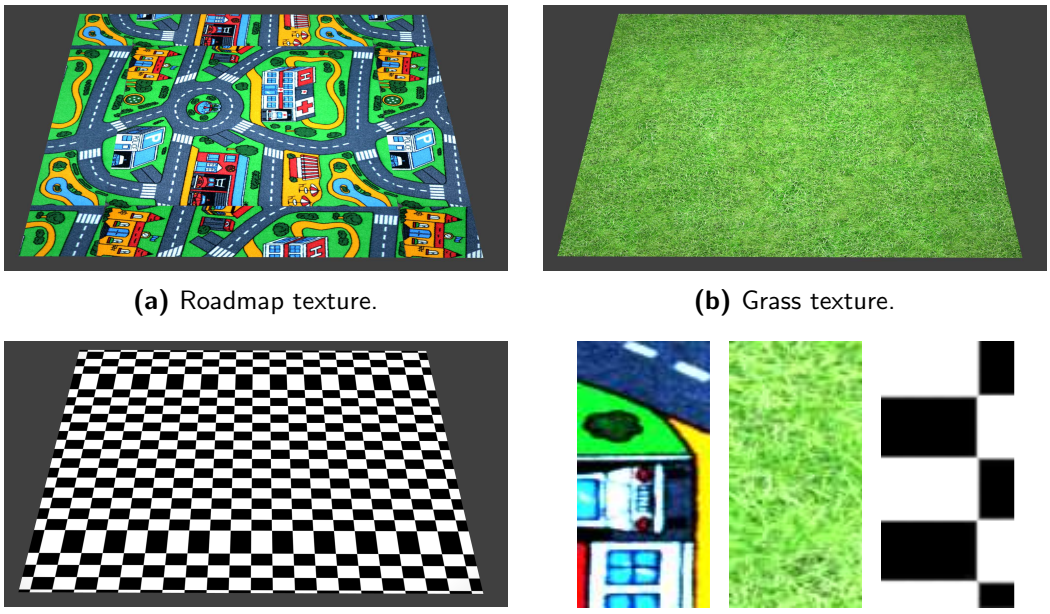
6-1 Outline of the analysis

As introduced in Chapter 2, the concept of ventral flow was defined as one of the visual cues that can be obtained from optical flow fields under the assumption of planar flow. Since the main intention of the work presented in this thesis is to be applied in a Micro Air Vehicle (MAV), the most likely situation in which this condition is fulfilled is when the aerial vehicle is equipped with a downward-looking vision sensor, the DVS in this case. Hence, the virtual

environments used in combination with the event-camera simulator only contain textured surfaces simulating the ground, as illustrated in Figures 6-1 and 6-3. For the generation of the dataset described in this chapter, the simulator needs to be provided with two main components: the scene from which it renders the images used for the extraction of events, and the trajectory of the camera. This trajectory, determined by camera pose and orientation, is the information from which ground truth ventral flow values are estimated.

The datasets generated for this preliminary analysis are divided into training, validation, and test sets. On the one hand, training and validation data are accessed during the learning phase. Here, the neural architecture adjusts its internal parameters (weights and biases) by means of comparing ground truth information with estimated ventral flows, always using data retrieved from the training set. The validation component, being generated from trajectories unseen during training, is employed to assess the network performance at fixed intervals, thus enabling to observe phenomenons such as learning and overfitting[1]. Moreover, based on the results obtained in this dataset, the so-called *hyperparameters* of the neural architecture (e.g., number of layers, size of convolutional filters) are fine-tuned until an acceptable configuration is found. On the other hand, once convergence is reached, the overall performance of the network is evaluated in a completely new part of the dataset, the test partition.

The neural networks evaluated in this analysis are implemented using TensorFlow (Abadi et al., 2016) and the Keras Python API. Further, all the simulations were performed on a Ubuntu 16.04 64-bit laptop equipped with an Intel Core i7-4790k quad-core CPU and an NVIDIA GeForce GTX 970M GPU. Note that the great majority of the operations are parallelized on the GPU by means of using the CUDA[2] libraries from NVIDIA.



(a) Roadmap texture.

(b) Grass texture.

(c) Checkerboard texture.

(d) Examples of rendered images.

Figure 6-1: Rendered images of the texture patterns used for data acquisition.

[1]Situation at which the deep neural architecture "memorizes" the training data, so that it is characterized by poor predictive performance in the validation and test datasets.

[2]CUDA documentation can be found in `https://developer.nvidia.com/cuda-toolkit`

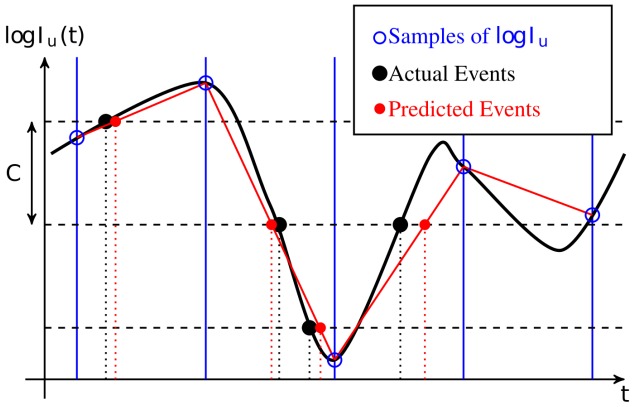


Figure 6-2: Working principle of the event-camera simulator. From Mueggler et al. (2017).

6-2 Dynamic Vision Sensor simulator

Instead of relying on advanced motion capture systems (e.g., OptiTrack, VICON), this preliminary analysis makes use of the event-camera simulator presented by Mueggler et al. (2017) for dataset generation. As mentioned before, using a three-dimensional virtual scene together with a predefined trajectory for the camera, this simulator renders intensity images at a high rate (1000 Hz) from which events are estimated. For this purpose, the computer graphics software Blender[3] is used in combination with the open-source Robot Operating System (ROS) package released by Mueggler et al. (2017). Figure 6-1 shows the textures employed in this preliminary analysis.

The working principle of the event-camera simulator is illustrated in Figure 6-2 for a single pixel $\boldsymbol{u} = (\hat{x}, \hat{y})^\top$ of the DVS. Inferring from this figure, the logarithmic brightness of the image at that particular pixel is sampled whenever an image is rendered from Blender. Once that more than one sample is available, these two values are linearly interpolated, and the resulting line is intersected with multiples of a predefined threshold C. Each of these intersections indicates the timestamp at which an event is generated. Further, the polarity is indicated by the slope of the estimated line, and the location is obviously given by the pixel at which this computation is performed.

There are two main concerns that need to be taken into account when working with this simulator. First of all, the default configuration of the virtual camera corresponds to the DAVIS sensor. Hence, before starting with the generation of the dataset, it is required to change the resolution and field of view of the camera to the correct values of the DVS, as indicated by iniLabs[4]. These modifications have to be directly made in Blender. For this preliminary analysis, a field of view of 70.8 degrees is used in combination with a pixel array of 128×128 in size. The second concern is about measurement noise. As stated by Mueggler et al. (2017), this simulator is noise-free due to the complexity of replicating the event noise that characterizes these sensors in real life. Event though this fact does not prevent this simulator from being a useful tool for the development of new event-based algorithms, special caution is needed when the transition between simulation and real life is performed. This is

[3]Blender documentation can be found in `https://www.blender.org/`
[4]DVS128 specifications can be found in: `https://inilabs.com/support/hardware/dvs128/`

especially true when the algorithm involves training or learning phases, as it is the case of the work presented in this preliminary thesis.

6-3 Determining ground truth ventral flow

As previously mentioned, the camera trajectory over the virtual textured surface is one of the inputs that have to be provided to the event-camera simulator for rendering a particular scene. Each of the entries of this trajectory file corresponds to the location and orientation of the camera, together with the timestep (in milliseconds) at which an image is obtained from the scene using Blender. Figure 6-3 illustrates the two frames of reference that are employed for the correct description of this information: the *inertial world reference frame*, $\mathcal{W}$; and the *body-fixed reference frame*, $\mathcal{B}$. Inferring from this figure, the location of $\mathcal{B}$ with respect to $\mathcal{W}$ is defined by the coordinates $(X_{\mathcal{W}}, Y_{\mathcal{W}}, Z_{\mathcal{W}})$. Further, its orientation results from a sequence of three rotational transformations defined by the Euler angles ψ, θ, ϕ, in order of rotation. The product of these three rotation matrices is denoted as $\mathbb{T}$. Since the simulator needs this orientation to be defined using quaternions (Diebel, 2006), the relation between these two forms of attitude representation is given by Eq. (6-1). Here, the shorthand notation $c_x = \cos(x)$ and $s_x = \sin(x)$ is introduced.

$$\begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} c_{\phi/2}c_{\theta/2}c_{\psi/2} + s_{\phi/2}s_{\theta/2}s_{\psi/2} \\ s_{\phi/2}c_{\theta/2}c_{\psi/2} - c_{\phi/2}s_{\theta/2}s_{\psi/2} \\ c_{\phi/2}s_{\theta/2}c_{\psi/2} + s_{\phi/2}c_{\theta/2}s_{\psi/2} \\ c_{\phi/2}c_{\theta/2}s_{\psi/2} - c_{\phi/2}s_{\theta/2}c_{\psi/2} \end{bmatrix} \tag{6-1}$$

Further, based on this quaternion representation, the linear transform $\mathbb{T}$ can be defined as:

$$\mathbb{T} = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2\left(q_x q_y + q_0 q_z\right) & 2\left(q_x q_z - q_0 q_y\right) \\ 2\left(q_x q_y - q_0 q_z\right) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2\left(q_y q_z + q_0 q_x\right) \\ 2\left(q_x q_z + q_0 q_y\right) & 2\left(q_y q_z - q_0 q_x\right) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{6-2}$$

Denoting the trajectory as $\boldsymbol{T}$, the structure of a particular entry i to this array is given by:

$$\boldsymbol{T}_i = \begin{bmatrix} t_i & X_{\mathcal{W}} & Y_{\mathcal{W}} & Z_{\mathcal{W}} & q_0 & q_x & q_y & q_z \end{bmatrix} \tag{6-3}$$

Due to the high temporal resolution at which the images are rendered from Blender, it is possible to accurately estimate the velocity of the DVS throughout the trajectory just by assuming constant speed between two successive frames. This operation leads to the definition of the body velocity components $(U_{\mathcal{W}}, V_{\mathcal{W}}, W_{\mathcal{W}})$ in the inertial frame of reference $\mathcal{W}$. These velocities are easily obtained in $\mathcal{B}$ as following:

$$\begin{bmatrix} U_{\mathcal{B}} \\ V_{\mathcal{B}} \\ W_{\mathcal{B}} \end{bmatrix} = \mathbb{T} \begin{bmatrix} U_{\mathcal{W}} \\ V_{\mathcal{W}} \\ W_{\mathcal{W}} \end{bmatrix} \tag{6-4}$$

For the computation of ground truth ventral flow values, the planar optical flow relations proposed by de Croon et al. (2013), and introduced in Section 2-1-2, are applied. However,

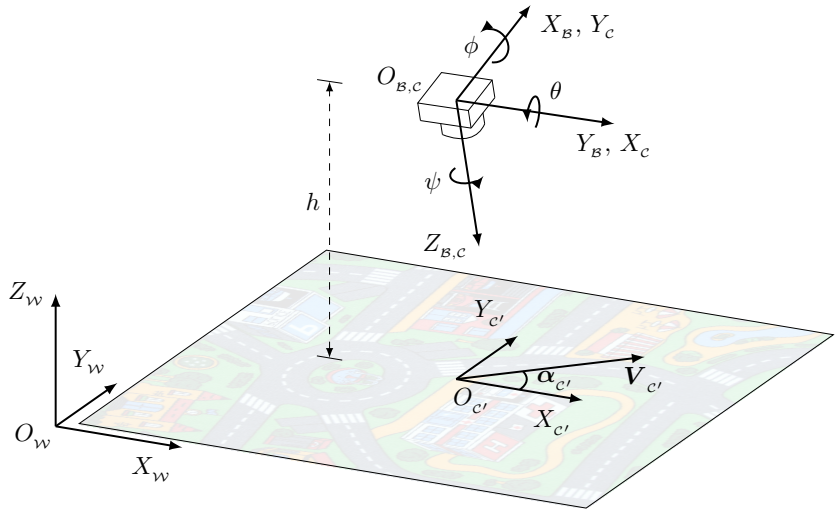


Figure 6-3: Schematic of the reference frames employed in this preliminary analysis.

since the derivation of these expression is based on the optical flow formulation of Longuet-Higgins & Prazdny (1980), a new frame of reference that matches the definition of pixel coordinates needs to be introduced. Hence, the *camera reference frame*, $\mathcal{C}$, is similarly defined as $\mathcal{B}$, but with the X- and Y-axis interchanged, as illustrated in Figure 6-3. Based on this, the DVS velocity components can now be computed in $\mathcal{C}$ as:

$$\begin{bmatrix} V_{\mathcal{C}} \\ U_{\mathcal{C}} \\ W_{\mathcal{C}} \end{bmatrix} = \begin{bmatrix} U_{\mathcal{B}} \\ V_{\mathcal{B}} \\ W_{\mathcal{B}} \end{bmatrix} = \mathbb{T} \begin{bmatrix} U_{\mathcal{W}} \\ V_{\mathcal{W}} \\ W_{\mathcal{W}} \end{bmatrix} \tag{6-5}$$

Finally, let Z_0 denote the distance to the planar surface along the optical axis $Z_{\mathcal{C}}$. Taking into account the fact that the camera does not always have to be perfectly aligned to the ground, this distance is computed using the following geometrical relation:

$$Z_0 = \frac{Z_{\mathcal{W}}}{\cos(\phi)\cos(\theta)} \tag{6-6}$$

According to de Croon et al. (2013), this distance is used for the computation of the main visual observables that characterize the camera's ego-motion under the assumptions that this formulation implies. These are the horizontal ventral flows and divergence, and they are obtained as follows:

$$\omega_x = -\frac{U_{\mathcal{C}}}{Z_0}, \quad \omega_y = -\frac{V_{\mathcal{C}}}{Z_0}, \quad D = -2\frac{W_{\mathcal{C}}}{Z_0} \tag{6-7}$$

Note that, as previously mentioned, this preliminary analysis is only focused on the estimation of ventral flow components.

6-4 Dataset description

Five datasets were recorded for this preliminary analysis by configuring the virtual DVS to move above a flat textured surface at a distance of $h = 0.5$ meters, as shown in Figure 6-3. Each of these datasets consists of multiple sequences of events, together with the corresponding ground truth ventral flow values derived with the expressions presented in Section 6-3. Depending on the intended use, these datasets are further divided into three different categories: training, validation, and test. As also illustrated in Figure 6-3, the reference frame $\mathcal{C}'$ is introduced as the projection of $\mathcal{C}$ onto the $X_{\mathcal{W}}Y_{\mathcal{W}}$-plane. Since the optical axis of the DVS is perfectly orthogonal to the surface in the experiments conducted for this analysis, the definition of this frame of reference enables a better understanding of how these datasets are obtained.

First of all, training and validation partitions are analyzed together since they are generated using the same principle and the roadmap texture as flat surface. The idea behind this approach is that the DVS, and therefore $\mathcal{C}$ and $\mathcal{C}'$, is located at a particular location defined by the coordinates $(X_{\mathcal{W}}, Y_{\mathcal{W}}, h)$, from which multiple event sequences are computed. These sequences are defined based on the combinations of the components of the velocity and angle vectors, $\boldsymbol{V_c}$ and $\boldsymbol{\alpha_c}$ respectively. For training, the dataset referred to as "Set 1" is characterized by a velocity vector with six components, ranging from 0.25 to 1.5 m/s; 24 orientations in the $X_{\mathcal{W}}Y_{\mathcal{W}}$-plane; and three different origins. Furthermore, all the trajectories last exactly one second. Hence, "Set 1" is defined as a set of $6 \times 24 \times 3 = 432$ sequences of events of one second of duration and constant speed. As previously mentioned in this chapter, the validation dataset is employed to assess the performance of the deep neural architecture during the training process. Hence, "Set 2" is comprised of event sequences from an unseen configuration of this approach. Again, this dataset is characterized by straight trajectories that last one second, but now the number of sequences comes down to 60. For a better understanding of the limits of these sets, Figure 6-4 shows the evolution of the ventral flow components as a function of the orientation angle.

Regarding the datasets corresponding to the test category, "Set 3", "Set 4", and "Set 5" are defined by a group of five sequences of events of six seconds of duration, generated from moving the virtual DVS through circular trajectories of radii $r \in [0.5, 1.5]$ meters, as shown

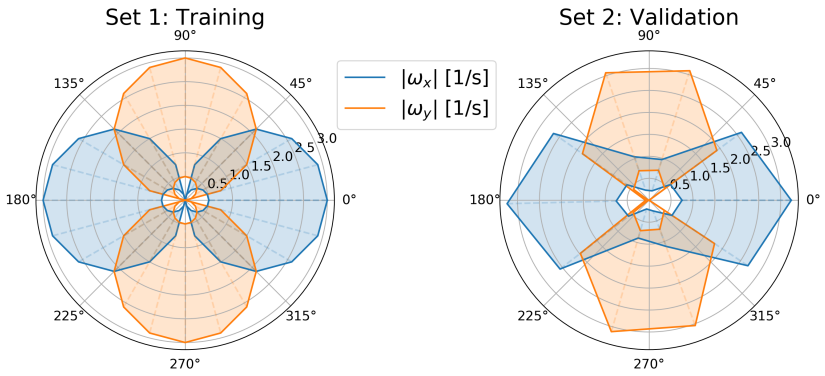


Figure 6-4: Range of ventral flow values of the training and validation datasets, as a function of the orientation angle.

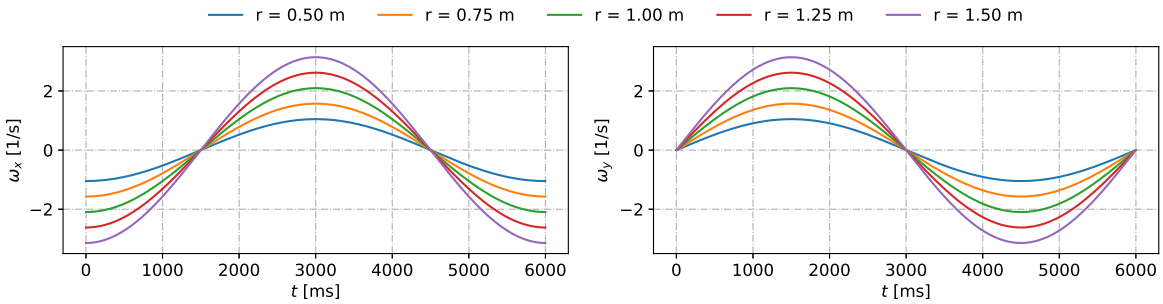


Figure 6-5: Temporal evolution of the ground truth ventral flow components of the Sets 3 and 4 of the dataset.

in Figure 6-5. For the correct assessment of the performance of the neural architecture, these trajectories are also simulated at an altitude of $h = 0.5$ meters. The only difference between these sets is due to the textured surface from which the simulator estimates the events, as explained in Section 6-2. Similarly to Sets 1 and 2, Set 3 makes use of the roadmap texture (Figure 6-1a), while the events from Set 4 correspond to a more natural environment defined by a grass pattern (Figure 6-1b), and those from Set 5 to the checkerboard (Figure 6-1c).

Figure 6-5 shows the temporal evolution of the ground truth ventral flow components of these datasets. Inferring from this figure, most of the combinations of ventral flow values remain within the limits of the training dataset, as illustrated in Figure 6-4. However, the main challenges that the network has to face with these test partitions are the fact that the trajectory is not straight anymore, but circular; and that two new textures are introduced. The results of this preliminary analysis will be of great help to decide whether more trajectories and textures need to be used during training for a better accuracy.

An overview of the main characteristics of these datasets is provided in Table 6-1.

Table 6-1: Summary of the main characteristics of the event datasets used in the preliminary analysis of this thesis.

	Texture	Num. sequences	Motion type	Duration [s]	Num. events (avg.)
Set 1	Roadmap	432	Straight	1.0	2.31e6
Set 2	Roadmap	60	Straight	1.0	2.41e6
Set 3	Roadmap	5	Circular	6.0	16.85e6
Set 4	Grass	5	Circular	6.0	8.86e6
Set 5	Checkerboard	5	Circular	6.0	25.78e6

6-4-1 From events to images

As introduced in Chapter 4, one of the main differences between Spiking Neural Networks (SNNs) and Artificial Neural Networks (ANNs) lies in the format in which input data is provided to these architectures. On the one hand, due to their asynchronous nature, SNNs receive temporally-separated pulses that, in computer vision, may indicate the occurrence of

an event at a particular spatial location in the image plane. Note that, here, the information is not contained in the amplitude of the pulse, but in the time at which it is generated. On the other hand, ANNs do not consider this temporal aspect. Instead, the input to these networks is represented as an array of numbers that is updated at fixed time intervals. When these networks are applied for vision purposes, this array normally represents an image, i.e., each element corresponds to the brightness value of a specific pixel. Since this preliminary analysis evaluates the possibility of learning to estimate motion using deep ANNs and the DVS, it is necessary to use suitable encoding schemes for representing events as static images.

Based on the working principles of the event-based optical flow estimation methods introduced in Section 3-2, the approach chosen for the generation of input images from DVS data consists in the accumulation of events over a temporal window of length Δt_f. These images, which are equal in size to the pixel array of the DVS, are initialized as empty canvasses every time the previous image is stored or processed by the network. Further, the brightness levels of their pixels are directly dependent on the polarity of the events that are within the temporal boundaries of the image under analysis. Based on this principle, there are two main factors to be specified that may have an impact on the performance of the neural architecture: the length of the accumulation window Δt_f, and the manner in which the polarity is encoded. Regarding the former parameter, temporal windows of 1.0, 2.5, 5.0, and 10.0 milliseconds of duration are employed in the preliminary analysis presented in this work.

For the polarity encoding scheme, three different configurations have been evaluated, as shown in Figure 6-6. Starting from the left, the Different-Channel-Same-Brightness (DCSB) representation generates images with two channels, each corresponding to a particular polarity. For visualization purposes, DCSB images are shown as colored frames in which the green channel encodes ON events, the red does the same with OFF events, and the blue channel is left empty. Continuing with the description, Same-Channel-Different-Brightness (SCDB) images are characterized by a single channel in which ON and OFF events are encoded using different brightness levels and a gray canvas. Finally, Same-Channel-Same-Brightness (SCSB) frames accumulate events in a single channel independently of their polarity.

Note that, due to the aforementioned input limitations of ANNs, the datasets summarized in Table 6-1 are always pre-processed by one of this three representation methods before serving as input to the neural architectures.

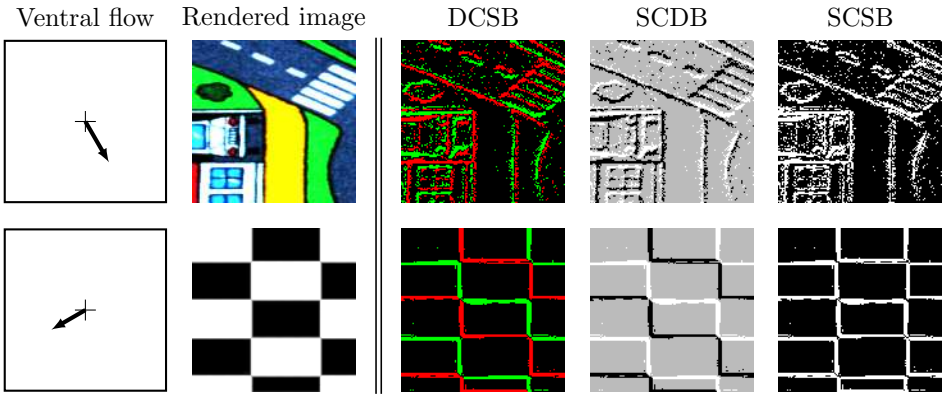


Figure 6-6: Visualization of the events accumulated over a temporal window of $\Delta t_f = 1.0$ ms. when a particular motion is performed over the roadmap and checkerboard textures.

Chapter 7

Ventral Flow Estimation using Deep Learning Architectures

Once an overview of the main pre-processing steps is obtained, this chapter focuses on the performance evaluation of DL techniques when applied for ventral flow estimation from event-based data. For this purpose, and to limit the scope of this analysis, two neural architectures are chosen based on state-of-the-art methods found in literature (see Chapter 4). All the reported experiments are conducted using these networks, whose details are presented in Section 7-1. Further, this section also introduces the general characteristics of the simulations run for training and testing these architectures according to the requirements of each experiment. Finally, Section 7-2 includes the results used for the aforementioned performance evaluation, and for the impact analysis of factors such as the length of the temporal window or the polarity encoding scheme.

7-1 Neural architectures and simulation details

This section serves as an introduction to the implementation details of the networks that, based on methods found in literature, have been designed for this preliminary analysis. Furthermore, the main aspects of the simulations used for training and testing this architecture under different conditions are also covered in detail.

7-1-1 Neural architectures for ventral flow estimation

The literature study conducted in Part II of this work revealed that, with the exception of a couple of proposals (e.g., Teney & Hebert, 2016; Pillai & Leonard, 2017), the great majority of DL architectures addressing the problems of visual odometry and optical flow estimation use the networks presented by Dosovitskiy et al. (2015) as a basis. Not only the neural structure is replicated, but also their weights and biases are normally transferred to the new architectures. As argued by Wen (2016) and Wang et al. (2017), the benefits of transfer

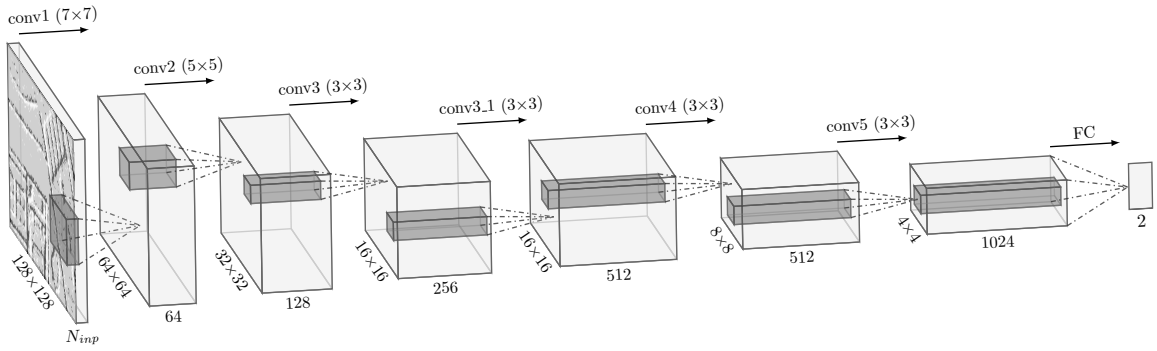


Figure 7-1: Architecture of the deep convolutional network employed in this preliminary evaluation of ventral flow estimation.

learning[1] come from the fact that, by applying it, a part of the system directly becomes optimal for the task under analysis before performing any training. As a consequence, only a small portion of the network needs to be notably adjusted, thus reducing the computational load of the training phase.

Based on this reasoning, the neural networks employed throughout this preliminary analysis gain inspiration from the convolutional segment of the FlowNetSimple network presented by Dosovitskiy et al. (2015). Apart from illustrating this architecture, Figure 7-1 also clearly describes its structure. Inferring from it, the network basically consists of six convolutional layers that, arranged in a feed-forward fashion, decrease the spatial dimension of the input while increasing the number and complexity of the extracted high-level features. Similarly to the work proposed by Dosovitskiy et al. (2015), the input is comprised by a set of N_{inp} stacked frames that, in this case, are generated by encoding DVS events as images (see Section 6-4). This input is fed through the deep network until two scalar values are computed, corresponding to the horizontal ventral flow components (ω_x, ω_y). The resulting architecture is categorized as a deep Convolutional Neural Network (CNN).

Apart from the configuration shown in Figure 7-1, there are three more factors to be specified about the structure of this architecture. First of all, in order to control the size of output feature maps, a common approach is to pad input volumes with a set of zeros around the border. For each layer of this convolutional architecture, the size of this *zero-padding* is chosen so that the size of the spatial dimensions of the input and output feature maps coincides. Secondly, instead of using pooling layers to progressively reduce the dimensionality of the input data, this network is characterized by *strided convolutions*. With the exception of "conv3_1", all the convolutional layers employed in this network use a stride[2] of two pixels. The combination of these strided convolutions and the aforementioned zero-padding results in output maps with half the size of the input spatial dimensions. Finally, all the convolutional layers are followed by ReLU layers (Nair & Hinton, 2010) in order to introduce non-linearities in the network.

An overview of the overall structure of this neural network is provided in Table 7-1.

[1]The concept of transferring the knowledge learned while solving a particular problem, to a model addressing a different but related problem using a similar –if not the same– architecture.

[2]Number of pixels that a convolutional kernel "jumps" at a time when applied over a particular feature map. A convolutional layer is said to be strided if this parameter is greater than one.

Table 7-1: Detailed structure of the deep convolutional network employed in this analysis.

Layer	Size [px^2]	Padding [px]	Stride [px]	Channels [-]
conv1	7×7	3	2	64
conv2	5×5	2	2	128
conv3	3×3	1	2	256
conv3_1	3×3	1	1	512
conv4	3×3	1	2	512
conv5	3×3	1	2	1024

Adding recurrency with LSTM layers

The deep convolutional architecture depicted in Figure 7-1 is meant to process an input volume comprised by different frames, and, by doing so, to estimate the horizontal ventral flow components that characterize the ego-motion of the simulated DVS. Since the weights and biases of this network remain constant in time after training, the estimated magnitudes of ventral flow only depend on the input under analysis. Neither previous inputs nor previous outputs have an impact on the current computations carried out by the network. For these reasons, this pure convolutional architecture is not capable of performing what is referred to as *sequential modeling*: the fact of finding temporal correlations among features extracted from a sequence of successive inputs.

However, as argued by Wang et al. (2017), useful information for the problem of estimating ventral flow may be encoded in the high-level features extracted from previous inputs, when the model is evaluated in a temporal sequence. For the assessment of whether addressing this problem as sequential is beneficial, recurrency is added to the deep convolutional network by means of using Long Short-Term Memory (LSTM) layers. The resulting architecture is categorized as a deep Recurrent Convolutional Neural Network (RCNN), and its structure is illustrated in Figure 7-2. Inspired by the proposals of Wang et al. (2017) and Nguyen et al. (2017), the main difference between this network and the aforementioned neural architecture is that two LSTMs layers, of 1,000 hidden units each, have been included in between of the convolutional segment, described in Table 7-1, and the final dense layer. The performance of this recurrent convolutional network and the aforementioned CNN is assessed in Sections 7-1 and 7-2.

7-1-2 Simulation and training details

For the evaluation of the previously introduced neural architectures, a simulation environment was created using Keras as the Python interface of a TensorFlow backend engine (Abadi et al., 2016). For a better understanding of the work, this section provides an overview of the three main aspects of this environment: data preparation, training, and testing.

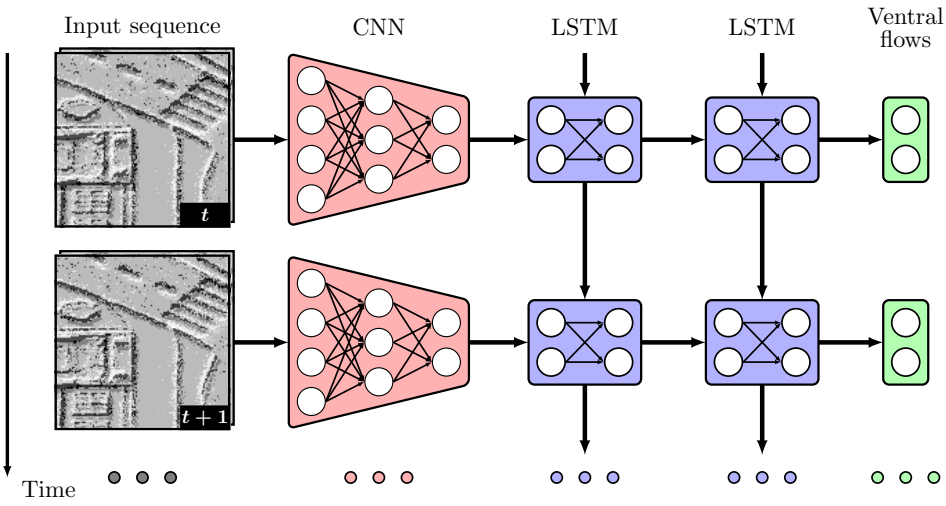


Figure 7-2: Schematic of the deep recurrent convolutional network employed for evaluating the impact of sequential modeling in the estimation of ventral flow. For visualization purposes, the SCDB encoding scheme is used to represent DVS events as one-channel images.

Data preparation

A detailed description of the datasets employed in this preliminary analysis was given in Section 6-4. Here, it became clear that there are two main parameters describing the pre-processing steps of these datasets: the length of the temporal window used for accumulating DVS events, denoted as Δt_f; and the encoding scheme chosen for representing these events as images. Each of the experiments conducted for this preliminary thesis is characterized by a set of pairs of these parameters (see Section 7-2).

Once the image-based representation of the datasets is obtained, the understanding of how these images are fed into the networks is of relevant importance. As previously mentioned, the neural architectures employed in this analysis are meant to process (in a single forward pass) a set of N_{inp} stacked frames for the estimation of ventral flow. Due to the high temporal resolution of the DVS, consecutive images may look too similar for the network to accurately estimate these observables. For this reason, the *frame-to-frame temporal separation*, Δt_{f2f}, is introduced as a parameter controlling which frames are selected in experiments with $N_{inp} > 1$. Once the selection is made, these images are normalized to the interval $[0, 1]$, and they are stacked together forming an input of size $(128, 128, N_{inp})$, where N_{inp} corresponds to the number of channels. Note that, in case of using the DCSB scheme for representing DVS events as images (see Section 6-4-1), the number of channels is doubled since ON and OFF events are encoded separately.

For successive forward passes of the network, the selection of the images to be fed into the network can be performed randomly or sequentially. On the one hand, if the network is characterized for not having recurrent connections (see Figure 7-1), frames from different passes are independent (i.e., they correspond to different trajectories within the same dataset). Hence, this selection is performed randomly. On the other hand, if the network has recurrent layers (e.g., LSTM), successive images need to be employed since these layers maintain an internal state vector with information extracted from previous time instants. Therefore, in this case, the selection of the input is sequential.

Training

The concept of training a neural architecture refers to the adjustment of the weights and biases that define its internal structure. Depending on whether ground truth data is used for tuning these internal parameters or not, this training can be categorized as supervised or unsupervised, respectively. Similarly to methods found in literature, the training performed in this preliminary analysis is supervised; and more specifically, the Adaptive Moment Estimation (Adam) optimization algorithm (Kingma & Ba, 2014) is employed in combination with the well-known error backpropagation method.

Supervised training using error backpropagation can be understood as a two-step process. First of all, the input is propagated through the neural architecture until the estimated values of the output, $\hat{\boldsymbol{\omega}}_i$, are obtained. This step corresponds to the *forward pass*. Having reached this point, $\hat{\boldsymbol{\omega}}_i$ is compared to the ground truth data associated to the input under analysis, $\boldsymbol{\omega}_i$, by means of using a *loss function*. Thereafter, the gradients of this function are computed with respect to all the internal parameters of the network in what is known as the *backward pass* of the error backpropagation technique. These gradients indicate the direction and magnitude (in the *weight space*) of the updates that have to be applied to each of these parameters in order to minimize the estimation error, ε_ω. Based on this reason, the optimization algorithms exploiting this information are categorized as *gradient-descent methods*. Among these algorithms, Adam (Kingma & Ba, 2014) stands out by the possibility of computing adaptive learning rates for each parameter. This is accomplished by keeping track of the temporal evolution of the mean and variance of each gradient. According to Ruder (2016), this optimization algorithm outperforms other adaptive-learning methods such as Adagrad (Duchi et al., 2011) and Adadelta (Zeiler, 2012); as well variations of the well-known Stochastic Gradient Descent (SGD). For more information on neural network optimizers, please refer to the work presented by Ruder (2016).

Regarding the specific details of the training simulations of this preliminary analysis, first of all, the loss function is defined as the Mean Squared Error (MSE) of the horizontal ventral flow components. As shown in Eq. (7-1), the optimal configuration of the network, $\boldsymbol{\theta}^*$, corresponds to the internal parameters that minimize this expression.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{N_b} \sum_{i=1}^{N_b} ||\hat{\boldsymbol{\omega}}_i - \boldsymbol{\omega}_i||_2^2 \tag{7-1}$$

From Eq. (7-1), N_b refers to the so-called *batch size*, i.e., the number of input samples used for the computation of the training loss, and thus of the gradients employed for the optimization. A trade-off is needed with N_b since small values of this parameter imply a loss in the accuracy of the estimated gradients, but large values require more memory and the networks become hard to train. For this analysis, $N_b = 16$ was found to be an appropriate value.

The process of computing the MSE over a particular batch and performing a parameter update based on this error is referred to as a *step* of the learning phase. The input data used in these steps is retrieved from the training dataset (Set 1). Once a certain number of training steps are completed, the validation phase commences. Here, the information is retrieved from the validation dataset (Set 2) in batches of N_b input volumes, and the network performance is evaluated using Eq. (7-1) as loss function. This process is repeated until a certain number of

validation steps is reached. Then, the average of the MSE values is presented (further referred to as *validation loss*), as it is indicative of the generalizability of the network to unseen data. Note that no parameter update is conducted in this part of the process.

When the validation phase finishes, the network is said to have completed an *epoch*. For this preliminary analysis, an epoch consists of 1,000 training and 500 validation steps. There is no limitation on the number of epochs per simulation, since it automatically stops when the validation loss does not improve for a period of ten epochs. According to Caruana et al. (2001), this technique, known as *early stopping*, is one of the most-used methods for avoiding overfitting in neural architectures.

Testing

As introduced in Section 6-1, once convergence is reached in the training process, the performance of the network is again evaluated, but now using the testing datasets: Sets 3, 4, and 5. During this process, instead of using the loss function defined by Eq. (7-1), the estimated ventral flow components are stored, and then compared to the corresponding ground truth data. The *mean absolute error* and the *error variance* are the measures used for assessing accuracy. Note that, for a fair comparison of the error metrics over the trajectories comprising the testing datasets (i.e., characterized by different radii), these parameters are commonly normalized in this analysis by the corresponding mean absolute flow, denoted by $\overline{\omega}_{x,y}$, of each trajectory.

7-2 Results

This section presents the results obtained from the experiments conducted for this preliminary analysis. These experiments are divided in two main groups depending on the number of frames comprising the input volume, N_{inp}. Firstly, in Section 7-2-1, the possibility of accurately estimating ventral flow information from multiple DVS images stacked together ($N_{inp} > 1$) is assessed using the previously introduced CNN (see Figure 7-1) and RCNN (see Figure 7-2). Moreover, a small sensitivity analysis is presented to evaluate the impact that some parameters, such as the length of the temporal window Δt_f and the encoding scheme, have on the overall performance of the networks. Secondly, in Section 7-2-2, the ventral flow estimation problem is approached again using the same neural architectures, but this time, with just a single DVS image as input ($N_{inp} = 1$).

7-2-1 Ventral flow estimation from multiple DVS images

As mentioned before, the convolutional segment of the neural architectures employed in this preliminary analysis is inspired by the FlowNetSimple architecture proposed by Dosovitskiy et al. (2015). With an input consisting of a pair of successive images, their network was shown to estimate dense optical flow fields with competitive levels of accuracy at a rate of 10.0 FPS. Since two consecutive images are temporally separated by a considerable time interval with this implementation, motion aspects (e.g., direction, speed) are clearly discernible to the naked eye, and thus to a neural network. Regarding the case of this analysis, due to the high

temporal resolution of the sensor, images can be obtained at a much faster rate than with a conventional frame-based camera independently of the length of the accumulation window Δt_f. Since two successive DVS images could be almost indistinguishable from each other, a new method of forming this input is needed.

In order to find a suitable configuration for the networks under analysis, the following two types of input volumes are evaluated: (1) an input format consisting of two stacked frames that are temporally separated by Δt_{f2f} milliseconds; and (2), an input that does not consist of two but N_{inp} successive images stacked together. These configurations are further referred to as "TS-I", and "C-I", respectively. The results of this analysis are presented in Table 7-2. Note that, for this first experiment, only the performance of the CNN is assessed, and this architecture was trained using Sets 1 and 2 with $\Delta t_{f,\text{T}} = 1.0$ ms.

Table 7-2: Ventral flow estimation errors (mean absolute error and variance) when the CNN is applied to Set 3 using different input formats. The lowest error in each of the columns is highlighted. The computation time (t_C) of each configuration is shown as well.

Set 3: CNN, DCSB, $\Delta t_{f,\text{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, $r = 1.0$ m

	N_{inp} [-]	Δt_{f2f} [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		t_C [ms]
			Mean*	Var*	Mean*	Var*	
TS-I_1	2	1.0	0.1038	0.0304	0.2016	0.1047	7.621
TS-I_5	2	5.0	0.0919	0.0237	0.0972	0.0242	7.621
TS-I_{10}	2	10.0	0.0655	0.0127	0.1110	0.0309	7.621
TS-I_{15}	2	15.0	0.0633	0.0119	0.0998	0.0263	7.621
TS-I_{20}	2	20.0	**0.0609**	**0.0098**	0.1146	0.0274	7.621
TS-I_{25}	2	25.0	0.0664	0.0108	0.1103	0.0337	7.621
C-I_5	5	1.0	0.0779	0.0171	0.0991	0.0240	9.248
C-I_{10}	10	1.0	0.1089	0.0329	0.1356	0.0435	10.297
C-I_{15}	15	1.0	0.0834	0.0188	0.0869	0.0151	11.017
C-I_{20}	20	1.0	0.0863	0.0196	0.0940	0.0195	11.865
C-I_{25}	25	1.0	0.0679	0.01191	**0.0786**	**0.0141**	12.353

*Normalized by $\overline{\omega}_{x,y} = 1.83$ 1/s

Based on the results presented in Table 7-2, it is important to remark that the convolutional network presented in Section 7-1-1 manages to learn the input-output mapping needed for ventral flow estimation under the conditions of this experiment, at least when evaluated in an environment characterized by the same texture as the training dataset (see Table 6-1). However, as expected, the network performance varies depending on the input format. On the one hand, with C-I, these results confirm the idea that motion information can be extracted from long sequences of consecutive images, even though the level of similarity among them is high. The main drawback of this approach is that, obviously, the computational requirements increase with the volume of the input. On the other hand, with TS-I, these results confirm that temporal separation is a key factor if motion wants to be estimated from the comparison of just a pair of images. Since the estimation errors of both configurations are similar, the input format selected for being the basis of the remaining part of this preliminary analysis corresponds to a TS-I configuration, more specifically, the TS-I_{15}.

Once an input configuration is selected, both the CNN and RCNN are now evaluated on the rest of the circular trajectories that comprise Set 3. This experiment is conducted especially for assessing the performance of these neural architectures when different combinations of (ω_x, ω_y) have to be estimated. Quantitative and qualitative results are presented in Table 7-3 and Figures 7-3 and 7-4, respectively.

Table 7-3: Ventral flow estimation errors obtained by evaluating both the CNN and RCNN on all the circular trajectories from Set 3. The mean absolute flow ($\overline{\omega}_{x,y}$) is shown for normalization.

Set 3: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, TS-I$_{15}$

	r [m]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
CNN	0.50	0.1375	0.0306	0.1367	0.0329	1.17
– " –	0.75	0.0907	0.0219	0.1152	0.0295	1.49
– " –	1.00	**0.0638**	**0.0119**	0.0995	**0.0263**	1.83
– " –	1.25	0.0679	0.0155	**0.0862**	0.0279	2.17
– " –	1.50	0.0809	0.0243	0.1037	0.0365	2.49
RCNN	0.50	0.1055	0.0230	0.2002	0.0702	1.17
– " –	0.75	0.1670	0.1252	0.1482	0.0526	1.49
– " –	1.00	0.1298	0.0540	0.1274	0.0412	1.83
– " –	1.25	**0.0938**	**0.0266**	0.0951	0.0312	2.17
– " –	1.50	0.0992	0.0451	**0.0698**	**0.0198**	2.49

*Normalized by $\overline{\omega}_{x,y}$

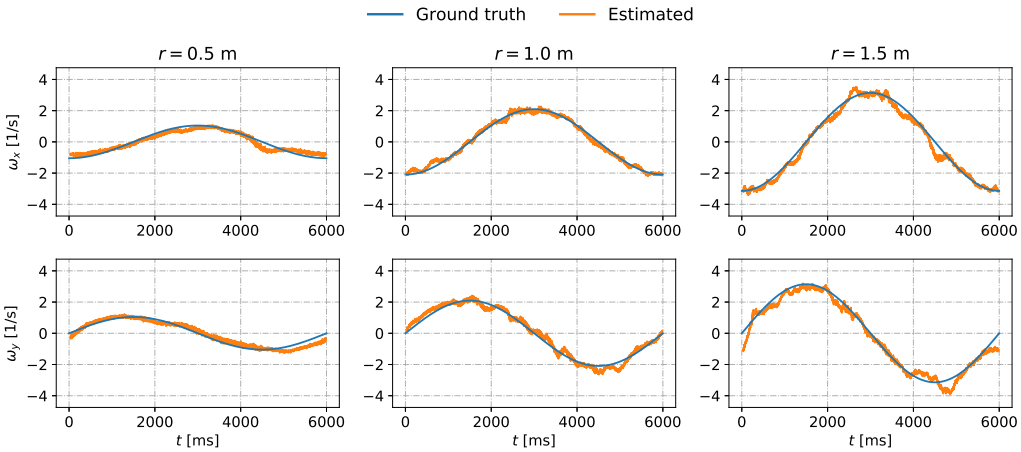


Figure 7-3: Temporal evolution of the ventral flow components estimated with the CNN on Set 3. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, TS-I$_{15}$.

From these results, it is possible to derive several conclusions. First of all, using the work presented by Hordijk et al. (2018) as a reference, the results shown in Table 7-3 confirm the validity of this approach under the conditions of this experiment, even though the error metrics obtained are considerably higher. In Hordijk et al. (2018), the authors reported a mean absolute error of 0.0878 1/s (approx.) and a variance of 0.0051 1/s^2 (approx.) for both ω_x and ω_y in a horizontal circular trajectory above a checkerboard surface. This trajectory

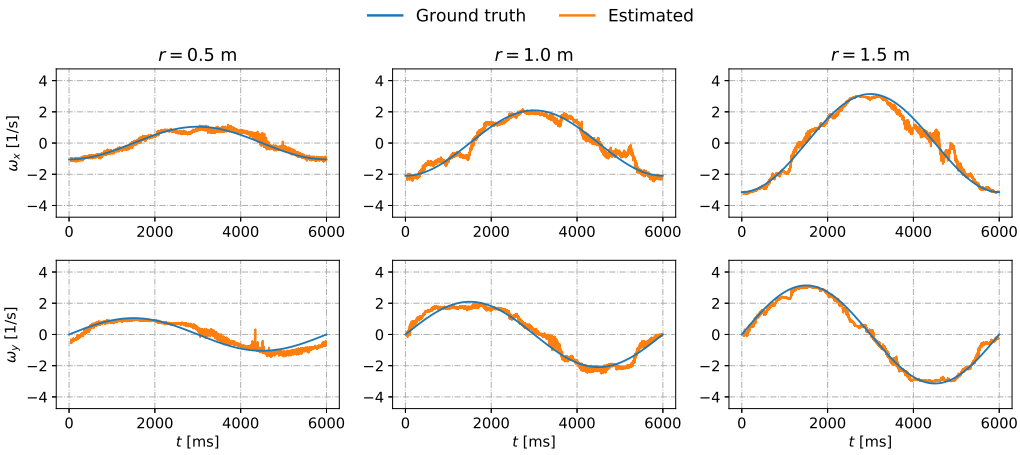


Figure 7-4: Temporal evolution of the ventral flow components estimated with the RCNN on Set 3. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, TS-I$_{15}$.

was characterized by an $r \approx 0.5$ meters, and a full circle was completed in approximately eight seconds at an altitude of 0.8 meters. Even though these results are not fully comparable with the data from Table 7-3, they serve as a state-of-the-art reference for the experiments conducted in this preliminary analysis.

The second conclusion that can be derived is that the performance of these neural networks, represented by the normalized error metrics in Table 7-3, is not directly related to the mean absolute flow that characterizes a particular trajectory, $\overline{\omega}_{x,y}$, when the evaluation is carried out on this set. These are expected results since Set 3 is based on the same texture as the training dataset: the roadmap (see Figure 6-1). Moreover, according to Figure 7-5, the circular trajectories used for testing are characterized by combinations of (ω_x, ω_y) that, in their majority, lie within the regions determined by Set 1. Hence, the visual structure of the images employed in this experiment is similar to that of those from the training dataset, and thus, the performance of the network is ensured on Set 3. For these reasons, the impact of $\overline{\omega}_{x,y}$ is subsequently assessed on Sets 4 and 5, which are characterized by textures unseen during training.

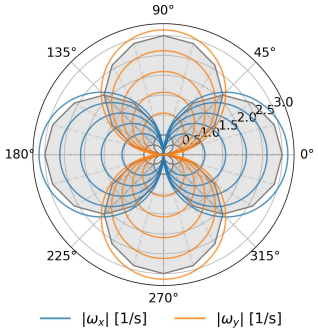


Figure 7-5: Comparison between ground truth ventral flow values from Set 3 (color lines) and training data (gray regions), as a function of the direction of motion.

Finally, from these results it is still not possible to determine whether approaching the ventral flow estimation problem in a sequential fashion is beneficial or not in terms of network performance. As it is possible to infer from Table 7-3, the normalized error metrics obtained with the CNN are very similar to those from RCNN. However, this could be a consequence of the fact that, as mentioned before, Set 3 is based on the same texture as the training dataset. Therefore, this comparison is also subsequently performed on Sets 4 and 5.

The rest of this section presents small sensitivity analysis in order to evaluate the impact that different parameters have on the performance of these neural architectures. The parameters under analysis are: (1) the texture of the surface and the length of the temporal window used

for accumulating DVS events, Δt_f; (2) the encoding scheme employed for representing these events as static images; and (3), the number of filters that characterize the convolutional segment of both networks.

Sensitivity analysis - Network generalizability and Δt_f

Until now, the capabilities and limitations of the neural networks have been addressed using Set 3, i.e., a dataset based on the same texture as the training data: the roadmap (see Figure 6-1a). However, in order to evaluate their generalizability, the performance of these architectures has to be evaluated on trajectories generated from new environments. For this reason, both the CNN and RCNN are now tested on Set 4, characterized by the grass pattern (Figure 6-1b), and on Set 5, characterized by the checkerboard texture (Figure 6-1c).

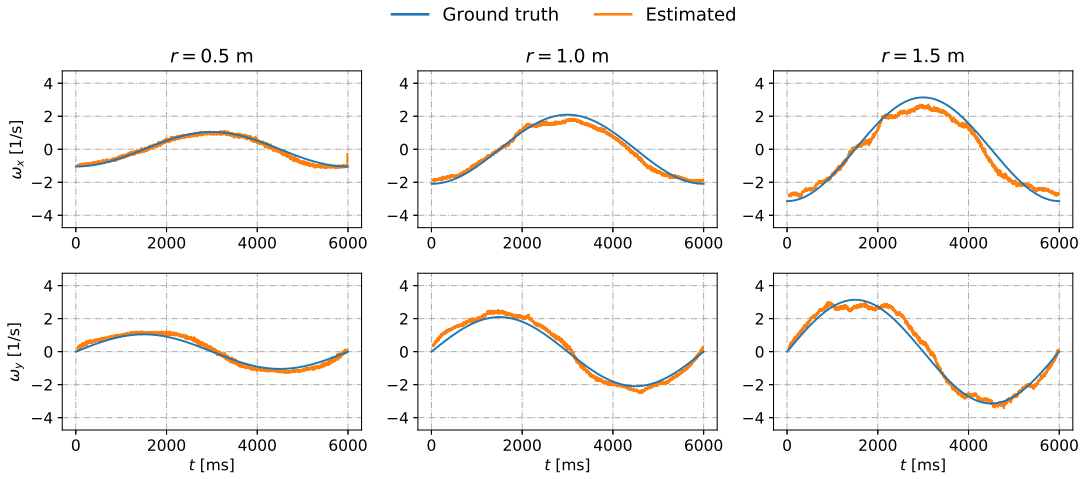


Figure 7-6: Temporal evolution of the ventral flow components estimated with the CNN on Set 4. Config.: DCSB, $\Delta t_{f,\mathsf{T}} = 1.0$ ms, $\Delta t_f = 5.0$ ms, TS-I$_{15}$.

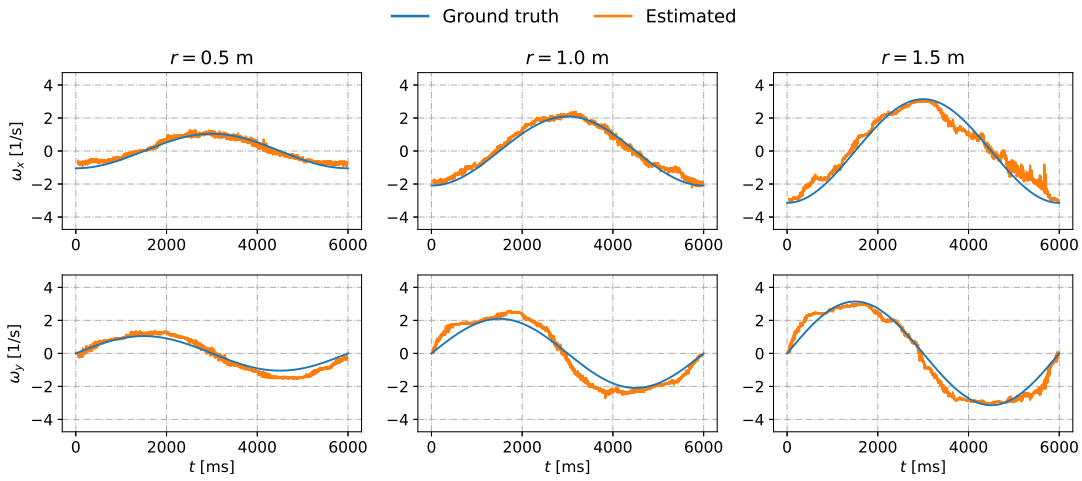


Figure 7-7: Temporal evolution of the ventral flow components estimated with the RCNN on Set 4. Config.: DCSB, $\Delta t_{f,\mathsf{T}} = 1.0$ ms, $\Delta t_f = 5.0$ ms, TS-I$_{15}$.

Starting with Set 4, quantitative and qualitative results are presented in Tables 7-4 and 7-5, and Figures 7-6 and 7-7, respectively, for the circular trajectories defined by $r = [0.5, 1.0, 1.5]$ meters. Note that different values for Δt_f are evaluated in both experiments due to the differences in texture and appearance between the images obtained from these environments and from the roadmap.

Table 7-4: Sensitivity results for ventral flow estimation errors on Set 4 with the CNN, using Δt_f as dependent variable. The lowest errors of each trajectory are highlighted.

Set 4: CNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, TS-I_{15}

r [m]	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
0.5	1.0	0.5074	0.3634	0.4526	0.2924	1.17
0.5	2.5	0.2706	0.1049	0.1454	**0.0288**	1.17
0.5	5.0	**0.0804**	**0.0104**	**0.1426**	0.0289	1.17
0.5	10.0	0.1992	0.0617	0.3377	0.1474	1.17
1.0	1.0	0.5228	0.6172	0.4088	0.3706	1.83
1.0	2.5	0.2687	0.1632	**0.1341**	**0.0403**	1.83
1.0	5.0	**0.1258**	**0.0317**	0.1415	0.0411	1.83
1.0	10.0	0.1309	0.0473	0.3392	0.2293	1.83
1.5	1.0	0.5282	0.8631	0.4599	0.6579	2.49
1.5	2.5	0.2984	0.26799	0.1921	0.1266	2.49
1.5	5.0	0.1626	0.0694	**0.1114**	**0.0410**	2.49
1.5	10.0	**0.1202**	**0.0513**	0.1675	0.0941	2.49

*Normalized by $\overline{\omega}_{x,y}$

Table 7-5: Sensitivity results for ventral flow estimation errors on Set 4 with the RCNN, using Δt_f as dependent variable. The lowest errors of each trajectory are highlighted.

Set 4: RCNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, TS-I_{15}

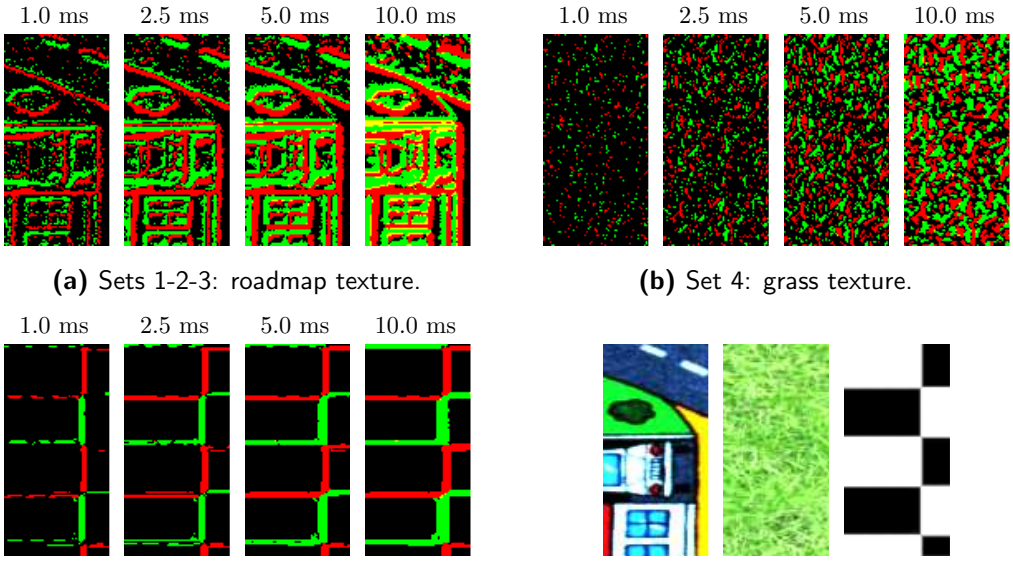
r [m]	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
0.5	1.0	0.4830	0.3406	0.3939	0.2076	1.17
0.5	2.5	0.2854	0.1314	**0.1040**	**0.0178**	1.17
0.5	5.0	**0.1293**	**0.0252**	0.2036	0.0615	1.17
0.5	10.0	0.1543	0.0316	0.4006	0.2140	1.17
1.0	1.0	0.5023	0.5871	0.3718	0.3197	1.83
1.0	2.5	0.2003	0.1006	**0.0928**	**0.0242**	1.83
1.0	5.0	**0.1056**	**0.0216**	0.2084	0.1128	1.83
1.0	10.0	0.1439	0.0307	0.2922	0.1908	1.83
1.5	1.0	0.4169	0.5406	0.3007	0.3169	2.49
1.5	2.5	0.1817	0.1197	**0.1247**	**0.0465**	2.49
1.5	5.0	**0.1598**	**0.0891**	0.1261	0.0585	2.49
1.5	10.0	0.1951	0.1197	0.1382	0.0636	2.49

*Normalized by $\overline{\omega}_{x,y}$

Inferring from Tables 7-4 and 7-5, Δt_f has a significant impact on the performance of both the CNN and RCNN when evaluated on Set 4. For all the trajectories evaluated in this experiment, the temporal window leading to the lowest (normalized) estimation error is $\Delta t_f = 5.0$, which differ significantly from the one used for training $\Delta t_{f,\mathrm{T}} = 1.0$ ms. These results allow the derivation of an important conclusion: for these neural networks to be generalizable to new environments, Δt_f needs to be adapted according to the visual structure of the scene under analysis (e.g., type of texture) and to the ego-motion states of the sensor (e.g., altitude, attitude, speed) in such a way that the image statistics (e.g., *density* of events per image) are similar to those from the training dataset.

This need of varying Δt_f can be qualitatively understood from an image appearance perspective. Figure 7-8 shows the clear dissimilarities between the images used during training (Set 1) and those employed for the evaluation on Set 4. On the one hand, in Figure 7-8a, it is possible to observe that, with the roadmap, the visual structure of the image can be extracted even with $\Delta t_f = 1.0$ ms, for the conditions of this experiment. On the other hand, because of the cluttered texture and the low resolution of the DVS (see Figure 7-8d), larger values of Δt_f are needed until some structure can be discerned on Set 4, as shown in Figure 7-8b. This is the main reason why, since the convolutional segment of both networks learns to extract features from Set 1 with $\Delta t_{f,\mathrm{T}} = 1.0$ ms, a Δt_f different from $\Delta t_{f,\mathrm{T}}$ is required for the correct performance of the these two architectures on this particular environment.

Even though adequate values of Δt_f were found for the conditions of the evaluation on Set 4, the robustness (i.e., generalizability) of these networks can only be ensured if there exists a measurable parameter (or parameters), referred to as *image descriptor*, that is (are) indicative of the Δt_f needed for each new environment and for each possible combination of ego-motion states. In this preliminary analysis, the density of events, represented as the variance of



(a) Sets 1-2-3: roadmap texture.

(b) Set 4: grass texture.

(c) Set 5: checkerboard texture.

(d) Examples of rendered images.

Figure 7-8: Illustration of the impact that Δt_f has on the image appearance depending on the texture from which they are generated. Only half of each image is shown. Config.: DCSB, $r = 1.0$ m, $h = 0.5$ m.

the image intensity and denoted by $\sigma^2(I)$, is proposed as image descriptor based on the conclusions derived from Tables 7-4 and 7-5 and Figure 7-8. In order to confirm the validity of this parameter, new training, validation, and testing datasets are generated according to the principles described in Chapter 6, but this time, varying Δt_f in such a way that all the DVS images are characterized by the same variance. Based on the image statistics of data from Set 4 and $\Delta t_f = 5.0$ ms, the target value for this descriptor is set to $\sigma^2(I) = 0.125$ for this experiment. Quantitative and qualitative results are presented in Table 7-6 and Figure 7-9. Note that this evaluation is only performed for the case of the CNN due to the similar results obtained so far with this architecture in comparison to the RCNN.

Table 7-6: Sensitivity results for ventral flow estimation errors on Set 4 with the CNN, using Δt_f as adaptive variable and the density of events as image descriptor. The mean Δt_f value for each trajectory ($\overline{\Delta} t_f$) is shown as well.

Set 4: CNN, DCSB, adaptive $\Delta t_{f,\mathrm{T}}$, TS-I$_{15}$

r [m]	$\overline{\Delta} t_f$ [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
0.50	7.34	0.1201	**0.0122**	**0.0909**	**0.0085**	1.17
0.75	4.93	**0.0854**	0.0125	0.0999	0.0138	1.49
1.00	3.73	0.0898	0.0180	0.1146	0.0197	1.83
1.25	2.93	0.1202	0.0387	0.1495	0.0395	2.17
1.50	2.42	0.1624	0.0826	0.2044	0.1194	2.49

*Normalized by $\overline{\omega}_{x,y}$

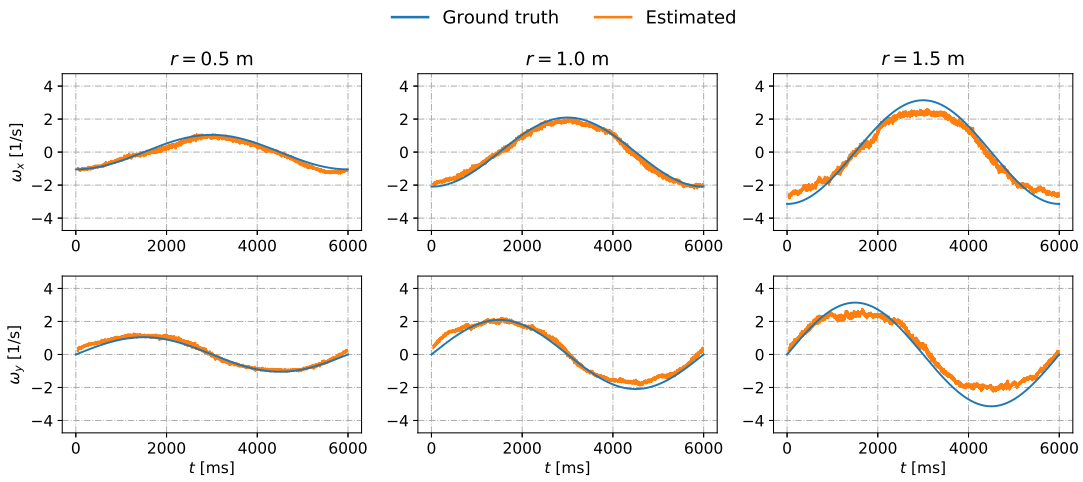


Figure 7-9: Temporal evolution of the ventral flow components estimated with the CNN on Set 4. Config.: DCSB, adaptive $\Delta t_{f,\mathrm{T}}$, adaptive Δt_f ms, TS-I$_{15}$.

From these results, it is possible to derive an important conclusion. If optical flow observables are to be estimated directly from the output of an event-based vision sensor using a neural network and by stacking two or more (event-based) frames as input volume, then these frames have to be similar to the training data in terms of appearance. By adapting the length of Δt_f in such a way that each DVS image is generated with approximately the same value for the event density, this condition is met only if the training dataset is also generated according to

this principle. If compared to previous results (see Table 7-4 and Figure 7-6) Table 7-6 and Figure 7-9 show the positive impact that this approach has on the network performance and generalizability to new environments. Note that, as expected, the value of Δt_f needed for generating DVS images with the required level of $\sigma^2(I)$ is inversely proportional to the mean absolute flow of each trajectory.

Once the results obtained from the performance evaluation on Set 4 have been analyzed, the same experiment is now conducted on Set 5 (i.e., checkerboard texture) in order to confirm, firstly, that approaching the ventral flow estimation problem sequentially with the RCNN

Table 7-7: Sensitivity results for ventral flow estimation errors on Set 5 with the CNN, using Δt_f as dependent variable. The lowest errors of each trajectory are highlighted.

Set 5: CNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, TS-I_{15}

r [m]	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
0.5	1.0	**0.2862**	**0.1200**	**0.4746**	**0.3154**	1.17
0.5	2.5	0.3321	0.1685	0.6574	0.4023	1.17
0.5	5.0	0.3541	0.1991	0.7550	0.7601	1.17
0.5	10.0	0.3706	0.2280	0.8491	0.9513	1.17
1.0	1.0	**0.2534**	**0.1756**	**0.3755**	**0.3139**	1.83
1.0	2.5	0.2554	0.1826	0.5032	0.5504	1.83
1.0	5.0	0.2591	0.1907	0.5815	0.7235	1.83
1.0	10.0	0.2673	0.2009	0.6497	0.8907	1.83
1.5	1.0	0.2976	0.2925	**0.2795**	**0.2675**	2.49
1.5	2.5	0.2731	0.2671	0.3900	0.4910	2.49
1.5	5.0	0.2632	0.2559	0.4548	0.6456	2.49
1.5	10.0	**0.2588**	**0.2441**	0.5039	0.7682	2.49

*Normalized by $\overline{\omega}_{x,y}$

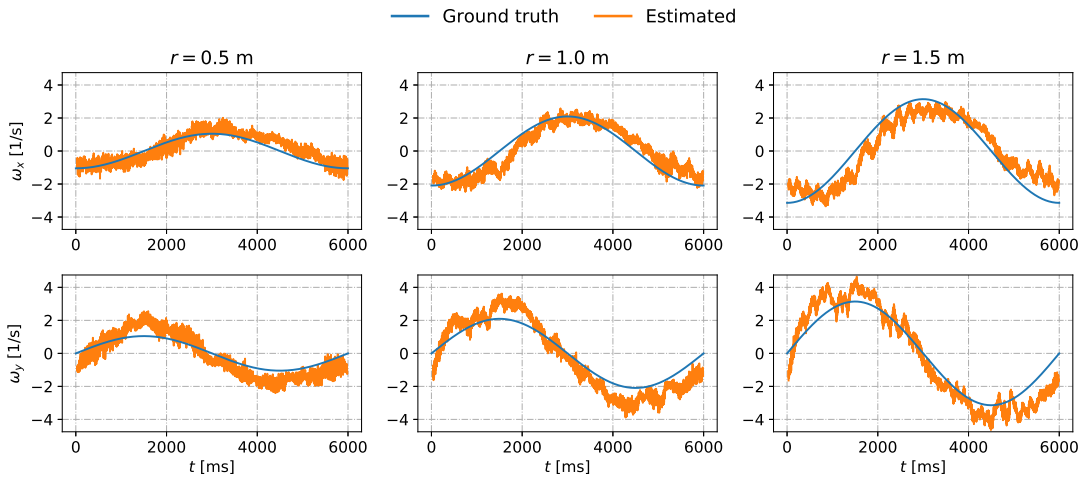


Figure 7-10: Temporal evolution of the ventral flow components estimated with the CNN on Set 5. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, TS-I_{15}.

does not lead to a significant improvement of the performance; and secondly, the validity of the event density as image descriptor for the adaptation of Δt_f on this new environment. Quantitative and qualitative results for the case of the CNN are presented in Table 7-7 and Figure 7-10, respectively; and in Table 7-7 and Figure 7-10 for the case of the RCNN.

Table 7-8: Sensitivity results for ventral flow estimation errors on Set 5 with the RCNN, using Δt_f as dependent variable. The lowest errors of each trajectory are highlighted.

Set 5: RCNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, TS-I_{15}

r [m]	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
0.5	1.0	**0.3024**	**0.1613**	**0.5959**	**0.4704**	1.17
0.5	2.5	0.3197	0.1741	0.7241	0.6568	1.17
0.5	5.0	0.3270	0.1814	0.7642	0.7174	1.17
0.5	10.0	0.3432	0.1989	0.7857	0.7594	1.17
1.0	1.0	**0.2086**	0.1237	**0.3002**	**0.2063**	1.83
1.0	2.5	0.2146	0.1262	0.3652	0.2876	1.83
1.0	5.0	0.2157	0.1261	0.3936	0.3267	1.83
1.0	10.0	0.2142	**0.1203**	0.4101	0.3516	1.83
1.5	1.0	**0.2666**	**0.2314**	**0.1652**	**0.1132**	2.49
1.5	2.5	0.2735	0.2398	0.1875	0.1458	2.49
1.5	5.0	0.2783	0.2463	0.1976	0.1611	2.49
1.5	10.0	0.2860	0.2573	0.1995	0.1655	2.49

*Normalized by $\overline{\omega}_{x,y}$

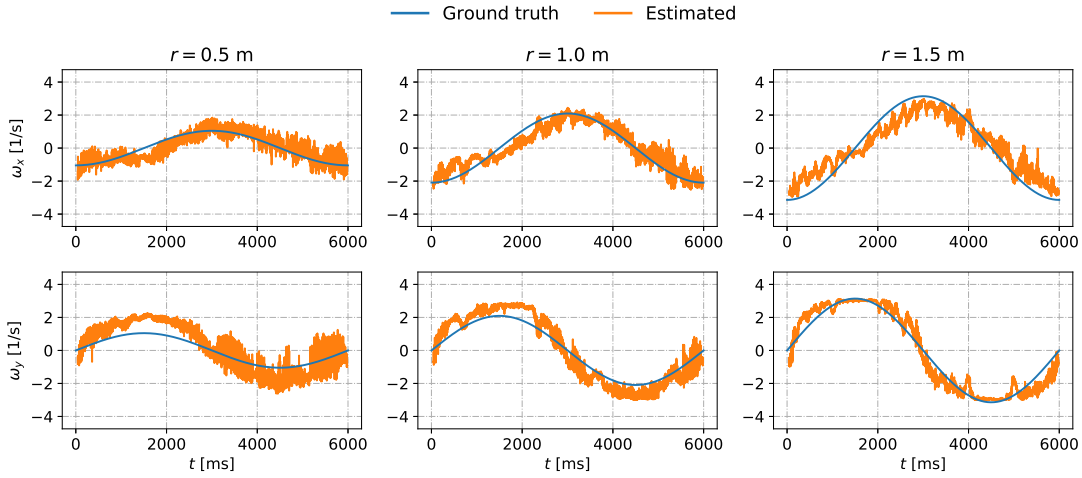


Figure 7-11: Temporal evolution of the ventral flow components estimated with the RCNN on Set 5. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, TS-I_{15}.

As it is possible to infer from these results, the estimation errors for both ω_x and ω_y are considerably higher than when the same networks are evaluated on Set 3 with $\Delta t_f = 1.0$ ms (see Table 7-3), or on Set 4 with $\Delta t_f = 5.0$ ms (see Tables 7-4 and 7-4), independently of the Δt_f employed. Moreover, from Figures 7-10 and 7-11, these networks are capable of tracking ground truth data but with a high variance. These results indicate that these

errors are not due to inadequate values of Δt_f, but due to the fact that the networks do not generalize properly to this particular environment, probably because of the peculiar features that characterize the checkerboard pattern. Additionally, from the results of this experiment, and from those obtained from the evaluation of both the CNN and RCNN on Set 4, it is possible to derive the conclusion that, at least when optical flow visual observables are to be estimated from an input volume consisting of two or more DVS images, approaching this problem in a sequential fashion (i.e., with the RCNN) does not have a significant impact on the performance. Based on this, the rest of the sensitivity analysis is conducted with just the convolutional architecture.

Finally, the last experiment in this section of the sensitivity analysis consists in the evaluation of the performance of the CNN on Set 5 using the aforementioned mechanism for adapting Δt_f based on the density of DVS events, $\sigma^2(I)$. For the correct assessment of the results, the target value for this image descriptor is set to $\sigma^2(I) = 0.125$, as done for the previous experiment of this kind. Quantitative and qualitative results are presented in Table 7-9 and Figure 7-12, respectively.

Table 7-9: Sensitivity results for ventral flow estimation errors on Set 5 with the CNN, using Δt_f as adaptive variable and the density of events as image descriptor. The mean Δt_f value for each trajectory ($\overline{\Delta} t_f$) is shown as well.

Set 5: CNN, DCSB, adaptive $\Delta t_{f,\mathrm{T}}$, TS-I_{15}

r [m]	$\overline{\Delta} t_f$ [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
		Mean*	Var*	Mean*	Var*	
0.50	29.0356	**0.4700**	**0.3188**	**0.2484**	**0.0937**	1.17
0.75	24.3965	0.5353	0.5305	0.3129	0.1928	1.49
1.00	18.0872	0.5656	0.7390	0.3567	0.3088	1.83
1.25	14.5253	0.5858	0.9440	0.3707	0.4044	2.17
1.50	11.4542	0.6078	1.1820	0.3996	0.5420	2.49

*Normalized by $\overline{\omega}_{x,y}$

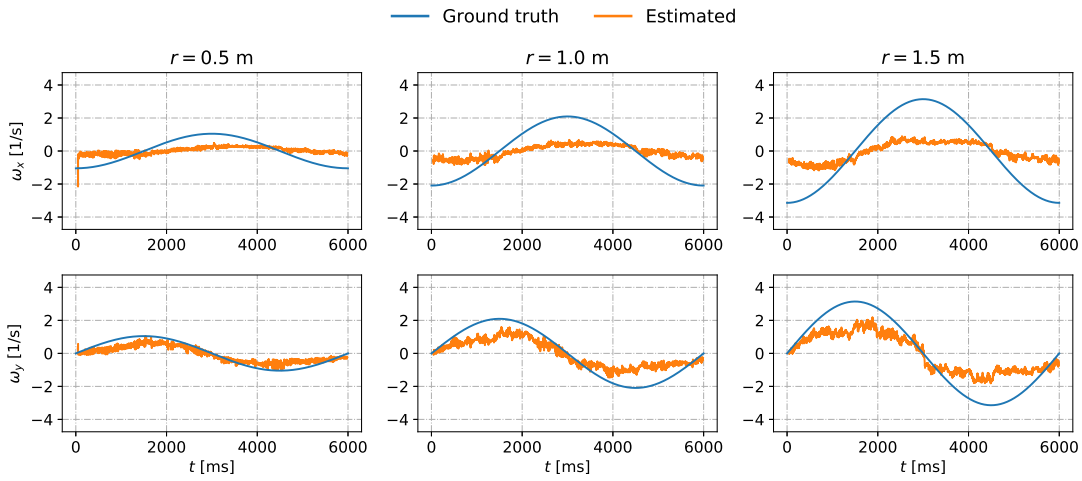
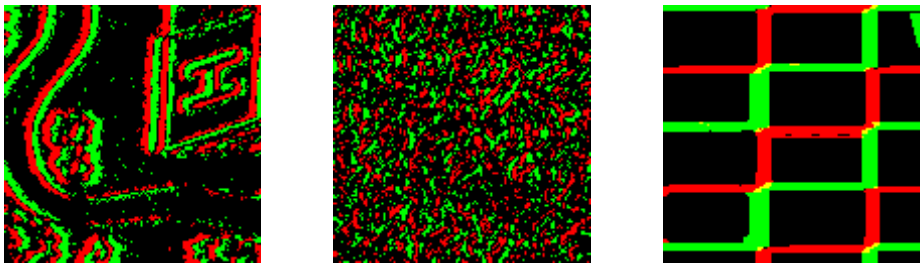


Figure 7-12: Temporal evolution of the ventral flow components estimated with the CNN on Set 5. Config.: DCSB, adaptive $\Delta t_{f,\mathrm{T}}$, adaptive Δt_f ms, TS-I_{15}.



Sets 1-2-3 Set 4 Set 5

Figure 7-13: Illustration of the impact that target event density of $\sigma^2(I) = 0.125$ has on the image appearance depending on the texture from which they are generated. Config.: DCSB, $r = 1.0$ m, $h = 0.5$ m.

Since the use of the event density as image descriptor for the adaptation of Δt_f has already been validated on Set 4 (i.e., grass texture), the poor performance on this new environment can only be due to significant differences between the appearance of the images comprising the training dataset and the checkerboard texture that characterizes Set 5. These differences are clearly discernible from Figure 7-13. Based on this reason, it is important to remark that a different target value for $\sigma^2(I)$ is not a potential solution for increasing the performance on this environment. A new value for $\sigma^2(I)$ requires the network to be re-trained on a dataset generated accordingly, and thus, the differences on the appearance of the image features that can be extracted by the CNN remain. To the best of my knowledge, a better performance can only by expected if the training dataset is expanded with a more diverse set of textures. Note that the evaluation of whether this hypothesis is true or not remains out of the scope of the present work.

Sensitivity analysis - Encoding scheme

The second parameter evaluated in this sensitivity analysis is the encoding scheme employed for representing sets of accumulated events as static images. Three types of schemes are tested in this experiment: DCSB, SCDB, and SCSB. A detailed description of these encoding methods is given in Section 6-4-1, and, furthermore, a sample of their corresponding images is illustrated in Figure 6-6. Quantitative results from the evaluation of these three configurations on Set 3 (i.e., roadmap) are presented in Table 7-10. Note that, obviously, three different CNN architectures were optimized for this experiment, one for each encoding scheme.

From this table, the main conclusion that can be derived is that some sort of differentiation between ON/OFF events is needed for the convolutional network to perform accurately. Both the DCSB and the SCDB schemes make this distinction, and the ventral flow estimation errors obtained with them are very similar. As an hypothesis, the difference between these two representations may come from the contrast of their corresponding images. Since events are separately encoded in different channels according to their polarity, the images generated with the DCSB method are characterized by a higher contrast than the ones obtained with SCDB. This last method uses a single channel based on a gray canvas, and encodes events as black or white pixels. Regarding the results obtained with the SCSB format, it is interesting to see that, even though the network performs worse, it is still possible to estimate the desired visual cues with decent levels of accuracy.

Table 7-10: Sensitivity results for ventral flow estimation errors with the encoding scheme as dependent variable.

Set 3: CNN, $\Delta t_f = 1.0$ ms, $r = 1.0$ m, TS-I_{15}

	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]	
	Mean*	Var*	Mean*	Var*
DCSB	**0.0609**	**0.0119**	**0.0998**	**0.0264**
SCDB	0.0742	0.0172	0.1060	0.0304
SCSB	0.1493	0.0598	0.1093	0.0333

*Normalized by $\overline{\omega}_{x,y} = 1.83$ 1/s

Sensitivity analysis - Number of convolutional filters

Finally, the last parameter evaluated in this sensitivity analysis corresponds to the number of convolutional filters that characterize the two neural networks employed in this section. Briefly speaking, this number is an indicator of the computational power of the architectures, and it determines the number of features that can be extracted at each layer. This experiment assesses the performance impact of this parameter using three configurations: the original (shown in Table 7-1), one with half the number of filters (0.5x), and one with the double (2x). After training three CNNs according to these settings, the evaluation is performed on Sets 3, 4, 5, as shown in Table 7-11. Note that the value of Δt_f employed for each dataset is selected in agreement with the results of the sensitivity analysis conducted with this parameter (i.e., Δt_f) as dependent variable.

From Table 7-11, the estimation errors seem to be inversely proportional to the number of convolutional filters, at least with the configurations analyzed in this experiment. On Set 3, it is possible to observe a slight performance improvement with the 2x configuration in respect of the normal architecture, especially in the estimation of ω_y. However, this change is not very significant. The same variation is appreciated on Set 4. The benefits of increasing the number of filters become more clear when the evaluation is conducted on Set 5. Here, both the mean absolute error and the error variance decrease considerably, thus improving the generalizability of the network to this environment. These results lead to the conclusion that increasing the number of convolutional filters has a noticeable impact on the network performance when evaluated on environments where the normal architecture performs poorly independently of the Δt_f used. The additional features extracted by the network from the training dataset (i.e., Set 1) when this number is incremented are responsible for this behavior.

Despite of the results obtained, note that the practice of increasing the number of filters is not always beneficial. Firstly, complex neural architectures tend to overfit (i.e., to memorize the training data), and thus, their generalization to new data is not ensured (Goodfellow et al., 2016). Secondly, the fact of increasing the number of internal parameters of a network entails an increase in the computational load of the algorithm, represented in Table 7-11 by t_C. According to this, and based on the similar performance of the 0.5x and 1.0x configurations on Sets 3 and 4, an efficient and accurate solution may result from using the 0.5x (or even smaller) structure, in combination with a more diverse training dataset for improving the generalizability of the network. The evaluation of whether this hypothesis is true or not remains out of the scope of the present work.

Table 7-11: Sensitivity results for ventral flow estimation errors with the number of convolutional filters as dependent variable. The lowest error in each column and dataset is highlighted. The number of trainable parameters (N_{param}) and the computation time (t_C) of each configuration are shown as well.

Sets 3-4-5: CNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $r = 1.0$ m, TS-I_{15}

	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		N_{param} [-]	t_C [ms]
		Mean*	Var*	Mean*	Var*		
Set 3							
0.5x	1.0	0.0704	0.0151	0.1001	0.0280	2,200,930	2.702
1.0x	1.0	**0.0633**	**0.0119**	0.0999	0.0263	8,870,482	7.621
2.0x	1.0	0.0724	0.0141	**0.0796**	**0.0166**	35,075,458	36.11
Set 4							
0.5x	5.0	0.1606	0.0494	0.1398	0.0434	2,200,930	2.702
1.0x	5.0	**0.1258**	**0.0318**	0.1416	0.0410	8,870,482	7.621
2.0x	5.0	0.1398	0.0357	**0.1198**	**0.0355**	35,075,458	36.11
Set 5							
0.5x	1.0	0.3644	0.3706	0.3800	0.3065	2,200,930	2.702
1.0x	1.0	0.2534	0.1758	0.3755	0.3139	8,870,482	7.621
2.0x	1.0	**0.2187**	**0.1232**	**0.2385**	**0.1533**	35,075,458	36.11

*Normalized by $\overline{\omega}_{x,y} = 1.83$ 1/s

7-2-2 Ventral flow estimation from a single DVS image

The final experiments conducted on this preliminary analysis examine the possibility of estimating the desired optical flow visual cues from just a single DVS image using the CNN and RCNN architectures introduced in Sections 7-1-1 and 7-1-2, respectively. This input configuration is further referred to as "S-I". Note that the mechanism introduced in the previous section for adapting the value of Δt_f based on the event density, $\sigma^2(I)$, is only employed for the case of the RCNN, as well as the fixed-Δt_f evaluation.

Convolutional Neural Network - Fixed-Δt_f

Starting with the analysis of the CNN, this network is now evaluated on all the circular trajectories comprising Set 3. The reason for conducting this experiment is that, since this testing environment and the training dataset are based on the same texture (i.e., roadmap), the performance of the convolutional network under these conditions can help understand whether ventral flow information can accurately be estimated from a single DVS image. Quantitative and qualitative results are presented in Table 7-12 and Figure 7-14.

Inferring from these results, it is possible to conclude that this input configuration is not sufficient for an accurate estimation of the desired optical flow observables, at least if the CNN is employed under the conditions of this experiment. This statement can be confirmed if these results are compared to those obtained with the TS-I_{15} input configuration (i.e., two temporarily separated DVS images stacked together), which are presented in Table 7-3 and

Table 7-12: Ventral flow estimation errors obtained by evaluating the CNN on all the circular trajectories from Set 3.

Set 3: CNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, S-I

r [m]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
	Mean*	Var*	Mean*	Var*	
0.50	**0.3117**	**0.1476**	**0.2595**	**0.1024**	1.17
0.75	0.3837	0.2648	0.3139	0.2046	1.49
1.00	0.3803	0.3298	0.3923	0.3254	1.83
1.25	0.3957	0.4204	0.4007	0.4258	2.17
1.50	0.4577	0.6333	0.4337	0.5485	2.49

*Normalized by $\overline{\omega}_{x,y}$

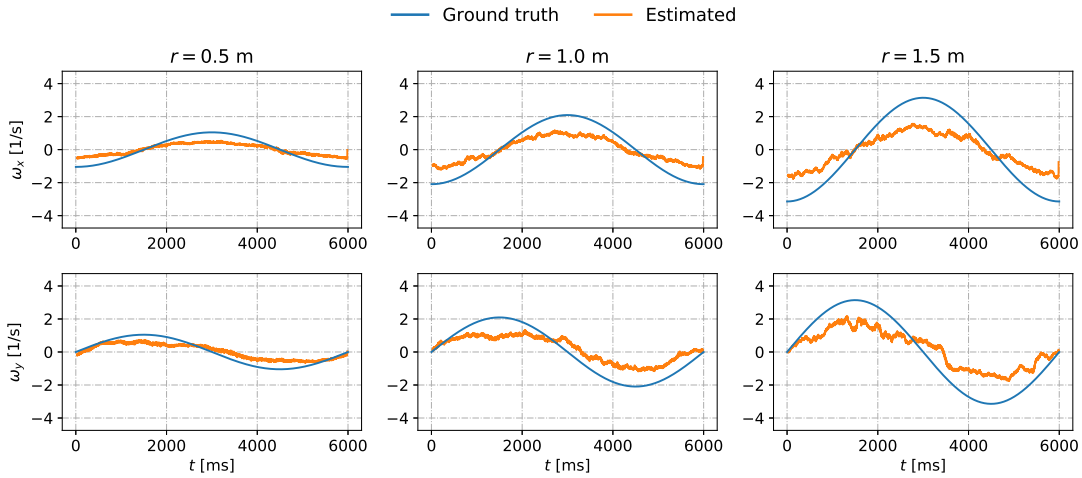


Figure 7-14: Temporal evolution of the ventral flow components estimated with the CNN on Set 3. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, S-I.

Figure 7-3. To the best knowledge of the author, the reason for this difference of performance relies on the fact that, when estimating motion from multiple DVS images, the network extracts most of the information from the disparity between these frames. However, in the case of a single DVS image, the input statistics (e.g., number of events, event density) are the only source of motion clues. Since these statistics vary based not only on the ego-motion states of the sensor (e.g., altitude, attitude, speed) but also on the visual structure of the scene (e.g., type of texture), the S-I is definitely not a valid input configuration for the CNN.

Once the results obtained from Set 3 have been analyzed, the same neural architecture is now evaluated on all the textures that comprise the testing dataset (i.e., Sets 3, 4, and 5) and using different configurations for the value of Δt_f, according to the results obtained in Section 7-2-1 (see Tables 7-4 and 7-7). Note that only the circular trajectory characterized by $r = 1.0$ meters is employed in this analysis. Quantitative and qualitative results are presented in Table 7-13 and Figure 7-15.

Table 7-13: Sensitivity results for ventral flow estimation errors on Sets 3, 4, and 5 with the CNN, using Δt_f as dependent variable. The lowest errors of each trajectory are highlighted.

Sets 3-4-5: CNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $r = 1.0$ m, S-I

	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]	
		Mean*	Var*	Mean*	Var*
Set 3	1.0	0.3803	0.3298	0.3923	0.3254
- " -	2.5	**0.1301**	**0.0422**	**0.2759**	**0.0872**
- " -	5.0	0.1975	0.0934	0.3493	0.1819
- " -	10.0	0.4165	0.3693	0.6144	0.6703
Set 4	1.0	0.7201	1.1831	0.6915	1.0552
- " -	2.5	0.7108	1.1852	0.5742	0.6966
- " -	5.0	0.6901	1.1522	**0.4253**	0.3680
- " -	10.0	**0.6803**	**1.1276**	0.5919	**0.1392**
Set 5	1.0	0.6912	1.0447	0.6273	1.0367
- " -	2.5	0.6633	0.9771	0.6077	0.8931
- " -	5.0	0.6441	0.9406	0.5951	**0.8672**
- " -	10.0	**0.6245**	**0.9169**	**0.5839**	0.8826

*Normalized by $\overline{\omega}_{x,y} = 1.83$ 1/s

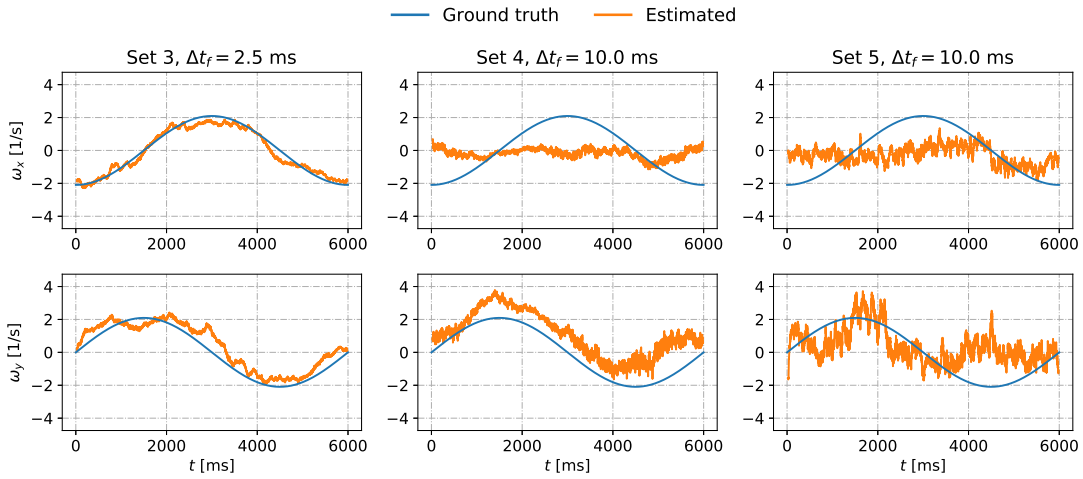


Figure 7-15: Temporal evolution of the ventral flow components estimated with the CNN on Sets 3, 4, and 5. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = [2.5, 10.0, 10.0]$ ms, $r = 1.0$ m, S-I.

From this experiment, it is possible to derive two conclusions. Firstly, despite the fact that the CNN was trained using an accumulation window of length $\Delta t_{f,\mathrm{T}} = 1.0$ ms, different Δt_f values could lead to a better performance. For instance, as shown in Table 7-13, even though Set 3 and the training dataset share the same texture (e.g., roadmap), the $\Delta t_f = 2.5$ ms and $\Delta t_f = 5.0$ ms configurations result in a significant decrease of the error metrics for the trajectory under analysis. Different values for the *optimal* Δt_f are expected if other circular trajectories are evaluated. The reason for this is that the accumulation window has a direct impact on the input statistics of the DVS images and, as mentioned before, they play a crucial role in the estimation of optical flow observables from a single frame.

Secondly, the generalizability of the CNN is also affected by the input configuration. As shown in Section 7-2-1, this convolutional network is capable of generalizing to new environments (e.g., grass, checkerboard) when multiple DVS images are employed as input volume. However, as it is clearly discernible from Figure 7-15, the performance of the architecture is not acceptable anymore when the S-I input configuration is employed. As mentioned before, the visual structure of the scene is another factor with a significant impact on the input statistics of these images.

Recurrent Convolutional Neural Network - Fixed-Δt_f

Once the performance of the CNN is assessed using just a single DVS image as input, the same experiments are now repeated but this time using the RCNN as neural model. Firstly, this network is evaluated on all the circular trajectories comprising Set 3. Quantitative and qualitative results are presented in Table 7-15 and Figure 7-17, respectively.

Table 7-14: Ventral flow estimation errors obtained by evaluating the RCNN on all the circular trajectories from Set 3.

Set 3: RCNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, S-I

r [m]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
	Mean*	Var*	Mean*	Var*	
0.50	0.2154	**0.0530**	0.2616	0.0997	1.17
0.75	0.2468	0.1224	0.1999	0.0856	1.49
1.00	0.1898	0.0914	0.1791	0.0901	1.83
1.25	**0.1327**	0.0593	0.1541	0.0749	2.17
1.50	0.1538	0.1096	**0.1351**	**0.0599**	2.49

*Normalized by $\overline{\omega}_{x,y}$

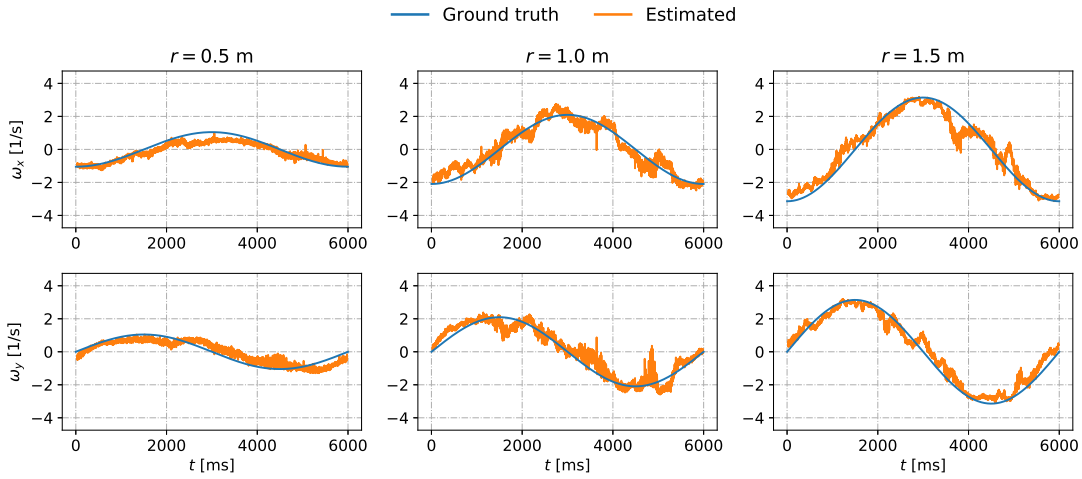


Figure 7-16: Temporal evolution of the ventral flow components estimated with the RCNN on Set 3. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = 1.0$ ms, S-I.

Inferring from these results, it is possible to conclude that the addition of the two LSTM units to the convolutional encoder (see Figure 7-2 for a schematic of the RCNN) has a significant

impact on the network performance when evaluated on Set 3. Even though the error metrics from Table 7-15 are higher than those obtained when this same architecture is evaluated in combination with the TS-I_{15} input configuration (see Table 7-3), Figure 7-17 confirms the validity of this approach. Therefore, these results remark the importance of approaching the ventral flow estimation problem with recurrent neural networks when just a single DVS image is employed as input. As shown in Figure 7-14, just the convolutional segment is not sufficient for a decent performance under the conditions of this experiment.

As done previously for the case of the CNN, the RCNN is now evaluated on all the textures comprising the testing dataset (i.e., Sets 3, 4, and 5) in order to evaluate the generalizability

Table 7-15: Sensitivity results for ventral flow estimation errors on Sets 3, 4, and 5 with the RCNN, using Δt_f as dependent variable. The lowest errors of each trajectory are highlighted.

Sets 3-4-5: RCNN, DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $r = 1.0$ m, S-I

	Δt_f [ms]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]	
		Mean*	Var*	Mean*	Var*
Set 3	1.0	**0.1898**	**0.0914**	**0.1791**	**0.0901**
– " –	2.5	0.3995	0.3664	0.5241	0.5203
– " –	5.0	0.4630	0.4550	0.7119	0.7734
– " –	10.0	0.5412	0.4921	1.0307	0.6836
Set 4	1.0	**0.7303**	**1.1983**	0.6761	1.0094
– " –	2.5	.0.7354	1.2647	**0.6566**	**0.9306**
– " –	5.0	0.7424	1.4258	0.8994	1.1513
– " –	10.0	0.7624	1.6009	1.4099	1.2168
Set 5	1.0	0.6140	**0.8224**	**0.9801**	1.5095
– " –	2.5	**0.5636**	0.8848	1.0029	**1.3734**
– " –	5.0	0.6028	0.9721	1.0046	1.4717
– " –	10.0	0.8039	1.2098	0.9939	1.6540

*Normalized by $\overline{\omega}_{x,y} = 1.83$ 1/s

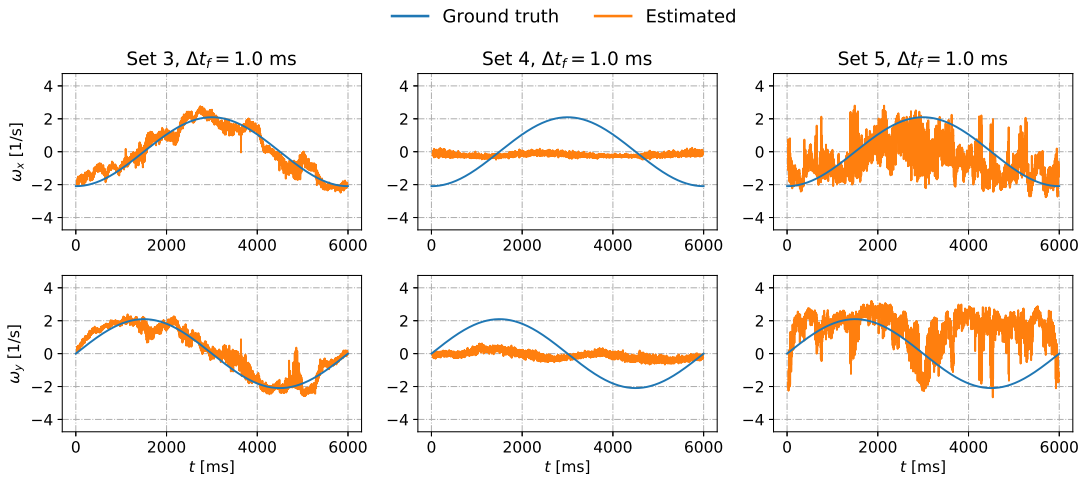


Figure 7-17: Temporal evolution of the ventral flow components estimated with the RCNN on Sets 3, 4, and 5. Config.: DCSB, $\Delta t_{f,\mathrm{T}} = 1.0$ ms, $\Delta t_f = [2.5, 10.0, 10.0]$ ms, $r = 1.0$ m, S-I.

of this architecture in combination with the S-I input configuration. Note that different Δt_f values are employed according to the results obtained in Section 7-2-1 (see Tables 7-4 and 7-7), and only the circular trajectory characterized by $r = 1.0$ meters is assessed. Quantitative and qualitative results are presented in Table 7-15 and Figure 7-17.

Similarly to the CNN, the outcome of this experiment indicates that, despite the decent performance on Set 3, the S-I input configuration has a significant negative impact on the generalizability of the RCNN. As shown in Section 7-2-1, this recurrent network is capable of generalizing to new environments (e.g., grass, checkerboard) when multiple DVS images are employed as input volume. However, as it is clearly discernible from Figure 7-17, the performance of the architecture is not acceptable anymore when just a single image is employed.

Recurrent Convolutional Neural Network - Adaptive Δt_f

In Section 7-2-1, it was possible to conclude that, when multiple DVS images are stacked together as input volume, varying Δt_f in such a way that all these images have similar input statistics (i.e., event density) is very beneficial for the accuracy of this network (and also for CNN) and its generalizability to new environments. This final experiment attempts to determine whether adapting the length of this accumulation window is also beneficial when the RCNN uses just a single DVS image as input. Quantitative and qualitative results are presented in Table 7-16 and Figure 7-18, respectively.

Table 7-16: Ventral flow estimation errors obtained by evaluating the RCNN on all the circular trajectories from Set 3, using Δt_f as adaptive variable.

Set 3: RCNN, DCSB, adaptive $\Delta t_{f,\mathrm{T}}$, S-I

r [m]	ε_{ω_x} [1/s]		ε_{ω_y} [1/s]		$\overline{\omega}_{x,y}$ [1/s]
	Mean*	Var*	Mean*	Var*	
0.50	0.4992	0.3722	0.7843	0.8254	1.17
0.75	0.2867	0.1846	0.4621	0.4573	1.49
1.00	**0.2077**	**0.1296**	**0.2049**	**0.1162**	1.83
1.25	0.2199	0.1499	0.2369	0.1858	2.17
1.50	0.3473	0.4125	0.3244	0.3448	2.49

*Normalized by $\overline{\omega}_{x,y}$

Inferring from these results, the adaptation of Δt_f is not helpful when the estimation of the desired ventral flow components is approached using a single DVS image. As previously introduced, when using a fixed Δt_f, the image statistics depend, among other things, on the ego-motion states of the sensor (e.g., altitude, attitude, speed). For this reason, they play a crucial role when motion clues are to be extracted from just one frame. Since these statistics are now fixed because of the varying Δt_f, the dependency on the sensor motion does no longer exists, thus leading to a bad performance of the RCNN. As it is clearly discernible from Figure 7-18, the network is not capable of distinguishing slow from fast motion.

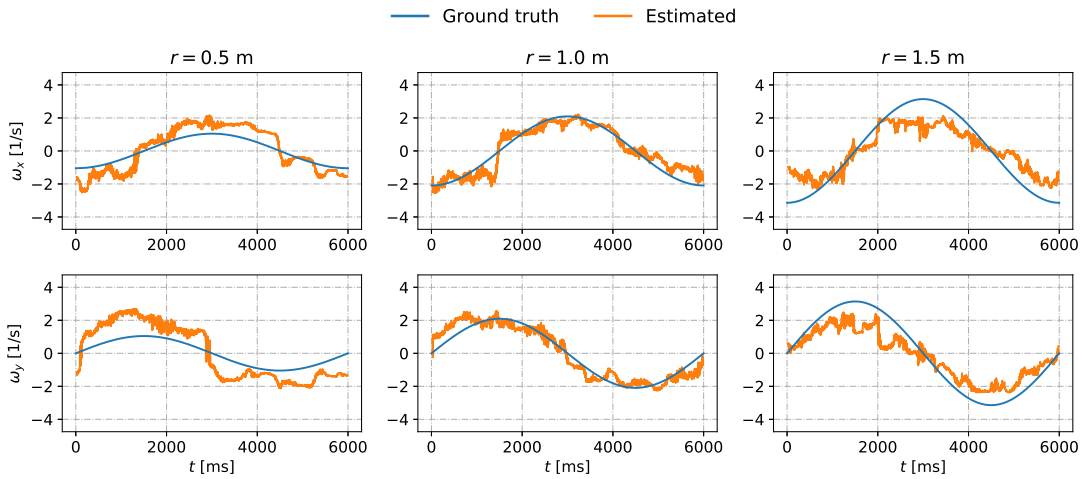


Figure 7-18: Temporal evolution of the ventral flow components estimated with the RCNN on Set 3. Config.: DCSB, adaptive $\Delta t_{f,\mathrm{T}}$, adaptive Δt_f, S-I.

Chapter 8

Discussion of Preliminary Results

In this preliminary analysis, a simulation framework is implemented for evaluating the possibility of estimating optical flow visual cues with DL neural architectures and the DVS as input sensor. Since large amounts of information are needed, an event-camera simulator (Mueggler et al., 2017) is combined with three-dimensional scenes modeled in Blender, for facilitating the process of constructing the required datasets. Images encoded from the event sequences defining these datasets are used for training, validating, and testing the neural architectures in the task of ventral flow estimation. The main results obtained in this preliminary analysis are collected and discussed in this chapter. Furthermore, the relation between these preliminary results and the final thesis work is also described.

8-1 Datasets

Five datasets were generated for training, validating, and testing the neural networks under analysis using three different textures: roadmap, grass, and checkerboard (see Figure 6-1 and Table 6-1). Although these datasets are convenient for this preliminary evaluation, several improvements are still possible. First of all, the training set is constructed only using the roadmap texture as a basis for all its event sequences. Increasing the number of textures of this particular dataset may lead to a better generalization of the networks to new environments. Second, since the trajectories from which these datasets are constructed are performed at a constant altitude, the ground truth data basically consists of pure ventral flow information. The definition of trajectories with combined motion in the horizontal and vertical axes would increase the level of realism of these simulations. Further, it would allow the estimation of additional optical flow cues such as divergence, D, and focus of expansion, FoE.

Due to the incompatibilities of conventional DL architectures with the asynchronous spatiotemporal data from the DVS, three different methods for encoding this information as static images were designed: the DCSB, SCDB, and SCSB schemes (see Figure 6-6). Firstly, the DCSB method generates images with two channels, each corresponding to a particular polarity. Two black canvases are employed, over which the events from a particular temporal

window of length Δt_f are represented as white pixels. Secondly, the SCDB scheme generates single-channel images by encoding ON and OFF events as white and black pixels, respectively, over a gray canvas. Finally, the SCSB method generates single-channel images based on a black canvas over which the events are represented as white pixels, no matter their polarity.

8-2 Performance of Deep Learning architectures

Based on the results obtained from this analysis, it is possible to confirm that, under the conditions of this preliminary evaluation, ventral flow can accurately be estimated from event-based data using conventional DL approaches, such as CNNs and RCNNs. However, the performance of these architectures varies considerably depending on factors such as the pre-processing settings, the type of input, the structure of the neural networks, or the environment over which the event sequences are constructed. This section summarizes the main findings of the experiments that have been conducted. This discussion covers, firstly, the results obtained when the input volume consists of multiple DVS images; and secondly, those corresponding to an input configuration comprised by a single image.

8-2-1 Ventral flow estimation from multiple DVS images

Two different input formats were proposed for the case when multiple DVS images are to be fed through the network simultaneously: the "TS-I" and "C-I" configurations. On the one hand, the TS-I approach consists of stacking two temporarily well-separated frames in such a way that motion clues are discernible from the disparity between these images. On the other hand, the C-I method generates input volumes by stacking long sequences of consecutive frames, despite the fact that the level of similarity among them is high due to the temporal resolution of the DVS. Even though similar results were obtained (see Table 7-2), the computational requirements of processing more than two frames led to the selection of the TS-I method as the optimal solution. More specifically, a temporal separation of 15 milliseconds, denoted by TS-I_{15}. This configuration was employed in all the experiments attempting to estimate ventral flow information from multiple DVS images.

The evaluation of the performance of both CNN and RCNN on Set 3 led to the derivation of two main conclusions. Firstly, the validity of these two neural architectures in the task of estimation optical flow observables is confirmed when the testing environment is based on the same texture as the training dataset (i.e., the roadmap). Secondly, it was shown that the accuracy of these networks is independent of the mean absolute flow of the circular trajectories that comprise this dataset (see Table 7-3). This is due to the fact that, in their majority, the combinations of (ω_x, ω_y) that characterize these sequences lie within the regions determined by Set 1 (see Figure 7-5).

For a generalizability assessment, the performance of these two networks was evaluated on Sets 4 and 5, both characterized by textures that are clearly different from the one employed in the training dataset (i.e., grass and checkerboard pattern, respectively).

From the evaluation on Set 4, it was possible to infer that the length of the accumulation window used for encoding DVS events as static images, Δt_f, has a significant impact on the accuracy of these networks. As shown in Tables 7-4 and 7-5, Δt_f values that differ from

the training configuration, $\Delta t_{f,T}$, need to be used for optimizing the performance. However, since the optimal Δt_f is directly related to the visual structure of the scene, there is no clear indicator for how to adapt this parameter when the training dataset has been generated using a fixed $\Delta t_{f,T}$. To overcome this problem, the density of events, $\sigma^2(I)$ is proposed as image descriptor, and new datasets are created varying Δt_f in such a way that all the images are characterized by the same variance. The experiments conducted with this mechanism led to the important conclusion that if optical flow observables are to be estimated from multiple DVS images, then these frames have to be similar to the training data in terms of appearance (see Table 7-6).

The only conclusion that can be derived from the evaluation of the neural networks on Set 5 is that they do not generalize properly to this particular environment, most probably because of the peculiar features that characterize the checkerboard pattern. Nevertheless, both architectures are capable of tracking ground truth data but with a high variance, as it is possible to infer from Figures 7-10 and 7-11.

Regarding the impact that the input encoding schemes have on the performance of the networks, it was found that some sort of differentiation between ON and OFF events leads to a more accurate estimation of ventral flow components (see Table 7-10). Hence, both the DCSB and SCDB methods are preferred over the SCSB configuration. However, note that, even though the networks perform worse with the latter scheme, it is still possible to estimate the desired visual cues with decent levels of accuracy.

With respect to the internal structure of the neural architectures, results showed that the fact of increasing the number of convolutional filters has a noticeable impact on the network performance when evaluated on environments where the normal architecture performs poorly, i.e., Set 5 in this case (see Table 7-11), independently of the Δt_f employed. The additional features extracted from the training dataset when this number is incremented are responsible for this behavior.

Finally, based on the results obtained from all the experiments conducted with this input configuration, it is possible to determine that approaching the ventral flow estimation problem in a sequential fashion with the RCNN does not have any significant impact on the performance if the input volume is comprised by multiple DVS images. For this reason, the CNN is preferred over the RCNN under these circumstances.

8-2-2 Ventral flow estimation from a single DVS image

Instead of relying on the disparity between input images, the temporal evolution of the input statistics (e.g., number of events, event density) is the only source of motion clues when a single DVS frame is employed. This input configuration is denoted as "S-I". For this reason, the lack of some kind of short-term memory makes the CNN unsuitable for the accurate estimation of optical flow observables with this input configuration, as it is shown in Figure 7-14. On the other hand, as a result of its recurrent connections, the RCNN reports error metrics that confirm the validity of this approach (see Table 7-14), despite being higher than those obtained with the same architecture and the TS-I_{15} input method (see Table 7-3). Based on this, it is possible to determine that, when ventral flow information is to be estimated from a single DVS image, it is crucial to employ a recurrent architecture. For this reason, the RCNN is preferred over the CNN under these circumstances.

Lastly, two additional conclusions can be derived from the experiments conducted with this input configuration. Firstly, despite the decent performance on Set 3 with the RCNN, the S-I method has a significant negative impact on the generalizability of this architecture to new environments (see Figure 7-17). This is a clear indicator that, if motion is to be estimated using this input configuration, the training dataset should be extended with additional textures. Secondly, the adaptation of Δt_f is not helpful when this problem is approached using the S-I method. This mechanism removes the dependency of the image statistics on the ego-motion states of the sensor. Therefore, the RCNN is no longer capable of distinguishing slow from fast motion, as it is possible to infer from Figure 7-18.

8-3 Implications of the analysis

The preliminary analysis conducted in this part serves as a basis for some of the scientific contributions of this thesis. This section describes the main implications of this analysis, and how they relate to the spiking neural architecture described in Part I.

Firstly, due to the large amount of data needed for training and evaluating neural networks regardless of the neuron model, the event-camera simulator presented by Mueggler et al. (2017) is still employed for the generation of event files instead of the DVS sensor itself. Ground truth information of optical flow visual observables is retrieved as explained in Section 6-3. Additionally, due to the superior performance of the DCSB encoding scheme, the final spiking architecture also processes input events separately depending on their polarity.

Secondly, from the deep neural networks evaluated in this preliminary analysis, convolutional layers (LeCun, 1989) seem to be a very efficient mechanism for dealing with input data that has a known grid-like topology. For this reason, the spiking network is mainly based in this type of layers. Note that, if the same computational capability is to be achieved with layers fully-connected neurons, a much larger number of cells and synaptic connections is required; and thus, a much more extensive training dataset (Goodfellow et al., 2016).

Finally, from the analysis conducted with a single DVS image as input, some sort of short-term memory is required for an accurate estimation of ventral flow information. The reason for this is that, with this input format, tracking features over time is the only source of motion clues. In this preliminary evaluation, LSTM layers were employed in order to provide the neural architectures with memory via recurrent connections. Instead of relying on recurrency, the bio-inspired architecture proposed in Part I achieves the same goal through the combination of the temporal dynamics of spiking neurons with synaptic connections of multiple delays.

Part IV

Appendices

Appendix A

Software and Implementation

cuda-snn[1] is a software library for numerical computation that aims to facilitate researchers the simulation of the spike-based communication and learning protocols of large multi-layer Spiking Neural Networks (SNNs). This library is written in its entirety in C++ and CUDA[2], thus ensuring computational efficiency by parallelizing the majority of the work load on the Graphics Processing Unit (GPU). This chapter provides an overview of the implementation details of the simulator. Note that, before going through this content, the reader is encouraged to have a clear understanding of the concepts introduced in Part I of this document.

CUDA engine architecture

The global architecture of the CUDA engine developed for the simulation of SNNs is shown in Figure A-1. The engine consists of several tasks, represented by blocks, arranged in a feed-forward fashion with a feedback connection. To ensure computational efficiency, each of these processes is implemented in a separate CUDA kernel[3], whose parallelization parameters are shown as well. These parameters determine which variables are parallelized, and hence the number of parallel threads required for the correct execution of each kernel. This section provides an overview of these processes.

The main workflow of the cuda-snn engine starts with the update of the synaptic trace of all neurons acting as presynaptic cells in the architecture. As described in Part I (Section IV-A), this parameter models the recent history of spikes transmitted, and plays an important role in the adaptation of neural dynamics, and in the Spike-Timing-Dependent Plasticity (STDP) mechanism. For its update, not only the events stored in the input file are employed, but also the postsynaptic spike trains of all layers in the network, hence the feedback connection. From Figure A-1, this is the only process of the workflow whose implementation is centered on presynaptic, rather than postsynaptic, neurons. Note that, in theory, it could be merged with

[1] Source code to be released in `https://github.com/tudelft`

[2] CUDA documentation can be found in `https://developer.nvidia.com/cuda-toolkit`

[3] Computational function compiled for high throughput accelerators such as GPUs.

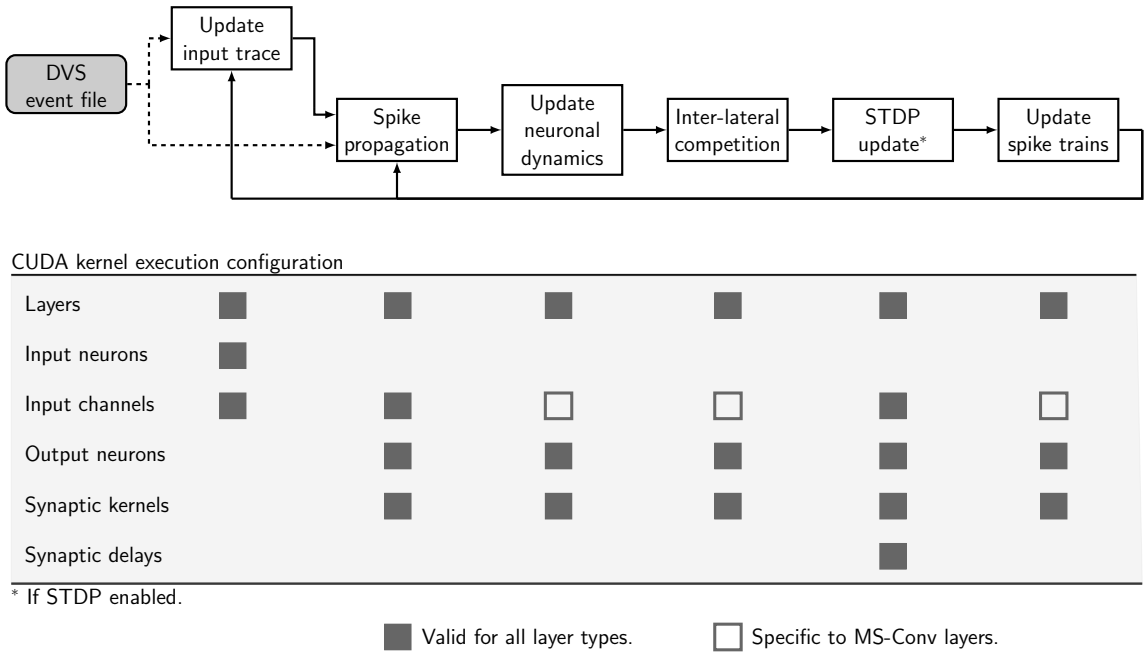


Figure A-1: High-level flowchart of the cuda-snn engine. Solid lines indicate the main workflow, while dashed lines represent interaction with external data storage. The parameters used for the GPU parallelization of each CUDA kernel are shown as well.

the spike propagation routine since they make use of the same input information. However, the overlapping nature of convolutional receptive fields would lead to computational redundancy.

With the synaptic trace of presynaptic neurons up to date, the spike propagation process focuses on the computation of the forcing functions driving the internal dynamics of all postsynaptic cells in the SNN architecture. Consequently, two main process are conducted in this kernel: firstly, presynaptic spike trains are scaled by the synaptic efficacy matrix of each layer, and their neuron-specific contributions are added up; and secondly, if homeostasis is enabled, the corresponding adaptive parameter of each neuron is computed. For an efficient implementation, apart from the layers, this routine parallelizes postsynaptic neurons and each presynaptic feature map to which they are connected.

Once the forcing function is computed, the cuda-snn engine updates the membrane potential of all the postsynaptic cells of the neural architecture according to the adaptive Leaky Integrate-and-Fire (LIF) neuron model introduced in Part I (Section IV-A). For this purpose, each neuron keeps track of an additional variable that indicates whether the cell is in the refractory period, and thus, whether their internal dynamics have to be updated. Note that, in this routine, the membrane potential is not yet reset in case the firing threshold is surpassed, since the competition mechanism relies on this parameter. With respect to the execution configuration of this kernel, apart from the layers, it parallelizes each postsynaptic neuron to be updated. If the MS-Conv layer (see Part I, Section V-C) is employed, the number of input channels is considered as well for the parallelization.

The implementation of the Winner-Take-All (WTA) competition routine (Part I, Section IV-B) differs depending on whether a particular layer is under training or not. If synaptic

plasticity is enabled, a recursive algorithm is employed in which the firing postsynaptic neuron with the highest membrane potential is the cell triggering the inhibitory mechanism first. This recursive approach is executed until all the non-inhibited firing neurons are analyzed. Note that what is parallelized is not the inhibitory mechanism, but the location-specific algorithm that sorts postsynaptic neurons based on their internal potential. Inhibition is conducted in a single thread. The need for this implementation comes from the fact that firing neurons from convolutional layers also reset cells within a small neighborhood around their spatial location. On the other hand, if a layer is in its inference phase, the entire location-specific competition routine is parallelized. Based on this, the execution configuration of this kernel is identical to the one employed by the process focused on updating the membrane potential of postsynaptic neurons.

Once the competition routine determines which postsynaptic neurons from a trainable layer trigger the weight update, the implementation of the STDP is performed as explained in Part I (Section IV-B). Since this kernel is centered on synaptic connections rather than on postsynaptic neurons, its execution configuration also considers the number of multisynaptic connections per pair of neurons for the parallelization. As a result, one parallel thread is launched per neural connection to be updated.

Finally, the last kernel of the cuda-snn engine updates the output spike train of each postsynaptic neuron in the architecture, and then resets the membrane potential and the refractory flag of the firing neurons. Accordingly, this kernel makes use of the same execution configuration as the routines focused on updating the membrane potential of postsynaptic neurons and on the application of the inter-lateral competition mechanism.

Source code, compilation details, and project dependencies

An overview of the source code of the cuda-snn simulator is given in Table A-1. As previously mentioned, the project is a combination of C++ (i.e., `.cpp`/`.hpp`) and CUDA (i.e., `.cu`/`.cuh`) files. On the one hand, C++ files cover a diverse set of tasks (e.g., importing or exporting data, hyperparameter definition, online visualization) for which a single thread on the Central Processing Unit (CPU) is employed. On the other hand, CUDA files contain the computational kernels introduced in Section A, which are heavily parallelized on the GPU. Due to these differences, the simulator makes use two different compilers at building time: `g++` for C++, and `nvcc` for CUDA files. To facilitate the compilation of the project, a makefile[4] is included. Hence, only the following two commands need to be detailed: `make clean` for deleting previously-compiled objects, and `make` for a new compilation.

An overview of the dependencies of the cuda-snn simulator on external third-party software libraries is given in Table A-2. At the moment of writing this document, the project is only compatible with Unix-based operating systems powered by at least one CUDA-enabled GPU[5]. Additionally, the NVIDIA CUDA Toolkit 8.0 or a newer release is required. The OpenGL and Gnuplot dependencies are labeled as "optional" since they are only used for the online visualization of the network activity. Similarly, the *cnpy* package is also optional since it is used exclusively for debugging the neural architecture in Python and for capturing snapshots

[4]Unix-specific file format that can be used to automatize the compilation of a project.
[5]Check compatibility at `https://developer.nvidia.com/cuda-gpus`

of the network activity. The aforementioned makefile of the project includes preprocessor macros that can be adjusted for enabling or disabling the use of these external libraries.

Table A-1: Overview of the source code of the cuda-snn simulator.

File name	Description
`main.cpp`	Being the main file of the simulator, it contains the hyperparameter settings, the initialization of the spiking network, and the training and validation loops. Further, it includes the preprocessing stage of input events conducted by the Input layer. This file serves as a link between the CUDA engine and input/output and visualization tools.
`data.cpp` /`.hpp`	Defines the functions used for importing (exporting) data to (from) the CUDA engine of the simulator. For instance, the functions used for reading event files from the database or for storing and loading synaptic weights are defined here.
`hyperparameters.cpp` /`.hpp`	Defines the data class used for transferring the hyperparameter settings into the CUDA engine.
`engine.cpp` /`.hpp`	Defines the host[6] base class of the CUDA engine.
`engine_gpu.cuh`	CUDA header file that defines "Engine_GPU": the main device[7] class of the CUDA engine of this simulator. This data structure contains the nested classes needed for the definition of the different types of neural layers supported.
`engine_gpu.cu`	Core CUDA file of the cuda-snn simulator. On the one hand, it contains the functions used for the initialization of the neural layers. On the other hand, this file launches the CUDA kernels of the main workflow, as depicted in Figure A-1.
`engine_gpu_kernels.cu` /`.cuh`	Defines the device functions launched by the CUDA kernels.
`plotter.cpp` /`.hpp`	Contains the online visualization tools of the cuda-snn simulator.

[6]In CUDA terminology, the host refers to the CPU and its memory.
[7]In CUDA terminology, the device refers to the GPU and its memory.

Table A-2: Dependencies of the cuda-snn simulator.

Package	Description
build-essential	It refers to all the packages that are required for the compilation of a Debian package. It comes with the `g++` compiler needed for the compilation of C++ files. To install it: `sudo apt-get install build-essential`.
NVIDIA CUDA Toolkit 8.0 (or newer)	CUDA is a parallel computing architecture that extends the capabilities of C++ and enables the development of code that can run on a GPU. The installation guide of these libraries can be found in `http://docs.nvidia.com/cuda/cuda-installation-guide-linux`.
OpenGL (optional)	Software package that serves as an interface with the GPU and that is used for graphics applications. To install it: `sudo apt-get install freeglut3`.
Gnuplot (optional)	Portable command-line driven graphing utility. To install it: `sudo apt-get install gnuplot`.
cnpy (optional)	Software library to read and write `.npy` files from C++. This format encodes Python NumPy[8] arrays into binary files. The installation guide of this library can be found in `https://github.com/rogersce/cnpy`.

[8]NumPy documentation can be found in `http://www.numpy.org/`.

Appendix B

Recommendations

This thesis presents the first hierarchical SNN architecture that is capable of learning the main functionalities of biological visual motion systems, namely local and global motion perception, from the events generated by an event-based vision sensor. Consequently, it lays the foundations for the application of these spiking architectures in the field of robotics for an efficient, biologically-plausible ego-motion estimation. In spite of this, there are still many open questions to be explored with respect to the main contributions of this work. This chapter summarizes these potential lines of research in the form of recommendations, which are grouped based on their main topics: general recommendations, the neuron model and synaptic plasticity learning rule, and the cuda-snn computational framework.

General

To show the capabilities of the proposed SNN, the architecture was trained on a very simplistic dataset. Future research on this topic should, firstly, train and assess the performance of an augmented version of the SNN on a more realistic dataset characterized by translational and rotational motion at different altitudes. This would lead to the identification of a more diverse set of spatial and spatiotemporal features, and to the emergence of neural selectivity to global motion in the three-dimensional space. If optical flow visual observables were to be estimated with a dataset of these characteristics, the proposed readout mechanism would have to be expanded to cope with the rotational motion. A potential solution would be to include measurements of sensor's attitude and angular rates, from an external rate gyro, as additional inputs in the readout architecture.

Once the performance of the spiking architecture is assessed in more realistic scenarios, a subsequent step could be the development of a reinforcement-based spiking learning mechanism that, inspired by the work of Rombouts, van Ooyen, et al. (2012) and Rombouts, Roelfsema, & Bohte (2012), exploits the motion information extracted by the SNN for a biologically plausible, end-to-end spiking optical-flow-based control. For onboard applications, such as the autonomous landing of a flying robot, effort should be made towards the implementation of this solution in neuromorphic hardware.

Neuron model and synaptic plasticity

One of the main difficulties in working with mathematical models of spiking neurons is the great number of coupled internal parameters that have to be manually tuned to get the desired response. Consequently, the LIF neuron model (Stein, 1965) was employed in this work as the simplest and hence most popular formulation; and still, the corresponding neural parameters of each layer were empirically tuned through a trial-and-error process. Further research on this subject should, firstly, conduct a structured study of the coupling of these parameters, and then evaluate the choice made for the SNN proposed in this work. Secondly, the performance of this motion-selective architecture should be assessed for different neuron models in order to see whether the use of more bio-realistic formulations leads to a stronger selectivity. Lastly, for a given neuron model, work should be done towards the application of an optimization method, such as genetic algorithms, for the selection of the most convenient neural parameters for the task at hand.

With respect to synaptic plasticity, two important recommendations are made. On the one hand, to fully exploit the benefits of the STDP implementation proposed in this thesis, the assumption of having distinct training and inference phases should be relaxed. Because of the stability of the rule, as the learning progresses, the weight updates converge to a (close-to-zero) equilibrium state; and therefore, there would be no need for differentiating between these two phases. However, in this work, the implementation of the inter-lateral competition mechanism varies depending on whether the network is under training or not, thus making this always-learning mode impracticable. Future research should focus on the reformulation of this mechanism, specifically for convolutional layers. Moreover, the addition of a forgetting factor to the STDP formulation could be a potential solution for a rapid adaptation to new environments.

On the other hand, the MS-Conv layer of the proposed SNN architecture relies on convolutional kernels comprised of multisynaptic connections with different transmission delays for the perception of the local motion of visual features. In this work, the delays are constant and equal for all kernels regardless of their tuning speed. A potentially beneficial line of work would be the evaluation of how the temporal range and number of multisynaptic connections affects the motion selectivity of the neural architecture. In case each tuning speed has its own set of optimal delays, a learning rule similar to STDP should be developed for the temporal adaptation of these parameters in an unsupervised fashion.

cuda-snn

Even though the cuda-snn simulator is already very useful and efficient at its current state, there are still several improvements to be made that can increase its impact significantly when released as an open-source project. In order of importance, two main recommendations are provided. First, a Python interface should be created. This would facilitate the use of the simulator by researchers unfamiliar with the C++ programming language or with CUDA. Second, for simple phenomenological neuron models, the computations carried out by the simulator could be done in an event-based fashion. In this manner, the update of neural dynamics is only conducted when receiving presynaptic information, and the solution of the model's differential equation is employed otherwise to compute its internal state at any other instant.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. Retrieved from `https://arxiv.org/abs/1603.04467`.

Abdul-Kreem, L. I., & Neumann, H. (2015). Neural mechanisms of cortical motion computation based on a neuromorphic sensory system. *Public Library of Science*, *10*(11), 1–33. doi: 10.1371/journal.pone.0142488.

Adelson, E. H., & Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America*, *2*(2), 284–299. doi: 10.1364/josaa.2.000284.

Adiv, G. (1985). Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *7*(4), 384–401. doi: 10.1109/tpami.1985.4767678.

Albright, T. D. (1984). Direction and orientation selectivity of neurons in visual area MT of the macaque. *Journal of Neurophysiology*, *52*(6), 1106–1130.

Alkowatly, M. T., Becerra, V. M., & Holderbaum, W. (2014). Bioinspired autonomous visual vertical control of a quadrotor unmanned aerial vehicle. *Journal of Guidance, Control, and Dynamics*, *38*(2), 249–262. doi: 10.2514/1.g000634.

Baird, E., Boeddeker, N., Ibbotson, M. R., & Srinivasan, M. V. (2013). A universal strategy for visually guided landing. *Proceedings of the National Academy of Sciences*, *110*(46), 18686–18691. doi: 10.1073/pnas.1314311110.

Barlow, H. B., & Levick, W. R. (1965). The mechanism of directionally selective units in rabbit's retina. *The Journal of Physiology*, *178*(3), 477–504. doi: 10.1113/jphysiol.1965.sp007638.

Barron, J. L., Fleet, D. J., Beauchemin, S. S., & Burkitt, T. (1992). Performance of optical flow techniques. In *Proceedings of the 1992 IEEE Conputer Society Conference on Computer Vision and Pattern Recognition* (pp. 236–242). doi: 10.1007/bf01420984.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., et al. (2012). Theano: New features and speed improvements. Retrieved from `https://arxiv.org/abs/1211.5590`.

Baudry, M. (1998). Synaptic plasticity and learning and memory: 15 years of progress. *Neurobiology of Learning and Memory*, *70*(1), 113–118. doi: 10.1006/nlme.1998.3842.

Beauchemin, S. S., & Barron, J. L. (1995). The computation of optical flow. *ACM Computing Surveys (CSUR)*, *27*(3), 433–466. doi: 10.1145/212094.212141.

Belatreche, A., Maguire, L. P., McGinnity, M., & Wu, Q. (2003). A method for supervised training of spiking neural networks. In *Proceedings of IEEE Cybernetics Intelligence-Challenges and Advances* (pp. 39–44).

Benosman, R., Clercq, C., Lagorce, X., Ieng, S., & Bartolozzi, C. (2014). Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(2), 407–417. doi: 10.1109/tnnls.2013.2273537.

Benosman, R., Ieng, S., Clercq, C., Bartolozzi, C., & Srinivasan, M. (2012). Asynchronous frameless event-based optical flow. *Neural Networks*, *27*, 32–37. doi: 10.1016/j.neunet.2011.11.001.

Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, *18*(24), 10464–10472.

Bi, G. Q., & Poo, M. M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, *24*(1), 139–166. doi: 10.1146/annurev.neuro.24.1.139.

Bichler, O., Querlioz, D., Thorpe, S. J., Bourgoin, J. P., & Gamrat, C. (2012). Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, *32*, 339–348. doi: 10.1016/j.neunet.2012.02.022.

Bishop, C. M. (1994). Mixture Density Networks. Unpublished manuscript.

Bloesch, M., Omari, S., Hutter, M., & Siegwart, R. (2015). Robust visual inertial odometry using a direct EKF-based approach. In *Proceedins of the 2015 IEEE/RSJ Internaitonal Conference on Intelligent Robots and Systems* (pp. 298–304). doi: 10.1109/iros.2015.7353389.

Boahen, K. A. (2000). Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *47*(5), 416–434. doi: 10.1109/82.842110.

Bohte, S. M., Kok, J. N., & La Poutré, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1), 17–37. doi: 10.1016/s0925-2312(01)00658-0.

Borst, A. (2011). Fly vision: Moving into the motion detection circuit. *Current Biology*, *21*(24), R990–R992. doi: 10.1016/j.cub.2011.10.045.

Borst, A., & Euler, T. (2011). Seeing things in motion: Models, circuits, and mechanisms. *Neuron*, *71*(6), 974–994. doi: 10.1016/j.neuron.2011.08.031.

Borst, A., Haag, J., & Reiff, D. F. (2010). Fly motion vision. *Annual Review of Neuroscience*, *33*, 49–70. doi: 10.1146/annurev-neuro-060909-153155.

Borst, A., & Helmstaedter, M. (2015). Common circuit design in fly and mammalian motion vision. *Nature Neuroscience*, *18*(8), 1067–1076. doi: 10.1038/nn.4050.

Bower, J. M., & Beeman, D. (2012). *The book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System*. Springer Science & Business Media.

Brandli, C., Berner, R., Yang, M., Liu, S., & Delbruck, T. (2014). A 240x180 130 dB 3 µs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, *49*(10), 2333–2341. doi: 10.1109/JSSC.2014.2342715.

Brosch, T., & Neumann, H. (2014). Computing with a canonical neural circuits model with pool normalization and modulating feedback. *Neural Computation*, *26*, 2735–2789. doi: 10.1162/neco_a_00675.

Brosch, T., & Neumann, H. (2016). Event-based optical flow on neuromorphic hardware. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies* (pp. 551–558). doi: 10.4108/eai.3-12-2015.2262447.

Brosch, T., Tschechne, S., & Neumann, H. (2015). On event-based optical flow detection. *Frontiers in Neuroscience*, *9*, 1–15. doi: 10.3389/fnins.2015.00137.

Camus, T. (1997). Real-time quantized optical flow. *Real-Time Imaging*, *3*(2), 71–86. doi: 10.1006/rtim.1996.0048.

Caruana, R., Lawrence, S., & Giles, C. L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems* (pp. 402–408).

Censi, A., & Scaramuzza, D. (2014). Low-latency event-based visual odometry. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation* (pp. 703–710). doi: 10.1109/icra.2014.6906931.

Chahl, J. S., Srinivasan, M. V., & Zhang, S. W. (2004). Landing strategies in honeybees and applications to uninhabited airborne vehicles. *The International Journal of Robotics Research*, *23*(2), 101–110. doi: 10.1177/0278364904041320.

Cho, D. D., & Lee, T. (2015). A review of bioinspired vision sensors and their applications. *Sensors and Materials*, *27*(6), 447–463. doi: 10.18494/sam.2015.1083.

Conradt, J., Berner, R., Cook, M., & Delbruck, T. (2009). An embedded AER dynamic vision sensor for low-latency pole balancing. In *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops* (pp. 780–785).

Costante, G., Mancini, M., Valigi, P., & Ciarfuglia, T. A. (2016). Exploring representation learning with CNNs for frame-to-frame ego-motion estimation. *IEEE Robotics and Automation Letters*, *1*(1), 18–25. doi: 10.1109/lra.2015.2505717.

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

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6). doi: 10.1109/tpami.2007.1049.

de Croon, G. C. H. E. (2016). Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy. *Bioinspiration & Biomimetics*, *11*(1). doi: 10.1088/1748-3190/11/1/016004.

de Croon, G. C. H. E., Ho, H., De Wagter, C., Van Kampen, E., Remes, B., & Chu, Q. P. (2013). Optic-flow based slope estimation for autonomous landing. *International Journal of Micro Air Vehicles*, *5*(4), 287–297. doi: 10.1260/1756-8293.5.4.287.

Delbruck, T. (2007). *jAER Open Source Project.* Retrieved from `https://sourceforge.net/projects/jaer/`.

Delbruck, T., & Lang, M. (2013). Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*, *7*, 1–7.

Delorme, A., Gautrais, J., Van Rullen, R., & Thorpe, S. (1999). SpikeNET: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, *26*, 989–996. doi: 10.1016/s0925-2312(99)00095-8.

Demuth, H. B., Beale, M. H., De Jess, O., & Hagan, M. T. (2014). *Neural network design.* Martin Hagan.

De Valois, R. L., Cottaris, N. P., Mahon, L. E., Elfar, S. D., & Wilson, J. A. (2000). Spatial and temporal receptive fields of geniculate and cortical cells and directional selectivity. *Vision Research*, *40*(27), 3685–3702. doi: 10.1016/s0042-6989(00)00210-8.

De Wagter, C., Tijmons, S., Remes, B. D. W., & de Croon, G. C. H. E. (2014). Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system. In *Proceedings of the 2014 IEEE international conference on robotics and automation* (pp. 4982–4987).

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, *58*(15-16), 1–35.

Diehl, P. U., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, *9*, 1–9. doi: 10.3389/fncom.2015.00099.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S. C., & Pfeiffer, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proceedings of the 2015 International Joint Conference on Neural Networks* (pp. 1–8). doi: 10.1109/ijcnn.2015.7280696.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., et al. (2015). FlowNet: Learning optical flow with convolutional networks. In *Proceedings*

of the 2015 IEEE International Conference on Computer Vision (pp. 2758–2766). doi: 10.1109/iccv.2015.316.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Eichner, H., Joesch, M., Schnell, B., Reiff, D. F., & Borst, A. (2011). Internal structure of the fly elementary motion detector. *Neuron*, *70*(6), 1155–1164. doi: 10.1016/j.neuron.2011.03.028.

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems* (pp. 2366–2374).

Expert, F., & Ruffier, F. (2012). Controlling docking, altitude and speed in a circular high-roofed tunnel thanks to the optic flow. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1125–1132). doi: 10.1109/iros.2012.6385946.

Expert, F., & Ruffier, F. (2015). Flying over uneven moving terrain based on optic-flow cues without any need for reference frames or accelerometers. *Bioinspiration & Biomimetics*, *10*(2). doi: 10.1088/1748-3182/10/2/026003.

Feng, J. (2003). *Computational neuroscience: A comprehensive approach*. CRC press.

Ferster, D., & Spruston, N. (1995). Cracking the neuronal code. *Science*, *270*(5237), 756. doi: 10.1126/science.270.5237.756.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fleet, D. J. (2012). *Measurement of image velocity* (Vol. 169). Springer Science & Business Media.

Fleet, D. J., & Jepson, A. D. (1990). Computation of component image velocity from local phase information. *International Journal of Computer Vision*, *5*(1), 77–104. doi: 10.1007/bf00056772.

Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation* (pp. 15–22). doi: 10.1109/icra.2014.690658.

Fortun, D., Bouthemy, P., & Kervrann, C. (2015). Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, *134*, 1–21.

Franceschini, N., Riehle, A., & Le Nestour, A. (1989). Directionally selective motion detection by insect neurons. In *Facets of Vision* (pp. 360–390). Springer. doi: 10.1007/978-3-642-74082-4_17.

Fraundorfer, F., Heng, L., Honegger, D., Lee, G. H., Meier, L., Tanskanen, P., et al. (2012). Vision-based autonomous mapping and exploration using a quadrotor mav. In *Proceedings of the 2012 IEEE International Conference on Intelligent Robots and Systems* (pp. 4557–4564). doi: 10.1109/iros.2012.6385934.

Fuentes-Pacheco, J., Ruiz-Ascencio, J., & Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, *43*(1), 55–81. doi: 10.1007/s10462-012-9365-8.

Gallego, G., Lund, J. E. A., Mueggler, E., Rebecq, H., Delbruck, T., & Scaramuzza, D. (2017). Event-based, 6-DOF camera tracking from photometric depth maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–11.

Gautama, T., & van Hulle, M. A. (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. doi: 10.1109/tnn.2002.1031944.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3354–3361). doi: 10.1109/cvpr.2012.6248074.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press. doi: 10.1017/cbo9780511815706.

Gewaltig, M. O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*(4), 1430. doi: 10.4249/scholarpedia.1430.

Geyer, C., & Daniilidis, K. (2000). A unifying theory for central panoramic systems and practical implications. *European Conference on Computer Vision*, 445–461. doi: 10.1007/3-540-45053-x_29.

Ghosh-Dastidar, S., & Adeli, H. (2009). Spiking neural networks. *International Journal of Neural Systems*, *19*(04), 295–308. doi: 10.1142/s0129065709002002.

Gibson, J. J. (1950). The perception of the visual world.

Giulioni, M., Camilleri, P., Dante, V., Badoni, D., Indiveri, G., Braun, J., et al. (2008). A VLSI network of spiking neurons with plastic fully configurable "stop-learning" synapses. In *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems* (pp. 678–681). doi: 10.1109/ICECS.2008.4674944.

Giulioni, M., Lagorce, X., Galluppi, F., & Benosman, R. B. (2016). Event-based computation of motion flow on a neuromorphic analog neural platform. *Frontiers in Neuroscience*, *10*, 1–13. doi: 10.3389/fnins.2016.00035.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goodman, D., & Brette, R. (2008). Brian: a simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics*. doi: 10.3389/neuro.11.005.2008.

Grabe, V., Bülthoff, H. H., Scaramuzza, D., & Giordano, P. R. (2015). Nonlinear ego-motion estimation from optical flow for online control of a quadrotor uav. *The International Journal of Robotics Research*, *34*(8), 1114–1135.

Graves, A. (2013). Generating sequences with recurrent neural networks. Retrieved from `https://arxiv.org/abs/1308.0850`.

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). doi: 10.1109/icassp.2013.6638947.

Grossman, E., Donnelly, M., Price, R., Pickens, D., Morgan, V., Neighbor, G., et al. (2000). Brain areas involved in perception of biological motion. *Journal of Cognitive Neuroscience*, *12*(5), 711–720. doi: 10.1162/089892900562417.

Grzywacz, N. M., & Yuille, A. L. (1990). A model for the estimate of local image velocity by cells in the visual cortex. *Proceedings of the Royal Society of London: Biological Sciences*, *239*(1295), 129–161. doi: 10.1098/rspb.1990.0012.

Güney, F., & Geiger, A. (2016). Deep discrete flow. In *Asian Conference on Computer Vision* (pp. 207–224). doi: 10.1007/978-3-319-54190-7_13.

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference* (Vol. 15, pp. 10–5244). doi: 10.5244/c.2.23.

Hassenstein, B., & Reichardt, W. (1956). Systemtheoretische analyse der zeit-, reihenfolgen- und vorzeichenauswertung bei der bewegungsperzeption des rüsselkäfers chlorophanus. *Zeitschrift für Naturforschung*, *11*(9-10), 513–524. doi: 10.1515/znb-1956-9-1004.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). doi: 10.1109/cvpr.2016.90.

Hebb, D. O. (1952). *The organisation of behaviour: A neuropsychological theory*. Wiley.

Heeger, D. J. (1987). Model for the extraction of image flow. *Journal of the Optical Society of America*, *4*(8), 1455–1471. doi: 10.1364/josaa.4.001455.

Heeger, D. J. (1988). Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, *1*(4), 279–302. doi: 10.1007/bf00133568.

Herissé, B., Hamel, T., Mahony, R., & Russotto, F. X. (2012). Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, *28*(1), 77–89. doi: 10.1109/tro.2011.2163435.

Hines, M. L., & Carnevale, N. T. (2006). The NEURON simulation environment. *Neural Computation*, *9*(6). doi: 10.1007/978-1-4614-7320-6_795-1.

Ho, H. W., & de Croon, G. C. H. E. (2016). Characterization of flow field divergence for MAVs vertical control landing. In *AIAA Guidance, Navigation, and Control Conference* (pp. 1–13). doi: 10.2514/6.2016-0106.

Ho, H. W., de Croon, G. C. H. E., & Chu, Q. P. (2017). Distance and velocity estimation using optical flow from a monocular camera. *International Journal of Micro Air Vehicles*. doi: 10.1177/1756829317695566.

Ho, H. W., de Croon, G. C. H. E., van Kampen, E., Chu, Q. P., & Mulder, M. (2016). Adaptive control strategy for constant optical flow divergence landing. Retrieved from `https://arxiv.org/abs/1609.06767`.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, *117*(4), 25–71. doi: 10.1007/bf02459568.

Hordijk, B. J. P., Scheper, K. Y. W., & de Croon, G. C. H. E. (2018). Vertical landing for micro air vehicles using event-based optical flow. *Journal of Field Robotics*, *35*(1), 69–90. doi: 10.1007/bf02459568.

Horn, B. K. P., & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, *17*(1-3), 185–203. doi: 10.1016/0004-3702(81)90024-2.

Iakymchuk, T., Rosado-Muñoz, A., Guerrero-Martínez, J. F., Bataller-Mompeán, M., & Francés-Víllora, J. V. (2015). Simplified spiking neural network architecture and STDP learning algorithm applied to image classification. *EURASIP Journal on Image and Video Processing*, *2015*(1), 1–11.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 2, pp. 1647–1655). doi: 10.1109/cvpr.2017.179.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572. doi: 10.1109/tnn.2003.820440.

Izhikevich, E. M. (2007). *Dynamical systems in neuroscience*. MIT Press.

Izzo, D., & de Croon, G. C. H. E. (2012). Landing with time-to-contact and ventral optic flow estimates. *Journal of Guidance, Control, and Dynamics*, *35*(4), 1362–1367. doi: 10.2514/1.56598.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia* (pp. 675–678). doi: 10.1145/2647868.2654889.

Kendall, A., Grimes, M., & Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of the 2015 IEEE International Conference on Computer Vision* (pp. 2938–2946). doi: 10.1109/iccv.2015.336.

Kendoul, F. (2014). Four-dimensional guidance and control of movement using time-to-contact: Application to automated docking and landing of unmanned rotorcraft systems. *The International Journal of Robotics Research*, *33*(2), 237–267. doi: 10.1177/0278364913509496.

Kendoul, F., Fantoni, I., & Nonami, K. (2009). Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles. *Robotics and Autonomous Systems*, *57*(6), 591–602. doi: 10.1016/j.robot.2009.02.001.

Kendoul, F., Yu, Z., & Nonami, K. (2010). Guidance and nonlinear control system for autonomous flight of minirotorcraft unmanned aerial vehicles. *Journal of Field Robotics*, *27*(3), 311–334. doi: 10.1002/rob.20327.

Khan, M. M., Lester, D. R., Plana, L. A., Rast, A., Jin, X., Painkras, E., et al. (2008). SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks* (pp. 2849–2856). doi: 10.1109/ijcnn.2008.4634199.

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., & Masquelier, T. (2016). STDP-based spiking deep neural networks for object recognition. Retrieved from `https://arxiv.org/abs/1611.01421`.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. Retrieved from `https://arxiv.org/abs/1412.6980`. Available from `https://arxiv.org/abs/1412.6980`

Kirkwood, A., & Bear, M. F. (1994). Hebbian synapses in visual cortex. *Journal of Neuroscience*, *14*(3), 1634–1645.

Kistler, W. M., Gerstner, W., & van Hemmen, J. L. (1997). Reduction of the Hodgkin-Huxley equations to a single-variable threshold model. *Neural Computation*, *9*(5), 1015–1045. doi: 10.1162/neco.1997.9.5.1015.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. 225–234). doi: 10.1109/ismar.2007.4538852.

Konda, K. R., & Memisevic, R. (2015). Learning visual odometry with a convolutional network. In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications* (pp. 486–490). doi: 10.5220/0005299304860490.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet: Classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105). doi: 10.1145/3065386.

Kueng, B., Mueggler, E., Gallego, G., & Scaramuzza, D. (2016). Low-latency visual odometry using event-based feature tracks. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 16–23). doi: 10.1109/iros.2016.7758089.

Lai, W. S., Huang, J. B., & Yang, M. H. (2017). Semi-supervised learning for optical flow with generative adversarial networks. In *Advances in Neural Information Processing Systems* (pp. 353–363).

Lappe, M., & Rauschecker, J. P. (1993). A neural network for the processing of optic flow from ego-motion in man and higher mammals. *Neural Computation*, *5*(3), 374–391. doi: 10.1162/neco.1993.5.3.374.

LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in Perspective*, 143–155.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lee, D. N. (1976). A theory of visual control of braking based on information about time-to-collision. *Perception*, *5*(4), 437–459. doi: 10.1068/p050437.

Lee, D. N., Davies, M. N. O., & Green, P. R. (1993). Visual control of velocity of approach by pigeons when landing. *Journal of Experimental Biology*, *180*(1), 85–104.

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, *10*. doi: 10.3389/fnins.2016.00508.

Legenstein, R., Pecevski, D., & Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *Public Library of Science: Computational Biology*, *4*(10), e1000180. doi: 10.1371/journal.pcbi.1000180.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128x128 120 dB 15 µs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, *43*(2), 566–576. doi: 10.1109/JSSC.2007.914337.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11–26. doi: 10.1016/j.neucom.2016.12.038.

Longinotti, L. (2014). *cAER: A framework for event-based processing on embedded systems*. BSc Thesis. University of Zürich. Retrieved 2017-05-15, from `http://sourceforge.net/projects/jaer/files/cAER/`.

Longuet-Higgins, H. C., & Prazdny, K. (1980). The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B: Biological Sciences*, *208*(1173), 385–397. doi: 10.1098/rspb.1980.0057.

Lucas, B. D., & Kanade, T. (1981). An iterative technique of image registration and its application to stereo. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 674–679).

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671. doi: 10.1016/s0893-6080(97)00011-7.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, *275*(5297), 213–215. doi: 10.1126/science.275.5297.213.

Masquelier, T., & Thorpe, S. J. (2007). Unsupervised learning of visual features through spike timing dependent plasticity. *Public Library of Science: Computational Biology*, *3*(2), 247–257. doi: 10.1371/journal.pcbi.0030031.

McCarthy, C., & Bames, N. (2004). Performance of optical flow techniques for indoor navigation with a mobile robot. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation* (Vol. 5, pp. 5093–5098). doi: 10.1109/robot.2004.1302525.

McCarthy, C., Barnes, N., & Mahony, R. (2008). A robust docking strategy for a mobile robot using flow field divergence. *IEEE Transactions on Robotics*, *24*(4), 832–842. doi: 10.1109/tro.2008.926871.

McGuire, K., de Croon, G. C. H. E., De Wagter, C., Tuyls, K., & Kappen, H. (2017). Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics and Automation Letters*, *2*(2), 1070–1076. doi: 10.1109/LRA.2017.2658940.

Medathati, N. V. K., Neumann, H., Masson, G. S., & Kornprobst, P. (2016). Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision. *Computer Vision and Image Understanding*, *150*, 1–30. doi: 10.1016/j.cviu.2016.04.009.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*(6197), 668–673. doi: 10.1126/science.1254642.

Mostafa, H. (2016). Supervised learning based on temporal coding in spiking neural networks. Retrieved from `https://arxiv.org/abs/1606.08165`.

Mouraud, A., Puzenat, D., & Paugam-Moisy, H. (2005). DAMNED: A distributed and multithreaded neural event-driven simulation framework. Retrieved from `https://arxiv.org/abs/cs/0512018`.

Mueggler, E., Huber, B., & Scaramuzza, D. (2014). Event-based, 6-DOF pose tracking for high-speed maneuvers. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2761–2768). doi: 10.1109/iros.2014.6942940.

Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., & Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research*, *36*(2), 142–149. doi: 10.1177/0278364917691115.

Nagel, H. H. (1990). Extending the oriented smoothness constraint into the temporal domain and the estimation of derivatives of optical flow. In *European Conference on Computer Vision* (pp. 139–148). doi: 10.1007/BFb0014860.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 807–814). doi: 10.1.1.165.6419.

Nguyen, A., Do, T. T., Caldwell, D. G., & Tsagarakis, N. G. (2017). Real-time pose estimation for event cameras with stacked spatial LSTM networks. Retrieved from `https://arxiv.org/abs/1708.09011`.

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 756–770. doi: 10.1109/TPAMI.2004.17.

Nistér, D., Naroditsky, O., & Bergen, J. (2004). Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 1, pp. 652–659). doi: 10.1109/CVPR.2004.1315094.

O'Connor, P., Neil, D., Liu, S. C., Delbruck, T., & Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, *7*. doi: 10.3389/fnins.2013.00178.

Olah, C. (2015). Understanding LSTM networks. (Blog post) Retrieved 2017-06-12, from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Orchard, G., Benosman, R., Etienne-Cummings, R., & Thakor, N. V. (2013). A spiking neural network architecture for visual motion estimation. In *Proceedings of the 2013 IEEE Biomedical Circuits and Systems Conference* (pp. 298–301). doi: 10.1109/biocas.2013.6679698.

Orchard, G., & Etienne-Cummings, R. (2014). Bioinspired visual motion estimation. *Proceedings of the IEEE*, *102*(10), 1520–1536.

Orchard, G., Jayawant, A., Cohen, G. K., & Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, *9*. doi: 10.3389/fnins.2015.00437.

Orchard, G., Thakor, N. V., & Etienne-Cummings, R. (2013). Real-time motion estimation using spatiotemporal filtering in FPGA. In *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE* (pp. 306–309). doi: 10.1109/biocas.2013.6679700.

Paugam-Moisy, H. (2006). *Spiking neuron networks: A survey* (Tech. Rep.). Idiap Research Institute.

Pillai, S., & Leonard, J. J. (2017). Towards visual ego-motion learning in robots. Retrieved from `https://arxiv.org/abs/1705.10279`.

Ponulak, F., & Kasinski, A. (2006). ReSuMe learning method for spiking neural networks dedicated to neuroprostheses control. In *Proceedings of the EPFL LATSIS Symposium 2006 on Dynamical Principles for Neuroscience and Intelligent Biomimetic Devices* (pp. 119–120).

Posch, C., Matolin, D., & 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. doi: 10.1109/jssc.2010.2085952.

Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., & Delbruck, T. (2014). Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proceedings of the IEEE*, *102*(10), 1470–1484. doi: 10.1109/jproc.2014.2346153.

Ramón y Cajal, S., & Azoulay, L. (1955). *Histologie du système nerveux de l'homme & des vertébrés*. Consejo Superior de Investigaciones Cientificas, Instituto Ramón y Cajal.

Ranjan, A., & Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* (Vol. 2, pp. 2720–2729). doi: 10.1109/cvpr.2017.291.

Rao, R. P. N., & Sejnowski, T. J. (2001). Spike-timing-dependent hebbian plasticity as temporal difference learning. *Neural Computation*, *13*(10), 2221–2237. doi: 10.1162/089976601750541787.

Rebecq, H., Horstschaefer, T., Gallego, G., & Scaramuzza, D. (2017). EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, *2*(2), 593–600. doi: 10.1109/lra.2016.2645143.

Reichardt, W. (1961). Autocorrelation, a principle for the evaluation of sensory information by the central nervous system. *Sensory Communication*, 303–317. doi: 10.7551/mitpress/9780262518420.003.0017.

Ren, Z., Yan, J., Ni, B., Liu, B., Yang, X., & Zha, H. (2017). Unsupervised deep learning for optical flow estimation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence* (pp. 1495–1501).

Revaud, J., Weinzaepfel, P., Harchaoui, Z., & Schmid, C. (2015). EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1164–1172). doi: 10.1109/cvpr.2015.7298720.

Rombouts, J. O., Roelfsema, P. R., & Bohte, S. M. (2012). Neurally plausible reinforcement learning of working memory tasks. In *Advances in Neural Information Processing Systems* (pp. 1871–1879).

Rombouts, J. O., van Ooyen, A., Roelfsema, P. R., & Bohte, S. M. (2012). Biologically plausible multi-dimensional reinforcement learning in neural networks. In *International Conference on Artificial Neural Networks* (pp. 443–450).

Rosten, E., Porter, R., & Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(1), 105–119. doi: 10.1109/tpami.2008.275.

Rousselet, G. A., Fabre-Thorpe, M., & Thorpe, S. J. (2002). Parallel processing in high-level categorization of natural images. *Nature Neuroscience*, *5*(7), 629–630. doi: 10.1038/nn866.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. Retrieved from `https://arxiv.org/abs/1609.04747`.

Ruffier, F., & Franceschini, N. (2005). Optic flow regulation: The key to aircraft automatic guidance. *Robotics and Autonomous Systems*, *50*(4), 177–194. doi: 10.1016/j.robot.2004.09.016.

Ruffier, F., & Franceschini, N. (2015). Optic flow regulation in unsteady environments: A tethered MAV achieves terrain following and targeted landing over a moving platform. *Journal of Intelligent & Robotic Systems*, *79*(2), 275–293. doi: 10.1007/s10846-014-0062-5.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, *5*(3), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. doi: 10.1007/s11263-015-0816-y.

Rust, N. C., Mante, V., Simoncelli, E. P., & Movshon, J. A. (2006). How MT cells analyze the motion of visual patterns. *Nature Neuroscience*, *9*(11), 1421–1431. doi: 10.1038/nn1786.

Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry Part I: The first 30 years and fundamentals. *IEEE Robotics & Automation Magazine*, *18*(4), 80–92. doi: 10.1109/mra.2011.943233.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117. doi: 10.1016/j.neunet.2014.09.003.

Simoncelli, E. P., Adelson, E. H., & Heeger, D. J. (1991). Probability distributions of optical flow. In *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 310–315). doi: 10.1109/CVPR.1991.139707.

Simoncelli, E. P., & Heeger, D. J. (1998). A model of neuronal responses in visual area MT. *Vision Research*, *38*(5), 743–761. doi: 10.1016/s0042-6989(97)00183-1.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. Retrieved from `https://arxiv.org/abs/1409.1556`.

Srinivasan, M. V. (1994). An image-interpolation technique for the computation of optic flow and egomotion. *Biological Cybernetics*, *71*(5), 401–415. doi: 10.1007/s004220050100.

Srinivasan, M. V., Zhang, S., Lehrer, M., & Collett, T. (1996). Honeybee navigation en route to the goal: Visual flight control and odometry. *Journal of Experimental Biology*, *199*(1), 237–244. doi: 10.1006/anbe.1998.0897.

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

Sutton, M. A., Wolters, W. J., Peters, W. H., Ranson, W. F., & McNeill, S. R. (1983). Determination of displacements using an improved digital correlation method. *Image and Vision Computing*, *1*(3), 133–139.

Sze, B., Chen, Y. H., Yang, T. J., & Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. Retrieved from `https://arxiv.org/abs/1703.09039`.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9). doi: 10.1109/cvpr.2015.7298594.

Szeliski, R. (2010). *Computer vision: Algorithms and applications*. Springer Science & Business Media. doi: 10.1007/978-1-84882-935-0.

Tavanaei, A., & Maida, A. S. (2017). Multi-layer unsupervised learning in a spiking convolutional neural network. In *Proceedings of the 2017 IEEE International Joint Conference on Neural Networks* (pp. 2023–2030).

Teney, D., & Hebert, M. (2016). Learning to extract motion from videos in convolutional neural networks. Retrieved from `https://arxiv.org/abs/1601.07532`.

Thorpe, S., Delorme, A., & Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks*, *14*(6), 715–725. doi: 10.1016/s0893-6080(01)00083-1.

Tschechne, S., Sailer, R., & Neumann, H. (2014). Bio-inspired optic flow from event-based neuromorphic sensor input. In *Proceedings of the 6th IAPR Workshop on Artificial Neural Networks in Pattern Recogniftion* (pp. 171–182). doi: 10.1007/978-3-319-11656-3_16.

Ullman, S. (1979). *The interpretation of visual motion.* MIT Press. doi: 10.2307/1575119.

Wang, S., Clark, R., Wen, H., & Trigoni, N. (2017). DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Proceedings of the 2017 IEEE international Conference on Robotics and Automation.*

Weber, J., & Malik, J. (1995). Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, *14*(1), 67–81. doi: 10.1007/bf01421489.

Weikersdorfer, D., & Conradt, J. (2012). Event-based particle filtering for robot self-localization. In *Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics* (pp. 866–870). doi: 10.1109/robio.2012.6491077.

Weinzaepfel, P., Revaud, J., Harchaoui, Z., & Schmid, C. (2013). DeepFlow: Large displacement optical flow with deep matching. In *Proceedings of the 2013 IEEE International Conference on Computer Vision* (pp. 1385–1392). doi: 10.1109/iccv.2013.175.

Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., & Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation* (pp. 957–964). doi: 10.1109/icra.2012.6225147.

Weiss, S., Brockers, R., & Matthies, L. (2013). 4-DOF drift free navigation using inertial cues and optical flow. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4180–4186). doi: 10.1109/iros.2013.6696955.

Wen, H. (2016). VINet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the 31st AAAI International Conference on Artificial and Intelligence.*

Wiesel, T. N. (1982). The postnatal development of the visual cortex and the influence of environment. *Bioscience Reports*, *2*(6), 351–377.

Wu, Y., Deng, L., Li, G., Zhu, J., & Shi, L. (2017). Spatio-temporal backpropagation for training high-performance spiking neural networks. Retrieved from `https://arxiv.org/abs/1706.02609`.

Yang, Z., Murray, A., Worgotter, F., Cameron, K., & Boonsobhak, V. (2006). A neuromorphic depth-from-motion vision model with STDP adaptation. *IEEE Transactions on Neural Networks*, *17*(2), 482–495. doi: 10.1109/tnn.2006.871711.

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. Retrieved from `https://arxiv.org/abs/1511.07122`.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. Retrieved from `https://arxiv.org/abs/1212.5701`.

Zenke, F., & Ganguli, S. (2017). SuperSpike: Supervised learning in multi-layer spiking neural networks. Retrieved from `https://arxiv.org/abs/1705.11146`.

Zhao, S., Li, X., & Bourahla, O. E. F. (2017). Deep optical flow estimation via multi-scale correspondence structure learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (pp. 3490–3496). doi: 10.24963/ijcai.2017/488.

Zweig, S., & Wolf, L. (2016). InterpoNet, A brain inspired neural network for optical flow dense interpolation. Retrieved from `https://arxiv.org/abs/1611.09803`.

"To succeed, planning alone is insufficient. One must improvise as well. I'll improvise."

"Salvor Hardin" in Foundation (1951), by Isaac Asimov.