Autonomous learning in a neuromorphic vision system : an active efficient coding spiking model applied to vision

Thomas Barbier

ÉCOLE DOCTORALE
DES SCIENCES POUR L'INGÉNIEUR
Université Clermont Auvergne

Autonomous learning in a neuromorphic vision system.

An active efficient coding spiking model applied to vision.

Thèse présentée par **Thomas Barbier**

Pour obtenir le grade de **Docteur d'Université**

Spécialité : **Électronique et Systèmes ou Informatique**

Soutenue publiquement le **19 Janvier 2023** devant le jury composé de

Alain Dutech	Rapporteur	Chargé de Recherche, HDR LORIA.
Laurent Perrinet	Rapporteur	Directeur de Recherche INT.
Yulia Sandamirskaya	Examinatrice	Ingénieur de Recherche Intel Labs.
Benoit Cottereau	Examinateur	Directeur de Recherche CerCo.
Fréderic Chausse	Examinateur	Professeur des Universités Université Clermont Auvergne.
Vincent Barra	Examinateur	Professeur des Universités Université Clermont Auvergne.
Jochen Triesch	Directeur	Professeur des Universités Frankfurt Institute for Advanced Studies.
Céline Teulière	Encadrante	Maîtresse de Conférences Université Clermont Auvergne.

Remerciements

Ce manuscrit conclut 3 années de recherches intense mais passionnantes à l'Institut Pascal au sein de l'équipe de vision par ordinateur ComSee. J'ai eu l'occasion pendant cette thèse de découvrir de nombreux domaines qui ne m'étaient pas familier, tel que la neuroscience ou l'ingénierie neuromorphique, qui s'avèrent être des domaines passionnants.

Je tiens tout d'abord à remercier les rapporteurs et examinateurs qui ont pris la peine de s'intéresser à ma recherche et de venir pour certain en présentiel à ma soutenance de thèse.

Je remercie ensuite mes superviseurs, Céline et Jochen, sans qui cette thèse n'aurait pas été possible. Merci pour tout ce que vous m'avez apporté, tant sur le plan scientifique qu'humain. Vous avez toujours su me pousser dans mes raisonnements et m'aider alors que je me confrontait parfois à des problèmes difficiles. Votre aide précieuse m'as permis de mener à bout ma recherche et je vous en suit vraiment reconnaissant.

Je remercie aussi mes collègues et amis . Merci pour toute ces discussions, rires et autre moments de partage, parfois autour d'un café le midi, parfois autour d'une bière le soir. Votre soutien moral direct ou indirect a été essentiel, et cette thèse serait bien plus morose sans vous.

Enfin, merci à ma famille, tout particulièrement mes parents et ma sœur, qui m'ont soutenu inconditionnellement durant ces trois ans. Merci pour leur soutien moral qui m'as été essentiel pour mener à bien ma recherche tout en gardant le sourire.

Abstract

Biological systems continuously adapt their neural representations to the statistics of their sensory input signals to operate efficiently. However, those statistics are shaped directly by the organism's behavior when sampling the environment. In the case of vision, the organism must therefore solve a complex problem of jointly learning visual encoding and eye control without external supervision. This autonomous joint learning of visual representations and actions has been previously modeled in the Active Efficient Coding (AEC) framework and implemented using traditional frame-based cameras as visual sensory inputs. This type of sensor is very well studied and used in many visual applications. Nevertheless, its performance in terms of the acquisition rate, dynamic range, or power consumption remains far from the capability of biological vision systems.

Event-based cameras are a new type of vision sensor. Based on the mammalian eyes, they imitate the early visual pathways, such as the retina. Each pixel unit is independent and emits a signal when it detects a high enough change in light intensity. It operates at a very short timescale (the order of a few microseconds), allowing it to capture swift movements with high precision. A static scene will not create visual feedback, thus avoiding redundant information. Finally, the asynchronous nature of the sensor allows it to drastically reduce the power consumption (a few milliwatts) and increase the dynamic range (more than 120 dB). All those features come with a challenge; the asynchronous output of event-based cameras is not well suited to be used with conventional computer vision. Many popular algorithms, such as artificial neural networks, depend on discrete operations, making them incompatible with this new type of sensor. Spiking Neural Network (SNN)s are bio-inspired networks that try to reproduce the computations of biological neuronal systems. They are especially well suited to be used with event-based cameras.

Building on the AEC framework and using those novel event-based sensors as sensory input, we want to create a system capable of learning smooth eye pursuit and vergence based on efficient coding representations of its environment. Our model is composed of 3 main blocks. The first stage comprises the sensory inputs, carried out by a stereoscopic pair of event-based cameras mounted on a moving robotic head. The second stage comprises a two-layer SNN, which encodes the sensory inputs into an efficient visual representation. This visual representation is fed into the

third stage, composed of a spiking reinforcement learner. This stage is responsible for learning motor commands to maximize a reward signal. This reward signal is computed directly from the activity levels of the efficient coding layer, which is modulated by plastic inhibitory connections learned on specific visual patterns.

Our work on the second stage has been extensively described in [1] and [2]. The spiking neural network is capable of learning orientation, disparity, and motion representations and efficiently tuning to the statistics of the scene using a modified Spike-Timing Dependent Plasticity (STDP) rule. It is composed of two layers, the simple and complex cells, directly inspired by neurons found in the early mammalian visual pathways.

We based the reinforcement framework on a reformulation of the traditional TD-Learning framework for use in spiking continuous applications. It is, therefore, well adapted to work with the efficient coding layer. It comprises a population of critic cells, whose role is to estimate a value function, while the actor cells are directly linked to motor cells. Both populations are directly wired to the first two layers of the SNN, which acts as state representations for the visual inputs.

Taking inspiration from predictive coding ideas, we generate the intrinsic reward directly from the efficient coding layer and modulate the cells' activity using inhibition connections between the cells. Using both a lateral and top-down inhibition scheme, we learn to lower the activity of the network when presented with specific visual patterns. If those patterns appear more often during training, they will naturally be more suppressed. From there, we derive a reward signal inversely proportional to the global activity of the simple cell layer.

Finally, we combine all the blocks to solve visual tasks such as tracking and orientation stabilization. The process is performed in the spike domain, from visual input to motor commands, without needing external supervision.

Keywords: Active Efficient Coding, Spiking Neural Networks, Reinforcement Learning, Intrinsic Reward.

Résumé

Les systèmes biologiques adaptent continuellement leurs représentations neuronales aux statistiques de leurs signaux d'entrée sensoriels pour fonctionner efficacement. Cependant, ces statistiques sont façonnées directement par le comportement de l'organisme lorsqu'il échantillonne l'environnement. Dans le cas de la vision, l'organisme doit donc résoudre un problème complexe d'apprentissage conjoint du codage visuel et du contrôle oculaire sans supervision externe. Cet apprentissage autonome conjoint des représentations visuelles et des actions a été précédemment modélisé dans le cadre Active Efficient Coding (AEC) et mis en œuvre en utilisant des caméras traditionnelles basées sur les images comme entrées sensorielles visuelles. Ce type de capteur est très bien étudié et utilisé dans de nombreuses applications visuelles. Néanmoins, ses performances en termes de taux d'acquisition, de plage dynamique ou de consommation d'énergie restent loin des capacités des systèmes de vision biologiques.

Les caméras événementielles constituent un nouveau type de capteur de vision. Basées sur les yeux des mammifères, elles imitent les premières voies visuelles, telles que la rétine. Chaque pixel est indépendant et émet un signal lorsqu'elle détecte une variation suffisamment importante de l'intensité lumineuse. Il fonctionne à une échelle de temps très courte (de l'ordre de quelques microsecondes), ce qui lui permet de capturer des mouvements rapides avec une grande précision. Une scène statique ne créera pas de retour visuel, évitant ainsi les informations redondantes. Enfin, la nature asynchrone du capteur permet de réduire drastiquement la consommation électrique (quelques milliwatts) et d'augmenter la plage dynamique (plus de 120 dB). Toutes ces caractéristiques s'accompagnent d'un défi : la sortie asynchrone des caméras événementielles n'est pas bien adaptée à la vision par ordinateur conventionnelle. De nombreux algorithmes populaires, tels que les réseaux neuronaux artificiels, dépendent d'opérations discrètes, ce qui les rend incompatibles avec ce nouveau type de capteur. Les réseaux neuronaux artificiels sont des réseaux bio-inspirés qui tentent de reproduire les calculs des systèmes neuronaux biologiques. Ils sont particulièrement bien adaptés pour être utilisés avec des caméras événementielles.

En se basant sur le cadre AEC et en utilisant ces nouveaux capteurs événementiels comme entrée sensorielle, nous voulons créer un système capable d'apprendre la poursuite oculaire et la vergence en douceur en se basant sur des représentations codantes

efficaces de son environnement. Notre modèle est composé de trois blocs principaux. Le premier comprend l'entrée sensorielle, effectuées par une paire stéréoscopique de caméras événementielles, montées sur une tête robotique mobile. La deuxième étape comprend un SNN à deux couches, qui encode les entrées sensorielles en une représentation visuelle efficace. Cette représentation visuelle est introduite dans le troisième bloc, composée d'un apprentissage par renforcement impulsionel. Cet étage est responsable de l'apprentissage des commandes motrices afin de maximiser un signal de récompense. Ce signal de récompense est calculé directement à partir des niveaux d'activité du deuxième bloc, qui est modulée par des connexions inhibitrices plastiques apprises sur des motifs visuels bien spécifiques.

Nos travaux sur le deuxième bloc ont été largement décrits dans [1] et [2]. Le réseau de neurones à impulsions est capable d'apprendre des représentations d'orientation, de disparité et de mouvement et de s'accorder efficacement aux statistiques de la scène en utilisant une règle Spike-Timing Dependent Plasticity (STDP) modifiée. Il est composé de deux couches, les cellules simples et complexes, directement inspirées des neurones présents dans les premières voies visuelles des mammifères.

Nous avons basé le cadre de renforcement sur une reformulation du cadre traditionnel de TD-Learning pour l'utiliser dans des applications à temps continu. Il est donc bien adapté pour fonctionner avec la couche de codage. Il comprend une population de cellules critiques, dont le rôle est d'estimer une fonction de valeur, tandis que les cellules actrices sont directement liées aux cellules motrices. Les deux populations sont directement reliées aux deux premières couches du SNN, qui agit comme des représentations d'état pour les entrées visuelles.

En nous inspirant des idées de codage prédictif, nous générons la récompense intrinsèque directement à partir de la couche de codage efficace et modulons l'activité des cellules en utilisant des connexions d'inhibition entre les cellules. En utilisant un schéma d'inhibition à la fois latéral et descendant, nous apprenons à réduire l'activité du réseau lorsque des motifs visuels spécifiques sont présentés. Si ces motifs apparaissent plus souvent au cours de la formation, ils seront naturellement davantage supprimés. De là, nous dérivons un signal de récompense inversement proportionnel à l'activité globale de la couche de cellules simples.

Enfin, nous combinons tous les blocs pour résoudre des tâches visuelles telles que le suivi et la stabilisation de l'orientation. Le processus est réalisé dans le domaine impulsionel, de l'entrée visuelle aux commandes motrices, sans nécessiter de supervision externe.

Mots-Clés : Codage actif et efficace, réseau de neurones impulsionnels, apprentissage par renforcement, récompense intrinsèque.

Contents

List of Figures

List of Tables

Glossary

binocular In biology, binocular vision is a type of vision in which an animal has two eyes capable of facing the same direction to perceive a single three-dimensional image of its surroundings x, xvi, 7, 17, 18, 37, 43, 44, 78, 84, 98, 131

depth perception Depth perception is the ability to perceive distance to objects in the world using the visual system and visual perception. 56, 92

depth estimation Depth Estimation is the task of measuring the distance of each pixel relative to the camera. 5, 6, 25, 26, 82, 85

dynamic range Dynamic range describes the ratio between the brightest and darkest parts of an image, from pure black to brightest white. It is measured either as a ratio or as a base-10 (decibel) or base-2 (doublings, bits or stops) logarithmic value of the difference between the smallest and largest signal values. 5, 22

extrinsic An extrinsic property is not essential or inherent to the subject that is being characterized. xx, xxi, 21, 108, 109, 115, 119, 120, 121, 124, 125, 127

feature detection Feature detection includes methods for computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not 23

Field-Programmable Gate Array A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing. xxix, 35

grating A grating is any regularly spaced collection of essentially identical, parallel, elongated elements. xix, xxi, xxii, 28, 57, 88, 89, 90, 118, 119, 122, 124, 125

intrinsic In science and engineering, an intrinsic property is a property of a specified subject that exists itself or within the subject. xi, xii, xv, xxii, xxv, 4, 6, 7, 8, 10, 19, 21, 29, 35, 85, 93, 95, 96, 97, 98, 99, 105, 108, 109, 121, 122, 123, 124, 125, 126, 127, 129, 130

latency A measure of the time delay experienced by a system. 4, 5, 21, 23, 24, 26

Long-Term Potentiation In neuroscience, long-term potentiation (LTP) is a persistent strengthening of synapses based on recent patterns of activity. xxix, 32

Long-Term Depression In neursocience, long-term depression (LTD) is an activity-dependent reduction in the efficacy of neuronal synapses lasting hours or longer following a long patterned stimulus. xxix, 32

optical flow Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene xvi, 24, 25, 42, 43, 55, 73, 78, 92

pathway In neuroanatomy, a neural pathway is the connection formed by axons that project from neurons to make synapses onto neurons in another location, to enable neurotransmission (the sending of a signal from one region of the nervous system to another). x, 5, 6, 7, 8, 11, 19, 26, 27, 38, 49

receptive field The receptive field, or sensory space, is a delimited medium where some physiological stimuli can evoke a sensory neuronal response in specific organisms. xi, xvi, xvii, xviii, xix, 10, 17, 28, 37, 38, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 57, 58, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 92, 98, 111, 121

stereoscopic Stereoscopy (also called stereoscopics, or stereo imaging) is a technique for creating or enhancing the illusion of depth in an image by means of stereopsis for binocular vision. xvii, xxv, 43, 52, 56, 57, 78, 80, 82

tracking describes several different methods of extracting camera motion information from a motion picture. xii, xx, xxi, xxii, xxv, 5, 6, 10, 13, 23, 24, 92, 93, 95, 96, 105, 109, 110, 111, 114, 115, 116, 117, 119, 120, 121, 124, 125, 127, 130, 132, 133, 151

vergence Vergence is the simultaneous movement of both eyes in opposite directions to obtain or maintain single binocular vision. xx, 6, 7, 13, 96, 98, 99, 105, 131, 132, 133, 151

Very-Large-Scale Integration Very large-scale integration (VLSI) is the process of creating an integrated circuit by combining millions or billions of MOS transistors onto a single chip. xxx, 18

Acronyms

AEC Active Efficient Coding v, vii, xx, 6, 7, 8, 9, 12, 17, 18, 35, 92, 94, 95, 98, 99, 100, 121, 129, 131
AI Artificial Intelligence 94
ANN Artificial Neural Network 31, 35, 39, 70, 75

CNN Convolutional Neural Network 24
CPU central processing unit 34, 52, 53, 131, 132

ESN Echo State Network 41

FPGA Field-Programmable Gate Array 35, 96
FSV Frequency-normalized Spread Vector 43

GASSOM Generative Adaptive Subspace Self-Organising Map 41
GPU graphics processing unit 34, 35, 53, 95, 96, 132
GUI Graphical User Interface 146, 147, 149, 150

HOTS Hierarchy of event-based Time Surfaces xv, 25

LIF Leaky Integrate and Fire 30, 41, 45, 49, 51
LSTM Long Short-Term Memory 25
LTD Long-Term Depression 32, 48
LTP Long-Term Potentiation 32, 99

MDP Markov Decision Process 13, 14, 15
MSE Mean Squared Error 66, 68, 69, 82

POMDP Partially Observable Markov Decision Process 15

R-STDP Reward Modulated STDP xvi, 7, 10, 33, 34, 95, 96, 97, 102, 114
ROS Robot Operating System xiii, 145, 146, 147, 148, 151, 152

SDC Stochastic-Deterministic Coordinated 97
SNN Spiking Neural Network v, vi, viii, x, xvi, 5, 8, 10, 11, 12, 26, 30, 31, 32, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 49, 52, 53, 58, 65, 66, 75, 92, 95, 96, 100, 102, 105, 126, 129, 130, 132, 147, 152
STDP Spike-Timing Dependent Plasticity vi, viii, x, xvi, xx, 6, 10, 11, 32, 33, 34, 36, 40, 41, 42, 43, 44, 47, 48, 51, 73, 75, 91, 92, 95, 96, 99, 106, 121, 130

TD Temporal Difference xx, 8, 10, 15, 16, 95, 97, 99, 100, 102, 103, 114, 126, 150

VLSI Very-Large-Scale Integration 18

WTA Winner-Take-All 10, 40, 41, 102

Close your eyes.
Count to one.
That's how long forever feels.

– *Kurzgesagt*

Chapter 1

General introduction

1.1 Motivation

In many computer vision applications, efficiency of results is the primary motivating factor for the choice of algorithms and architectures. More and more people turn to deep learning frameworks as they often provide the best results on many classical datasets. However, we often forget that those methods have many drawbacks that make them impractical at best when implementing them in real-world scenarios.

One of the major flaws is simply energy consumption. For instance, training a large natural language processing model such as GPT-3 consumes over 1000 megawatt-hours. That is enough energy to power a small town for a day [3]. In contrast, the brain uses only about 20 watts to operate [4] while performing many more cognitive functions than simply language processing. We still have many things to learn from biological systems.

We are interested in getting closer to the impressive performances of the human brain. The strong adaptation capacity, low energy use, and resilience of biological models are some of the main arguments for trying to create bio-inspired frameworks. The development of event-based cameras follows those principles. It is a sensor that shares many of the advantages of the biological vision system. We think it is perfectly adapted as the sensory organ for a complete spiking visual model.

Autonomous frameworks based on intrinsic motivation offer simple yet convincing ways to learn control tasks without external supervision. Direct feedback from the environment or other biological organisms is limited, even for very social animals. It seems natural that vision can develop without the need for constant external feedback. Curiosity and other attention mechanisms can be a way to induce emergent behaviors. They propose a convincing way of learning effective control commands in complex environments such as vision. Combined with spiking neural networks for learning efficient representations in an unsupervised manner, those autonomous models can provide concrete results on solving visual tasks and exciting insights into how the brain and, more specifically, the visual system operates.

1.2 Context

Biological vision systems During millions of years of evolution, vision has become an essential tool for biological survival in many species. Most rely heavily on vision to search for food, escape predators, or bond with other individuals. Biological vision systems have developed to become highly performant. An eagle can detect prey at more than 3 kilometers in good weather conditions. Flying insects have impressively fast escape response time when subjected to a looming threat, all based on visual feedback.

Mammals and humans possess very versatile vision systems, especially suited for working with others in a group. Humans have been able to thrive partly due to their social capabilities, which are heavily tied to the ability to recognize individuals and their emotions from facial structures.

It is undeniable that vision plays an essential role in the biological system. Therefore, it is no surprise that vision is one of the significant research areas in human history. We developed many ways to improve our visual capabilities with complex optical tools. More recently, the invention of photography has durably changed our society. Cameras are everywhere, with an estimated 4.7 billion images posted online daily.

Computer vision systems Computer vision has progressed tremendously in the last few decades. Applications have flourished in autonomous driving, manufacturing, and surveillance. There is a heavy incentive to develop algorithms with good performance, low latency, power consumption, and ability to work in varying lighting conditions. In that regard, computer vision is still far behind the abilities of biological vision systems.

The human visual system provides unparalleled versatility, the ability to perform tracking, object recognition, depth estimation, and much more in a fraction of a second. Even better, it only consumes a small proportion of the power cameras and computers need to operate.

Standard frame-based cameras are an excellent tool for computer vision. They are cheap, easy to manufacture, and easy to use, especially compared to other technical vision sensors such as lidars or radars. Still, their limitation compared to biological systems have pushed to create more performant alternatives.

Event-based cameras In the last decades, a new type of sensor called an event-based camera has been introduced and promises powerful capabilities compared to frame-based cameras. It offers interesting new approaches for models of vision. Event-based cameras capture information as a continuous stream, excluding most of the spatial redundancy in a scene. They have been designed to resemble the mammalian retina, which makes them a natural choice for designing vision models closer to biology. They have very low latency, high temporal resolution, low power consumption, and high dynamic range. Therefore they are a promising tool for creating a bio-inspired vision system.

Spiking neural networks Processing visual cues can be done in many different ways. We are interested in developing a fully autonomous model that can learn to adapt to visual statistics without needing external supervision. Neural networks have been demonstrated to be practical tools for learning in various applications, vision included. However, their dynamics are very different compared to the neuronal interactions present in the brain. They do not match well with the asynchronous nature of event-based camera output. Spiking Neural Network (SNN) can be designed to mimic some of the intricate properties of neural circuits. They can be easily connected to event-based cameras.

The literature shows two main ways of modeling spike-based neural networks. One focuses on the rate at which neurons spike to encode information, the rate-based approach. The other also considers the Spatio-temporal structure of spike patterns as an essential vector of information. Guyonneau et al. [5] showed how the second approach could effectively learn efficient visual representations and be more robust to fast visual stimuli. They also note the importance of sparsity in neural codes. We are particularly interested in this Spatio-temporal approach to learning visual representation.

Humans learn both the neural representations and control of eye movements without the requirement for external supervision. The visual pathways self-calibrate using only the statistical properties of the input. That can be replicated in models

using self-supervised spiking learning rules such as the popular Spike-Timing Dependent Plasticity (STDP) rule. Nevertheless, when learning motor behaviors, STDP alone is not enough. The agent's actions must be evaluated to tune the system properly. In traditional reinforcement learning, this is done using a reward extracted from the environment. However, this implies using outside supervision. Even though humans use outside supervision, such as imitation or direct feedback, this is only true for more abstract and complex behaviors requiring external judgment. Many vision tasks such as tracking or quick depth estimation are fast processes that cannot rely on external feedback alone.

Active Efficient Coding framework We know that the human visual system changes tremendously in its early stages. As a child, developing vision is an essential task that requires much time. It is an active process combining learning efficient neural representations with accurate movements of the eyes. We are interested in similar vision models that can learn autonomously to capture the visual statistics of a scene and adapt their behaviors accordingly.

The Active Efficient Coding (AEC) hypothesis stipulates that biological systems learn efficient neural representations and behaviors jointly. In the case of vision, eye movements generate specific stimuli that are then learned and stored as an efficient code in the early visual pathways of the brain. Those codes are then used to generate eye motor commands, which may change the visual behavior and thus the visual statistics, implying new stimuli to learn. This continuous and cyclic process helps acquire a robust visual system. The AEC framework proposes a complete and coherent hypothesis on how early vision develops in humans and mammals. It can explain the adaptation of neural circuits to visual inputs by jointly tuning the neural representation and eye behaviors.

The AEC model also proposes that some behaviors can emerge from rewarding efficient representation. The model will naturally select behaviors that promote a better encoding of the input. For that reason, the reward is generated intrinsically from the efficient coding model and transmitted to the reinforcement learner. It is directly tied to the encoding quality, so better encoding generates a higher reward promoting the behavior that leads to improving the encoding.

Such models have been proposed in [6]–[8]. However, they focus on traditional frame-based stimuli, using conventional neural networks to perform accurate vergence control. However, the brain uses fundamentally different ways of processing information. It comprises many groups of neurons arranged in densely inter-connected networks. The transmission of information between neurons is a continuous stream of spikes. In this work, we are interested in extending the AEC framework focusing on a more biologically-plausible approach that mimics the transmission of information in the brain.

Works such as [9]–[11] propose spiking AEC models of binocular vision for vergence control. They extract coarse and fine-scale image patches from a binocular pair of cameras. A correctly verged pair of eyes will create redundant stimuli from the left and right fields, which the sparse coding model will efficiently encode. Therefore, the agent that seeks to maximize coding efficiency will naturally verge his eyes on the target stimulus. Those works are an important basis on which the efficient coding hypothesis has been developed. Nevertheless, they lack some biological plausibility. Most notably, they use rate coding approaches with traditional frame-based cameras, discarding temporal cues that play an important role in biological vision.

For biological systems, being able to encode a signal using minimal activity is primordial, as it means reducing the energy consumed in doing so. An efficient encoding is an encoding that produces minimal cell activity while still extracting meaningful information from the input. Generally, the activity reduction in neural systems is made using specialized inhibitory cells. They can learn to inhibit excitatory cells when presented with a specific stimulus, reducing overall activity. We based our model on a similar assumption. An efficient encoding mechanism associates the most frequent input with the smallest codes. In this work, we learn to strongly inhibit frequent visual stimuli via inhibitory connections, lowering the network's activity for those stimuli. The intrinsic reward is then generated proportionally to the network efficient layer activity; the lowest the activity, the highest the reward.

In the brain, reward signals are often associated with neurochemicals, such as dopamine. They can be transmitted a while after an action has been selected, which in turn affects the synaptic strength related to the selection of that action. We propose a similar model, where the reward generated from the first encoding layers is transmitted to the reinforcement learner as a neuromodulator. The agent learns from a combination of the inputs from the efficient coding layer and the intrinsic reward using a three-factor learning rule, often called Reward Modulated STDP (R-STDP) [12].

1.3 Contributions

We propose a fully spiking neural network composed of 3 main components as depicted in Fig. 1.1. The sensory block is made of event-based cameras that serve as the visual inputs to the system. The second is an efficient coding block, composed of a dual layer mimicking the cells in the early stage of the human visual pathways. It learns effective representations of the visual inputs and transmits them to the next component, the reinforcement learner. It uses a modified, fully spiking TD learning framework to learn effective policies. Finally, the reward the reinforcement learner

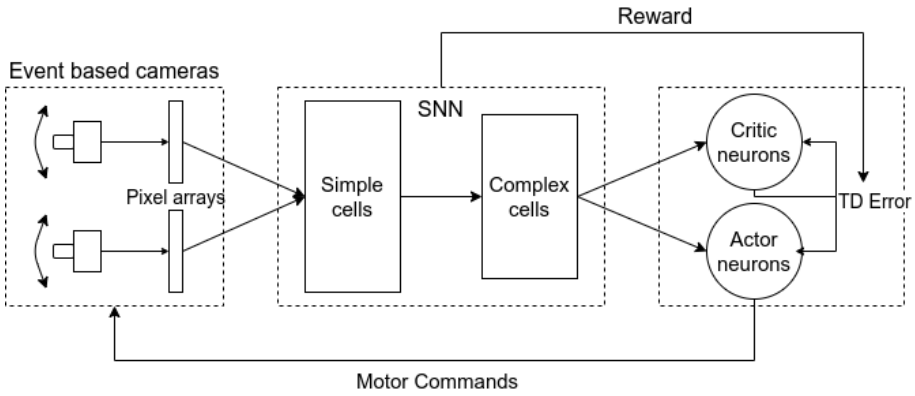


Figure 1.1: AEC model composed of 3 main blocks, the sensory input, efficient coding model, and reinforcement learner. The reward is intrinsically generated from the efficient coding layer.

uses intrinsically generated rewards from the encoding layer through learning lateral and recurrent inhibitory connections within and between the cell layers. The system is built upon the AEC hypothesis. It jointly learns the neural representations as well as effective visual behaviors without the need for any external supervision.

In this manuscript, we present our contribution in two parts, namely:

Efficient visual encoding with a SNN – *partially published as* [1] *and* [2]
We propose a novel dual-layer SNN based on event-based camera inputs that learn efficient visual representation from the scene. We demonstrate the ability of the network to learn Gabor-like representations that are very similar to the ones found in simple cells in the human visual system. We also show that those representations are well-tuned to the statistics of the scene. We validate that hypothesis using a driving dataset recorded with an event-based camera.

Then, we focus on showing the properties of the second layer, akin to the complex cells also found in the brain in the early stages of the visual pathways. We compare our neuronal model orientation response and selectivity to biological findings. We also demonstrate the ability of our cells to learn more complex representations, such as motion and disparity. We test this hypothesis on event sequences taken outside on a mobile robotic platform.

Spiking Reinforcement Learning with an Actor Critic framework – *to be submitted to an international journal*
We propose a fully spiking reinforcement learning network based on the previous contribution. We use the capabilities of the dual layer SNN to learn an efficient visual representation, which is the visual encoding basis for our second part, the reinforcement learning model. This model is based on a continuous TD framework combined with an actor-critic model. We learn an efficient value and policy to tackle complex tasks in a simulated environment. We removed

supervision entirely by generating our reward from the activity of the simple cell layer based on unsupervised learning of inhibitory connections between the cells. It allows the network to be close to fully autonomous in solving the task.

Details on publications and communications are given in Appendix B. All our source code and datasets have been made publicly available on the GitHub of Institut Pascal, comsee-research, for reproducibility and broad accessibility, as:

Neuvisys is an open-source C++ spiking neural network library.
Available at `https://github.com/comsee-research/Neuvisys`.

Neuvisys-simulator is an open-source event-based simulator based on the CoppeliaSim framework.
Available at `https://github.com/comsee-research/Neuvisys-simulator`.

Neuvisys-analysis is an open-source python cluster of useful methods to create, modify, process, and analyze raw events as well as the neuvisy spiking neural network library.
Available at `https://github.com/comsee-research/Neuvisys-analysis`.

More details can be found in Appendix C.

1.4 Manuscript outline

The manuscript is organized as follows:

Chapter 2 provides background information and motivation for our contribution. Following the 3 part architecture, we give detailed information on the AEC reinforcement learning framework, event-based cameras, and spiking neural networks. We first introduce the general reinforcement learning concept and how the AEC model fits into it. Then, we present event-based cameras, a novel sensor type. After a short historical overview, we describe in detail their operating principles. We specifically focus on their differences from traditional frame-based cameras, the dominant vision sensor in the market today. We then describe the critical relationship between biological vision and event-based cameras and how the latter can be used as an accurate human retina model in bio-inspired computer models. Finally, we quickly review how they are used in concrete applications and compare the main ways of processing event streams. In the second part, we introduce the concept of neuromorphic engineering and describe some of the leading models for creating bio-inspired

neural networks. We detail the different architectures as well as learning mechanisms and hardware implementations.

Chapter 3 covers the problem of how to efficiently encode visual information and extract visual features such as orientation, motion, or depth from event-based outputs in an unsupervised manner. We describe our work constructing a two-layer SNN based on two biological cells found in the early human visual system, the simple and complex cells. We insist on the vital role of those cells as early feature detectors and show that our computer model can perform similarly. We describe in detail the cell model that we built our SNN on, as well as all the bio-inspired mechanisms that help regulate the activity and learned representation. We also describe the modified STDP rule that we are using to learn efficient representations in an unsupervised manner. Then, we analyze the resulting learned receptive fields of both the simple and complex cells and compare them to biological findings. Finally, we present applications of the framework on a driving dataset or using our robotic platform to perform depth analysis.

Chapter 4 covers the problem of solving closed-loop visual tasks involving robotics. We present a fully spiking reinforcement learning framework based on a continuous formulation of the discrete TD learning model. We describe the two main drivers of the reinforcement learning agent, the critic and actor neuron population. The first try to estimate a value function based on the continuous state described by the efficient coding layer in the form of simple and complex cell activation. The actor neurons are cells connected to specific motor actions. Through a Winner-Take-All (WTA) mechanism, the cells with the most activation will decide the following action to perform. Both populations learn using a three-factor rule called R-STDP, a mix of a fully unsupervised STDP learning rule and an external reward signal. The reward signal from the environment can be replaced by an intrinsic reward signal, learned using a novel inhibition scheme in the efficient coding layer. We describe in more detail how the network can regulate its activity by learning those inhibitory connections to produce an effective intrinsic reward signal for its reinforcement learning agent. Finally, we test the reinforcement learning framework in simulated environments on a tracking and stabilization task.

Chapter 5 provides a general conclusion with discussions about our contributions and perspectives for improvements and future works.

Chapter 2

Background

Introduction

This Chapter provides the theoretical and mathematical background of our work. It describes the fundamental knowledge required to understand our model and some of its motivations.

We introduce the concept of neuromorphic engineering and how our work fits in that domain. We explain the main arguments that motivated us to create a fully spiking and autonomous vision system based on the AEC framework. We detail the core principles of the AEC framework and how it evolved from traditional reinforcement learning.

Then, this Chapter provides insight into the operating principles of event-based cameras. Similarly to how the brain is adapted to work with the eyes, we constructed the whole processing and application framework based on event-based cameras. They are the sensing tool that allows the framework to get information on the scene. For that reason, understanding their operation principle, as well as their advantages and limitation, is primordial. We will focus on how they operate without getting into technical details about the inner circuitry itself. We will compare them to its biggest competitor, frame-based cameras, the dominant types of visual sensors in today's market. We will review a few traditional computer vision applications made with event-based cameras.

Finally, we will explore some of the main biological concepts and models that inspired the neuromorphic community and us. We will see how event-based cameras have been designed to resemble the mammalian retina and produce similar outputs. We will describe some biological visual systems components and their mathematical model. We focus on the design of SNN and their learning mechanisms. We also discuss the possible hardware implementation strategies for those algorithms.

2.1 Active Efficient Coding

Vision is an active process that can be framed as a reinforcement learning problem. We will first provide some background on reinforcement learning in a discrete environment and from there introduce the AEC framework.

2.1.1 Background on Reinforcement Learning

Reinforcement learning is a general framework that model sequential decision processes. It is traditionally described as an agent acting in an environment. The agent receives information from the environment through state and reward. Then, depending on that information, the agent takes action on the environment. The process usually repeats itself until the agent can solve a specific task. Figure 2.1 resumes that process.

Vision fits well in that framework. The agent is the eyes that perceive the environment as light information. The actions would usually be the movements of the eyes that allow the agent to perceive new information. The reward can be anything linked to visual tasks such as fixation, tracking, and vergence.

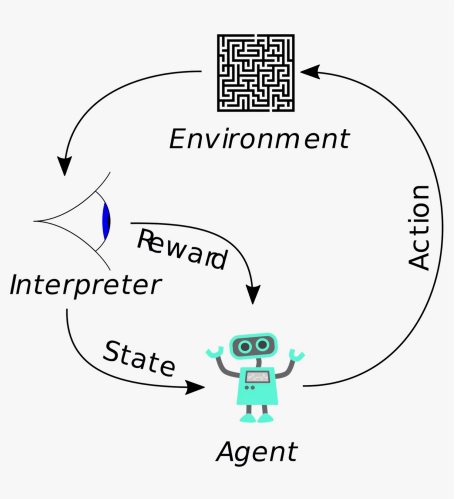


Figure 2.1: Reinforcement learning diagram of an agent environment interaction. The agent receives a state and reward from the environment and can act on the latter via actions. It is a cyclic process.

2.1.1.1 Markov Decision Process

In classic reinforcement learning, the environment is often described as a Markov Decision Process (MDP). A MDP is defined by an environment where the states s follow this property:

$$\mathbb{P}(s_{t+1}|s_0, s_1, ..., s_t) = \mathbb{P}(s_{t+1}|s_t) \tag{2.1}$$

meaning that the knowledge of the previous state is enough to determine the next state fully.

In a discrete formulation, the agent moves from state to state every time step t. The framework is composed of the following:

- A set of environment and agent states S.
- A set of actions A.
- The transition T from one state s to another s' under action a defined by the stochastic probability:

$$\mathbb{P}(s, a, s') = \mathbb{P}_r\left(s_{t+1} = s'|s_t = s, a_t = a\right). \tag{2.2}$$

- The reward function $R(s, a, s')$ obtained when transitioning from one state s to s' under action a.

In a MDP, the goal of the agent is to find a policy $\pi(a, s)$ that gives the probability of taking action a when in state s:

$$\pi : A \times S \to [0, 1], \pi(a, s) = \mathbb{P}_r(a_t = a|s_t = s). \tag{2.3}$$

that maximizes the expected return.

For MDP in continuous space, the environment contains an infinity of states s and actions a. The transition T become a stochastic probability density written $T'(s, a, s')$ with $\int_S T(s, a, s')\, ds' = 1$.

State and Action-Value function In order to compare policies, we use the value function $V^\pi(s)$ as a criterion that associates a state with a value, indicating how desirable it is to be in that state. It is defined as the expected sum of all the discounted future rewards when following the policy π:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \tag{2.4}$$

where r_t is the reward at step t and γ is the discount factor ($\gamma \in [0, 1)$]). The discount factor balances the importance of future rewards compared to immediate ones.

While the state value function estimates the quality of a state alone, the action-value function estimates it for a state and action pair. It is defined as:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]. \tag{2.5}$$

The value function can be used to compare two policies. It is said that π_a dominates π_b if:

$$V_a^\pi(s) \geq V_b^\pi(s). \tag{2.6}$$

In that framework, the agent's purpose is to learn an optimal policy. The optimal policy π^* is achieved when selecting the best action in each state, leading to maximum overall reward. There exist no other policies that dominate it. It respects the following constraint:

$$\pi^* \in \arg\max_\pi V^\pi. \tag{2.7}$$

In the same way, the optimal state value and action-value functions can be defined as:

$$V^*(s) = V^{\pi^*}(s) = \max_\pi V^\pi(s), \tag{2.8}$$

$$Q^*(s, a) = Q^{\pi^*}(s, a) = \max_\pi Q^\pi(s, a). \tag{2.9}$$

Bellman equation The Bellman equation is a necessary optimality condition in dynamic programming methods. It decomposes the value function in two parts, an immediate payoff and the discounted value from successive states:

$$V^\pi(s) = \mathbb{E}_{s'|s} \left[R(s, \pi(s), s') + \gamma V^\pi(s') \right], \tag{2.10}$$

$$Q^\pi(s, a) = \mathbb{E}_{s'|s,a} \left[R(s, a, s') + \gamma Q^\pi(s', a) \right]. \tag{2.11}$$

While the Bellman optimality equations are written as:

$$V^*(s) = \max_{a \in A} \mathbb{E}_{s'|s,a} \left[R(s, \pi(s), s') + \gamma V^*(s') \right], \tag{2.12}$$

$$Q^*(s,a) = \mathbb{E}_{s'|s,a} \left[R(s,a,s') + \gamma \max_{b \in A} Q^*(s', b) \right]. \tag{2.13}$$

with a and b two separate actions that belong to the action space A.

The Bellman optimality equations help break the dynamic optimization problem into a sequence of more straightforward problems.

2.1.1.2 Resolution methods

Fully and partially observable MDP In a fully observable MDP, the agent possesses full knowledge of the states information. Otherwise, the environment is said to be partially observable. Most environments in vision tasks are only partially observable since the eyes rarely cover the entirety of the visual environment. In a Partially Observable Markov Decision Process (POMDP), the Markov hypothesis defined in equation (2.1) is not necessarily verified. However, it is still possible to solve a POMDP by assuming that the environment follows the properties of a fully observable MDP.

In a finite and discrete environment where the environment models T and R are known, it is possible to use dynamic programming methods to solve the MDP. Otherwise, the agent must learn effective representations of the environment while interacting with the latter. Two main categories of algorithm exist, **model-based** methods that use a model of the environment and **model-free** methods that do not.

There exist three classes of resolution methods, *Critic-Only*, *Actor-Only*, and *Actor-Critic*. Critic-only learns value functions and then deduces policies from them. Actor-only methods try to directly learn a better policy than the previous one, sometimes by estimating value functions without learning them. Finally, the actor-critic method learns both value functions and policies at the same time.

We will focus here on actor-critic methods since our reinforcement learner is based on this class of methods.

Temporal difference method The Temporal Difference (TD) learning method is based on the Bellman equations. It uses an estimated value function $\hat{V}(s_t)$, a parametric estimator of the true value function $V^\pi(s_t)$. In a MDP, considering the transition (s_t, a_t, s_{t+1}), the Bellman equation can be written as follow:

$$V^\pi(s_t) \approx r_t + \gamma V^\pi(s_{t+1}) \tag{2.14}$$

An estimator $\hat{r}$ of the reward r_t will therefore be defined by:

$$\hat{r}_t \approx \hat{V}^{\pi}(s_t) - \gamma \hat{V}^{\pi}(s_{t+1}) \tag{2.15}$$

From there, we can define the temporal difference error as the difference between the observed reward and its estimation $\delta_t = r_t - \hat{r}_t$. The temporal difference error can then be written:

$$\delta_t = \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t) + r_t. \tag{2.16}$$

The estimation of the value function can be corrected using the following:

$$\hat{V}^{\pi}(s) \leftarrow \hat{V}^{\pi}(s) + \alpha_t \delta_t \tag{2.17}$$

with α_t a learning rate parameter.

The algorithm for the TD learning algorithm is the following:

Algorithm 1 TD algorithm

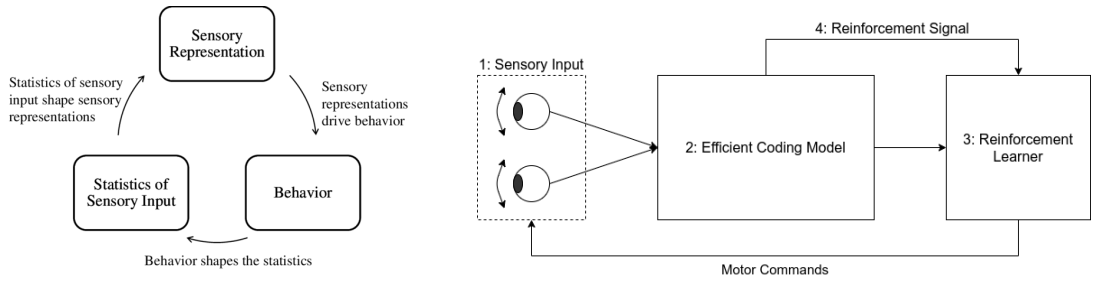
Initialize $\hat{V}$
for $t = 1, 2, 3, \ldots$ **do**
  Observe (s_t, s_{t+1}, r_t)
  $\delta_t = \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t) + r_t$
  $\hat{V}^{\pi}(s) \leftarrow \hat{V}^{\pi}(s) + \alpha_t \delta_t$ if $s = s_t$
end for

The presented algorithm allows the agent to obtain an efficient state estimation in the form of the value function estimation. We then need to improve the policy to learn to solve the task. We describe this process in Chapter 4. We based our reinforcement learning framework on a TD actor-critic method. However, since our framework operates in continuous time, we redefined the TD framework to work with a fully spiking reinforcement learning formulation.

2.1.2 Active Efficient Coding

2.1.2.1 Efficient coding hypothesis

Information coding is an essential tool in the brain as the latter has limited resources. The brain alone is responsible for about 20% of energy consumption [13]. Therefore, tuning neural systems for energy efficiency presents a clear evolutionary advantage. The efficient coding hypothesis states that the brain can save much energy by learning sensory representation tuned to the statistics of sensory signals. It is a theoretical model that describes how information is encoded by neural systems and was first developed in 1961 by Horace Barlow [14].



(a) Active Efficient Coding hypothesis. (b)

Figure 2.2: (a) Active efficient coding hypothesis (b) Active efficient coding general architecture.

Since then, numerous experiments have shown that neurons can learn efficient representation directly correlated to the sensory inputs [15], [16]. In the visual system, neuronal receptive fields can be explained using unsupervised models that learn to encode natural images [17], [18] efficiently.

The efficient coding hypothesis implies that neural codes directly depend on the statistics of the sensory signal. Those statistics are a function of the environment and the organism's behavior. However, the efficient coding hypothesis limits itself to environmental information and does not necessarily consider the organism's behavior. The AEC framework is an addition to the efficient coding model. It extends efficient coding to active perception by considering motor actions such as eye movements for vision. It states that perception and behavior are closely intertwined in an action-perception cycle, similar to reinforcement learning.

2.1.2.2 AEC extension

The AEC framework states that biological agents jointly optimize their neural coding and behavior in a closed-loop system to create an efficient sensory representation of the environment. Figure. 2.2 present in a concise manner both the AEC hypothesis and its architecture. Models for the development of active binocular vision have been proposed in [6]–[8], [19].

Biological systems can directly shape the statistics of their sensory inputs from their behavior. The organism can maximize its coding efficiency by tuning neural representations and behavior simultaneously. This additional degree of control gives them an undeniable advantage.

Contrary to traditional reinforcement learning, the AEC framework does not rely on external supervision to learn. We saw that reinforcement learning methods use a reward to optimize their policy. In AEC, the goal is to optimize neural representation to maximize coding efficiency. The latter can be estimated directly from those neural representations without needing external feedback to judge the

learned representation's quality. For instance, in active binocular vision systems, the reward is generated from a reconstruction error between the left and right visual encoding.

An organism's ability to self-calibrate without external supervision is also a clear evolutionary advantage. Even for mammals that heavily rely on mimicking or teaching behaviors, those feedbacks are limited and cannot explain the efficiency of control systems. The AEC ability to self-calibrate is a convincing argument.

Active Inference The development of sensory encoding and behaviors in biological systems has been proposed in other coding hypotheses. Active inference is a framework that encompasses encoding and behavior adaptation through a model of minimizing sensory input surprise [20]. It is an approach to understanding the brain framed as a single imperative to minimize free energy [21]. This hypothesis has extended many models, such as [22], [23]. However, we will not focus on it in this manuscript as we are primarily interested in developing a spiking AEC framework instead.

2.2 Event-based cameras

In this section, we introduce event-based cameras and how they have redefined the field of computer vision. In the first place, however, we will introduce the concept of neuromorphic engineering. Then, we detail the operating principles of event-based cameras. We compare their novel way of sensing light information to the more traditional frame-based approaches [24]. A review of many of those sensors can be found here [25].

2.2.1 Neuromorphic engineering

The word neuromorphic designates using analog or digital circuits in Very-Large-Scale Integration (VLSI) systems to mimic neuro-biological architectures. It is a field of science dedicated to studying the brain and finding ways of replicating some of its underlying mechanisms. The term has been extended with time to work using software to mimic biological architectures. The neural systems can have many purposes, such as perception, motor control, or multisensory integration.

The neuromorphic community works actively to understand the morphology of neural systems, their interactions, the representation of information, and how they learn. They build systems for capturing and processing sensory information, such

as vision and auditory systems, as well as their robotic integration. It is an active interdisciplinary field of research taking inspiration from domains such as biology, physic, mathematics, computer science, and electrical engineering.

Neuromorphic vision Vision is of primary importance in neuromorphic engineering [26]. Many essential applications in society rely on the ability to perceive the environment. The development of event-based cameras offers the perfect opportunity for the neuromorphic community to develop bio-inspired algorithms that mimic the visual pathways of mammals and humans. This novel sensor has redefined the field of neuromorphic vision and sparked considerable interest even beyond it. Many models have emerged recently using that sensor as the basis for visual sensing. We will discuss some of them in the following two chapters.

2.2.2 Quick historical overview

Event-based cameras are a recent development in computer vision. The prototype was featured on the cover of the journal Scientific American[27] in May 1991, presented in Fig. 2.3.

Developed by the graduate student Misha Mahowald at Caltech University, this Silicon Retina, though still very limited, was the first attempt at mimicking the eye's neural architecture from a joint biological and engineering perspective. Since then, event-based camera designs have changed a lot. With each iteration, we observe increasing performances and resolutions. We present a few event-based cameras in Fig. 2.13.

The development of event-based cameras is intrinsically tied to the one of neuromorphic engineering.

Figure 2.3: Cover of the Scientific American of May, Volume 264, Issue 5, featuring the picture of a cat taken with the Silicon Eye.

2.2.3 Event representation

Conventional camera capture visual information at specific rates depending on an external clock. It works by letting light accumulate on a sensor, outputting a frame reflecting the amount of light that exists in a scene. It means there is an extended time when a frame-based camera will not output any information.

In contrast, event-based cameras respond to brightness changes in the scene. Each pixel continuously records the amount of light change, and if the brightness log

(a) (b)

Figure 2.4: (a) Metavision EVK4 – HD Event-based camera from Prophesee. (b) From left to right, DVXplorer Mini, DVXplorer Lite, DVXplorer, and Davis346 event-based cameras from Inivation.

intensity exceeds a specified threshold, this pixel activates (spikes) and outputs an event. Four variables represent an event: t, the time at which the change happened, x and y pixel location, and p, the 1-bit polarity indicating the type of change ("ON" means brightness increase, "OFF" means brightness decrease). The timestamps t are precise in the order of a microsecond. One event can usually be stored with less than two bytes.

Thus, event-based cameras have variable data-rate. A still scene without any light change will not create any output, whereas high-density scenes with lots of movements might create very high output rates.

2.2.4 Mathematical model for event generation

Event-based sensors consist of an array of independent pixels that react to light intensity variation. Events generated from a specific pixel are written as the tuple $e_k = (x, y, t, p)$, with x and y the pixel coordinates, t the timestamp, and p the polarity. One pixel responds to its log photocurrent,

$$L = \log(I) \tag{2.18}$$

with L the photocurrent and I the light intensity. An event is generated if the brightness change

$$\Delta L\left(x, y, t\right) = L\left(x, y, t\right) - L\left(x, y, t - \Delta t\right) \tag{2.19}$$

exceeds a temporal contrast threshold C

$$\Delta L\left(x, y, t\right) = PC \tag{2.20}$$

with $C > 0$ and $p \in \{+1, -1\}$. The contrast sensitivity C can be adapted depending on brightness conditions, sensor noise... ON and OFF polarity each have its contrast sensitivity C^+ and C^-.

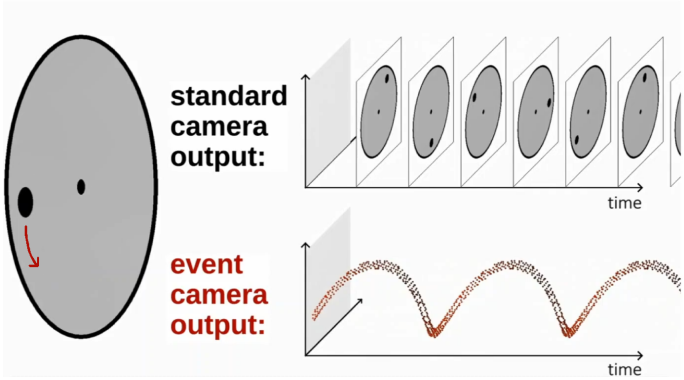


Figure 2.5: Difference between a frame-based and event-based output from Gallego et al. [28].

Event generation can measure the temporal derivative of brightness instead of the absolute value in frame-based cameras. It is written as:

$$\Delta L\left(x, y, t\right) \approx \frac{\partial L}{\partial t}\left(x, t\right) \Delta t \tag{2.21}$$

$$\frac{\partial L}{\partial t} \approx \frac{pC}{\Delta t} \tag{2.22}$$

This model does not consider some extrinsic or intrinsic phenomena such as sensor noise. However, it is an effective way of relating events to the physical illumination of the scene. An example of event generation from a rotating dot is displayed in Fig. 2.5.

2.2.5 Advantages compared to frame-based cameras

Event-based cameras, by their asynchronous nature and engineering specificity, offer many advantages over traditional frame-based cameras:

High temporal resolution The monitoring brightness being independent for each pixel, it is possible to create high-speed circuitry with fast readouts. Events are detected with microseconds resolutions, much higher than most frame-based cameras, and offer exact temporal information even for fast motions. It also almost eliminates the motion blur limitation of traditional frame-based cameras.

Low latency By removing the global exposure time of a frame, event-based cameras can send information as soon as they receive some, which could result in latency as low as a few microseconds. In reality, due to limitations in information transmission, most event-based cameras on the market have about sub-milliseconds performance.

High dynamic range The logarithmic scale used by the photoreceptors in the pixel, as well as their independent nature, offers event-based cameras a very high dynamic range, often above 120dB. It easily exceeds the performance of even a good-quality frame-based camera, averaging around 60dB. It gives event-based cameras remarkable performance in low-light applications or very contrasted scenes.

Low power Finally, removing redundant data in a scene allows event-based cameras to function with low power consumption. Most market event-based camera use about 10mW, which makes them ideal for embedded applications that require strict power limitations.

As we showed here, event-based cameras present several desirable properties. Frame-based cameras can perform similarly to event-based cameras when explicitly designed for performance. High-speed cameras achieve extreme frame rates of a few thousand frames per second, or industrial high dynamic range cameras reach up to 140dB. However, those specific designs can be expensive, bulky, or energy-intensive, while event-based cameras achieve well on all points while being small and cheap, considering they are still prototypes.

2.2.6 Challenges of a new sensing paradigm

The asynchronous nature of event-based cameras poses a real challenge compared to frame-based cameras. Most historic vision algorithms and processing methods are based on frame-based images and matrix computations. Even other less common visual sensors, such as lidars or radars, which also present sparse visual representations, rely on gathering data at fixed intervals of time. On the other hand, event-based cameras output a continuous flow of sparse events, making them incompatible with traditional vision processing algorithms. Another significant difference is the information encoded itself. Frame-based cameras encode the light intensity as a grayscale value, whereas event-based cameras only record the change in light intensity as binary information. Finally, noise is still a real handicap for event-based cameras. The sensor's high sensitivity combined with inherent transistor circuit noise means some events will inevitably be generated even though there were no changes in the scene. It is especially problematic in low light conditions, where noise increases dramatically.

All those differences must be considered when designing a system with event-based cameras. As it is still a novel sensor, we need more extensive research. It benefits frame-based camera processing methods. However, the advantages that confer event-based cameras are interesting enough that many people have already started implementing novel vision algorithms specifically for the asynchronous nature of events. As time passes, the performance gap between the two sensing paradigms

is closing for some of the most critical applications and datasets. It also paves the way for algorithms that might be used with other types of asynchronous sensors. For instance, brain interfaces for bio-medical settings might benefit significantly from those approaches.

2.2.7 Applications

We will review applications related to the work done in this research manuscript, namely feature detection and tracking, as well as depth and optimal flow estimation. We, however, exclude bio-inspired methods that will be discussed in the next Chapter. We will describe some traditional computer vision algorithms used with event-based cameras. Those applications can give a good understanding of what it is possible to do with the performances of event-based cameras and as a benchmark to compare bio-inspired approaches. We selected a few exciting applications but did not intend to be exhaustive. We focused on works that have laid the foundation for further research and innovative approaches in the field. The reader can refer to [28] for a more exhaustive list of applications.

2.2.7.1 Feature detection and tracking

Event-based cameras are handy in tracking applications. Their very low latency allows them to output information on the scene more often than frame-based cameras. It is a significant advantage when tracking an object as precisely as possible. However, in the case of a non-static camera, events will be produced both by the object's movements in the scene and the camera's movements. It is essential to distinguish between both to perform efficient tracking.

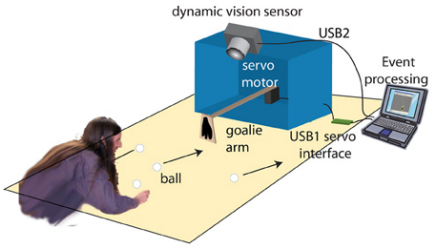


Figure 2.6: Robotic goalie with super-human reaction time by Delbruck et al. [29]

Simple tracking methods are designed around the principle that events generated by an object in motion will be close in time and space. Therefore, clustering methods have been demonstrated in [30]–[32]. Delbruck et al. [29] presented a robotic goalie with 3ms reaction time, shown in Fig. 2.6. Schraml et al. [33] showed a person-tracking solution in real-time, and Conradt et al. [34] demonstrated a pencil balancer using two event-based cameras and an event-driven fitting of the pencil model. Those applications use straightforward algorithms but demonstrate that the extremely low latency of event-based cameras can be used to solve challenging real-time applications.

Later, more complex algorithms were developed that are more robust to sensor noise and camera movement. Rea et al. [35] proposed a visual attention mechanism using two event-based cameras mounted on an iCub robot. It shows a faster reaction time than using frame-based cameras. Similarly, Glover et al. [36] also use an Icub robot combined with a particle filter for a visual tracking algorithm. Lagorce et al. [37] show high-speed tracking of multiple visual features simultaneously. Ni et al. [38] performed an object-tracking algorithm based on the geometric transformation between a model and the events from the object.

Some approaches use event- and frame-based sensors to improve the tracking. It is possible to combine the low latency of event-based cameras with the temporal correspondence of frames to create effective tracking algorithms, such as in [40]. Li et al. [41] base their method on a correlation filter on the event stream and event integration over time to recreate frame-based representations that are then used in a traditional Convolutional Neural Network (CNN).

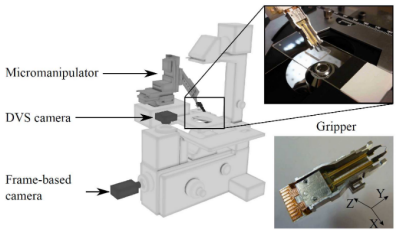


Figure 2.7: Micromanipulator combining event-based and frame-based camera by Ni et al. [39]

Ni et al. [39] present a visual shape-tracking algorithm for micro-robotics using an iterative closest-point approach 2.7. They combine the fast event-based output for direct haptic feedback with the precise spatial output of a frame-based camera for retrieving the object position.

Finally, Chin et al. [42] demonstrated that it is possible to use event-based cameras mounted on space-related optics to track stars and other celestial bodies.

2.2.7.2 Visual motion and optical flow estimation

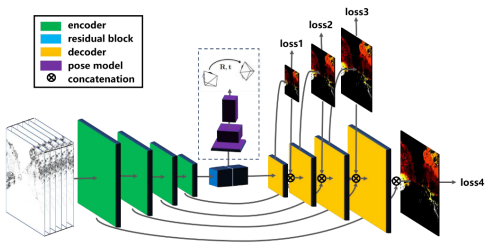


Figure 2.8: Deep neural network for motion estimation with event-based cameras by Zhu et al. [43]

The fundamental principle of event-based cameras is to detect dynamic changes in a scene. That makes them especially suited to estimate local and global motion and the optical flow in a dynamic environment. Much optical flow processing is based on using spatial neighboring pixel information to determine the local optic flow. However, with events, when one pixel is triggered, there is no guarantee that its neighbor will be too in a short temporal window.

First methods such as [44], [45] were based on using surfaces of active events to compute the normal flow. However, they are limited to that component only. Measuring the full optical flow on the tangential and normal component is more challenging, but has been done in [43], [46]–[48]. Many of those approaches adapt the event representation to work with deep neural networks. For instance, Monforte et al. [49] transform the event input flow in a target center of mass and then transmit it to a Long Short-Term Memory (LSTM) for trajectory prediction. Figure 2.8 presents one of those models.

2.2.7.3 3D depth estimation and reconstruction

There are many different ways to estimate depth from visual input. We will focus on stereo setups here since we are interested in comparing them to human vision, which uses both eyes to estimate depth.

It is possible to use spatial and temporal information to try and recreate a 3D scene. In 3D reconstruction, both cameras are rigidly attached to a common axis, which allows some simplification, such as assuming that the left and right pixels sit on the same epipolar line. Also, by synchronizing the clocks of both cameras, the event should share a similar timestamp between the two cameras.

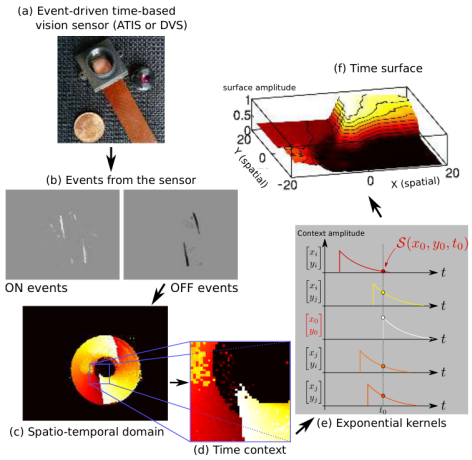


Figure 2.9: HOTS method for the Spatio-temporal processing of event streams by Lagorce et al. [50]

Earlier attempts at stereo vision used the classical computer vision solution, solving the correspondence problem across both image planes and then triangulating the 3D point location. Some work proceeded on aggregating the events into frame-based representation to do so [33], [51], while others use event-by-event methods such as [52]. Lagorce et al. [50] developed Hierarchy of event-based Time Surfaces (HOTS), presented in Fig. 2.9, a general purpose event by event method for event-based processing that can be used in many applications such as depth estimation. Grimaldi et al. [53] have extended that framework recently.

By using additional information, like event polarity, epipolar constraints, or local scene edge orientation, it is possible to improve event matching and get better performances [54]–[56].

By using only global information instead of local ones, cooperative approaches are pretty successful at reconstructing 3D information [57], [58]. Once again, the

low latency of event-based data makes it possible to perform 3D depth estimation at very high rates. It can be critical in real-time applications that require fast depth estimation for further processing. However, most techniques described here happen on the visual scene with limited movements and often static cameras.

2.3 Spiking Neural Networks

SNNs are based on biological mechanisms and processes. We will therefore describe how biological neurons operate, then study some of the mathematical models used to describe them, and finally see their integration in larger systems such as spiking neural networks.

2.3.1 Biological vision pathway

The human retinal system comprises three main parts: the photoreceptors, bipolar cells, and ganglion cells [25], [59]–[62]. They are separated as the outer, middle, and inner layers, as depicted in Fig. 2.10. Photoreceptors convert light into electrical pulses transmitted to bipolar and ganglion cells before entering optical fibers and the V1 area.

The retina is an essential part of the visual system, as it does an essential pre-processing of visual information. At least 12 bipolar cells are thought to encode different types of visual information and transmit them to specialized ganglion cells [63]. Ganglion cells are divided into two subgroups, X and Y cells. X cells are focused in the fovea and carry information such as color, patterns, and spatial details. On the other hand, Y cells are found around the fovea and carry information on the changes in the scene, such as object movement, speed, and distance.

These Y cells, part of the transient pathways, are dedicated to processing dynamic visual cues. They represent about 30% of the visual system. The transient pathways extend quite deep into the brain, from the retina up to the thalamus and V1 areas.

2.3.2 A model inspired by the early visual system

Event-based cameras mimic the properties of the biological retina [63]. Figure 2.10 present a simplified structure of the retina. Each pixel's internal circuitry comprises a photoreceptor part that captures the light intensity, a differencing circuit that acts as a thresholding mechanism on the light intensity change, and a comparator to determine the polarity of the change.

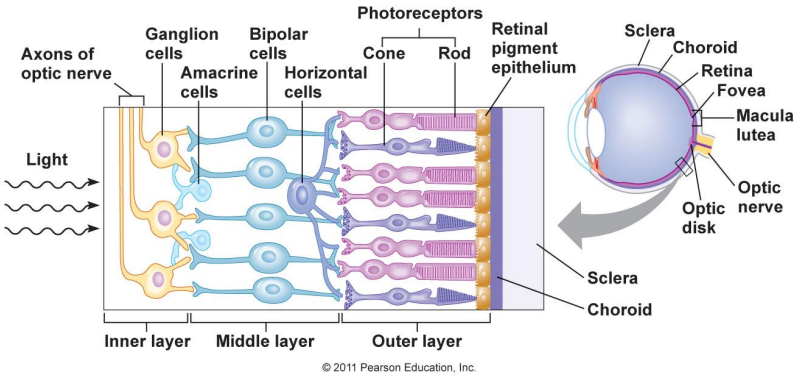


Figure 2.10: Simplified diagram of the retina.

Event-based camera work very similarly to the Y ganglion cells. They correspond to the transient pathways up to the retinal ganglion cells. They can detect changes in light intensity which relay information on scene changes.

2.3.3 Biological neuron models

Neurons are cells found in the nervous system of biological systems. They are considered the central information-processing unit of the nervous system.

Neurons are usually represented with three main parts, **dentrites**, an **axon**, and a cell body, the **soma**, as shown in Fig. 2.11. The dendrites are where the neuron receives inputs from other neurons. Dendritic trees are believed to be responsible for many early processing and non-linearities. Meanwhile, the axon acts as the output towards other neurons, while the neuron's nucleus lies in the soma, where the DNA lies.

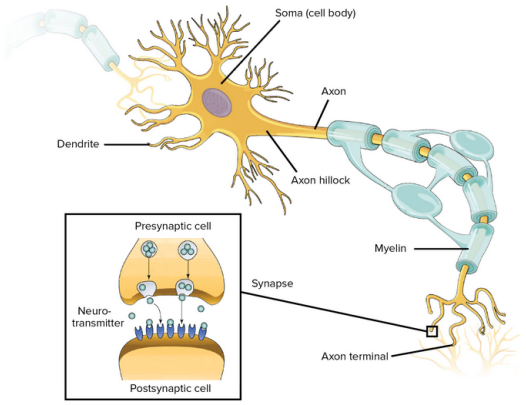


Figure 2.11: Neuron and synapse diagram from Khanacademy.org.

Neurons use electrical signals and chemical bindings to communicate with each other. When a neuron receives action potentials from other neurons, it accumulates them in its membrane potential inside the soma. When the input action potential accumulation exceeds a threshold, the neuron is excited. It fires an action potential of its own that is transmitted to other neurons via the axon.

The end of the axon connects to other neurons' dendrites through **synapses**. The transmission of information in this region is done via chemical neurotransmitters.

Therefore, neurons communicate through electrical impulses in the axon and dendrites and via chemical bonding in the synapses.

We call **pre-synaptic** the neuron that sends the action potential, and **post-synaptic** the one that receives that action potential relative to a synapse.

Simple cells Simple cells [64] are neurons found in the primary visual cortex (V1). They are one of the earliest visual detectors after the retina. Their main characteristic is that they are selective to specific orientations. They only respond to well-defined oriented edges and gratings.

We often define their receptive field as a Gabor function [40], [65], a popular filter for visual processing. Fig. 2.12 represents such a Gabor filter, with the blue region indicating inhibition and the red facilitation.

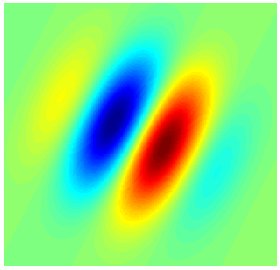


Figure 2.12: Example of a Gabor function.

Complex cells Complex cells [64] are neurons found in the primary visual cortex (V1) and secondary visual cortex (V2). They also respond to oriented edges and gratings with a degree of spatial invariance. They receive inputs from multiple simple cells. Their receptive field cannot be mapped into fixed excitatory and inhibitory zones.

2.3.3.1 Excitation and inhibition

Action potentials do not necessarily result in increasing the membrane potential. Some neurons are specialized cells that send inhibitory signals that result in a decrease of the membrane potential of the post-synaptic neuron.

Inhibition in biological systems is considered essential for the system's stability. Interneurons responsible for inhibition are only a fraction of the totality of neurons in the brain, and yet they play an essential role in regulating the activity of principal cells [66], [67]. Furthermore, it has been shown that neurons' selectivity to stimulus is also partly due to inhibition [68]. The ratio between the two is the Excitation/Inhibition balance.

2.3.3.2 Homeostatis

Homeostasis refers to how living organisms regulate their internal physical and chemical conditions to maintain optimal functions. In the case of neuronal systems,

homeostatic mechanisms refer to principles in the brain that help neurons regulate their activity over time. [69], [70]

For instance, inhibition can be a homeostatic mechanism to keep neuronal activity levels between certain bounds. Many homeostatic processes exist that act on the intrinsic and synaptic properties of neurons to maintain target electrical activity.

2.3.3.3 Synaptic plasticity

Synaptic plasticity is a crucial element for neural systems. It is the core mechanism through which neurons can learn and develop memory in the brain.

Synaptic plasticity controls the strength of communication between two neurons. It happens at the synaptic connection between a pre-synaptic and post-synaptic cell. By altering the number of neurotransmitter receptors on a synapse, a neuron can tune the amount of potential that will be transmitted during an action potential.

There are two types of synaptic plasticity, long-term and short-term. Short-term plasticity occurs on a sub-second timescale. It helps control the information a neuron receives but reverts to pre-change levels a few moments later. Long-term plasticity, however, can last for much longer, up to hours, days, or even years. It is the dominant model of how neural systems store information.

Synaptic plasticity exists both for excitatory and inhibitory synapses.

2.3.4 Mathematical neuron models

Although relatively small, neurons are complicated biological machines with many properties that are still poorly understood. Nevertheless, when abstracting some of their more intricate functions, it is possible to create mathematical models of how neurons operate that are precise enough to operate computer algorithms. We will describe some of the most used ones.

Perfect Integrate and Fire The perfect integrate and fire neuron model is one of the earliest and simplest models, dating back to Louis Lapicque in 1907. A neuron is represented by its membrane voltage potential V which evolves following:

$$I(t) = C_m \frac{dV_m}{dt} \tag{2.23}$$

with C the membrane potential capacitance and I the flowing current. If the membrane voltage exceeds a threshold V_θ, it is reset to its resting potential V_{rest}, and a spike is produced. The firing frequency of these neurons increases, therefore, linearly with the input voltage.

Leaky Integrate and Fire The Leaky Integrate and Fire (LIF) neuron model is very similar to the perfect integrate and fire. It is one of the most popular neuron models and offers great simplicity compared to more detailed bio-physical models [71] while still providing an accurate depiction of biological principles. The main difference with the perfect integrate and fire is that leakage is added to the model as:

$$C_m \frac{dV_m}{dt} = I(t) - \frac{V_m(t)}{R_m}. \tag{2.24}$$

V_m is the voltage across the cell membrane, and R_m is the membrane resistance. If no input current is fed into the neuron, the membrane potential will naturally decrease to the resting potential.

Hodgkin–Huxley The Hodgkin–Huxley model was described in 1952 by Alan Hodgkin and Andrew Huxley. It is a complete formulation for a neuron model that includes the relationship between the flow of ionic currents. They earned a Nobel prize in Physiology or Medicine in 1963 for their work on that model. It describes how action potentials are initiated and propagated using a set of nonlinear differentiable equations.

The voltage-current (V_m, I_i) relationship can be written as:

$$C_m \frac{dV_m}{dt} = -\sum_i I_i(t, V) \tag{2.25}$$

with C_m the cell membrane capacity. Each current is given by:

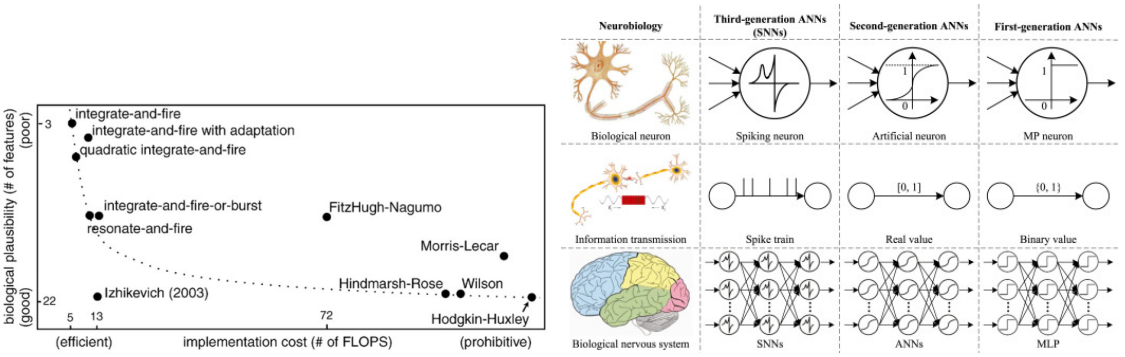
$$I_i(t, V) = g(t, V).(V - V_i) \tag{2.26}$$

where $g(t, V)$ is the conductance of ion channels, and V_i is the reversal potential of the specific ion channel. The typical ionic currents used are Calcium Ca^{2+} and Sodium Na^+ input currents, as well as varieties of Potassium K^+ outward currents.

Many other more advanced biophysical models of neurons exist, but we will not describe them here. Please refer to Fig. 2.13a or [64], [72] for a more exhaustive list.

2.3.5 Spiking neural networks

SNNs are interconnected networks of the previously defined neuronal models. When a neuron spikes by exceeding its membrane potential threshold, it sends signals to other connected neurons via synapses. A synapse transmits information from one neuron to another. When a neuron receives an input from a synapse (called a "pre-synaptic" input) at a time t, its membrane potential changes by the amount w_i. If this change excites the next neuron, then the process repeats itself. The strength



(a) (b)

Figure 2.13: (a) Computation cost versus biological plausibility of different neuron models from [72]. (b) Different generations of neural networks from [73].

of a synapse w_i is plastic, i.e., it can change with time. Modifying the synaptic weights is the primary way to control the behavior of a SNN.

For a fixed input, the network dynamic will depend on many factors. The network connection architecture, the type of neuron mathematical model, their internal parameters, and the strength of their synaptic connections will all determine the behavior of a SNN.

Contrary to Artificial Neural Network (ANN)s, SNN incorporate the concept of time into their operating model. We observe those operating differences in Fig. 2.13b. There is no fixed propagation cycle at which the information is transmitted. Instead, they continuously transmit spikes in their internal neuronal layers. The output will, therefore, also be a sparse temporal code.

2.3.6 SNN learning mechanisms

Learning in SNNs involves changing the weight in the synaptic connections between the neurons. ANNs rely mainly on learning using the backpropagation method. However, this rule is not easily transposable to SNNs. There exist three main ways of training SNNs:

- Supervised learning with gradient descent and spike backpropagation.
- Unsupervised learning through local learning rules.
- Reinforcement learning with reward modulated plasticity.

We will focus here on the last two as they are most closely related to our work.

2.3.7 Spike-Timing Dependent Plasticity

Spike-Timing Dependent Plasticity (STDP) is a fundamental learning mechanism of the nervous system. It is a temporally asymmetric form of Hebbian learning. M. M. Taylor suggested it in 1973 [74] while the first experiments were carried out by W. B. Levy and O. Steward [75] in 1983. However, the first demonstration of the classic STDP rule is attributed to Markram and Sakmann [76] in 1997.

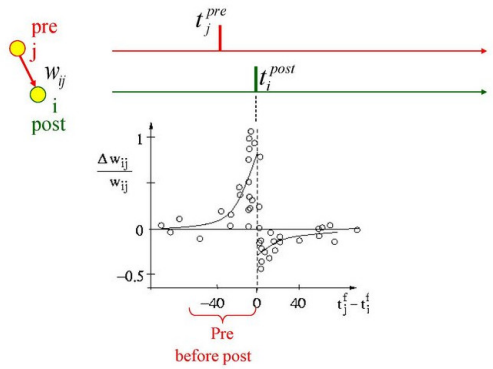


Figure 2.14: Exponential STDP interaction between a pre and post-synaptic neuron. It displays here an exponential STDP window. Image is taken from Scholarpedia.org.

The basic STDP model states that if a neuron's spike causes another neuron to spike, the synaptic connection from the first neuron to the second one will be strengthened. Conversely, if the post-synaptic neuron spikes before the pre-synaptic one, this generally leads to the weakening of the synaptic connection. This simple mechanism has given rise to one of the most popular techniques for unsupervised learning in SNNs via so-called STDP rules [77]. There are many different formulations of STDP, but the main idea stays the same. Synaptic weights evolve depending on the relative timing between pre and post-synaptic spikes. It is demonstrated in Fig. 2.14.

The sign and magnitude of a synaptic weight change depend on the relative timing of pre-and post-synaptic spikes. In the most common form of STDP, pre-synaptic spikes arriving shortly before a post-synaptic spike will lead to Long-Term Potentiation (LTP) of the synapse, while the reverse timing leads to Long-Term Depression (LTD).

The weight changes δ_{w_j} of a synapse from a pre-synaptic neuron j is linked to the relative timing between pre-synaptic spike arrivals t_j^f and post-synaptic spikes t_i^n. f and n are a count of the pre and post-synaptic spikes, respectively. The total weight change is written as:

$$\delta_{w_j} = \sum_{f=1}^{N} \sum_{n=1}^{N} W\left(t_i^n - t_j^f\right) \tag{2.27}$$

where $W(x)$ correspond to the STDP learning window. It is possible to use many types of windows, such as exponential, Gaussian or even triangle and step functions.

The above mathematical formulation considers all the pre and post-synaptic spikes when computing the weight change. It is called an all-to-all interaction and can be challenging to implement as it has a high computation cost. The usual technique

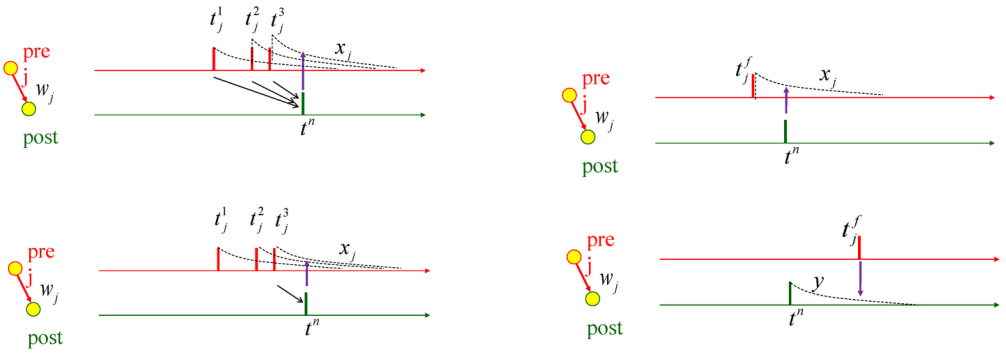


Figure 2.15: (a) Example of an all-to-all (top) versus nearest neighbor (bottom) STDP formulation. (b) Use of a synaptic trace to implement the STDP learning rule. Images is taken from Scholarpedia.org.

to solve that issue is the use of online update rules with the use of synaptic traces. When a pre-synaptic (respectively post-synaptic) spike arrives, it leaves a trace $x_j(t)$ (respectively $x_i(t)$), which decays exponentially after that. When the post-synaptic spike arrives, the weight is updated by the value of that trace. Fig. 2.15b presents an example of synaptic traces.

Other methods restrict the interactions between spikes to only their nearest neighbors. There are many ways of doing so. The most basic only considers the latest pre and post-synaptic spikes, while more advanced variations consider an arbitrary number of pre and post-synaptic spikes. Fig.2.15a shows an all-to-all interaction on top and the nearest neighbor interaction with only the latest pre-synaptic spike at the bottom.

2.3.8 Reward-modulated STDP

Reward Modulated STDP (R-STDP) is a learning rule based on two terms: an unsupervised STDP term similar to the one detailed above and a third-factor term that acts as the trigger and modulator for the synaptic changes.

In the R-STDP rule, the weight update is only performed when receiving information about the states visited recently. In biological systems, this is equivalent to neuromodulators, chemical substances that affect the synaptic transmission deployed after performing specific actions. Neuromodulators are either positive or negative depending on the type of reward the organism receives. Traditional positive and negative rewards, respectively, include the delivery of food or the infliction of pain when working with animals.

In order to keep track of all the usual changes that the STDP rule would create, the R-STDP rule is based on synaptic eligibility traces. Each synapse is given a small membrane potential that evolves every time the neuron spikes. We apply the usual STDP potentiation and depression to it, in addition to a decay term written:

$$\Delta w_{ei}(t + \Delta t) = w_{ei}(t)\frac{e^{-\Delta t}}{\tau_e} \tag{2.28}$$

with w_{ei} the eligibility trace and τ_e the eligibility decay time constant. As the agent moves in state space, the eligibility traces will keep track of the neuronal patterns coming from the representations layers. The time constant τ_e determines how far back in the past they will keep information. Then, once the reward is received, we apply the changes to the real weights. For that, we multiply the weights with the eligibility traces and the reward term:

$$\Delta w_i(t) = \eta w_{ei}(t) d(t) \tag{2.29}$$

with η the learning rate and $d(t)$ the third reward term that acts as the neuromodulator. The reward term indicates the quality of the previous agent's behavior. A negative reward indicates that the actions taken by the agent led to a worse state, while a positive one indicates an improvement in the agent's state. Therefore, the agent's behavior, reflected by the information stored in the eligibility traces, is either encouraged or punished.

With this rule, the neurons can learn even with sparse rewards since the information is stored in the eligibility traces. It is beneficial for tasks in continuous time or for experiments where the reward is distributed only at the end or after meaningful actions.

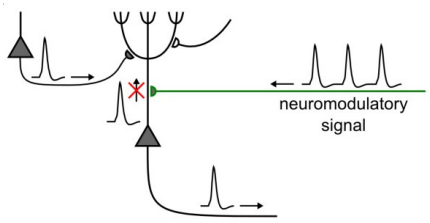
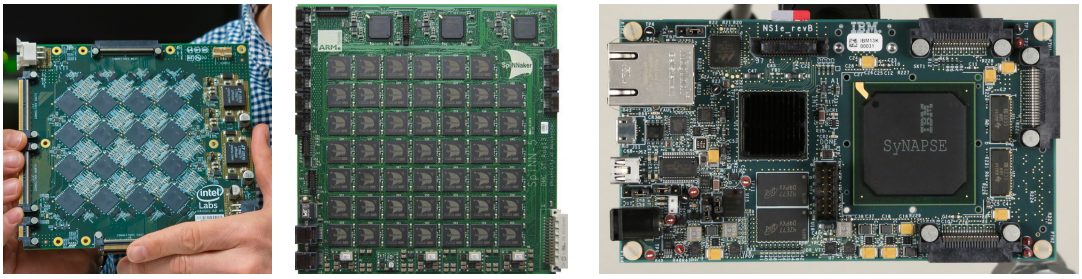


Figure 2.16: R-STDP learning rule. The neuromodulatory signal act as a third-factor rule on synapses. Illustration from [78].

2.3.9 Hardware implementation of event-based algorithms

Traditionally, computer algorithms are developed on modern computers with classical architectures equipped with central processing unit (CPU)s and graphics processing unit (GPU)s. CPUs are computing units designed to process operation sequentially, while GPUs are made for parallel computing. In that regard, CPUs are rarely a good option for event-based processing inputs since they are by nature asynchronous. However, the reality is that they remain the most accessible option when developing new methods since most programming languages are only compatible with them by default. More effort



(a) (b) (c)

Figure 2.17: (a) Loihi Intel's neuromorphic chip. (b) Spinnaker board. (c) Truenorth IBM's synapse board.

is required to make the code GPU compatible. However, the gain can be pretty interesting depending on the algorithm.

Bio-inspired approaches are complex to code on a GPU, although some SNN libraries offer the possibility to work with them. Nevertheless, even though GPUs are highly parallel, they are designed to work with dense representation, especially images. Therefore, converting the event stream into dense images and using ANNs is often a more accessible option.

Specific hardware can be designed to work directly with SNNs and event-based cameras for best performances. Extensive research has been done on jointly developing and running an algorithm on dedicated hardware such as a Field-Programmable Gate Array (FPGA) board. The advantage is that it is possible to create a board that is ideally suited to the need of the algorithm. However, it is a time and resource-intensive effort that cannot always be envisioned.

Neuromorphic hardware is the answer to that problem. Neuromorphic chips are versatile hardware requiring less effort than implementing an FPGA solution while significantly increasing running performances. They are well suited to work with the asynchronous flow of events processed into a SNN. They exist in many different configurations, either entirely analogous, numerical, or a mix of both. We present a few neuromorphic chips in Fig. 2.17.

Conclusion

In this Chapter, we presented the theoretical and mathematical background on which our framework has been developed. We based our framework on the AEC model, derived from a traditional reinforcement learning framework with an intrinsically generated reward. Motivated by using novel neuromorphic vision sensors

that are event-based cameras, we demonstrated that they possess exciting properties and are an excellent tool for bio-inspired research. Their asynchronous output is ideally suited to be used with SNNs and the STDP learning rule.

Event-based cameras offer an advantageous pre-processing mechanism that extracts objects' contours while removing redundant uniform surfaces. However, their asynchronous output varies depending on the observed scene, which makes them hard to use with traditional computer vision models. We demonstrated a few applications designed to process events showing promising results. In the next Chapter, we focus on the neuromorphic community's bio-inspired methods developed over time. Their low energy consumption and high adaptivity directly inspired by biological systems make them promising tools that fit our initial motivations. SNNs appear ideally suited to efficiently and autonomously compute the fast stream of events. We will show that it is possible to use SNNs with unsupervised learning techniques to extract efficient visual representations from event streams.

Chapter 3

Efficient visual encoding with a SNN

Introduction

We introduced in Chapter 2 how event-based cameras are an excellent tool for sensing visual information in a bio-inspired fashion. To fully exploit the benefits of these sensors, subsequent processing steps must take advantage of their low-latency asynchronous output. There is growing evidence that the visual system employs a sparse code to learn visual representations. Neuron's model based on efficient coding can be a biologically-plausible approach to learning visual codes [79]. Spiking Neural Network (SNN)s are ideally suited for this [80]–[82], since they follow the same architectural principles as used by the brain, using an arrangement of comparatively simple computational units ("neurons") connected via weighted connections ("synapses") and communicating asynchronously via spikes. At present, it is not yet fully understood how the connectivity of the visual cortex is set up through developmental and learning processes [83]. However, learning in the brain does not require direct supervision. Here, we attempt to mimic the unsupervised learning abilities of the visual cortex in a neuromorphic vision system.

To this end, we present a novel spiking neural network inspired by the mammalian visual cortex's so-called simple and complex cells [64]. To the best of our knowledge, it is the first SNN that learns complex cells via a timing-based rule from event-based visual input. We demonstrate that it learns feature detectors for local orientation, disparity, and motion from the asynchronous inputs of a pair of event-based vision sensors without supervision. The network combines an unsupervised learning mechanism: spike-timing-dependent plasticity, with homeostatic regulation of neuronal firing thresholds and multiplicative synaptic normalization to prevent unbounded growth of synapses. A simple, fast inhibition scheme decorrelates neural responses and effectively prevents multiple neurons from developing identical receptive fields. We present a dual-layer network inspired by simple and complex cells, biological neurons found in the brain's early visual pathways. We show that our network learns motion-sensitive receptive fields in an unsupervised fashion that qualitatively resembles receptive fields observed in the visual cortex. We demonstrate that the learned representations match biological findings and come to reflect the statistical properties of the input signals in terms of their distribution of orientation, motion, and disparity tuning. We validate our approach on simulated event sequences, an

event-based driving dataset in various road environments, and a mobile robotic platform in a recreated urban scene.

3.1 Related work

We saw in Chapter 2 a short insight on event-based processing algorithms and applications for event-based data. We first present a few SNN supervised learning methods for reference then introduce state of the art for bio-inspired SNN methods for vision. We focus on unsupervised learning for feature, motion, and disparity detection.

3.1.1 Supervised learning with SNN

One easy way to work with event-based data is to transform the continuous stream of information into a discrete one by integrating events into frame representation. That way, it is possible to use traditional computer vision methods such as ANN. We mentioned some of those methods in Chapter 2. However, we will focus here on methods that rely on SNNs to encode the visual information. Various attempts have been made at solving classification and recognition tasks using SNNs fed with event data. Often, they use spiking convolutional neural networks inspired by their frame-based counterparts.

There are two main ways of doing so. The first consists of training a standard ANN using backpropagation and then converting the network to an SNN [84], [85]. Though inferior to the ANN itself, this can lead to good results but is not very versatile, scalable, or biologically plausible. Fig. 3.1 presents such an architecture.

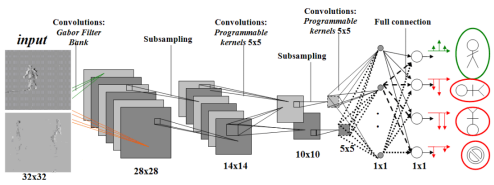


Figure 3.1: Convnet SNN architecture by Perez-Carrasco et al. [81]

The second involves learning directly in the spike domain, using a spike-based backpropagation method. Though those rules are still not fully biologically plausible, since they require extensive training data, they benefit from learning directly in the spike domain. We detailed those methods in Chapter 2.

Some research has pre-wired early visual feature detectors, sidestepping the learning problem. Chicca et al. [86] present a hard-wired neuronal circuit for orientation detection.

3.1.2 Unsupervised learning methods

The primary unsupervised learning approach to train SNN is to use a formulation of STDP as seen in Chapter 2.

Masquelier et al. [87] demonstrated unsupervised learning of hierarchical visual representations with STDP. They alternate simple cells layer for selectivity through a sum operation and complex cell layers for shift and scale invariance through a max operation. It allows them to learn increasingly complex visual features for adequate classification.

Kheradpisheh et al. [88] Propose a convolutional SNN architecture with up to 7 layers, shown in Fig. 3.2, taking inspiration from traditional deep learning. They learn to differentiate objects in images efficiently. Iakymchuk et al. [89] propose a simplified SNN which uses a spike response model combined with STDP architecture that learns on rate coded images of handwritten numbers. Srinivasan et al. [90] introduced a novel STDP-based convolution-over-time learning methodology in a convolutional network and applied it to handwritten digit recognition. Paulun et al. [91] used a bio-inspired down-sampling of the visual field fed to a complex brain-like Neucube architecture. The learning was done with a combination of STDP rules, and the last stage was comprised of a supervised SNN classifier.

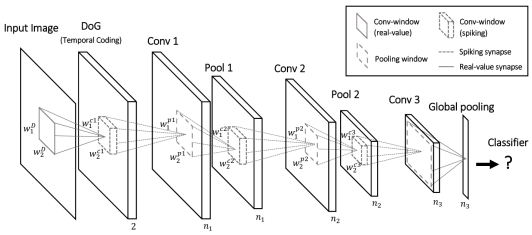


Figure 3.2: Deep SNN architecture for object classification by Kheradpisheh et al. [88]

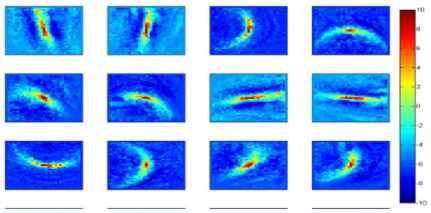


Figure 3.3: Example of receptive fields learned by Akolkar et al. [92]

In those approaches, the learned representations get increasingly complex the deeper the layer, from Gabors in the first layers to parts of the objects themselves in the last. Those deep spiking architectures are very performant but could be better suited to be used with vision control. They can be slow to learn, energy-intensive, or introduce processing delays. We are interested in more bio-inspired lightweight architectures that can be easily used in conjunction with a reinforcement learning agent.

Diehl et al. [80] use a combination of an excitatory and inhibitory layer based on a Winner-Take-All (WTA) mechanism for digit recognition. Their original architecture could be more scalable to larger and more complex inputs. Tavanaei et al. [93] demonstrated a one layer SNN which learns efficient visual representation on the MNIST dataset. Akolkar et al. [92] demonstrate the possibility of learning visual

receptive fields, shown in Fig. 3.3, in an event-driven framework that outperforms traditional Gabor filters.

Chandrapala et al. [94] demonstrate a 2-layered SNN that learns invariant feature representations similar to simple and complex cells in the brain. The learning uses Generative Adaptive Subspace Self-Organising Map (GASSOM) instead of STDP but gives somewhat similar results. Their architecture is shown in Fig. 3.5. They learn to differentiate digits from the MNIST-DVS dataset with 90% accuracy. Liu et al. [95] introduced a new "Multiscale Spatio-Temporal Feature" representation and applied it to recognition tasks such as gesture or digit recognition via STDP. Lagorce et al. [37] presented engaging unsupervised learning of visual representations based on an Echo State Network (ESN) with a WTA learning mechanism instead of the traditional SNN.

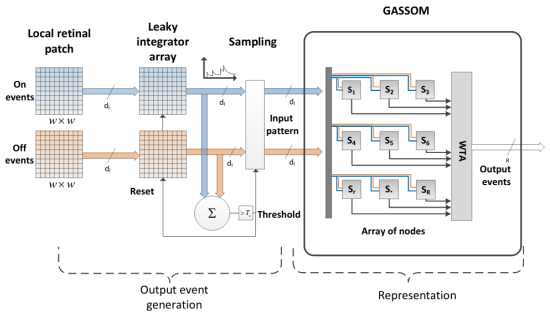


Figure 3.4: GASSOM architecture for learning simple and complex cells like representation by Chandrapala et al. [94]

Franciosini et al. [96] demonstrated a predictive SNN capable of learning simple and complex cells, but lacks ties to event-based cameras.

While all the mentioned architectures are based on the same fundamental operating principle, each has its design features. While most rely on LIF neurons, the dynamics of such models can change a lot depending on the parameters used. Similarly, there are many ways to formulate the STDP rule, which can give different results. Finally, additional homeostatic mechanisms such as inhibition connections or activity regulation features will also impact the behavior of a SNN.

Many of those works use the MNIST recognition task to compare their results. The main drawback of this dataset is that it is not an event-based dataset. They convert the frame representation to a spike code using a rate-coding approach. It limits the interest of using SNNs with precise timing capabilities.

Our network model shares architectural features with several of the works above. However, we focused on using actual event-based data for learning to fully exploit our model's precise temporal capabilities. Furthermore, we demonstrate the ability of our network to learn not only a spatial representation but also motion and disparity features.

3.1.3 Learning to capture motion

The primate visual system uses specific populations of neurons tuned to different motion directions and velocities to estimate object motion. Some work has been done on using event-driven data on bio-inspired ways of sensing motion. A survey of some of those methods can be found in [97]. We focus here on bio-inspired approaches using SNNs to estimate visual motion in the scene.

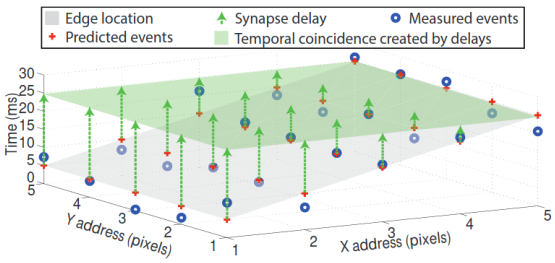


Figure 3.5: Receptive field with synaptic delays for motion estimation by Orchard et al. [102]

Tschechne et al. [98] estimate the optical flow of a scene with the help of a set of Spatio-temporal filters created by combining Gabor functions. Variants of the Barlow and Levick model, a direction-sensitive neuronal triplet, have been employed to create a network capable of estimating optic flow in [99] and [100]. The second proposes an implementation of IBM's TrueNorth neuro-synaptic system.

Li et al. [47] used motion energy detectors as a lateral geniculate model and Gabor kernels to estimate motion. Hopkins et al. [101] created a biologically inspired STDP-based network on the SpiNNaker neuromorphic chip. They detect motion using synaptic delays while successfully solving a more traditional recognition task by learning spatial representation.

Orchard et al. [102] have created a motion-sensing SNN using synaptic delays. Each neuron is designed to detect a specific motion direction and speed. The main limitation is that the delays and orientations for the whole population must be set by hand.

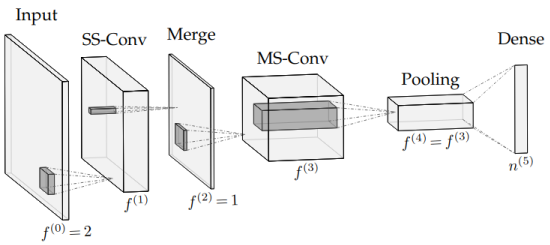


Figure 3.6: Convolutional SNN for optical flow estimation by Paredes-Vallés et al. [103]

All the mentioned works here do not autonomously learn to tune to motion. They have predetermined motion-sensitive filters instead. We use a similar technique with synaptic delays but learn the receptive fields with STDP so that the neurons are tuned to motion without external supervision.

Grimaldi et al. [104] presented a network with synaptic delays capable of learning temporal receptive fields for fast motion detection, as depicted in Fig. 3.7. After learning, their kernels become selective to different motion directions.

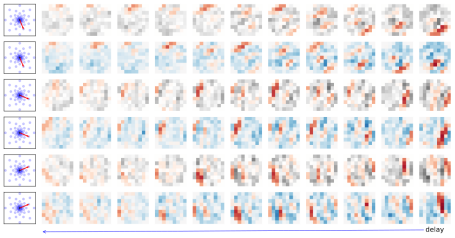


Figure 3.7: Example of a learned basis of receptive field from synaptic delays by Grimaldi et al. [104]

More recently, Paredes-Vallés et al. [103] have proposed a framework for learning motion-sensitive receptive fields from an event-based camera via a form of STDP in an unsupervised fashion. Their work is closely related to ours, but there are several differences. First, the depression part of their STDP rule does not require presynaptic input spikes to arrive shortly after a postsynaptic spike, an important feature of biological STDP. Second, they use a convolutional network architecture, which enforces the development of identical receptive fields in all parts of the visual field. It prevents the receptive fields at different locations from adapting to systematic differences in the statistics of optical flow signals across the visual field as is typical, e.g., during ego-motion. We propose a similar weight-sharing mechanism but with the ability to create visual regions to allow neurons to tune to the statistics of the different parts of the visual scene.

3.1.4 Learning binocular disparity

In the field of stereo vision, there have been attempts at creating networks to estimate binocular disparities between two event-based cameras. A survey of existing methods can be found in [105].

Dikov et al. [107] present a SNN that extracts depth information from a stereoscopic input stream. They implemented it on the SpiNNaker platform. Osswald et al. [108] demonstrate a two-stage architecture featuring coincidence and disparity detectors. Pozzi et al. [106] propose a clever arrangement of excitatory and inhibitory connections, shown in Fig. 3.8, in a triangular fashion to solve the correspondence problem.

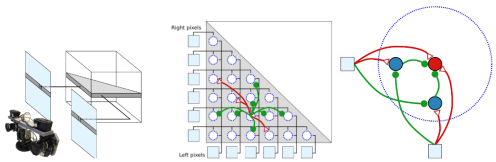


Figure 3.8: Triangular structure for disparity detection by Pozzi et al. [106]

These methods present interesting biological properties and cleverly designed network structures, but they lack a decisive trait: the ability to learn.

Chauhan et al. [109] propose a SNN with an abstract rank-based STDP rule for learning binocular disparity selective representations. Fig. 3.9 presents their architecture. They show that they learn receptive fields that respond rather closely to the ones of macaque when compared with a Frequency-normalized Spread Vector (FSV) metric [110]. It is engaging since they use a somewhat similar model to ours. The most significant difference is their STDP rule using only the first 10% spikes of

the most active cells for the potentiation window (rank-based) instead of our precise spike-timing rule.

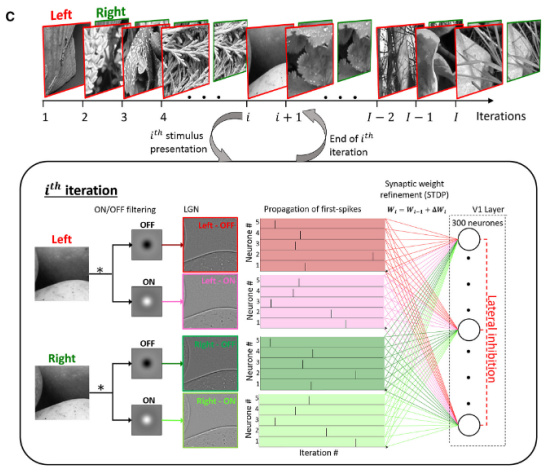


Figure 3.9: Learning binocular disparity selective representation by Chauhan et al. [109]

With this model, Debat et al. [111] designed a three-layer SNN combined with homeostatic mechanisms for estimating trajectories and outperformed human capabilities. Though they show how their learned representation efficiently captures the statistics of natural images, they use frame-based image datasets for that purpose, converting the grey scale values to spikes using a difference of Gaussian filters. A similar work applied to event-based data would be of great interest.

We are not aware of any other works able to learn binocular disparity selective receptive fields from event-based data via STDP. Most rely on hard-wired neural systems for solving the epipolar geometry constraint. We propose one of the first models for learning disparity from the natural statistics of the scene.

3.2 A dual-layered spiking neural network model

The objective of the efficient coding model is to process and encode the input stream as efficiently as possible. An efficient code is generally thought to compress the input without losing too much helpful information. Event streams are considered already not very redundant spatially but can contain much redundant temporal information because of their high temporal resolution. However, an efficient model can sparsify the data both spatially and temporally.

We based our coding model on a dual-layer network inspired by the early human visual system. More precisely, we find two types of cells in the brain that are considered to be the basic building bloc for visual information encoding, the simple and complex cells. They receive information directly from the retina, which in our case, can be approximated by the event-based camera. We tried to mimic some of the behaviors of those cells in our model.

3.2.1 Neuronal model

We use the LIF neuron model as the basic building block of our model. We detailed that model in Chapter 2. We reformulate here the membrane potential equation while using exponential decay. An input event from a pixel i creates an excitatory post-synaptic potential at time t, which increases the membrane potential according to the strength of the corresponding synapse $w_i(t)$. Let Δt be the time between the current input and the previous input to the neuron. Then the new membrane potential can be calculated directly upon arrival of the input as:

$$\tilde{V}(t + \Delta t) = V(t)\, e^{\frac{-\Delta t}{\tau_m}} + w_i(t) \tag{3.1}$$

where τ_m controls the exponential decay speed of the membrane potential.

If the membrane potential exceeds a threshold V_θ, the neuron is said to "spike". It generates an action potential, propagating to other neurons via synapses. Its membrane potential is then reset to $V_{rest} = 0$:

$$V(t + \Delta t) = \begin{cases} \tilde{V}(t + \Delta t) : \tilde{V}(t + \Delta t) < V_\theta \\ V_{\text{rest}} : \tilde{V}(t + \Delta t) \geq V_\theta \ . \end{cases} \tag{3.2}$$

3.2.2 Homeostatic mechanisms

The LIF neuron presented before is one of the simplest models. However, biological neurons exhibit much more complex mechanisms to adapt their behavior to varying situations. We replicate some of the most beneficial mechanisms to improve our spiking neural network's stability, adaptability, and robustness.

3.2.2.1 Refractory period

When a neuron spikes, it enters a period of low excitability called a refractory period. It strongly limits the spike frequency of neurons in the case of frequent and significant inputs. We model a refractory mechanism through a trace η_{RP} generated after each spike and then decaying exponentially back to zero at a rate defined by τ_{RP}. The membrane potential update becomes:

$$\tilde{V}(t + \Delta t) = V(t)\, e^{\frac{-\Delta t}{\tau_m}} - \eta_{\text{RP}}\, e^{-\frac{t+\Delta t-t_s}{\tau_{\text{RP}}}}, \tag{3.3}$$

where t_s is the time of the neuron's last spike.

3.2.2.2 Threshold and spike rate adaptation

Event-based cameras present an inherent variability in the output frequency depending on factors such as lighting conditions, the number of textures, or the relative

speed of objects. It means that neurons will be subjected to variable data rates during their operation, leading to high variability in their spike rates. Although this is somewhat unavoidable, keeping the spike rate of neurons in a reasonable range across different conditions is preferable. Biology handles that problem using a wide range of homeostasis mechanisms, be it at the heart of the neuron, the soma, or at the sites of its inputs, the synapses.

Homeostatic regulation of firing rates is a common feature of neural circuits allowing stable activity levels [112]. To avoid the necessity to fine-tune the neuron's firing thresholds, we use a homeostatic regulation to enforce a specific target firing rate $S^* = 0.75$ spikes s^{-1}. The threshold is adapted automatically every second from the difference between an estimate of the current spike rate $S(t)$ and the desired S^* as:

$$\Delta V_\theta = \eta_{\mathrm{TA}} \left(S(t) - S^*\right) , \tag{3.4}$$

with η_{TA} controlling the speed at which the threshold adapts. $S(t)$ is computed by counting the number of spikes in the previous 10 seconds. The threshold update happens every second and is a somewhat slow process intended to handle global illumination and speed conditions. We define a minimum threshold $V_{\theta\,\mathrm{min}}$ to avoid capturing camera noise in areas with very few inputs. This slow regulation process acts in the order of multiple seconds.

For local variations, we use a faster process called spike rate adaptation. Just like threshold adaptation, it relies on the neuron's activity. When a neuron spikes, a trace $V_{SRA}(t)$ is increased by a value η_{SRA}. This trace is subtracted for each pre-synaptic input and decays exponentially back to zero according to the parameter τ_{SRA}. The membrane potential internal update becomes:

$$\tilde{V}(t + \Delta t) = V(t)\, e^{\frac{-\Delta t}{\tau_m}} - V_{SRA}(t)\, e^{\frac{-\Delta t}{\tau_{SRA}}} - \eta_{\mathrm{RP}}\, e^{\frac{t_s - t - \Delta t}{\tau_{\mathrm{RP}}}} \tag{3.5}$$

This regulation mechanism acts immediately after a spike has happened. By choosing a relatively short time constant τ_{SRA} (in the order of a few hundred milliseconds), the effects of the spike rate adaptation mechanism are only visible on short periods, which complements the slower threshold update.

3.2.2.3 Static lateral inhibition

Inhibition is a crucial tool for biological systems. Inhibitory connections have been proven to serve multiple purposes: activity regulation in recurrent populations, suppression of entire brain areas, or prediction mechanisms.

We use a simple lateral inhibition mechanism to facilitate the learning of diverse receptive fields tuned to different orientations, phases, and movement directions and

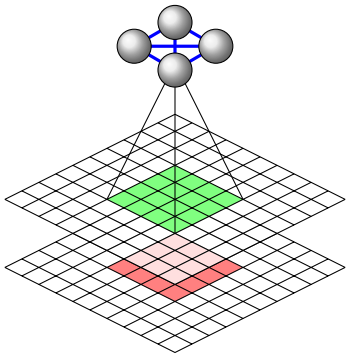


Figure 3.10: Groups of $N = 4$ neurons are connected to the same patch of input pixels providing ON (green) and OFF (red) events. The neurons are linked by inhibitory connections (blue). A neuron can be connected to pixels of its receptive field by D synapses with different delays to gain localized motion-sensing properties (not shown).

speeds. In the network, N neurons are connected to the same input pixels as shown in Fig. 3.10. Inhibitory connections link such neurons. We simplified the mechanism of inhibition compared to biology. Rather than assigning specific neurons to the role of inhibitory cells, we considered that some neurons could directly inhibit each other reciprocally. Specifically, when a neuron spikes, it instantly inhibits its neighbors by subtracting a fixed value η_{INH} from its membrane potential. This mechanism introduces intense competition between neighboring neurons. It prevents the ones that receive input from the same region of the sensor from firing at roughly the same time, implying similar synaptic weight updates and leading to similar receptive fields.

3.2.3 Learning through Spike Timing Dependent Plasticity

We presented the STDP learning rule in Chapter 2. In our implementation of STDP, as soon as a neuron spikes, its synaptic input connections will change depending on the timing of the last input they received. It is a symmetric interpretation, where we keep track of the timing of every pre-synaptic spike t_i and the two last post-synaptic spikes t_s and t_{s-1}. Each synapse undergoes an instantaneous change in weight depending on an exponential relationship between the time difference between the timestamp t_i of the last input arriving at synapse i and the time of the post-synaptic spike $t_s > t_i$:

$$\Delta w_i^{LTP} = \eta_{LTP}\, e^{\frac{t_i - t_s}{\tau_{LTP}}} \tag{3.6}$$

with η_{LTP} and τ_{LTP} controlling, respectively, the height and duration of the potentiation window. Any input spike arriving after a postsynaptic spike ($t_i > t_s$) leads to depression of the synaptic weight:

$$\Delta w_i^{LTD} = -\eta_{LTD}\, e^{\frac{t_{s-1} - t_i}{\tau_{LTD}}} \tag{3.7}$$

where η_{LTD} and τ_{LTD} control, respectively, the height and duration of the depression window. In this formulation, multiple pre-synaptic spikes can interact with the last post-synaptic spike to induce depression. Synapses whose weight would become negative due to LTD are set to zero. The STDP rule applies equally to all synapses with different time delays. The relevant time difference $|t_s - t_i|$ is always the one between the *arrival* of the pre-synaptic spike (occasionally delayed) and the moment of the post-synaptic spike.

3.2.3.1 Weight normalization

To avoid unbounded growth of synaptic weights, we use a simple weight normalization mechanism, which normalizes the weights projecting to a single neuron in a multiplicative fashion. After every spike of the neuron, all input synapses are multiplicatively rescaled such that the L_2 norms of the weights equal a parameter λ, chosen empirically. Such multiplicative rescaling of the efficacies of groups of synapses could result from local competition for synaptic building blocks such as neurotransmitter receptors [113]. Without this mechanism, the weights grow indefinitely, causing surges in neuron spike rates and unrealistic behaviors such as neurons being activated by only one spike.

3.2.3.2 Parallel synapses with different delays

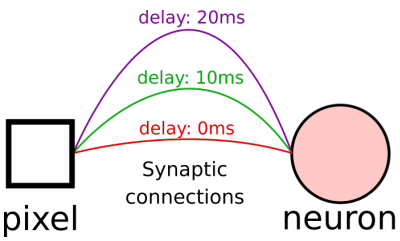


Figure 3.11: Diagram of the synaptic delays between a pixel to a simple cell

We have described all the mechanisms that define our artificial neurons. Changing the values of the thresholds, time constants, STDP functions, or homeostasis gains affects (sometimes drastically) the network dynamics and learning results. Network connectivity plays an equally important role. We refer to the *receptive field* of a neuron as the set of its pre-synaptic connections. The size and "depth" of this receptive field can significantly impact a cell's behavior. We define the set of input synapses of a neuron as a weight matrix W that will change as the network operates and synaptic weights are updated by the plasticity mechanisms.

To allow for the learning of motion-sensitive receptive fields, each pixel of the event-based camera can be coupled to a LIF neuron via D synapses with different delays, as depicted in Fig. 3.11. Therefore, a neuron can receive, e.g., the same on-event from the same pixel at D different times. Similarly, it can simultaneously receive events having occurred at different pixels at different times and therefore be sensitive to image motion. Signals with multiple delays from the same sensor

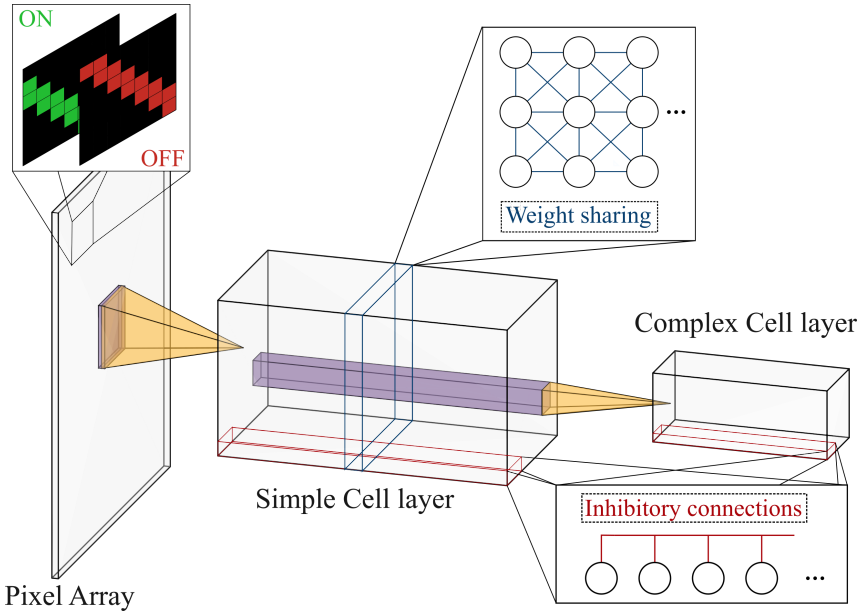


Figure 3.12: Proposed SNN architecture

are common in biological motion vision circuits. It is the classic idea behind the Reichardt detector, also known as Hassenstein-Reichardt detector model [114].

3.2.4 Spiking neural network architecture

3.2.4.1 Event-based pixel array

The event-based camera pixel array serves as the input layer of the SNN. Event-based cameras record the polarity of events, indicating the sign of the change in light intensity. We, therefore, separate the events in 2 maps depending on their polarity (ON and OFF) as depicted in Fig. 3.12.

3.2.4.2 Simple cell layer

Simple cells are neurons' local orientation filters found in the human early visual pathway. They typically form receptive fields in the shape of Gabor functions tuned to a specific orientation and possibly motion direction (see Sec. 2.3.3).

Simple cells have the important task of transforming event streams into more abstract information, such as orientations, motions, or depth while sparsifying the input as much as possible. They are based on the Leaky Integrate and Fire (LIF) neuron model described in Chapter 2.

The second layer comprises a set of simple cells connected to a specific region of the event-based camera via weighted synapses. The latter defines the neuron's

Parameters	*Unit*	**Simple cells**	**Complex cells**
V_{thresh}	mV	30	3
V_{reset}	mV	-20	-20
$V_{\text{thresh}}(min)$	mV	4	
η_{LTP}	mV	0.00077	0.2
η_{LTD}	mV	0.00021	0.2
η_{INH}	mV	25	25
η_{TA}	mV	1	
η_{RP}	mV	1	1
η_{SRA}	mV	0.6	
τ_{m}	ms	18	20
τ_{LTP}	ms	7	20
τ_{LTD}	ms	14	20
τ_{RP}	ms	20	30
τ_{SRA}	ms	100	
Synaptic Delay	ms	0	
S^*	$sp.s^{-1}$	0.75	
λ		4	10

Table 3.1: Simple and complex cell parameters.

receptive field. If one of the camera's pixels records an event, the neurons connected to it will receive an excitatory input depending on the synapse's strength. To differentiate ON and OFF events, neurons are connected with at least two synapses to a pixel, one for each event polarity. Furthermore, we allow pixels to connect to the spiking neurons with different synaptic delays (see section 3.2.3.2) to enable the development of motion tuning. Table 3.1 presents the parameters used during training. We chose parameters that are most plausible in biology.

We used a weight-sharing mechanism between simple cells to improve the diversity of learned receptive fields. Neurons looking at different visual field locations jointly learn the same set of synaptic weights. We define these neurons as belonging to the same "neuronal map". It is represented by the blue cross section in the simple cell layer in Fig. 3.12.

We increase the number of neuronal maps to obtain simple cells tuned to different orientations. We also connect neurons looking at similar locations of the visual field, but from different neuronal maps, with inhibitory connections. It is represented by the red inhibitory connections in Fig. 3.12.

3.2.4.3 Complex cell layer

Complex cells pool information from multiple simple cells and react to a more significant part of the visual field. They learn high-level representation from combining simple cells' receptive fields.

Our complex cell layer receives inputs from simple cells. They are also based on the LIF model, but with adapted parameters. They learn to represent oriented edges independently of the precise location of the edge, which requires a strongly non-linear behavior [115] [116]. Often, complex cell-like behavior is achieved via a max-pooling operation, but here we are interested in learning the non-linear behavior of complex cells with STDP. We adjust complex cell parameters to be more sensitive to inputs by giving them a lower spiking threshold (cf. Tab. 3.1). In our case, we removed the weight-sharing mechanism that would have a limited purpose since it would reduce the basis of complex cells receptive field learned while it is already small by design.

We also chose a different and simpler STDP window for the complex cells. Contrary to simple cells, the weight variation is always positive, following a step function. Unbounded growth is avoided using weight normalization. The STDP rule for complex cells is written as:

$$\begin{aligned} \Delta w_i^{\mathrm{LTP},c} &= \begin{cases} \eta_{LTP} : |t_i - t_s| \leq \tau_{\mathrm{LTP}} \\ 0 : |t_i - t_s| > \tau_{\mathrm{LTP}} \ . \end{cases} \\ \Delta w_i^{\mathrm{LTD},c} &= \begin{cases} \eta_{LTD} : |t_{s-1} - t_i| \leq \tau_{LTD} \\ 0 : |t_{s-1} - t_i| > \tau_{LTD} \ . \end{cases} \end{aligned} \tag{3.8}$$

We have also experimented with the following additional STDP windows for the complex cell learning rules:
Step left window:

$$\Delta w_i^{LTD} = \begin{cases} \eta_{LTD} : |t_{s-1} - t_i| \leq \tau_{LTD} \\ 0 : |t_{s-1} - t_i| > \tau_{LTD} \ . \end{cases} \tag{3.9}$$

Step right window:

$$\Delta w_i^{LTP} = \begin{cases} \eta_{LTP} : |t_i - t_s| \leq \tau_{LTP} \\ 0 : |t_i - t_s| > \tau_{LTP} \ . \end{cases} \tag{3.10}$$

Step symmetrical window:

$$\begin{aligned} \Delta w_i^{LTP} &= \begin{cases} \eta_{LTP} : |t_i - t_s| \leq \tau_{LTP} \\ 0 : |t_i - t_s| > \tau_{LTP} \ . \end{cases} \\ \Delta w_i^{LTD} &= \begin{cases} \eta_{LTD} : |t_{s-1} - t_i| \leq \tau_{LTD} \\ 0 : |t_{s-1} - t_i| > \tau_{LTD} \ . \end{cases} \end{aligned} \tag{3.11}$$

Linear window:

$$\begin{aligned}\Delta w_i^{LTP} &= \begin{cases}\eta_{LTP}(1-(t_i-t_s)) : |t_i-t_s| \leq \tau_{LTP} \\ 0 : |t_i-t_s| > \tau_{LTP}\ .\end{cases} \\ \Delta w_i^{LTD} &= \begin{cases}\eta_{LTD}(1-(t_{s-1}-t_i)) : |t_{s-1}-t_i| \leq \tau_{LTD} \\ 0 : |t_{s-1}-t_i| > \tau_{LTD}\ .\end{cases}\end{aligned} \tag{3.12}$$

Exponential window (similar to the rule for simple cells):

$$\begin{aligned}\Delta w_i^{LTP} &= \eta_{LTP}\, e^{\frac{t_i-t_s}{\tau_{LTP}}} \\ \Delta w_i^{LTD} &= -\eta_{LTD}\, e^{\frac{t_{s-1}-t_i}{\tau_{LTD}}}\ .\end{aligned} \tag{3.13}$$

3.2.4.4 Neuvisys, a SNN implementation

We decided early in this project to develop our implementation of a SNN and later spiking reinforcement learner. The motivations behind that were as follow:

- At the time, SNN libraries already existed but were not necessarily complete enough to provide all the neurons interactions we were interested in, such as synaptic delays, stereoscopic receptive fields or homeostatic mechanisms.
- The interface between the SNN and event-based cameras was not consistently implemented or would require additional efforts.
- We wanted to have total control of the software implementation to understand precisely the interactions between the neurons and the event-based camera.
- Neurons models and learning rules can be limited and hard to extend.
- The use of reinforcement learning was usually not considered, making the implementation of a spiking actor-critic framework difficult.

For that reason, we decided to implement our library. It is written in C++ and uses a Qt graphical interface to interact in real-time with the network. Everything runs on CPU. We present our code and how to run it in Appendix C.

Our implementation is fully asynchronous, contrary to some SNN libraries. Every spike is propagated at the exact time it was created, except for the events from the event-based camera. The latter are transmitted in packets (up to 10 000 events per packet) due to the bandwidth restriction in the USB connection.

The propagation of spikes is based on a depth-first algorithm, meaning that if a spike activates a neuron, we then process that neuron in priority and the subsequent neurons connected to it in the case where it would activate other deeper neurons.

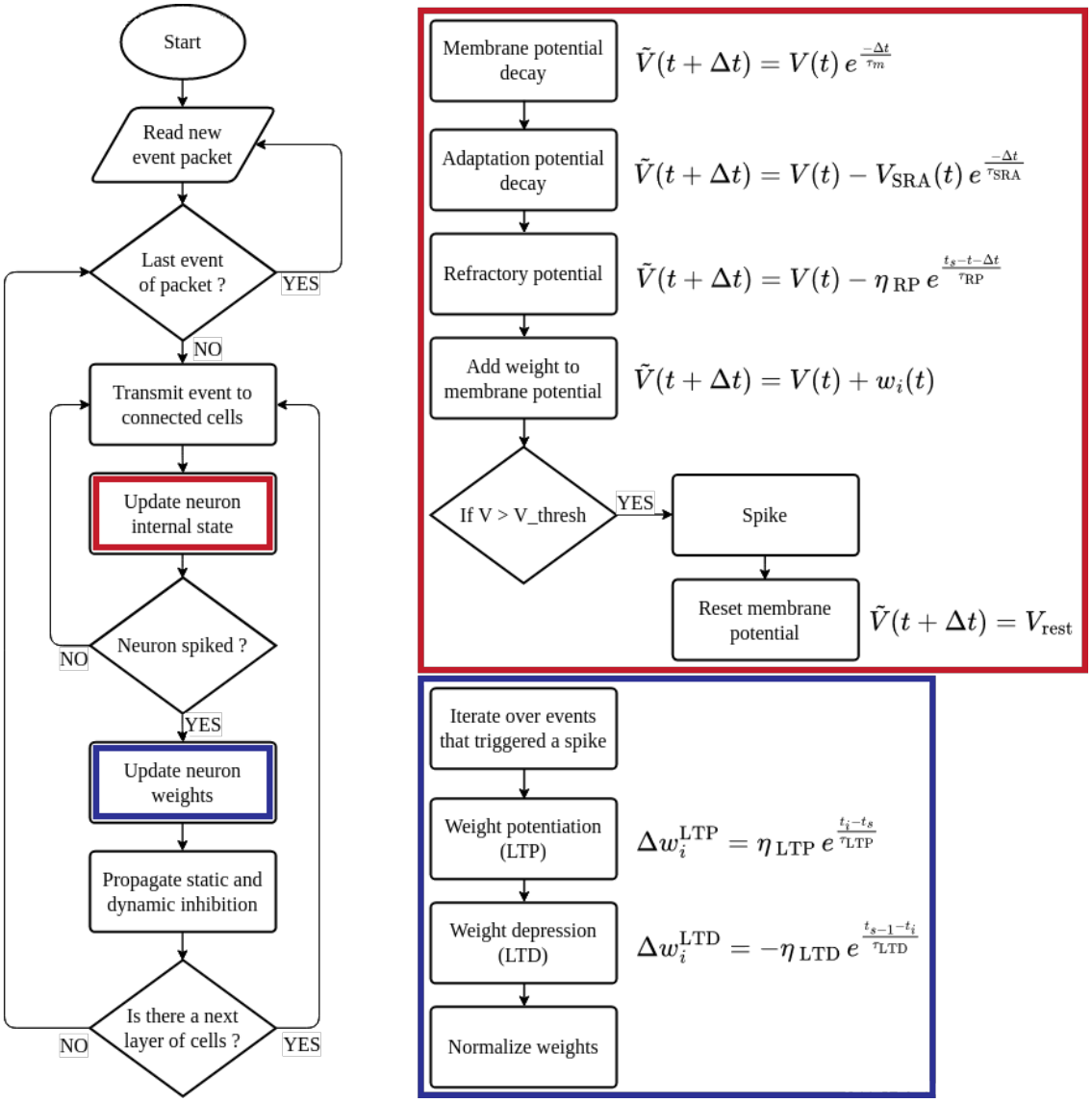


Figure 3.13: Flowchart of our SNN algorithm.

Figure 3.13 resumes the SNN algorithm along with some of the main equations.

Since our network runs on CPU, we cannot process multiple neurons simultaneously. Therefore we compute each event synchronously, which heavily limits our implementation performances. While small network architecture with a limited event rate runs near real-time, most of our experiments ran slower than real-time. The most complex network architecture can quickly get 100 times slower than real-time. It is one of the main limitations of our implementation. We considered a GPU implementation using the CUDA programming language, but the effort and time required were prohibitive. Neuromorphic hardware could be the best solution while not necessitating much change in our implementation.

Figure 3.14: Screen capture of four event recordings, shapes on a paper, an office, someone juggling, and a robotic platform in an urban environment. Blue and red, respectively, represent ON and OFF events.

3.3 Network activity analysis and visualization

We study in this section the simple and complex cell activity variation, spike patterns, and correlations in response to various types of event inputs. We are interested in neuron behaviors and differences before and after training.

3.3.1 Datasets of event-based recordings

For our experiments, we worked with multiple recordings made with an event-based camera.

3.3.1.1 Multi-purpose event-based recordings

For our experiments, we selected four event recordings in different environments. Shapes are drawn on a sheet of paper, a recording of an office, someone juggling with balls, and a mobile robotic platform in a small, recreated urban road environment. Each recording is a few seconds up to a few minutes long. We present a frame representation of the four of them in Fig. 3.14.

For those recordings, we used the default Davis346 [117], with the stock lens and parameters of the camera. We chose these four scenes to diversify the types

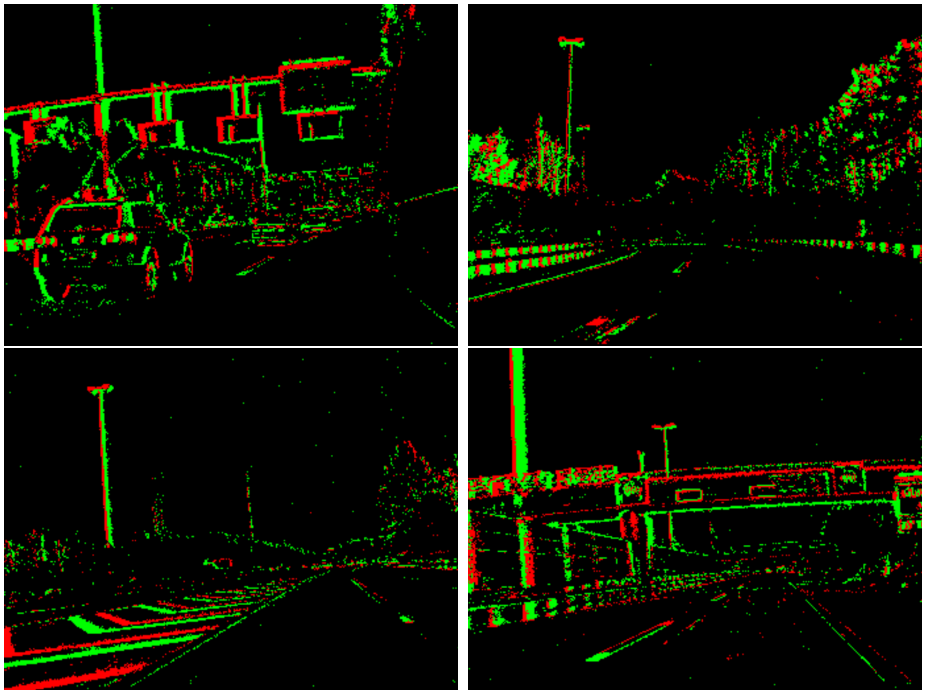


Figure 3.15: Frame representation of the input events from four of the driving sequences in the DDD17 driving dataset.

of visual inputs. The shape recording presents clearly defined edges with multiple orientations and motion directions. It is well adapted to test the learning of oriented filters. The recording is around 1 minute long. The office recording offers a more typical man-made environment with many straight edges, most of them vertical or horizontal. Similarly, the outside recording presents buildings with straight edges and some more natural objects, such as trees and bushes. They are both suited for learning in a typical urban environment. They are respectively 40 seconds and 7 minutes long. Finally, the juggling recording is the only one with a fixed camera setting, where the movements only come from the juggler and the balls. It is 2 minutes long.

3.3.1.2 DDD17 driving dataset

The DDD17: DAVIS Driving Dataset [118] features many hours of driving recorded on freeways, highways, and in cities with a DAVIS346B [117] event-based camera. The data set features various visual inputs such as cars, traffic signs, poles, trees, buildings, safety barriers, and road markings. Figure 3.27 shows four examples of short-time slices of events. It is a good dataset for motion estimation, as the car's movement will create very different optical flow in the scene depending on the position in the visual field. The speed of the car also varies greatly depending on the road situation.

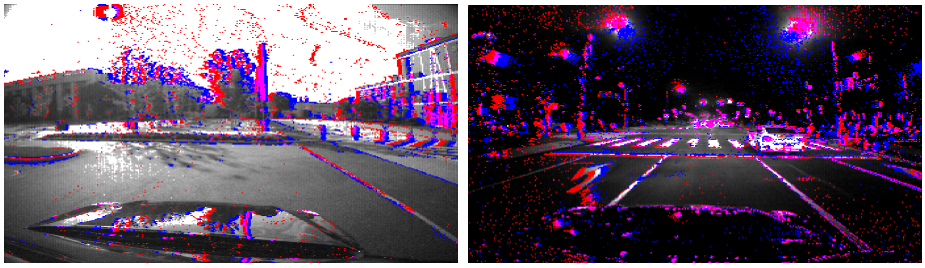


Figure 3.16: Event representation displayed on top of actual frames from two of the MVSEC's dataset driving sequences, one during the day and one during the night.

3.3.1.3 MVSEC driving dataset

The Multi Vehicle Stereo Event Camera dataset [43] is a collection of stereoscopic event-based recordings mounted on three different platforms, namely a car, a motorcycle, and a drone. We concentrated on the driving sequences somewhat similar to the DDD17 dataset. However, some recordings were also made at night, and the stereoscopic setup allows for different applications such as depth perception. Figure 3.16 presents two examples of recordings, one during the day and one during the night. All the event-based camera parameters and setup details are given in the dataset. It also includes frame-based videos since the setup includes frame-based cameras. It is advantageous for generating a ground truth for comparing event-based methods with traditional computer vision.

3.3.1.4 Robotic mobile platform in a urban environment

This dataset regroups multiple recordings made on a robotic mobile plaform mounted with a stereoscopic pair of event-based cameras. It can be used to test algorithms on stereo-driving scenarios at slow speeds in man-made environments resembling road systems.

We mounted two DAVIS346 event-based cameras on a stereoscopic rail. We used the stock lens and parameters on the cameras. The distance between the camera optical center is 50 cm. The recordings were made simultaneously with a synchronization cable to synchronize the event timestamps between the two cameras. The robot itself is a small 4-wheeled vehicle approximately 1 meter tall. Fig. 3.17 present the robotic vehicle we used with the cameras mounted on top. We used two CCTV cameras to limit the number of solar flares affecting event-based sensors. Despite that, there is still some present in the dataset.

3.3.2 Simulated sequences

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size rfs
		(x, y)				(x, y, z)	
Simple cells	static	none	1	1	yes	16,16,144	10,10,2
Complex cells	static	none	1	1	no	4,4,16	4,4,144

Table 3.2: Network architectural parameters

In addition to actual recordings from event-based sensors, we also created some simulated event sequences. The artificial videos are made using stereoscopic frames and are then converted to event-based streams using the ESIM simulator [119]. The simulation is accurate when given a high enough frame rate. However, the generated sequences do not present sensor noise. It allows us to generate specific stimuli that correspond to the exact need of an experiment. We primarily generated gratings and moving bars using that technique. We describe them in more detail in the experiment themselves.

Figure 3.17: Robotic mobile driving platform with two stereoscopic event-based cameras mounted on top.

3.3.3 Network parameters

Table 3.2 presents the network architecture used for the following experiments. It summarizes the main architectural parameters. We detail the number of synapses per neuron, the number of independent visual regions, the use of the weight-sharing mechanism, as well as the number of neurons per region and the size of their receptive fields. Finally, we give the type of inhibition used. We will discuss new inhibition mechanisms in the next Chapter, including the lateral inhibition range. For now, only static inhibition is used, described in section 3.2.2.3.

For simple cells, the receptive fields sizes are in pixels, with a depth of 2 for both ON and OFF polarity. For complex cells, receptive fields sizes represent the number of simple cells.

We created a network with $16 \times 16 \times 144 = 36864$ simple cells looking at a visual field of 160×160 pixels. The following layer comprises $4 \times 4 \times 16 = 256$ complex cells centered on the same visual field. We used the cell parameters detailed in Table 3.1.

3.3.4 Visualizing the network's behavior

This section analyzes how our SNN works and reacts to event-based inputs. As mentioned previously, SNN are computer models of biological brains. Over the years, neuroscientists have developed practical tools and visualization methods to describe how neuronal architectures react to various stimuli. We will use similar tools applied to our computer model.

We first learned the simple and complex cell receptive fields with the office event-based recording as input. We will describe in more detail the learned representation in the next Chapter.

Membrane potential variation Analyzing a cell is often done by looking at its membrane potential. The variation of potential indicates the cell's reaction to various inputs. Access to the membrane potential variation over time gives information on most of the dynamics regarding that cell.

However, recording the membrane potential of cells is an expensive process. Neurons can receive many events in short periods. It is possible to record the membrane potential only during the reception of a new event while extrapolating the decay. Nevertheless, even then, the very low latency of event-based cameras can create millions of events per second with microsecond timestamp accuracy. Recording all those changes requires expensive computation time and a lot of memory space. We cannot afford to do that when training networks, sometimes for many minutes or hours.

A simple solution to that problem is to record only when a neuron spikes. Contrary to events, neurons spike much less often. The storage and computational requirements for recording the neuron's activation time are much lower. Since spikes are the primary information medium in a SNN, looking at individual neurons' membrane potential is not necessarily helpful when trying to understand more global network dynamics.

Event and Raster plot An excellent way to represent spiking information is to use an event plot. This plot displays spike times in the form of a small bar for a specific neuron population. Looking at one neuron, we get information such as how active this cell is or if the spikes are close together or spread out in time. When looking at a group of neurons, we can see how correlated the spikes are from each other. This might indicate if the neuron has an equal representation or not. A raster plot is simply an event plot with added information, such as a histogram of cell activity variation on top and a mean cell firing rate on the right.

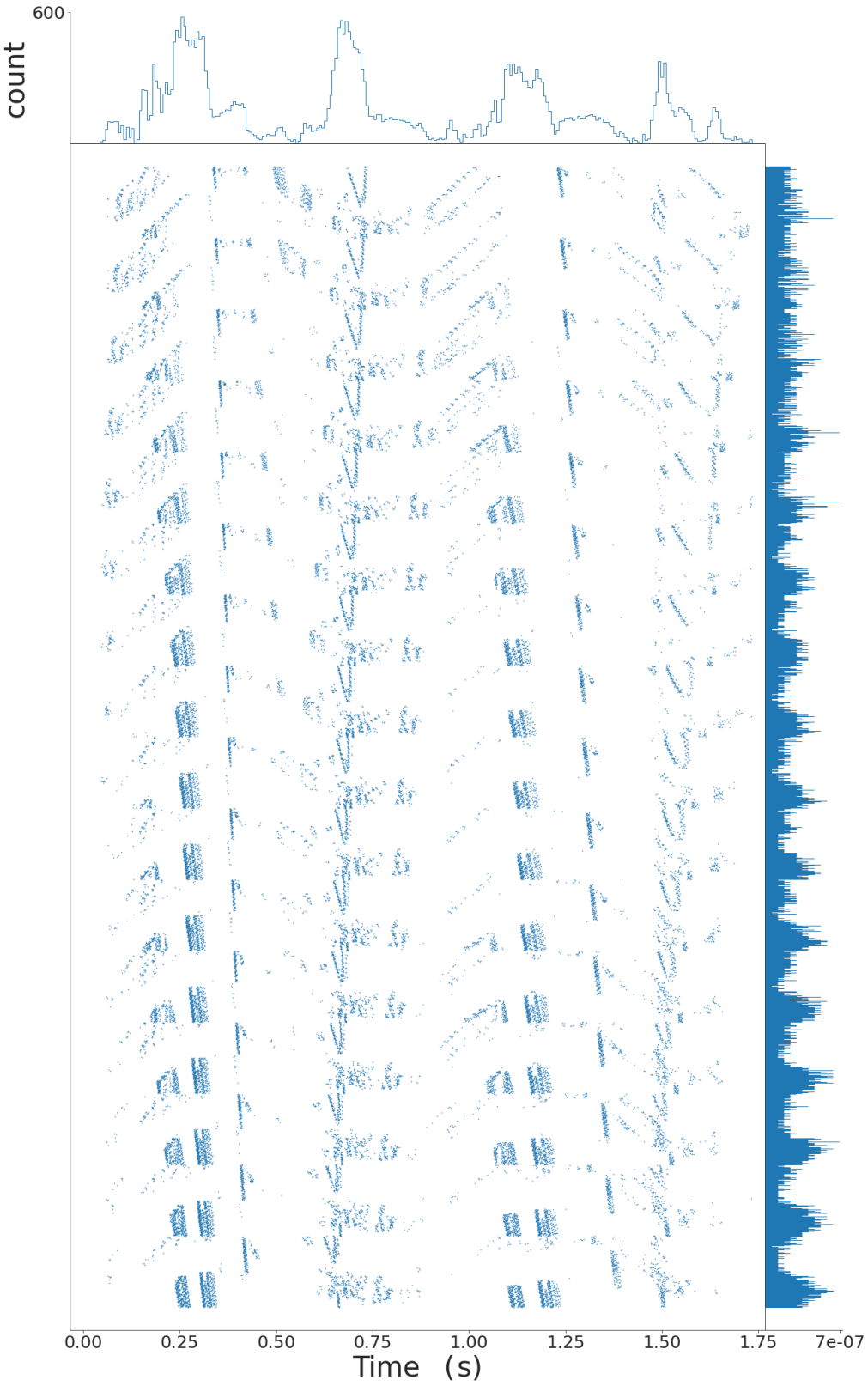


Figure 3.18: Raster plot of the simple cells on an event-based recording of an office.

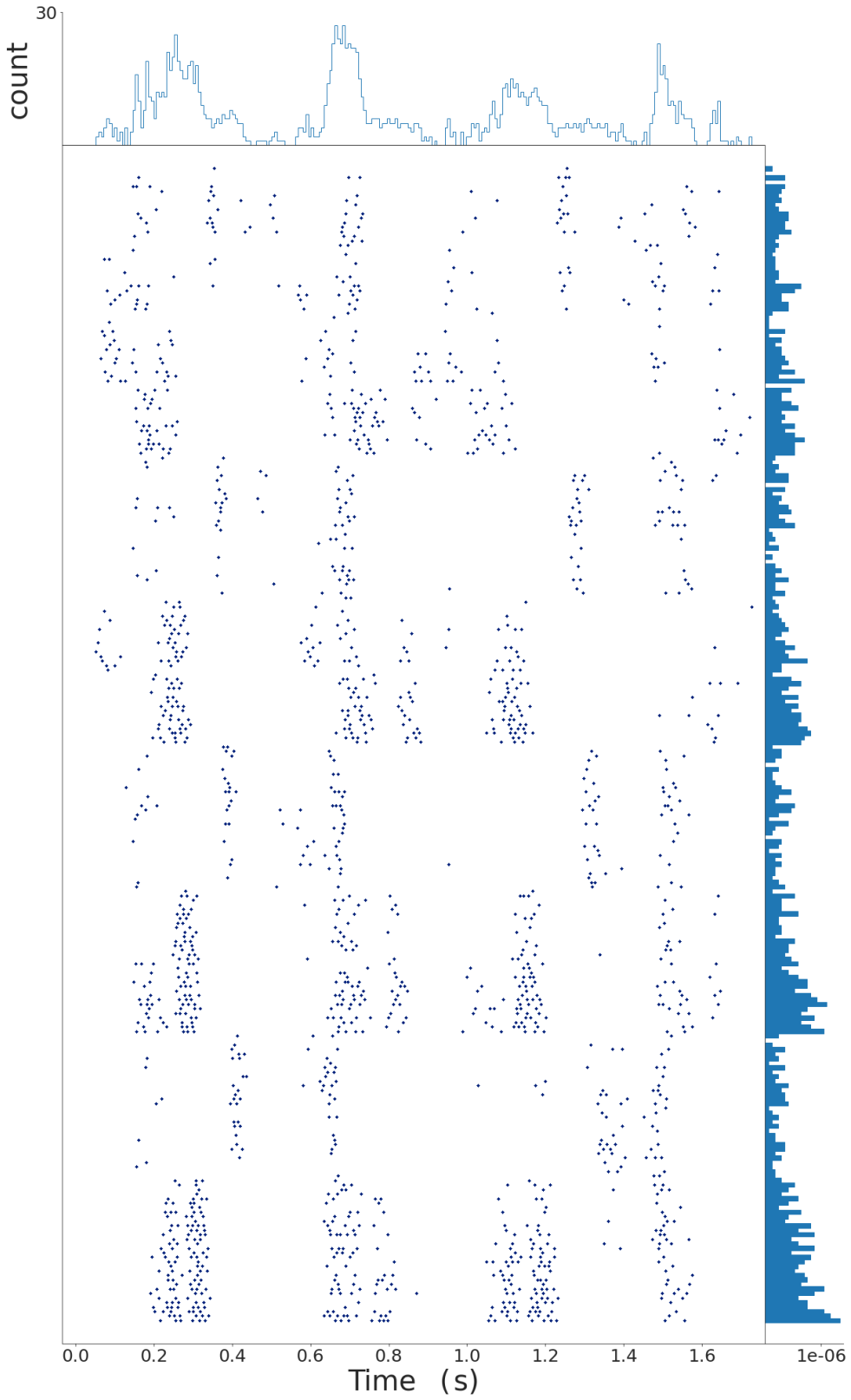


Figure 3.19: Raster plot of the complex cells on an event-based recording of an office.

Fig. 3.18 and Fig. 3.19 presents a raster plot of the network's simple and complex cells on the office recording presented in Sec. 3.3.1.1. As objects pass in front of the visual field, simple and complex cell spike in groups. Each line of the y-axis corresponds to a different cell with a unique firing pattern. This pattern depends mostly on two factors, the position in the visual field and the learned representation. The raster plot displays the cell in a flattened graph, even though they are arranged on a 2D plane, making it somewhat difficult to interpret precisely. Objects traversing the visual field generate similar firing patterns but with a delay.

We see that the cells in Fig. 3.18 display more precise firing patterns, while the complex cells in Fig. 3.19 pools information and have coarser firing patterns.

Instantaneous rate An instantaneous rate plot is very similar to an event plot, the main difference being that the spikes are first convoluted to a kernel before displaying it. Kernel convolution is defined as the integral of the product of a function f with a kernel g. It is a particular kind of integral transform where the kernel is reflected about the y-axis and shifted as follow:

$$(f * \kappa)(t) := \int_{-\infty}^{\infty} f(\tau)\kappa(t-\tau)d\tau \tag{3.14}$$

where $*$ represent the convolution operator, and τ is the axis of convolution.

Different kernel choices are possible. In this instance, we used a Gaussian kernel of the form $\kappa(t) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right)$. By doing so, we transform a discrete representation into a dense one. It is beneficial when applying traditional computer vision techniques that require dense matrices. The size of the kernel is an essential factor in the transformation process. Small kernels conserve more of the precise timing information of the spikes. In contrast, larger kernels make processing easier by reducing the representation size while losing more information. Fig. 3.20 and Fig. 3.21 present such a plot made with an exponential kernel of size 100ms. For the most part, we observe similar results to those in Fig. 3.18 and Fig. 3.19.

Time histogram A time histogram is similar to the instantaneous rates but applied to all the neuron spikes summed together. It gives a single output vector that gives the activity variation for the whole network layer. We lose local information at the neuron level, but this can be useful when analyzing the whole layer's behavior in response to some inputs. As with the instantaneous rate, we use a convolution window, which size determines the precision in time. Fig. 3.22 presents the network activity variation in the simple cell layer with a window size of 100ms. We can see that the activity varies quite a lot. We will see in Sec. 3.3.6 that this is primarily dependent on the input variations and that the two are heavily correlated. Simple and complex cells show similar variations in activity; however, the scales are different since the complex cells spike more on average.

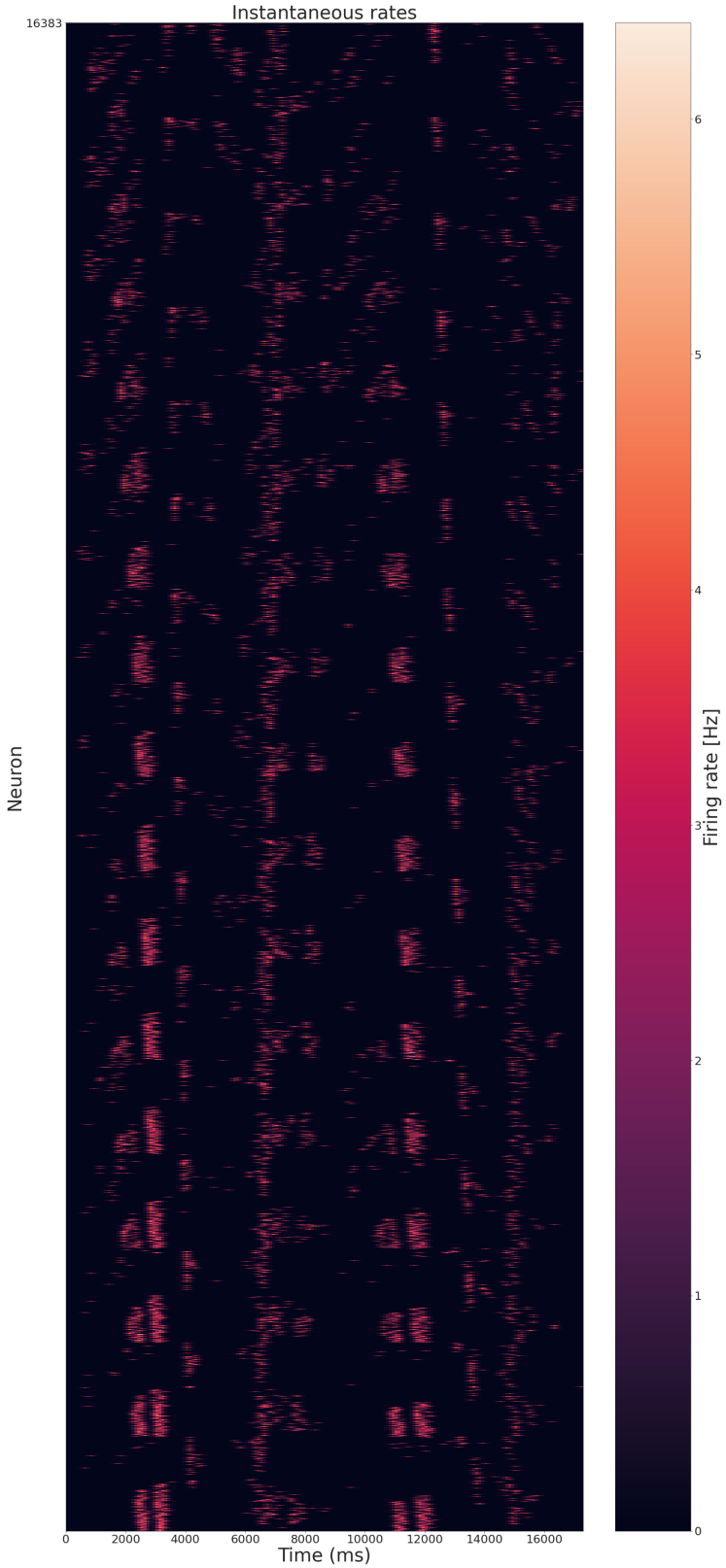


Figure 3.20: Instantaneous rates plot with a Gaussian kernel window of 100ms for the simple cell layer on an event-based recording of an office.

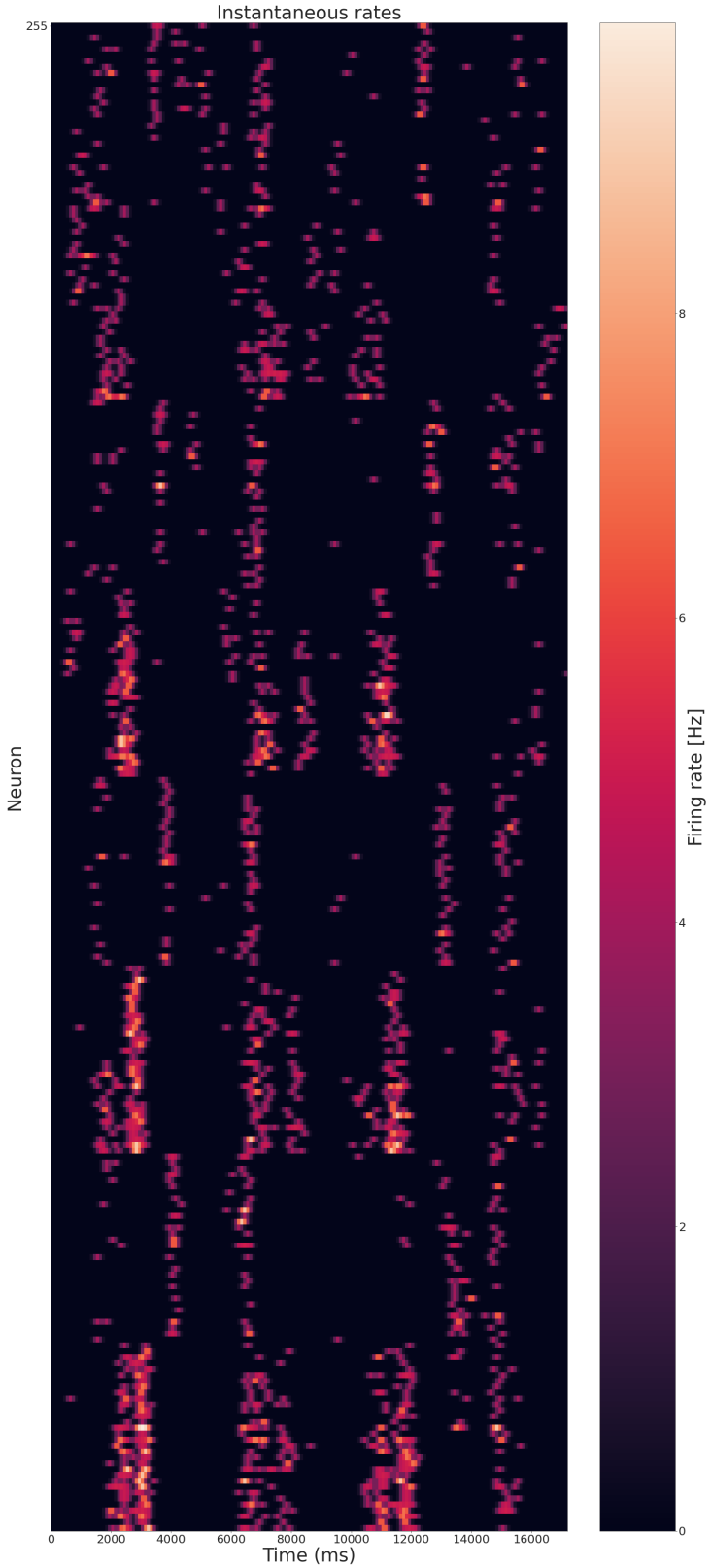
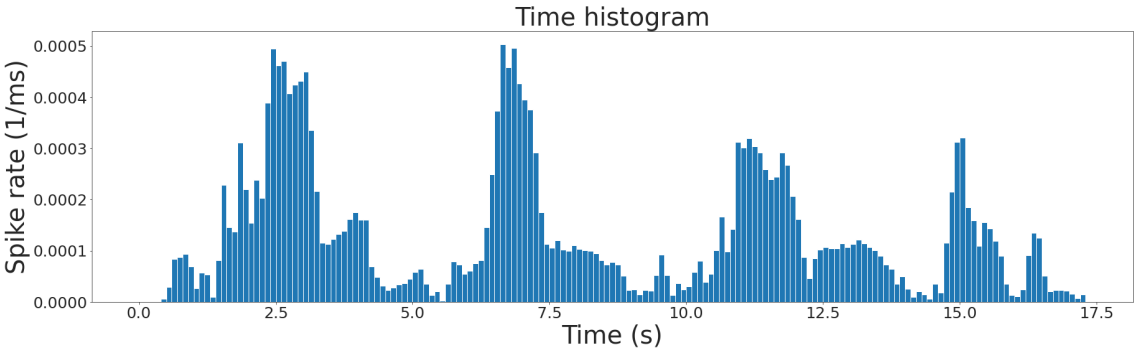
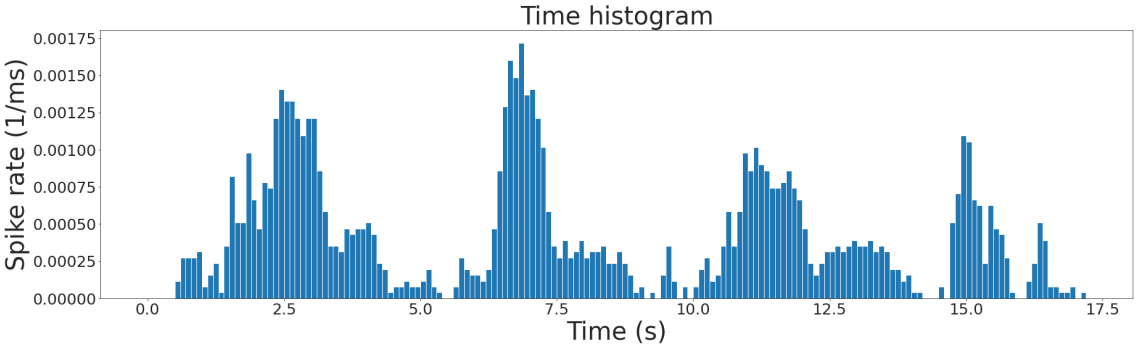


Figure 3.21: Instantaneous rates plot with a Gaussian kernel window of 100ms for the complex cell layer on an event-based recording of an office.

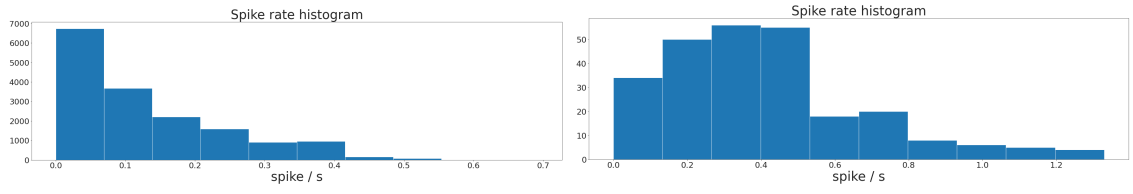


(a)



(b)

Figure 3.22: Time histogram on an event-based recording of an office. (a) Simple cell layer. (b) Complex cell layer. The scales are different for the two layers.



(a) (b)

Figure 3.23: Spike rate plots on an event-based recording of an office. (a) Simple cell layer. (b) Complex cell layer. The scales are different for each layer.

Scene	urban	office	shapes	juggling	mean
Number of events	11,8M	3,6M	6,5M	27,9M	
Random					
Simple cell activity reduction	257	189	125	137	177
Complex cell activity reduction	1359	825	1111	364	665
Simple cell correlation coefficient	0.74	0.87	0.97	0.90	0.87
Learned					
Simple cell activity reduction	173	112	67	112	116
Complex cell activity reduction	983	610	623	410	657
Simple cell correlation coefficient	0.84	0.90	0.98	0.92	0.91

Table 3.3: Sparsity analysis on four different event recordings. We show the result for two networks, one with learned representation and the other with random ones. We present the activity reduction, the number of events divided by the number of spikes in the cell layer. We also present the correlation coefficient between the events and cell activity variation.

Spike rate histogram The neurons' spike rates are an essential indicator of the network dynamics. The difference in spike rate between neurons can give some information about the learned representations. It depends on the input, but on average, we should expect neurons to stay in an acceptable range and follow some Gaussian distribution around a mean firing rate. Fig. 3.23 presents the simple cell firing rate when presented with the office recording. In that case, the cells firing rate is between 0 and 0.4 spikes s^{-1} for the simple cells and between 0 and 1.3 spikes s^{-1} for the complex cells. It is coherent with the previous time histogram. Complex cells spike more since we chose a lower threshold than simple cells, receiving inputs from many simple cells simultaneously.

3.3.5 Sparsity as an efficient coding mechanism

This section demonstrates that our SNN can sparsify event-based visual inputs. To do so, we fed the four event recordings described in Sec. 3.3.1.1 to 2 networks: (i) one that has not been trained and is initialized with uniformly sampled weights, (ii) the other previously trained on the drawn shapes recording.

Event streams are already sparse but still contain millions of events per second. The integrating process of simple cells reduces that number by an important factor. To demonstrate that concept, we recorded the number of spikes for the simple and complex cell layers and the difference between the normalized event activity and normalized simple cell activity. We present those results in Table. 3.3.

The activity reduction is the number of input events divided by the total number of spikes of the layer. It is a way of measuring the activity reduction induced by the

coding layers. Here, on average, the trained, efficient coding layer can reduce the input activity by a factor of 116 for the simple cell layer and 657 for the complex cell layer. This sparsity order is critical; it allows further layers to process the visual inputs much more efficiently. If spikes are the main power-intensive elements of a SNN, as suggested in [120], this would already represent a massive gain in total energy consumed.

3.3.6 Network activity variation

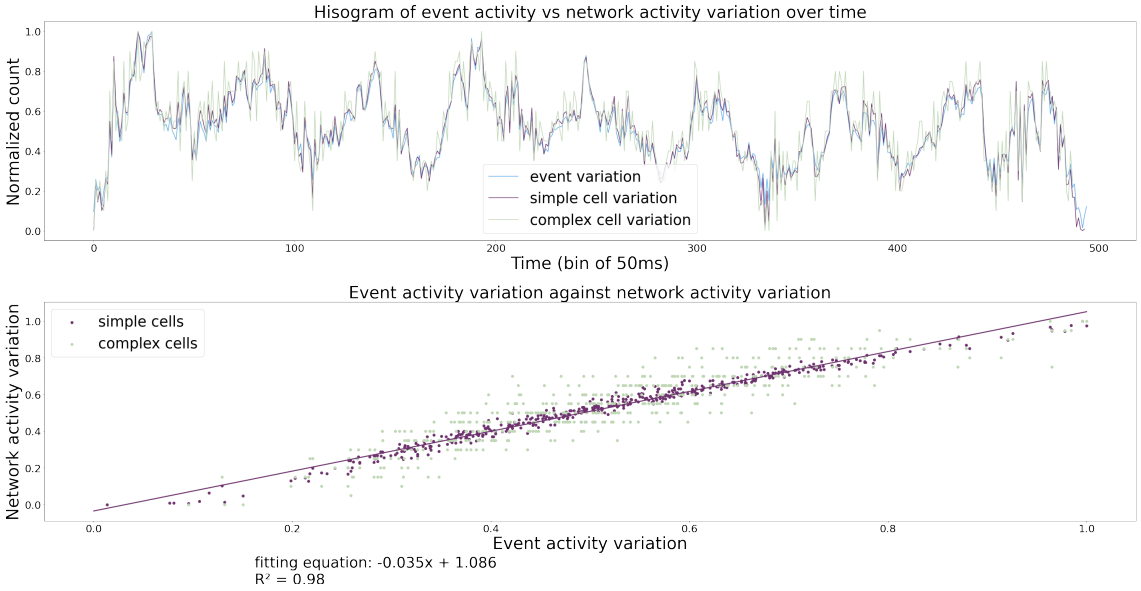
We observe the activity variation in the simple and complex cell layers function of the event-based inputs. Fig. 3.24 presents an example of the normalized activity variation between the input events in blue and the simple and complex layers in purple and green. It is for the shape recording described in Sec. 3.3.1.1 on the trained network. The weights have been learned on this specific recording, so the representation is especially well adapted to this visual input. It results in a high correlation between the event and network rates. The bottom figure shows a substantial correlation of 0.84 (MSE of 0.7) for that specific case. Oppositely, random weights give a lower correlation of 0.74 between the input and encoding signal (MSE of 3.37) for that recording. We present the same plots for the three other event-based recordings in the appendix as Fig. A.1, Fig. A.2, and Fig. A.3. Those figures demonstrate two points: the network activity variation for both layers is highly correlated to the input activity variation, even though the network activity is much more sparse. Moreover, after learning, that correlation tends to increase, pointing to the fact that learning a representation of the input is beneficial.

3.3.7 Independent spike responses

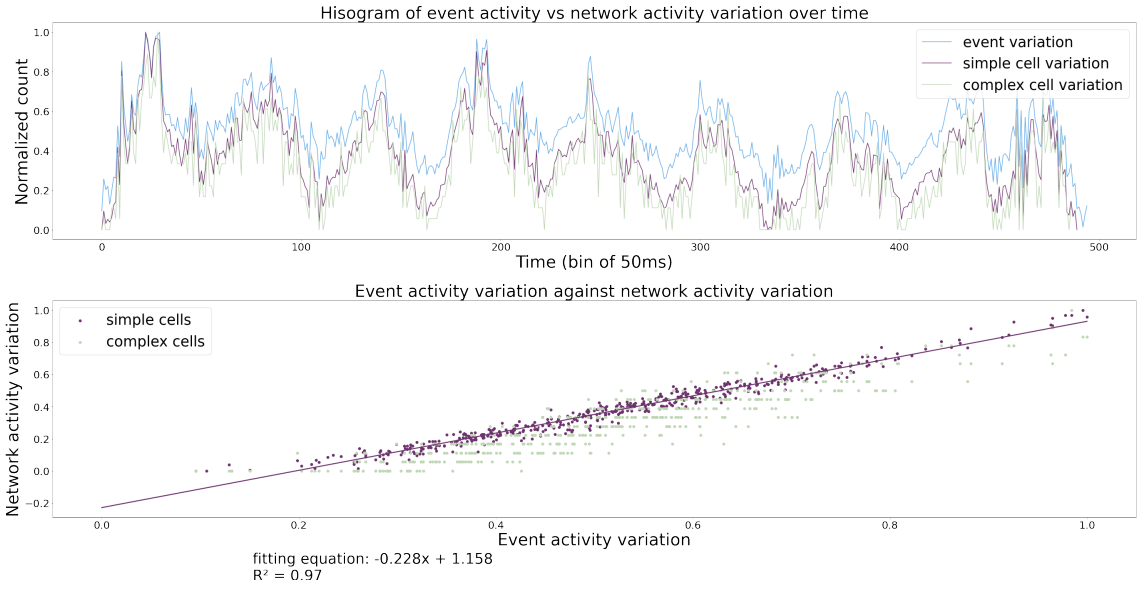
We demonstrated that the coding layer could significantly sparsify the input from the event-based camera. However, we must ensure that the coding layer retains most visual information during compression. We designed a simple experiment to prove that the network can efficiently distinguish between different visual features.

We used the same network described in Sec. 3.3.5, with a learned basis of different orientations. We want to submit that network to stimuli representing multiple orientations and see if the network can differentiate efficiently between them. For that, we used a simulated environment described in more detail in our reinforcement learning experiment Sec. 4.4.1. All we need to know for this experiment is that the environment is composed of a rotating camera looking at straight bars. As the camera rotates, the perceived orientation of the bars changes.

We presented the network with the oriented filters to this rotating stimulus and recorded the spike trains of the simple cells. We designed a simple supervised

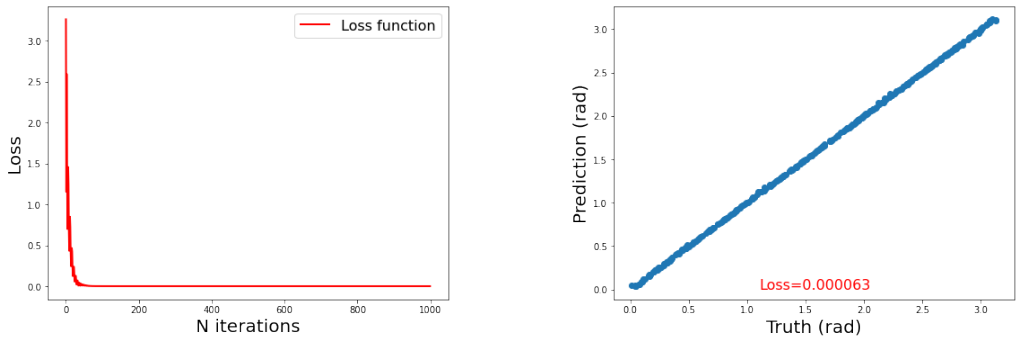


(a)



(b)

Figure 3.24: Top: Difference in activity between the input (event rate in blue), the simple cell layer (purple), and the complex cell layer (green). Bottom: scatter plot and correlation trends of the input rate function of the simple (purple) and complex cell (green) spike rates. We show the results of the shape recording. (a) With learned weights representation. (b) With random weights.



(a) (b)

Figure 3.25: (a) MSE Loss function over 1000 iterations. (b) The predicted angle is plotted against the true angle for the 200 test data points.

learning regression framework to associate the spike trains to a specific orientation of the bars. A spike train is a sparse and continuous representation; using it with a traditional supervised learning network is hard. Park et al. [121] showed that we could use kernel convolution methods to convert spike trains to more traditional feature vectors. We detailed the kernel convolution method earlier in Sec. 3.3.4. We used here an exponential kernel of the form:

$$\kappa(t) = \frac{1}{\sigma} e^{\frac{-t}{\sigma}} \tag{3.15}$$

with σ defining the width of the kernel. We convolute the spike trains with that kernel, using a σ of 10 and a sampling rate of 10 ms. It gives us 36864 features, one for each simple cell. We combined clockwise and counter-clockwise rotations of the camera for 10 seconds each to obtain a total of 2000 data points (since the sampling rate equals 10ms).

The final feature vector shape is $(2000, 36864)$. We recorded the orientation of the bars for each data point, which is stored in a label vector of size 2000. The orientation is bounded between 0 and π.

Finally, we designed a simple all-to-one linear readout for solving the regression task. We normalized, shuffled, and divided the data points into a train and test dataset containing 1800 and 200 points, respectively. We chose a MSE loss and Adam optimizer with a learning rate of 0.001. We trained for 1000 iterations.

Fig. 3.25a is the loss function over time. After only 50 iterations, the network converged. The training MSE loss after 1000 iterations is $1.2\,10^{-5}$. We present in Fig. 3.25b the test dataset with the network's prediction. The 200 predicted angles are plotted against the actual angle. All the points reside on the diagonal, which means that the angle is correctly predicted with great precision. The test MSE loss is minimal, only $6.3\,10^{-5}$.

To ensure that the learned representation has a real impact, we compare this supervised learning regression task to a network with randomly initialized weights. We kept the same data-gathering method and classifier parameters and trained for 1000 iterations. This time, the training MSE loss after training is equal to $9.1\,10^{-3}$, while the test MSE loss is $1.9\,10^{-3}$. The classifier can still mostly distinguish between different orientations but with less precision. It is not very surprising, considering the very high number of dimensions. With almost 37 000 neurons, we can expect a classifier to be able to solve such a simple classification task. Nevertheless, the important information is how well we can distinguish between orientations. The MSE is around two orders of magnitude higher than the loss from the network with trained representation.

3.4 Studying the receptive fields of simple and complex cells

In this section, we will analyze in detail the ability of our simple and complex cells to learn various representations. We show that they respond to orientation, motion, and disparity. We demonstrate the results on driving sequences from three different datasets.

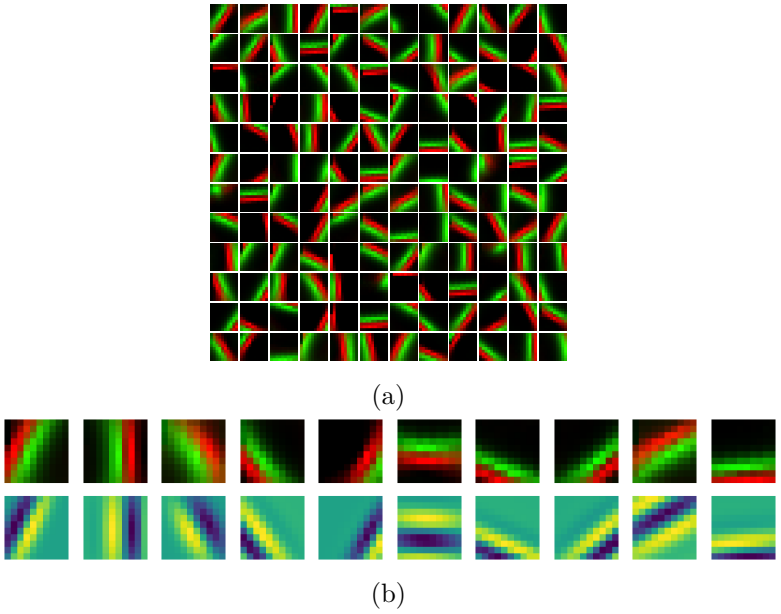
3.4.1 Learning simple cell receptive fields

This section demonstrates that our simple cell layer can capture different visual properties depending on the parameters and synaptic connection, such as orientation, motion, and disparity.

3.4.1.1 Learning on moving shapes

We first test the network on the shape inputs from Sec. 3.3.1.1. In that case, events represent the moving edges of the shapes. An edge can have many properties, such as orientation, speed, or disparity, in the case of stereo-vision. A good coding representation can capture that information.

To increase variability, we perform data augmentation by presenting the same video with artificial rotations and mirroring effects during training. It ensures that the neurons are presented with edges of all possible orientations. We used the first layer of simple cells without multi-synaptic connections and trained the network for



(a)

(b)

Figure 3.26: (a) Resulting receptive fields of simple cells learned with the moving shapes video sequence. (b) Examples of learned simple cell receptive fields with their matched Gabor function below it.

approximately 30 minutes. The network presents a similar architecture to the one detailed in Table 3.2 and cell parameters detailed in Table 3.1.

Fig. 3.26a presents the resulting receptive fields of the simple cells, forming a basis of oriented receptive fields. Green/red pixels represent synapses transmitting ON/OFF events, respectively. The color intensity represents the weight strength. Yellow areas indicate regions where the neuron is sensitive to both ON and OFF events. The cells learn well-defined receptive fields composed of two polarity edges. They show a wide range of orientation tuning. The learned representation is similar to Gabor filters. We further illustrate this by fitting Gabor functions to the receptive field as seen in Fig.3.26b. Those Gabors were then used and learned in many other applications, including traditional Artificial Neural Networks. They have proven to be effective early representations for solving complex visual tasks. We can observe that the simple cell receptive fields we learned are very similar to those of Gabor functions.

3.4.1.2 Evidence of visual statistical differences

To test the network's ability to develop diverse orientation-tuned receptive fields that match the statistics of the different parts of the visual scene, we used sequences from the DDD17: DAVIS Driving Dataset, described in Sec. 3.3.1.2.

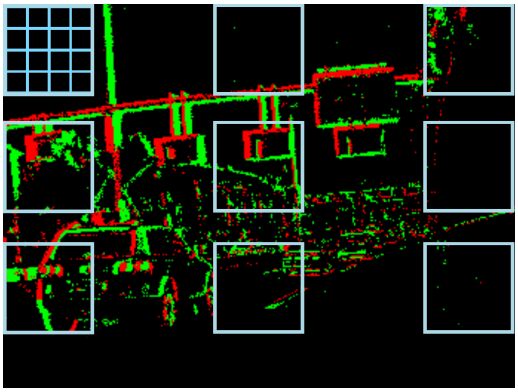


Figure 3.27: Examples of input events from the driving sequence. The blue squares indicate the locations of the nine different visual regions. Each region is then subdivided into 16 neurons' receptive fields that are indicated by a lighter shade of blue in the top left corner region.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	1	9	yes	4,4,144	10,10,2

Table 3.4: Network architectural parameters in the driving scenario with nine visual regions and weight sharing.

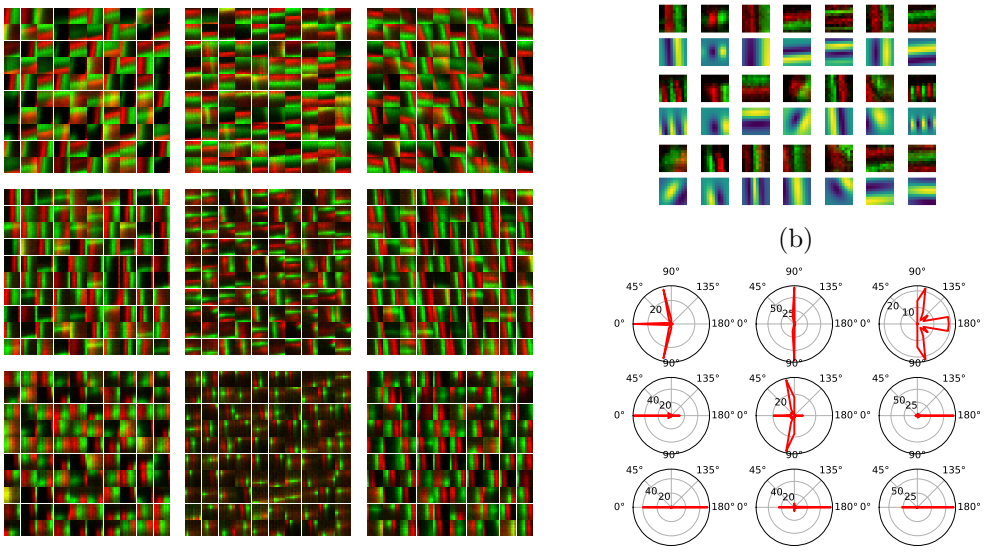


Figure 3.28: (a) Receptive fields learned for the 9 visual regions. (b) Selected examples of learned receptive fields (rows 1, 3, and 5) and corresponding Gabor fits (rows 2, 4, and 6) showing tuning to different orientations and scales. (c) Histogram of the network's receptive field orientations obtained from fitting Gabor functions to each visual region.

We changed a few architectural designs for that experiment. We show the network architecture in Table 3.4. The difference resides in the use of 9 visual regions, with fewer simple cells in them.

Inputs from different parts of the visual field have different statistical properties. An optimal representation of the visual inputs must take into account such effects [122]. To further explore statistical differences in visual inputs, we divided the pixel array into nine different regions (see Fig. 3.27). We offset the three bottom regions because the car's dashboard is visible in some driving sequences. Each region is subdivided into 16 smaller input tiles of 10×10 pixels, giving $9 \times 16 = 144$ tiles. Each input tile projects to $N = 100$ neurons connected by inhibitory synapses. The 16 neurons of the same region and slice N share the same synaptic weights. It implies a network size of $14,400$ neurons, but only 900 learned receptive fields.

In this experiment, we focused on the spatial structure of the learned receptive fields, and we thus considered a network without multiple synaptic delays (i.e., $D = 1$). The initial synaptic weights were drawn randomly from a uniform distribution. Figure 3.28 shows examples of learned receptive fields in the nine visual regions after 60 minutes of training on driving sequences alternating between highways, freeways, and cities.

Similarly to Sec. 3.4.1.1, we fitted Gabor functions to the simple cell receptive fields 93% of receptive fields obtained a good fit (sum of squared errors ≤ 5). Example fits are shown in Fig. 3.28b. They exemplify that filters of different orientations and

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	1	1	no	34,26,1	10,10,2

Table 3.5: Network architectural parameters in the driving scenario without weight sharing

scales are learned. It demonstrates that simple cells develop Gabor-like receptive fields for more natural and noisy environments such as driving scenes. Figure 3.28c shows the histograms of fitted Gabor orientations for each of the nine visual regions. Horizontal and vertical orientations are over-represented, resembling the oblique effect in visual perception [123]. Each region exhibits very different receptive field characteristics, including orientation preferences, as seen in Fig. 3.28c. The constant flow of objects from the center to the edges of the sensor generated by the car's forward motion is reflected in the preferred orientations of each visual region. The representation demonstrates that neurons will tune to the specific statistics of their visual field.

Removing the weight sharing mechanism In this experiment, we study the effect of the weight-sharing mechanism by deactivating it. We tested the STDP learning rule on the simple cells for the whole visual field. Here, every simple cell learns its representation independently of other surrounding cells. We created a network with 34 by 26 simple cells. It gives us a practical visual field of 340×260 pixels, close to the visual field of the DVS sensor of 346×260 pixels. The network architecture is detailed in Table 3.5.

Fig. 3.29 presents the learned receptive fields for all the simple cells. It was learned with one of the driving recordings, made on a suburban freeway with many buildings around, traffic signs, and tunnels. We can observe that the learned representation matches the driving sequence's statistics. Orientations are coherent with the optical flow of the scene since we expect horizontal orientation on the sides of buildings and lamp posts. On the top, horizontal orientation mostly comes from overhead traffic signs. At the bottom, orientations depend on road markings, such as zebras or demarcation lines. Zones without many visual inputs show no learning, such as in the bottom right portion of the visual field.

The weight-sharing mechanism helps the neurons to learn precise receptive fields by distributing the visual inputs on many different neurons. However, the mechanism is optional to learn. Compared to the network that learned with weight sharing, we can note that the receptive fields are, on average, noisier.

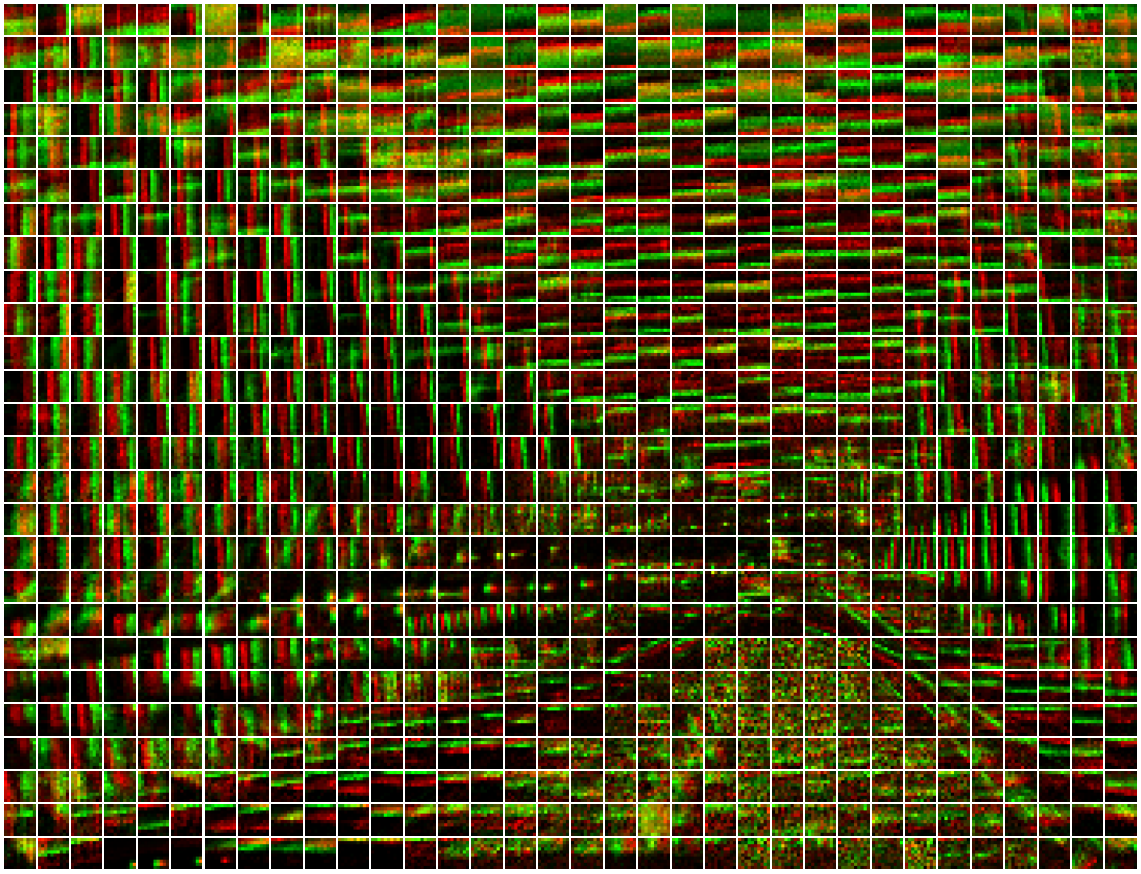
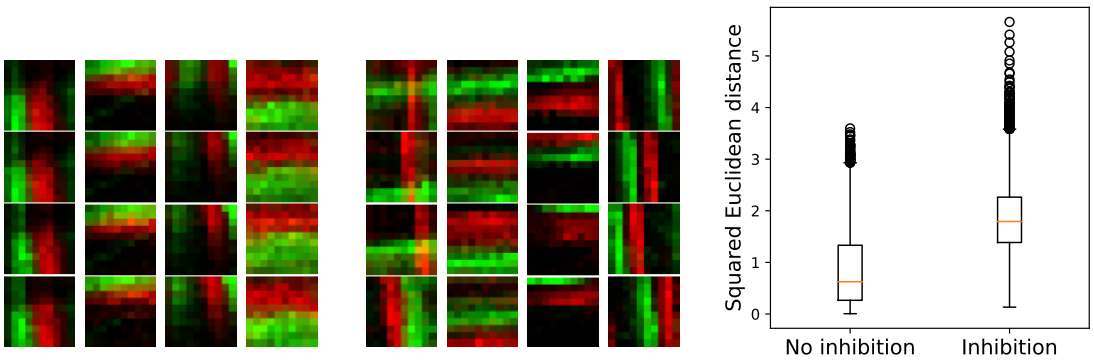


Figure 3.29: Receptive fields of a network without weight sharing learned on the entire visual field of the Davis346 event-based camera. The receptive fields were learned on one of the driving sequences in the DDD17 driving dataset.



(a) (b) (c)

Figure 3.30: Lateral inhibition diversifies receptive fields. Examples of 16 neurons' receptive fields learned without (a) and with (b) lateral inhibition. All neurons in a column receive the same inputs from the event sensor but start with a different random initialization of their synaptic weights. (c) Boxplot of the squared Euclidean distances between synaptic weights of neurons receiving similar inputs with and without lateral inhibition.

3.4.1.3 Lateral inhibition as a diversity mechanism

To test the importance of the lateral inhibition mechanism, we studied the effect of disabling it in Fig. 3.30. Each column in Fig. 3.30a represents the receptive fields of 4 neurons connected to the same input patch after learning without lateral inhibition. Even though the receptive field initialization was different, all four neurons have learned very similar receptive fields. In contrast, lateral inhibition leads to more diverse receptive fields (Fig. 3.30b). Figure 3.30c quantifies this effect by showing box plots of the distributions of all pairwise squared Euclidean distances between the receptive fields learned at the exact location. Inhibition significantly improves the diversity of receptive fields.

3.4.2 Weight evolution during training

Learning with SNNs is fundamentally different than with traditional ANNs. In a synchronous neural network, all the weights are updated simultaneously using some form of gradient descent optimization process. STDP is an unsupervised learning rule that only applies when a neuron spikes. It means the adaptation of neuron receptive fields depends on the cell dynamic rather than a continuous optimization process.

We present the evolution of the simple cells of a network with similar parameters presented in Sec. 3.4.1.1 while learning on the shapes recording detailed in Sec. 3.3.1.1. We start with random weights and present the recording to the network. Every 2s, we saved the weights and displayed them. We observe the evolution of those receptive fields in Fig. 3.31. We can see that with time, more and more receptive

fields learned a Gabor-like representation. However, only a fraction of the cells learn at a time. Most receptive fields remain random for a long time.

The last figure presents the receptive fields at the end of training (after showing the sequence 30 times, which corresponds to 450s). Interestingly, most receptive fields learned in the first minute have mostly stayed the same at the end of training. It shows that once a representation is learned, it remains relatively stable. We can also note that a few receptive fields still need to learn a well-defined receptive field at the end of the training. It is due to the amount of existing input pattern compared to the size of the receptive field basis. Here, there needs to be more diversity in the input, which implies that the learned basis is sufficient to encode the input well.

3.4.3 Development of motion and disparity tuned receptive fields

We evaluate here the ability of our network's simple cell layer to learn two more essential environment properties: motion and disparity.

3.4.3.1 Learning motion and disparity on synthetic inputs

We created a synthetic event video where we precisely controlled those two variables. The simplest type of stimuli consists of moving bars (from left to right) in front of a static camera. We chose a bright bar moving on a dark background, as depicted in Fig. 3.32a. It results in a leading positive event polarity edge, followed by a negative polarity edge. Because of the synthetic nature of the stimulus, we can choose precise bar orientations, motions, and disparities.

The learned simple receptive fields are shown in Figs. 3.32b and 3.32c. As already seen in Sec. 3.4.1.1, we observe again that the simple cells receptive fields are tuned to the orientation of the data sequence.

Specifically, simple cells are connected to the pixel array using three synaptic connections with different delays per pixel. It is represented in Fig. 3.32b by showing the corresponding three receptive subfields (one per delay) on top of one another. For each bar, we show two examples of learned receptive fields (colored frames indicate correspondence). The four bars move at speeds of (top to bottom) 420, 210, 140, and 105 pixels/s relative to the camera. To accurately capture this motion, we chose synaptic delays of 0, 10, and 20 ms for receptive fields of 10 by 10 pixels. This amounts to a velocity tuning of up to $\frac{10}{2.10^{-3}} = 500\,\text{pixels/s}$. Above that speed, the receptive fields would be too small (or the delays too big) to capture the bar motion. In Fig. 3.32b, we see that faster speeds become reflected in bigger shifts between the subfields corresponding to the different delays. For the fastest moving

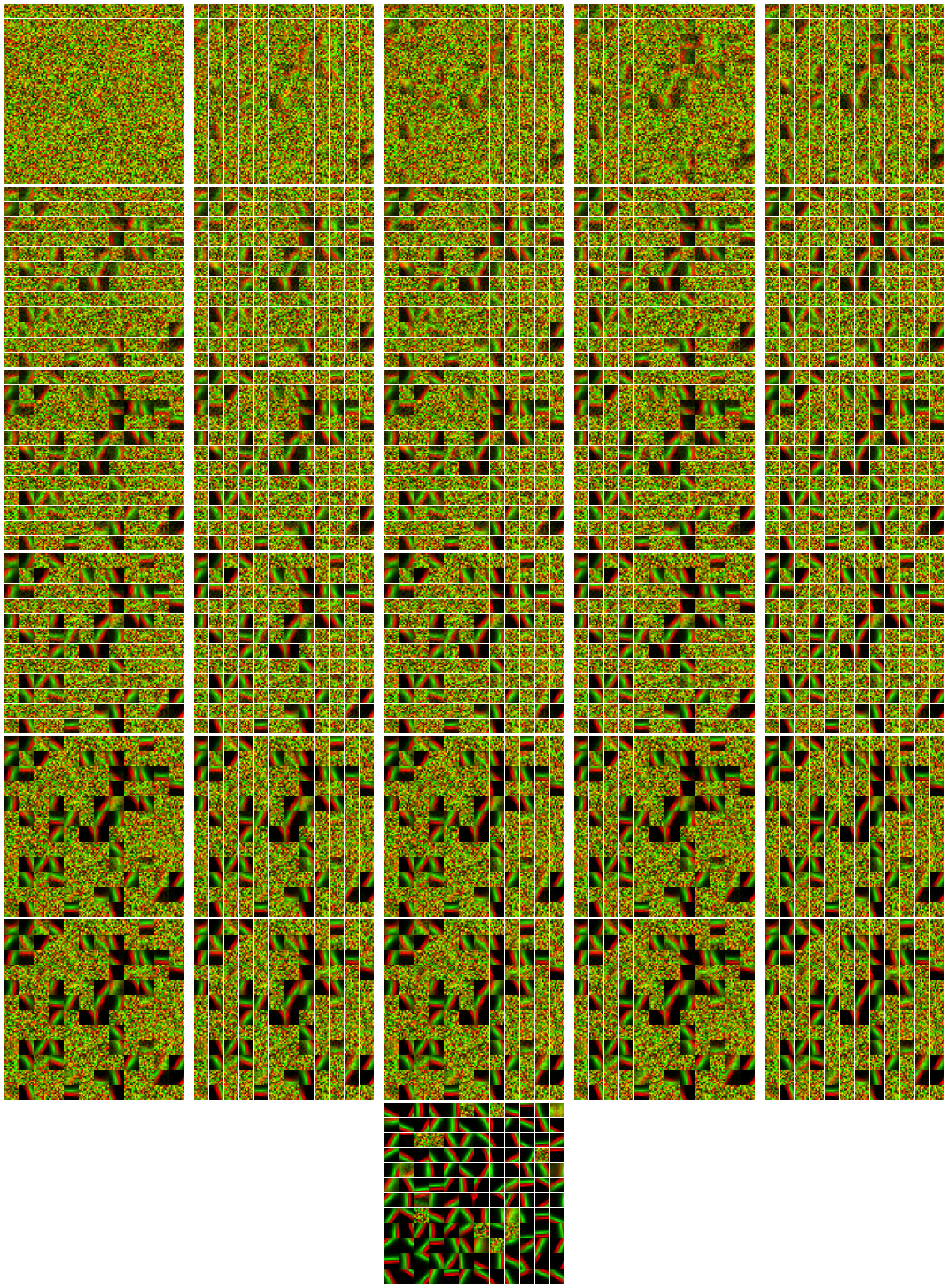


Figure 3.31: Evolution of the simple cell receptive fields during training on the shapes recording. Each figure is a snapshot of the weights made every 2s while showing event-based input from the shape recording. The last figure presents the weight at the end of training.

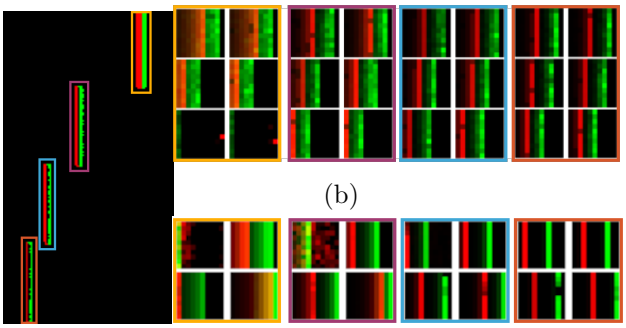


Figure 3.32: (a) Synthetic event video made of vertical bars moving from left to right. Their speed varies from the top (fastest) to the bottom (slowest). (b) Motion-sensitive cells (top) with three increasing synaptic delays (represented as three squared receptive field stack on top of each other). (c) Disparity-sensitive cells (bottom) are connected to a left and right "synthetic" camera, represented as two squared receptive fields stacked on top of each other. Two neurons are presented per moving bar, from the fastest (left) to the slowest (right).

bar (close to the 500 pixels/s limits), the receptive fields start to "break" for the 20 ms delay (bottom row in (b)). The first subfield with delay zero does not start completely to the right, pushing the shift for the 20 ms synaptic delay beyond the size of the receptive field. Due to the polarity-independent weight normalization (see Sec. 3.2.3.1), the weight to a random synaptic pixel becomes very strong (single red pixel). When adding multi-synaptic connections with multiple delays between the pixels and the simple cells, they learn a representation of the speeds of the bars.

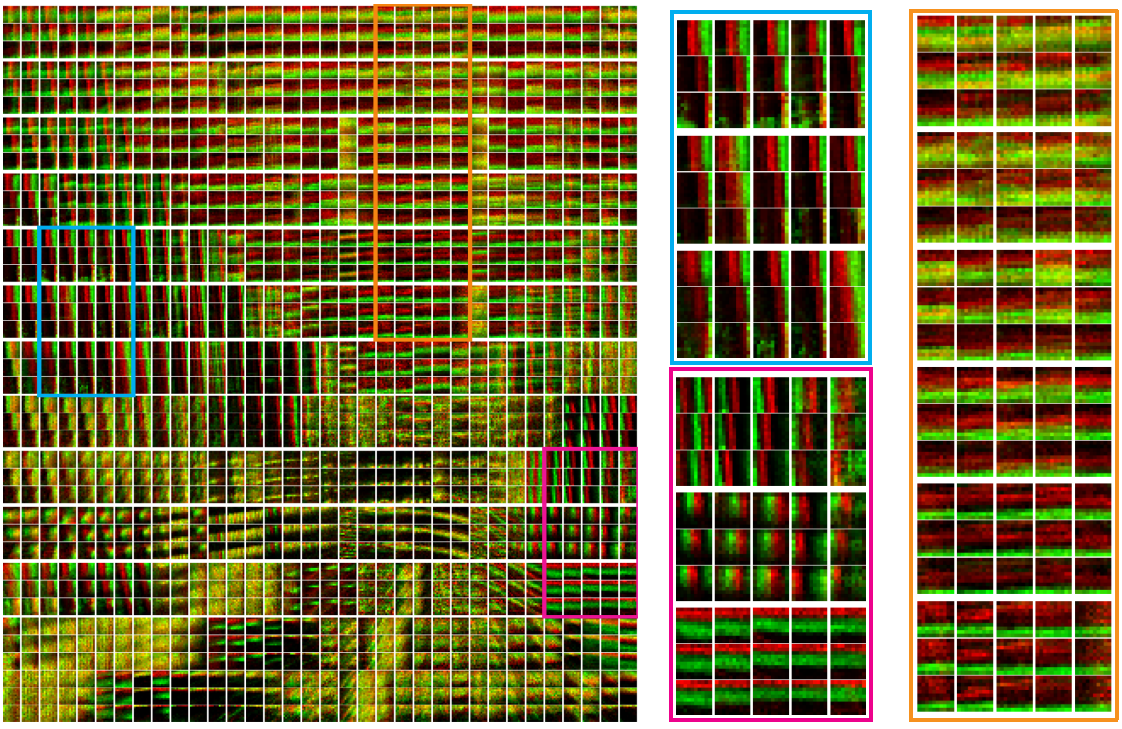
We extend this experiment by adding a second set of moving bars to form a stereoscopic setup. Each bar is slightly shifted in the second visual field to mimic different binocular disparities. Specifically, the bars have disparities of (top to bottom) 2, 4, 6, and 8 pixels. Here, single synaptic connections connect simple cells to the two stereoscopic visual fields. Fig. 3.32c depicts the resulting left and right receptive fields placed on top of each other. As expected, the shift between the left and right receptive subfields matches the disparity of the bar.

3.4.3.2 Motion tuning in an actual driving sequence

We further test the network's ability to develop motion-tuned receptive fields for natural input by training it on the driving sequence of Sec. 3.3.1.2. We used synapses with $D = 3$ different time delays of 0, 10, and 20 ms, respectively. We also returned to a complete tiling of the DVS array instead of the nine visual regions and disabled synaptic weight sharing. It allows neurons to tune to the optical flow that varies significantly in the scene depending on the visual field location. It is shown in Table 3.6. We were particularly interested in systematic differences in tuning

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	3	1	no	34,26,1	10,10,2

Table 3.6: Network architectural parameters in the driving scenario with multi-synaptic inputs.



(a) (b)

Figure 3.33: (a) Motion-tuned receptive fields learned across the entire field of view. Each receptive field has three sub-fields (arranged vertically) corresponding to different synaptic delays. Every second row of neurons has been removed in the figure to limit display size. (b) Enlarged view of marked groups of receptive fields in (a). See text for details.

properties across the visual field reflecting typical optic flow patterns occurring during driving. We chose a higher starting membrane potential threshold of $V_\theta = 150$ mV to compensate for the extra synapses as well as a lower target spiking rate of $S^* = 0.15$ spikes s^{-1}, which is sufficient in this sequence of fewer events. All other parameters were as in Table 3.1.

Learned Spatio-temporal receptive fields across the entire sensor array are shown in Fig. 3.33. We show only one of four neurons per location (compare Fig. 3.30). The receptive fields of some regions of the sensor are enlarged in Fig. 3.33b. Overall, a large variety of receptive fields tuned to different orientations, motion directions, and speeds have been learned. Importantly, we observe systematic differences in tuning properties across different parts of the visual field. In particular, the left and right regions of the network have mostly learned vertically tuned receptive fields.

In contrast, the top and bottom parts have developed horizontally tuned receptive fields (compare pink, blue, and orange regions). It is consistent with the expected statistics of the sensory input. The left and right regions of the data sequence contain many poles, trees, and buildings, which will generate vertically aligned event patterns moving horizontally due to the car's motion. In contrast, the top and bottom parts contain bridges, highway panels, and road markings, generating mostly horizontally aligned event patterns moving vertically. Furthermore, the shifts between receptive fields of different synaptic delays reflect the average speed of objects passing through that region of the sensor. We find more significant shifts in outer regions, and minor shifts in inner regions (comparing the top and bottom part of the orange region) correspond to large optic flow in the periphery and small optic flow in the center. It reflects the dominant optic flow pattern expected from forward ego-motion.

3.4.3.3 Learning disparity tuned simple cells on actual inputs

In this experiment, we extend the learning of disparity-tuned simple cell receptive fields to real event-based recordings. We used the drawn shape recording shown in Sec. 3.3.1.1 with a network architecture similar to Table 3.2, but with stereoscopic inputs. We generate disparities by duplicating the recording and generating disparities in various places of the visual field. We separated the visual field into nine different regions, each artificially displaced by a fixed disparity. Displacement ranges from -4 pixels to +4 pixels, from the top left to the bottom right of the visual field. That way, since we are using a weight-sharing mechanism, the simple cells receive visual inputs with multiple disparities at all times.

Fig. 3.34 presents the learned stereoscopic receptive fields of the simple cells. The left and right fields are displayed on top of each other for easier comparison. We can observe that cells learn to be tuned to a specific orientation and disparity. Fig. 3.35 shows the distribution of learned disparity-tuned receptive fields for that network.

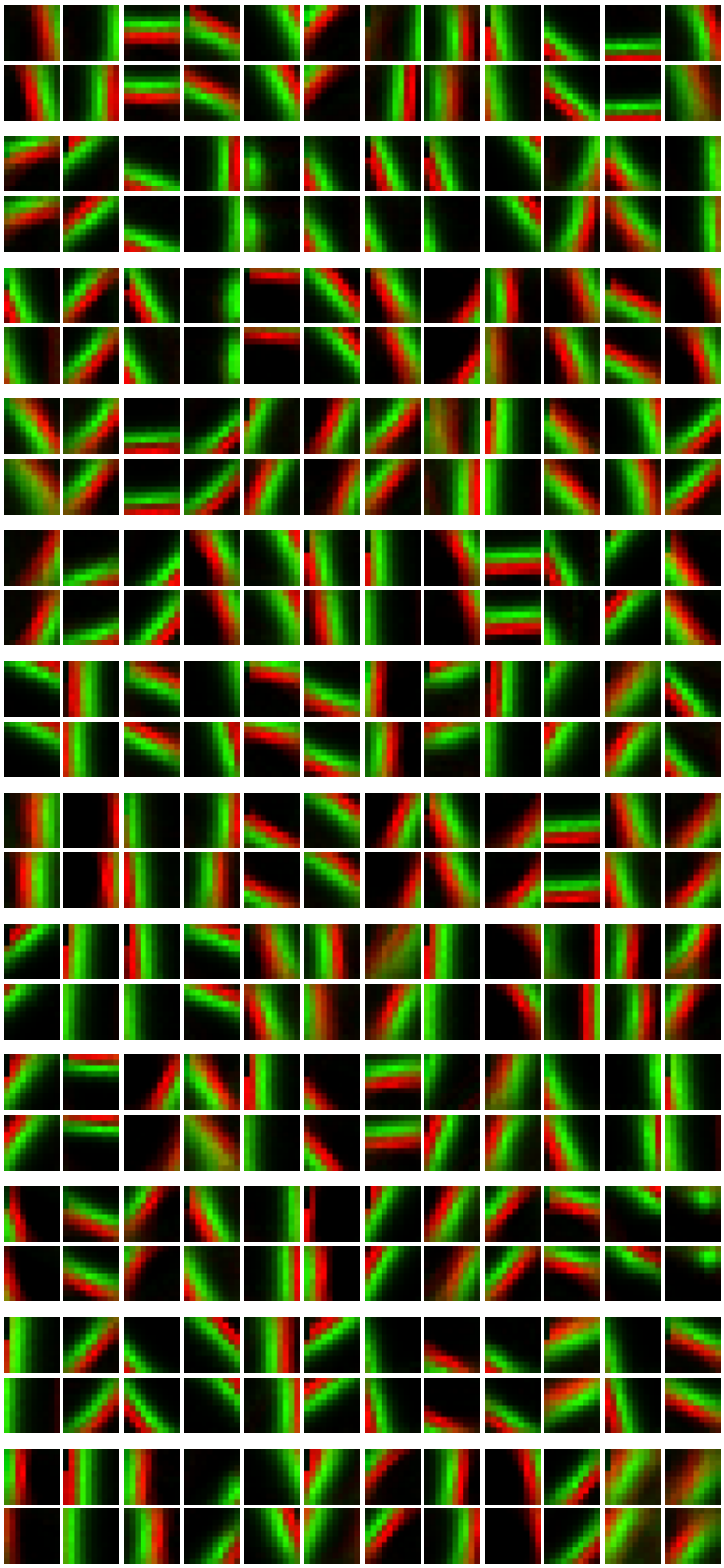


Figure 3.34: Basis of stereo simple cell receptive fields learned on a stereo recording of drawn shapes with added disparity. The figure displays the left and right receptive fields on top of each other.

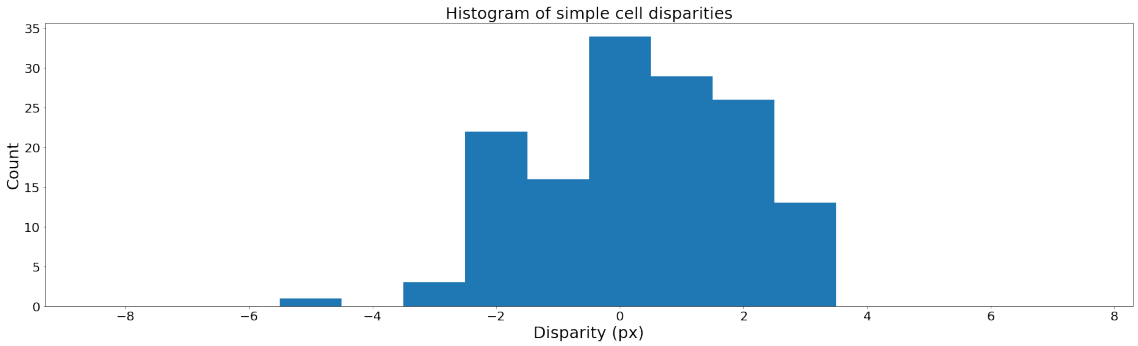


Figure 3.35: Histogram of learned disparities for the simple cells learned on a recording of the drawn shapes with added disparity.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	1	20	yes	3,3,49	10,10,2

Table 3.7: Network architectural parameters in a stereoscopic robotic mobile platform scenario.

We estimated the disparity using a simple matching scheme based on computing the MSE error between the left and right subfield. We shift the left subfield and select them according to disparity for the shift that produces the minimal MSE error. That way, each simple cell is attributed to a specific disparity. We then show the distribution for all the simple cells. The distribution is somewhat coherent with the disparity present in the event recording. The distribution is not uniform since there can be visual regions with more events and, therefore, more chance that a simple cell would tune to that disparity.

3.4.4 Estimating disparity from stereo driving scenes

In this section, we estimate the disparity in the visual scene by learning stereoscopic receptive field and comparing the left and right sub-field. We want to demonstrate that simple cells' basis should reflect the environment's disparity.

3.4.4.1 Depth estimation with a robotic mobile platform

We feed our network stereoscopic input from a pair of event-based cameras mounted on a mobile robotic platform operating in an urban outdoor environment (see Fig. 3.17). The disparity statistics vary greatly depending on the location within the

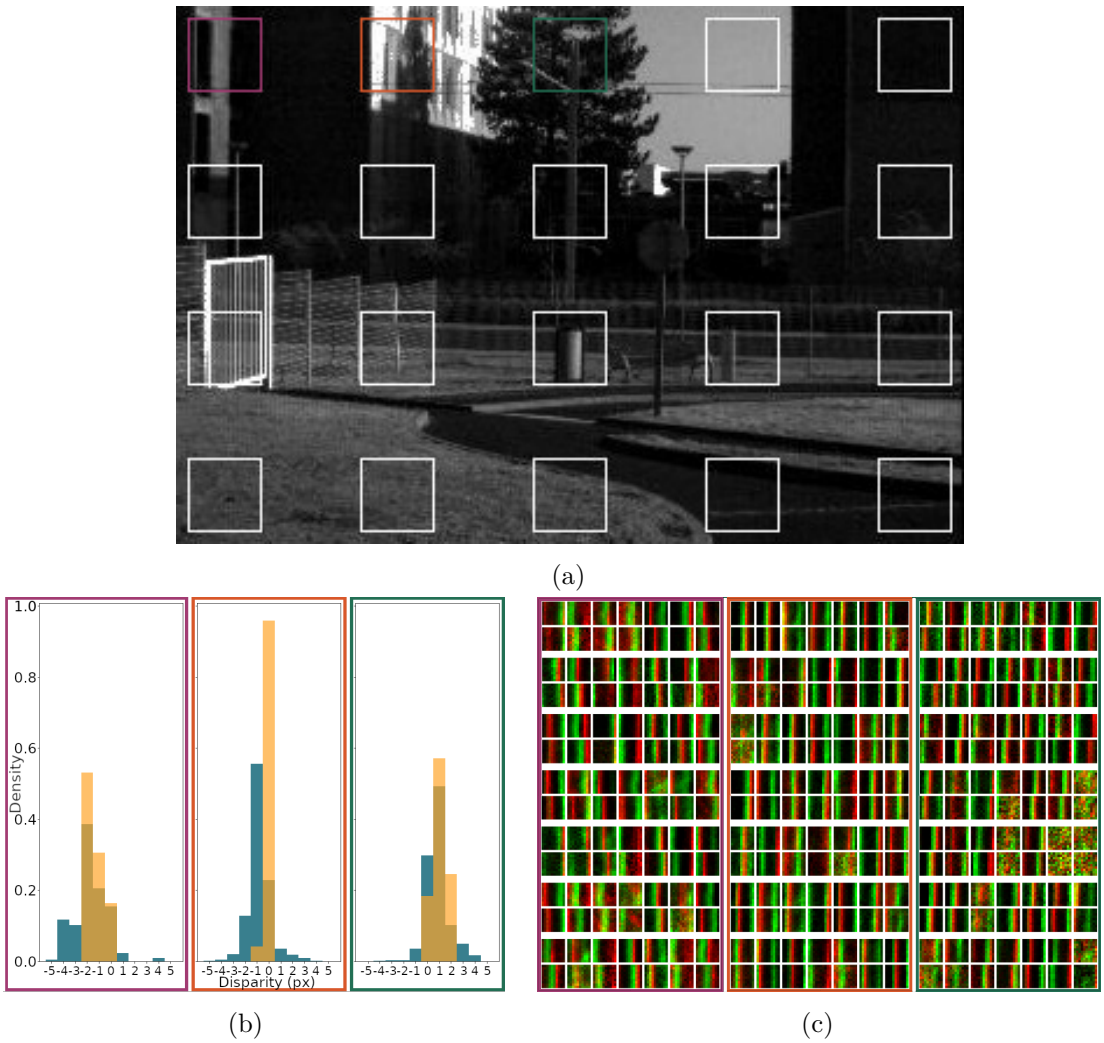


Figure 3.36: (a) Example of an image (left camera) taken from one of the outdoor sequences. The squares mark the different image regions. The colored squares in the upper left mark regions for which disparities and receptive fields are shown in (b) and (c). (b) Histograms of estimated disparities of the learned receptive fields (orange) and disparities estimated from corresponding image frames with conventional computer vision techniques (blue) for the three colored regions. (c) Learned left (top) and right (bottom) receptive subfields for the 49 neuron layers in the three colored regions.

visual field. We use a network of multiple patches of neurons looking at specific visual field regions. Neurons in different locations within a single region share their weights (similar to convolutional neural networks). Specifically, we consider 20 regions, each composed of an array of 3 by 3 simple cells, with 49 neuronal maps connected by inhibitory synapses. It is recapped in Table 3.7. Figure 3.36a shows a typical image from the training sequence. Squares mark the 20 regions.

Neurons are connected to the inputs of both the left and right sensors and receive events at slightly shifted pixel locations depending on the distance to the stimuli. The horizontal distance between the two event-based sensors induces disparities between the two images, which induces differences between the learned left and right receptive subfields of binocular neurons. We trained the network on a small repeated sequence of about 45 seconds to control the number of different disparities exposed to the neurons.

We find the preferred disparity of the neuron's learned receptive field by determining the smallest mean squared error (MSE) between the left and shifted versions of the right receptive subfields of the neuron. It is important to note that the maximum possible disparity is limited by the size of the receptive fields. That method is precise, up to a minimum of one pixel. The learned receptive fields are shown in Fig. 3.36c. We selected three regions most exposed to objects of similar distances so that a different disparity dominated input to each region. Only vertical receptive fields were learned for this input. It is likely due to the scene structure, which is dominated by vertical orientations. More generally, it is well-known that vertical and horizontal orientations are abundant in man-made environments [123]. Figure 3.36b presents a histogram (orange) of the computed disparities. We observe that most receptive fields in a region learned one specific disparity, up to a variation of 1 or 2 pixels. Interestingly, there is a systematic relation across the regions.

The learned disparity increases roughly linearly as we move from the left side to the center of the visual field. It is consistent with the scene's structure, where objects in the center present a smaller relative disparity. We compare the learned preferred disparities to disparities estimated via conventional computer vision techniques from image frames (that the sensor also produces) (blue histogram in Fig. 3.36b). The results are in reasonable agreement with the receptive fields learned by our spiking network. The inherent variability and noise in event-based data could explain the differences. Furthermore, the frame-based analysis considers information that might not appear in the event stream since uniform motifs tend not to create any events.

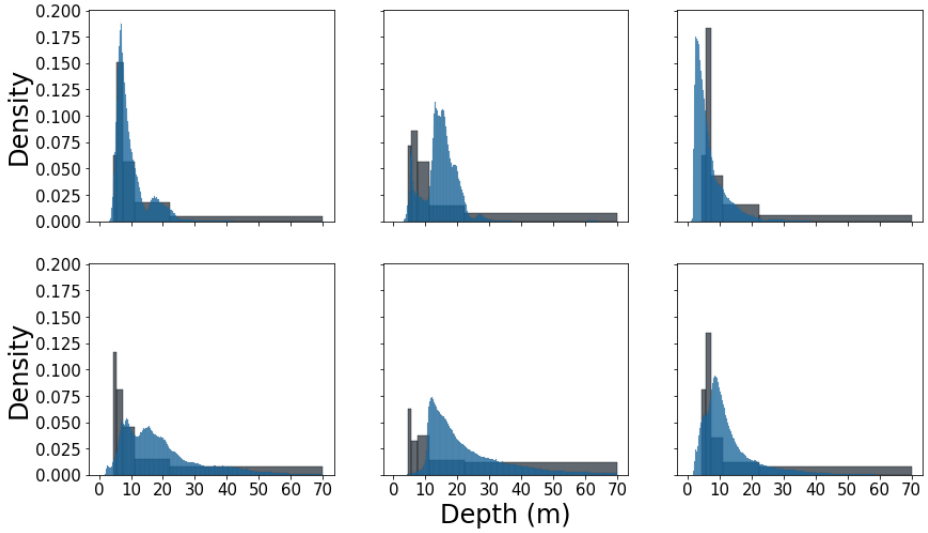


Figure 3.37: Depth histogram ground truth obtained from lidar information from the MVSEC dataset sequence for the first six visual regions (blue). Histogram of estimated depths of the learned receptive fields for the first six visual regions (grey).

3.4.4.2 Depth estimation from The Multi Vehicle Stereo Event Camera dataset

Similarly to the previous section, we tried replicating those results on a different dataset. We trained our network on the MVSEC dataset, detailed in Sec. 3.3.1.3. We focus on the sequence "outdoorday1data", a stereo urban driving scene. RFs are learned for nine different visual field regions (see Table 3.4). The learned preferred disparities for the different regions reflect the distribution of object distances across these regions roughly corresponding to the ground truth data of object distances.

Using the information provided in the dataset, we can estimate the network's ability to learn disparities. We have detailed information on the setup, including the baseline length B between the two event-based cameras (10 cm), the intrinsic focal length f in pixels (given by the calibration file, 223 pixels), and the depth of the scene D. According to this equation:

$$D = \frac{Bf}{d} \tag{3.16}$$

we can relate the disparity d in pixels of the learned simple cells' receptive fields to the scene's depth. Considering a 10 by 10 pixel receptive field, we estimated the disparity ranging from -5 to 5 pixels, depending on the relative shift between the left and right subfield. We use the absolute values of the disparity measures as we are only interested in depth computation. We can therefore detect depth ranging from $\frac{0.1 \times 223}{5} = 4.46$ meters up to theoretically infinite depth.

To have a comparison point, we used the ground truth depth maps present along the event-based sequence of the dataset obtained from lidar information. We present

those values for the six first regions in Fig. 3.37 in blue. We observe a wide depth range, ranging from a few meters up to 100 meters. We removed the three bottom regions, which are much more limited in depth since they are directed towards the road and the car (ranging from 2 to 5 meters). The central region (2^{nd} row 2^{nd} column in the figure) presents the most significant depths since it is oriented towards the center of the scene.

We plotted the computed depth histogram in grey using the learned receptive fields. To compare to the ground truth provided in the dataset, we used bins of unequal length using equation (3.16). We limited the last bin to 70 meters since it can theoretically represent any depth from 22.33 meters up to infinity. Comparing the two figures, we observe a reasonable visual correlation between the two distributions. Once again, frame-based and event-based information is not easily compared as they do not represent the same information. Moreover, our method to estimate receptive field disparity can be imprecise when the receptive field present some noise. It can explain the differences between the two histograms. Despite that, the set of simple cell receptive fields forms a coherent basis spanning a wide range of disparity tuning.

Figure 3.38 shows the simple cells' receptive fields learned on the driving sequence for the nine different regions. We observe a coherent tuning to orientation considering the learning scene, i.e., most oblique orientations in the corners and vertical and horizontal orientations for the sides and center. The set of receptive fields forms a solid basis that responds to multiple orientations and disparities consistent with the scene structure, making it an efficient processing stage for solving more complex dynamic tasks.

3.4.5 Learning complex cell receptive fields

We study in this section the response of complex cells to oriented stimuli and measure their selectivity in orientation and direction space. Complex cells learn somewhat different representations than simple cells. They cover larger zones of the visual fields and pool information from multiple simple cells without differentiating between polarities. It gives us receptive fields that are not sensitive to one specific motion direction but instead to more significant spatial features such as corners or edges while invariant to motion direction. We trained the network described by Table 3.8 on the shape recording Sec. 3.3.1.1. Fig. 3.39 presents a few examples of complex cells receptive fields. On top, we selected the maximum synaptic connection and displayed the simple cell receptive field associated with it. We also weighted the intensity of the receptive field display by the weight itself. It gives us the complex cell preference for a simple cell orientation filter. The bottom part presents the sum of all simple cells with the same receptive fields in the image plane. We observe that the top and bottom figures present very similar structures, often presenting one or

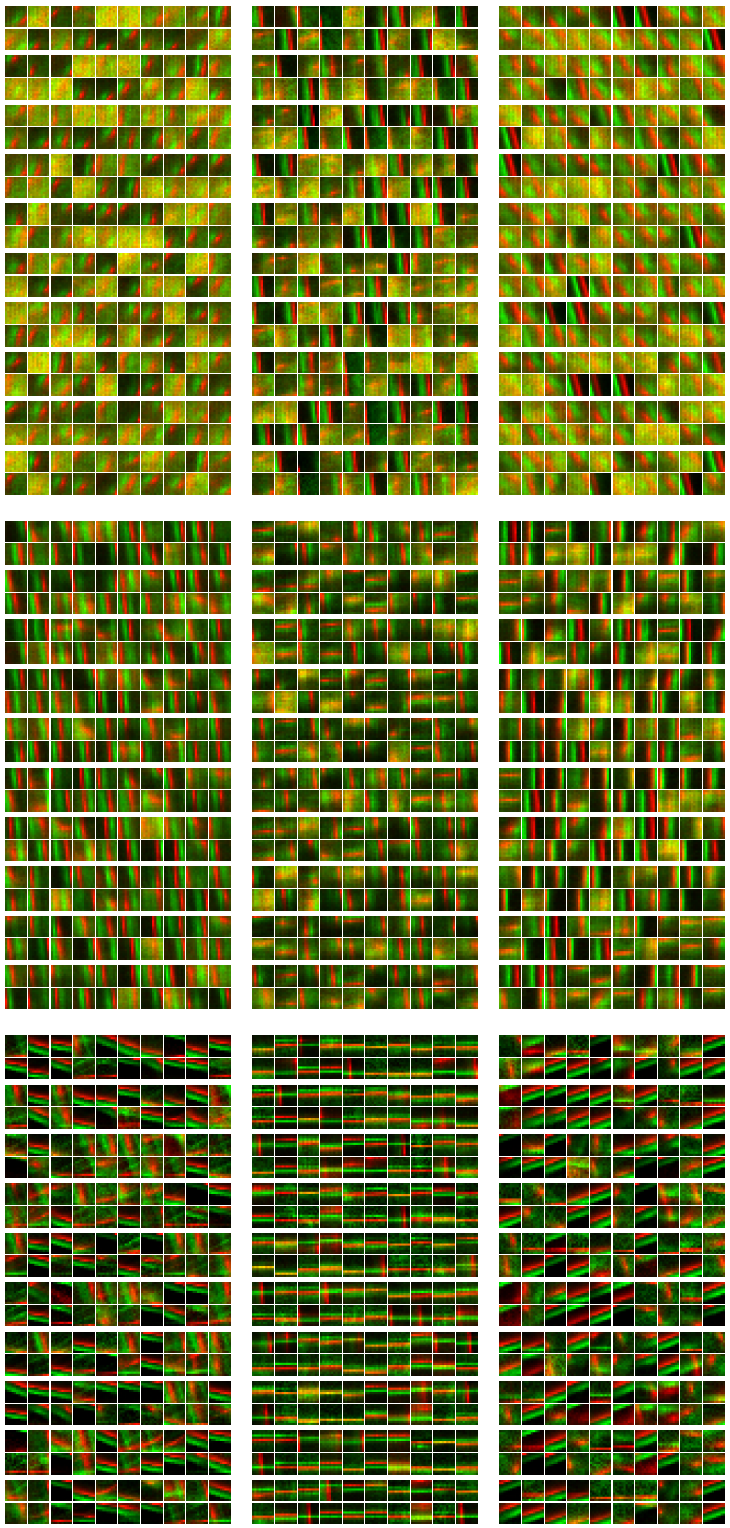


Figure 3.38: Simple cell receptive fields learned on nine different image regions of a driving sequence from the MVSEC dataset. We present the left and right subfields (from the left and right event-based cameras) on top of each other.

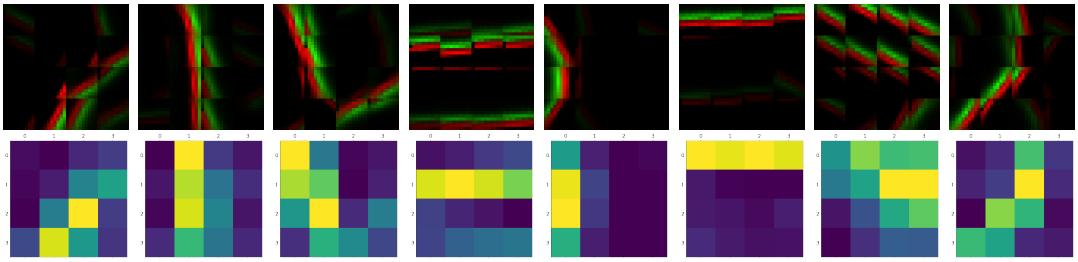


Figure 3.39: Examples of learned complex cell receptive fields. On top are the weighted simple cell receptive fields with the strongest connection to the complex cell. The bottom is the sum of all simple cells in the image plane with the same receptive fields.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size rfs
		(x, y)				(x, y, z)	
Simple cells	static	none	1	1	yes	12,12,100	10,10,2
Complex cells	static	none	1	1	no	3,3,16	4,4,100

Table 3.8: Network architectural parameters

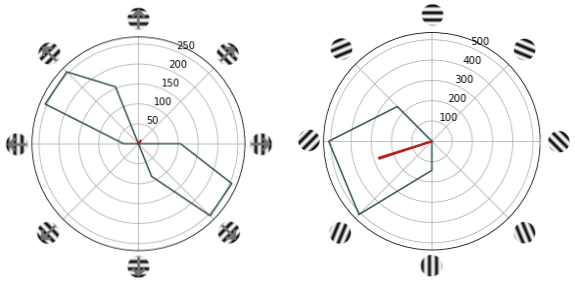
two (for corners) preferred orientations with substantial weights on a narrow portion of the complex cell receptive field.

3.4.5.1 Orientation and direction selectivity

Neuroscientists usually analyze the behaviors of these types of cells by observing their responses to oriented stimuli. A standard test is to show differently oriented gratings and measure a cell's response. We followed the same procedure by creating artificial event sequences as produced by gratings moving in 16 different directions (from 0 to 360 degrees by steps of 22.5 degrees). We measured the responses of our artificial neurons by counting the number of spikes produced during the presentation of the stimuli and averaging the numbers over 5 trials. We used the normalized vector length L to quantify orientation and direction tuning [124]:

$$\begin{aligned} L_{dir} &= \left| \frac{\sum_k R(\theta_k) \exp(i\theta_k)}{\sum_k R(\theta_k)} \right| \\ L_{ori} &= \left| \frac{\sum_k R(\theta_k) \exp(2i\theta_k)}{\sum_k R(\theta_k)} \right| , \end{aligned} \tag{3.17}$$

where $R(\theta_k)$ is the average number of spikes for stimulus direction θ_k. We summed together the number of spikes for the two opposite motion directions. The normalized



(a) (b)

Figure 3.40: (a) Complex cell response in direction space, made from counting the cell's spikes for a rotating grating stimulus. The red line corresponds to the normalized vector length and indicates the cell's selectivity strength and direction. (b) Complex cell response in orientation space, made by pooling over opposite directions.

vector length gives the sensitivity of a cell to a particular direction. A cell that reacts very strongly to only one direction is said to be highly direction selective. Whereas a cell reacting strongly to many different directions is not selective. A particular case emerges with cells being very selective in orientation space but not in direction space. It is due to cells responding strongly to opposing motion directions.

We visualize a cell's selectivity by plotting a circular histogram of its responses to different oriented stimuli. Figure 3.40 shows the response of an example complex cell in orientation and direction space. The red lines correspond to the normalized vector length. We observe that this cell is orientation selective in Fig. 3.40b but not direction selective in Fig. 3.40a. It exhibits two roughly symmetric lobes in direction space, which cause a low selectivity value. The cell will strongly respond to stimuli oriented at around 135° and 315° but not other orientations. Looking at the complete set of the 144 learned complex cells in direction space, we observe that most complex cells exhibit similar responses to the ones shown here, except for a few that are also direction-selective (uni-lobe).

We present a more extensive set of complex cell selectivity responses in Fig. 3.41. It shows the direction selectivity of 36 of the 144 trained complex cells. Most complex cells are sensitive to opposite motion directions (dual lobe). Only 2 out of the 36 cells presented have a unilateral response in direction space (4^{th} row, 3^{rd} column and 6^{th} row, 5^{th} column). The sizes and shapes of the lobes vary slightly. The population covers a wide range of orientations/directions, enabling it to represent various visual inputs.

We compute the normalized vector length for the whole batch of trained complex cells to quantify the results. The result is visualized in Fig. 3.42. It shows the overall orientation and direction selectivity. On average, cells are highly orientation-selective but not direction selective. Figure 3.42b represents the distribution of preferred orientations of the complex cells. It exhibits a preference for oblique orientation. It is

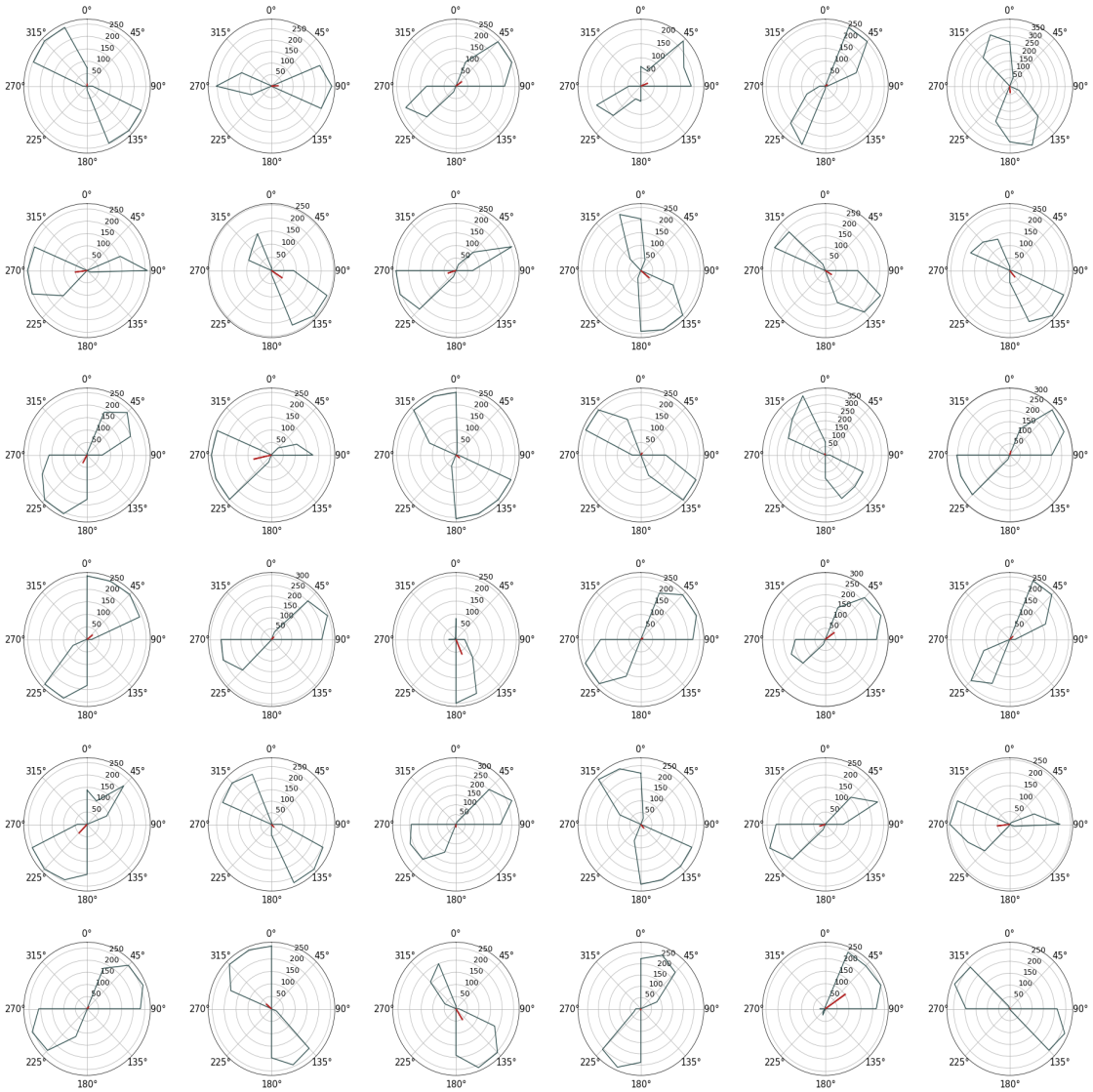
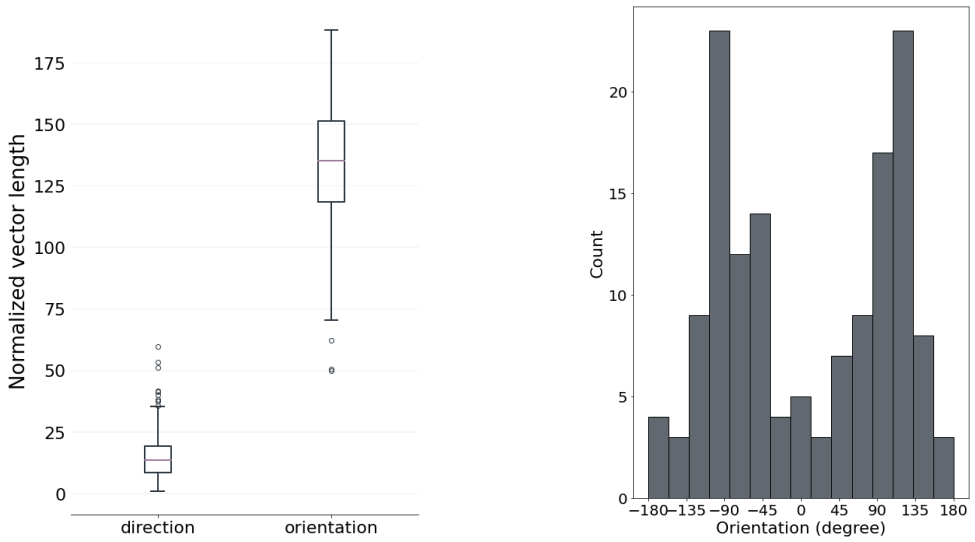


Figure 3.41: 36 first (out of 144) complex cell response in direction space. Results are obtained by measuring spike activity for gratings of multiple orientations.



(a) (b)

Figure 3.42: (a) Normalized vector length distribution of the network complex cells in direction and orientation space. (b) Histogram of normalized vector orientations in orientation space (0° corresponds to a horizontal orientation).

consistent with a slight overall preference for oblique orientations among simple cells seen in Fig. 3.26a. Overall, the complex cells have learned to respond to a wide range of orientations/motion directions. It is also coherent with biological complex cells that are orientation-selective but not direction selective, as mentioned in Sec. 3.2.4.3.

3.4.5.2 Analysis of additional STDP windows

We present the results obtained with the other STDP windows in Sec. 3.2.4.3. These windows lead to different degrees of orientation selectivity, as shown in Fig. 3.43. The simple step window used in our main results gives the highest amount of orientation selectivity.

Conclusion

We have presented a spiking neural network that learns orientation, motion, and disparity representations in a fully unsupervised fashion via STDP from the input of event-based cameras. Motion tuning arises from STDP with multiple synaptic delays combined with homeostatic mechanisms and a lateral inhibition scheme to diversify tuning properties.

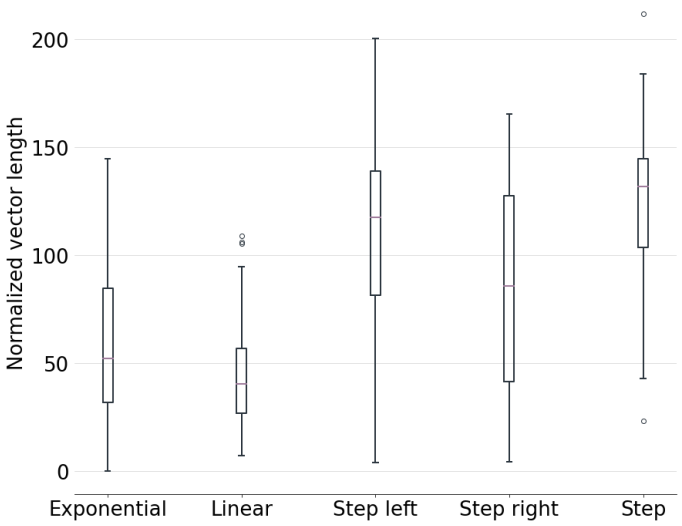


Figure 3.43: Orientation selectivity for 5 different complex cell STDP windows.

The learned representation shares many similarities to that observed in the brain, including simple and complex cells found in mammals' primary visual cortex. Furthermore, as observed in biology, the learned representation adapts to the statistics of the visual input. However, even though the mechanisms used are all biologically inspired, they are not intended as accurate models of biological reality. The biggest idealizations are the instantaneous lateral inhibition decorrelating responses of neurons with overlapping receptive fields and the normalization of synaptic inputs. However, the latter could be biologically plausible if the different groups of synapses were considered to reside on separate dendritic branches [113]. It is plausible for synapse groups with short vs. long delays, which correspond to inputs to more proximal vs. more distal dendritic branches, respectively. Similarly, On and OFF inputs could also be sorted into different dendritic branches or regions during development based on their correlations. Nevertheless, our model, while not fully bio-plausible, gives exciting insights into the processes of cells in the brain. It is one of the only complete formulations of a complex cell model for event-based inputs.

Compared to similar work, our network is very versatile. It can learn many different types of information, such as orientation, motion, and disparity, with minimal changes. It makes it practical for solving very different visual applications, such as tracking, optical flow estimation, or depth perception. We demonstrated that our coding layer presents sparse but informative representations of the environment. It can capture different visual properties of the scene, which makes our SNN well adapted as the first stage of a more complete AEC framework. We showed that the network significantly reduces the amount of input by compressing data without losing too much information. Simple and complex cell patterns are adequate representations of the input properties. They can be used as an efficient encoding layer for a second stage, such as a classifier or a reinforcement learner.

Chapter 4

Reinforcement Learning with intrinsic reward

Introduction

Reinforcement learning is getting increasingly popular in the Artificial Intelligence (AI) field. It has proven to be a compelling framework able to solve challenging tasks with impressive capabilities, often much better than a human could do. Algorithms such as Alphago or AlphaZero have revolutionized how chess and go are played, with the top players regularly training based on AIs suggestions. Their most significant limit, however, is how specialized they are. Reinforcement learning agents are trained on specific tasks and cannot transfer that knowledge to other problems. Reinforcement learning frameworks are beneficial when solving closed-loop motor control problems such as vision. They offer effective training procedures based on simple reward objectives without the need for extensive training data.

When it comes to biological systems, things get more complicated. There is clear evidence that some form of reinforcement behaviors exist in nature. It is possible to train animals to solve complex tasks, such as getting out of a maze, using some form of external reward, often by food distributed when the animal performed well [125]. It proves that biological systems work by interpreting external feedback and changing their neural architecture to maximize some reward. After enough repetition, visual information will trigger specific motor responses that lead to successfully solving the task.

For this behavior to appear, the animal must associate a positive reward with the actions that led to it. We know that some neuromodulators have an essential role in that case. Neuromodulators can influence synaptic plasticity, triggering changes in neuronal connections and therefore affecting the associated behavior. Dopamine levels rise sharply when the animal receives food, for instance. Even more interesting, dopamine can appear before the reward is given, as the animal expects to receive the reward after successfully performing the task.

In the AEC framework, the agent part is based on traditional reinforcement learning theory. The agent evolves in an environment, receiving information from it and acting on it via motor commands. In the case of vision, the main components are the visual perception of what is around the agent, and the main interactions are done by moving the eyes. From time to time, the agent receives a reward signal indicating how well it performs. The visual stream of information is fed to the efficient coding part, which role is to improve the data encoding to simplify the job of the reinforcement learner. We discussed that in detail in Chapter 3.

We are interested in how we can take inspiration from biological systems and design a reinforcement learning algorithm to process asynchronous visual data from event-based vision sensors. Since event streams are continuous flows of information,

traditional reinforcement learning methods cannot be used directly. We need to formulate algorithms adapted to processing neuronal spikes. The reward signal can act as a neuromodulator, triggering changes in the synaptic weights when combined with the STDP unsupervised learning rule.

However, the AEC framework does not require external feedback to optimize its behavior. In the formulation, the reward is intrinsically generated from the efficient coding layer. Basically, the better the encoding, the higher the reward will be. It is also true in biological systems. Vision is a fast and reactive process that cannot rely on external supervision alone. Efficient eye behaviors emerge from how well the brain can process visual data. Inefficient eye movements are quickly eliminated as they do not produce a suitable encoding.

To replicate this process, we derive the reward signal from the efficient coding layer activity. We introduce plastic inhibition connections in the efficient layer to act on cell activity. After learning, different visual inputs will trigger different amounts of activation, which can be used as a baseline to generate an intrinsic reward.

We propose a modified TD reinforcement learning algorithm based on an actor-critic formulation. The framework uses a three-factor learning rule called R-STDP to learn an effective mapping between the efficient coding layer and population of actor and critic neurons. We first describe the network using the environment's external reward. We demonstrate that the network can learn to solve complex visual tasks such as tracking or stabilization problems. We propose a novel method to generate an intrinsic reward from the activity of the simple cells based on the type of visual inputs. Inhibition connections are introduced laterally between the simple cells and recurrently from the complex cell to the simple cells. We show how to tune those connections to specific visual inputs using an STDP learning rule. We demonstrate that the average cell activity will be tightly correlated to the type of stimuli. We then derive an intrinsic reward signal from that activity and learn to solve those visual tasks again without needing external supervision.

4.1 Related work

Although SNNs are getting more and more popular, their use in reinforcement learning is still pretty rare. Bing et al. [126] propose a survey of work involving robotics control based on SNN.

Many applications involving motor control and SNNs are made with pre-wired circuits on neuromorphic chips or GPUs.

Jiang et al. [127] demonstrated a vision-based tracking for a snake robot with an event-based sensor. They combine an SNN for pipe-like object detection based on the Hough transform and integration of the snake body position. They demonstrate their results in simulation on a GPU. Linares-Barranco et al. [128] modeled an approach detection mechanism for obstacle avoidance. They mimick the approaching detection functionality of the retinal ganglion cells. They implement the algorithm on a FPGA connected to an event-based camera.

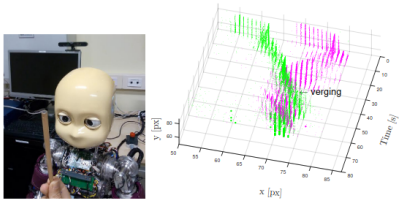


Figure 4.1: Icub robot performing vergence by Vasco et al. [129]

Vasco et al. [129] use an ICub robot, as shown in Fig. 4.1, to perform vergence control using Gabor filters with a phase shift model. Their work is inspired by the role of simple cells in disparity detection. However, they use predefined filters and do not learn any control behaviors.

While those work provide a complete framework and implementation of closed loop motor control, no learning is involved. It limits their applicability and biological relevance.

Learning effective policies using traditional reinforcement learning algorithm is difficult with an SNN due to their asynchronous and spiking nature. We can note some hybrid strategies, using traditional reinforcement learning frameworks to train the network, and then convert the weights to use with a SNN for fast and energy efficient inference, such as in [130].

Others such as [131] combine fast and energy efficient SNN for inference while learning with traditional deep reinforcement learning. However, we are interested in fully spiking strategies capable of learning in the spiking domain.

4.1.1 Spiking reinforcement learning

The brain relies heavily on external or intrinsic rewards to learn efficient behaviors. For instance, dopamine, a neuromodulator, has been studied extensively and is often associated with beneficial actions. The Reward Modulated STDP (R-STDP) learning rule tries to mimic that process by modulating the classic STDP rule with a third-factor reward.

Fremaux et al. and Florian et al. [12], [132] describe in detail this learning rule and its implementation. The reward can be introduced directly in the learning rule, or be used via eligibility traces [133] to delay the transmission of the reward and therefore associate it with past actions. R-STDP has been used in [134], [135] to tackle an MNIST classification problem, but lacks a real reinforcement learning framework in the traditional sense of having an agent perform actions in a controlled environment.

Yuan et al. [136] are combining stochastic and deterministic plasticity learning rules to form neuronal agents. They designed a unique Stochastic-Deterministic Coordinated (SDC) spiking reinforcement learning model capable of solving simple problems such as a 19-state random walk.

Most models of spiking reinforcement learning techniques use a TD learning actor-critic framework based on the R-STDP learning rule. Potjans et al. [137] was one of the first to propose such a reinforcement learning framework able to solve a simple grid-world task. Jitsev et al. [138] propose a model that mimics the basal ganglia functions that learn from both positive and negative rewards. Similarly, they demonstrate their results on a solving a grid-world environment. Nichols et al. [139] use a similar framework implemented on a more ambitious robotic agent. The robot learns to solve a wall-following task using laser and sonar proximity sensors.

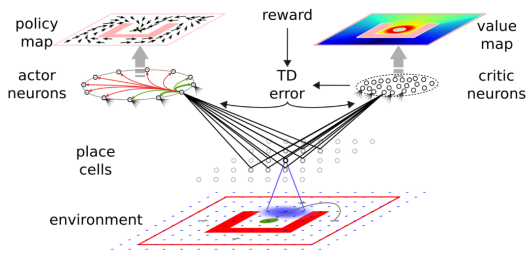


Figure 4.2: Actor-Critic spiking TD learning framework by Fremeaux et al. [140]

Similarly, Fremaux et al. [78], [140] have extended a traditional discrete Temporal Difference (TD) learning framework into a continuous spiking one. They use it with a fully spiking actor-critic based agent to solve simple maze environments, as presented in Fig. 4.2. Their work shows promising learning capabilities. However, they use fixed place cells to describe the environment and do not learn any efficient representations.

Weidel et al. [141] propose an actor-critic model with both a recurrent representation layer and a reward-modulated output layer. They learn the feed-forward connection to the representation layer using an unsupervised clustering method, and the output connections to the critic and actor population with a three-factor learning rule. They validate their model on the classic mountain car environment.

Recently, Anwar et al. [142] have created an impressive network for playing a modified pong game, shown in Fig. 4.3. As far as we know, they are the only ones solving 2D visual environments using a spiking reinforcement learning framework. They separated their network into three main regions, the visual cortex that transforms the frames into 8 motion directions and 1 spatial location, the association cortex that learns visual representations, and the motor cortex that outputs motor commands. The biggest limitation resides in the discretization in intervals of 20ms, and the rate-base coding of the frames to spike trains.

4.1.2 Intrinsic reward

One of the biggest limitations of reinforcement learning frameworks is the dependence on external rewards. In certain cases, this can be biologically plausible, such as animal experiments with humans delivering food as a reward when the action has been performed successfully. But in the case of learning to associate visual inputs with motor actions, this is not always clear how the brain creates such associations. There is good evidence that those cannot be learned solely from external triggers.

Jaegle et al. [143] showed that primates rely heavily on novelty and curiosity visual signals to generate interest and develop specific motor behaviors. Though this is more geared toward higher level behaviors such as food seeking for instance. It is believed that the human visual system is capable of generating intrinsic rewards based on how efficiently visual stimuli are encoded during eye gaze.

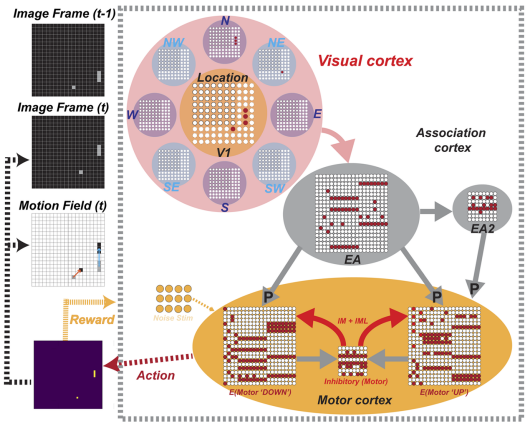


Figure 4.3: Actor-Critic spiking reinforcement learning for playing Pong by Anwar et al. [142]

The AEC framework is based on the efficient coding hypothesis. It stipulates that behaviors leading to more efficient coding of the information should be encouraged. In this framework, the reward is therefore generated directly from the efficient coding layer. Figure 4.4 presents a typical AEC architecture for vergence control. Work such as [6]–[8], [144] present a vergence control reinforcement learning framework. It is based on an intrinsic reward computed as the difference between visual patches from the left and right eye. However, their work uses traditional computer vision based on frame-based cameras.

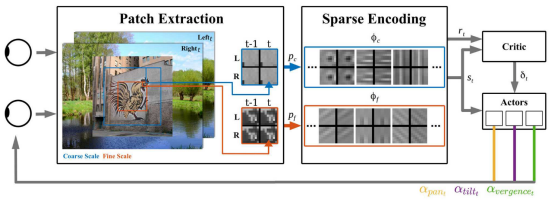


Figure 4.4: AEC vision architecture for vergence control by Lelais et al. [9]

More bio-inspired approaches can be found in [9]–[11]. They learn active binocular vision model based on sparse coding neurons on natural images. They compare scene statistics with their learned neurons receptive fields.

Similarly, Gibaldi et al. [145], [146] were able to learn vergence control of the eyes using an internal representation of the visual input, as shown in Fig. 4.5. They use disparity-tuned neurons close to the complex cells found in the brain to estimate object depth and design reinforcement learning strategies that use those cells to verge on the target object. Good gaze strategies are associated with efficient

visual encoding and are therefore reinforced over time. This is especially true for eye vergence, which brings both the left and right visual fields to produce very similar stimuli.

However, those methods do not work in the spiking domain and are not applied to event-based visual inputs. Furthermore, they do not learn the efficient coding part, using predefined filters found in the primate visual system. In a different domain, Chorley et al. [147] did try to predict dopamine signals from a competitive excitation/inhibition model, but do not link that to learning concrete visual strategies.

As far as we know, we are the first to propose a fully spiking reinforcement learning framework capable of solving control task while intrinsically generating the reward from its internal representation layers.

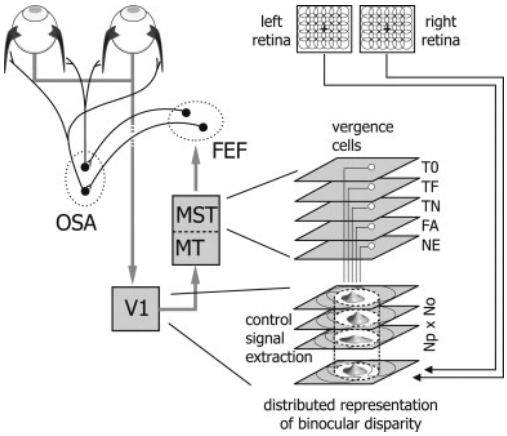


Figure 4.5: vergence control model from disparity-tuned neurons by Gibaldi et al. [145]

4.2 A fully spiking reinforcement learning framework

We gave some background on reinforcement learning and the discrete TD framework in Chapter 2. In this section, we present our reinforcement learning framework based on a TD learning actor-critic model working in continuous time with spiking inputs. Figure 4.6 presents our proposed AEC architecture.

4.2.1 Temporal difference error

Our reinforcement learning agent is based on a TD learning actor-critic framework formulated by [140]. They propose three different learning rules, namely TD-LTP, TD-STDP or r-max for the critic and actor population. We use the TD-STDP formulation since it is the one closest to our learning rule used in the efficient coding layer. We will note the differences in their framework as we explain it in more detail.

Contrary to the traditional discrete formulation, we use a fully spiking continuous reinforcement learning formulation. In a fully spiking environment, there are no real discrete steps of time t, or if there are (due to the limitation in temporal resolution

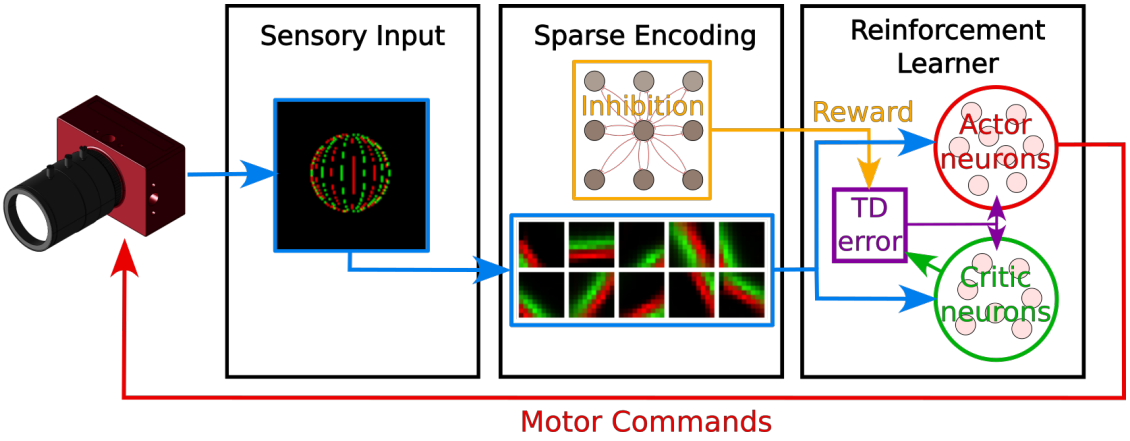


Figure 4.6: Proposed spiking AEC architecture.

of event-based cameras), they are too small to be effectively used. Therefore, we need to define a continuous TD error as follows:

$$\delta_t = \dot{V}(s_t) - \frac{1}{\tau_r} V(s_t) + r(s_t, a_t) \tag{4.1}$$

where $\dot{V}(s_t)$ is the derivative over time of the value function, and τ_r is the reward discount time constant that plays a similar role to the reward discount factor.

The TD error gives us an indication of how the network is performing. If the value function perfectly evaluates the expected result of a state action pair, the TD error should be equal to zero. Otherwise, it indicates if we overestimated the state action pair (negative δ_t) or underestimated it (positive δ_t).

4.2.2 Critic neurons

The value function estimation is essential for the agent to perform well. In SNNs, we cannot evaluate functions as easily as in traditional neural networks. Inputs are arriving continuously which forces us to look at the evolution of activity in a population of neurons instead.

The first layers of the SNN will serve as a state representation, i.e. their activation patterns represent the state in which the agent is. By connecting those cells to a population of other neurons, we can learn to associate specific neural activity with a state value function.

If we take one spiking neuron as a value estimator, we can define the value function as:

$$V(s_t) = \nu\rho(t) + V_0 \tag{4.2}$$

with ρ_t the firing rate of the neuron, V_0 the baseline for when the neuron has no activity, and ν a scaling factor.

With this equation, there is a linear relationship between neuron firing rate and value estimation. Since the firing rate of a neuron is always positive, we can obtain a negative value estimation by carefully selecting V_0 to be negative. This neuron will be called a critic neuron as it performs a similar task as the critic in the discrete formulation.

To make the system more robust, we use a whole population of such neurons. Therefore the equation (4.2) can be written as:

$$V(s_t) = \frac{\nu}{N_{critic}} \sum_{i=1}^{N_{critic}} \rho_i(t) + V_0 \tag{4.3}$$

We usually select $N_{critic} = 100$.

The firing rate of neurons evolves continuously over time. A simple option to evaluate it is to use a kernel function. We use an exponentially decaying kernel defined by:

$$\kappa(t) = \frac{e^{\frac{-t}{\tau_k}} - e^{\frac{-t}{\nu_k}}}{\tau_k - \nu_k} \tag{4.4}$$

with $\tau_k = 100$ ms and $\nu_k = 5$ ms.

Contrary to Fremeaux et al. [140] which were using the derivative of the critic kernel, we simply use a second-order numerical differentiation on the value to get an approximation of the value derivative $\dot{V}(s_t)$. This is because, in our framework, we had issues with instability in the derivative linked to the input natural variability of events rates. It can be written as follow

$$\dot{V}(s_t) = \frac{\eta_{\text{actor}}}{N} \sum_{i=N-t}^{t} \frac{V(s_{i+1}) - V(s_{i-1})}{2} \tag{4.5}$$

with η_{actor} a scaling factor and N the number of value points we take into account (since the simulation has a minimal time step of 1ms).

One limitation of the framework is that the value function estimation is directly proportional to the critic neuron spike rate. Neuronal spike rates are dependent on two major factors, the synaptic weight value of inputs, but also the amount of input itself. When submitted to many events, the network's simple and complex cells will inevitably spike more than when fewer events are present. This in turn will drive the critic neurons more, which will impact the value estimation. This is an important difference when compared to the work of Fremeaux et al. [140]. In their model, they ensure that the amount of activity in the population of state representation spiking neurons stays constant over time. But with event-based visual inputs, the amount of activity can vary a lot. We designed a few activity regulation mechanisms in our cells to limit those variations, but they were not sufficient when facing big variations in input rates.

We would want the value estimation to be dependent only on the values of the weights rather than the event rate. So a simple solution we found is to normalize the value function by the event rate itself. Cell rates in the network are highly correlated to the input rate, which allows us to normalize the event rate without introducing too much variation in the value function estimation.

4.2.3 Actor neurons

In a SNN, there are no regular time steps at which actions can be chosen from the output of a readout layer. But similarly to the critic neurons, we can assign spiking neurons to certain actions. In biology, neurons can trigger muscles they are connected to elicit fine-tuned movements. Fremeaux et al. [140] were using only one actor neuron per action, but to increase stability in the action selection, we assign multiple actor neurons to specific motor actions in our framework. We usually use 50 neurons per action.

As with the critic neurons, actor neurons are connected to the representation layers so that they can have access to the agent states' information. Then, we select actions at regular intervals by looking at the activity of those actor neurons. This process is somewhat of a simplification that is not very biologically plausible. We could envision selecting action only when a certain amount of activity has been registered in the actor neurons, but that would introduce a lot of variability in the control loop. The action selected is the one from the actor population that has the strongest activity. This is a simple WTA mechanism that can be found regularly in biological systems. For simplicity, we do not use a sliding window for computing the firing rate of actor neurons, but instead, simply count the number of spikes that happened since the last chosen action.

4.2.4 Three-factor learning rule

Learning associations between the representation layers and the critic and actor neurons is done using the R-STDP learning rule detailed in Sec. 2.3.8. In our actor-critic TD framework, the third factor term corresponds to the temporal difference error δ_t. Therefore, the weight update can be written as:

$$\Delta w_i(t) = \eta w_{ei}(t)\delta_t \tag{4.6}$$

Actions are selected at regular intervals predefined beforehand. We update both the critic and actor neurons weights every time an action is selected, using the last few TD errors before the last selected action. Importantly, we only update the actor neurons from which the previous action was selected since the TD error estimation arises from that specific action.

If the action contributed to increasing the TD error, then it was a positive one and we encourage the actor neurons association made with the representation layer. In the opposite case, a negative TD error will decrease the weights accordingly.

4.2.5 Exploration and exploitation strategy

Learning both the value and policy at the same time can be difficult. It is a classic problem in reinforcement learning. Biological systems face similar difficulties. A good policy can only be learned after an effective value approximation has been computed. Without any exploration, we risk getting stuck into invalid strategies or local minima if the policy converges before the value had time to properly explore the state space. To learn an effective policy, it is essential to make sure that a majority of states have been observed and associated with a correct value. To do that, we force the agent to first explore its environment, before slowly changing to an exploitation strategy. Managing exploration and exploitation is a classic problem in reinforcement learning. To allow the network to explore all the states at the beginning, the network selects random actions with a certain probability. As the learning progresses, we decrease that probability λ_{EXP}, which will force the agent to enter an exploitation phase. In that second phase, the agent has more time to fine-tune its policy to get closer to an optimal one.

Decay intervals We handle the change from exploration to exploitation using a simple decay mechanism. Every Δ_{decay} time interval, we decrease some of the learning parameters, including the exploration factor λ_{EXP}, action rate and critic and actor learning rate η following:

$$\Delta_{\eta} = \left(1 - \frac{\Delta_{\text{decay}} \eta_{\text{decay}}}{100}\right) \tag{4.7}$$

with η_{decay} the decay rate.

At the beginning of the learning, the value function has not yet learned properly. For that reason, it is preferable to reduce the actor learning rate slower than the critic learning rate, in order to give time to the critic neurons to learn a good value estimation first. We therefore use a smaller η_{decay} for the actor neurons.

Table 4.1 details all the cell parameters used in our framework. Table 4.2 presents the reinforcement learning parameters. We chose a discount time constant of one so that the value function update only depends on the previous state. We made this choice since our visual task do not require to take actions based on expected future rewards rather than the present one.

Figure 4.7 resumes our reinforcement algorithm along with some of the main equations.

Parameter	*Unit*	**Simple cells**	**Complex cells**	**Critic cells**	**Actor cells**
V_{thresh}	mV	30	3	2	2
V_{reset}	mV	-20	-20	-20	-20
$V_{\text{thresh}}(min)$	mV	4			
η_{LTP}	mV	0.00077	0.2	0.077	0.077
η_{LTD}	mV	0.00021	0.2	0.021	0.021
η_{INH}	mV	15	15		
η_{TA}	mV	1			
η_{RP}	mV	1	1		
η_{SRA}	mV	0.6			
η_{ILTP}	mV	0.77			
η_{ILTD}	mV	0.21			
τ_{m}	ms	18	20		
τ_{LTP}	ms	7	20	7	7
τ_{LTD}	ms	14	20	14	14
τ_{RP}	ms	20	30		
τ_{SRA}	ms	100			
S^*	$sp.s^{-1}$	0.75			
λ		4	10	4	4
λ_{lateral}		100			
λ_{topdown}		300			
η	mV			0.2	0.1
η_{decay}				5	1.66
ν_{k}	ms			5	
τ_{k}	ms			100	
τ_{e}	ms			250	250

Table 4.1: Parameters configuration for the network's cells in the reinforcement learning tasks.

	Action rate	Action rate (min)	λ_{EXP}	Δ_{decay}	V_0	τ_r	η_{actor}	ν	N
Unit	ms	ms	%	ms	mV	ms	mV		
Value	250	10	75	2000	-20	1	80	1000	100

Table 4.2: Reinforcement learning framework parameters.

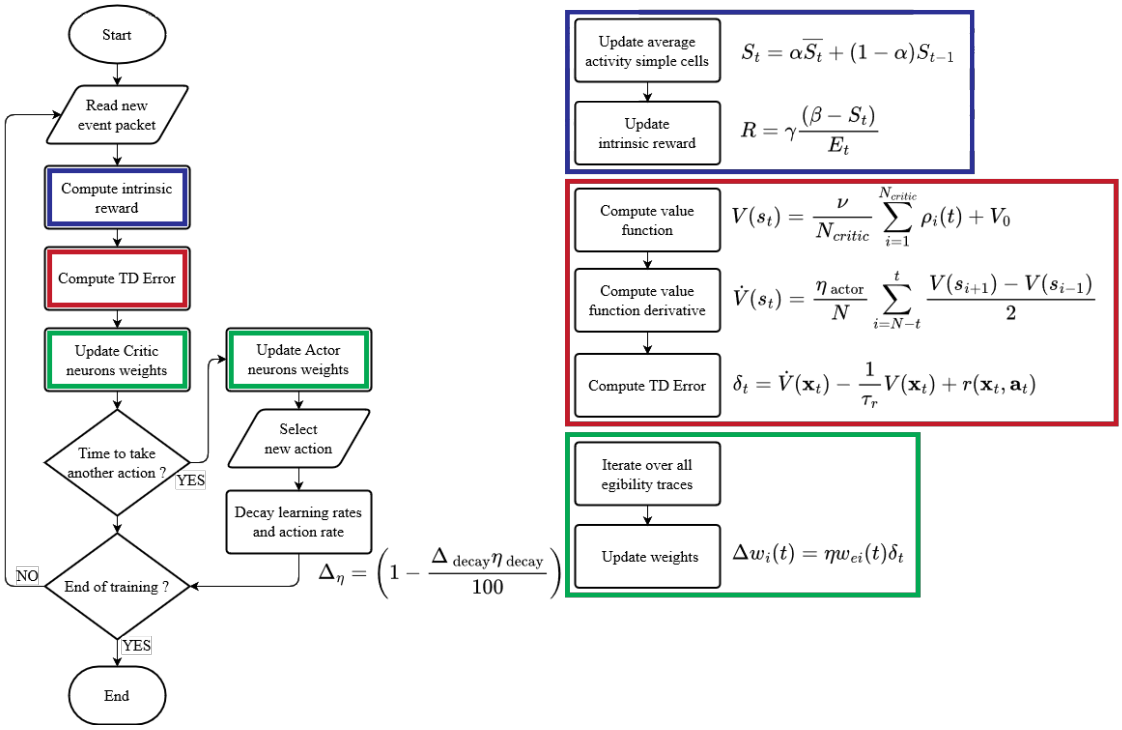


Figure 4.7: Flowchart of our reinforcement learning algorithm.

4.3 Intrinsic reward generation

Biological systems must be able to extract and refine reward signals from their environment to judge effectively the actions they undertake. In the case of visual stimuli, biological vision systems have to extract visual feedback linked to the most recent motor actions. This is not an easy task since it requires a profound understanding of the environment.

In the case of simple eye movements like tracking or vergence, the feedback does not require this high-level understanding. The changes in scene statistics are sufficient to produce an effective reward signal. For instance, verging two eyes on an object will highly correlate the left and right visual signals, which indicates that the action has been performed successfully.

Previous AEC models have used generative models for learning the sensory representation. This allows estimating the quality of the encoding via a reconstruction error. In contrast, our SNN is not generative: it does not try to explicitly reconstruct the input. Hence we cannot use a reconstruction error as the reward signal and need an alternative.

To address this problem, we designed an inhibition scheme that associates the quality of encoding with the relative amount of activity in the network. This is

achieved by learning lateral and recurrent inhibitory connections between the simple and complex cells. The rationale is that visual stimuli that are seen more often will trigger more inhibition. Therefore, the activity in the network will be significantly reduced when those stimuli are shown. We can then extract a reward that will be proportional to the activity of the encoding layers. High activity means a poor encoding of the visual stimuli, which is indicative of sub-optimal eye movements. On the contrary, low activity means good encoding and effective eye movements.

We use two different types of inhibitory connections, top-down connections from the complex cells to the simple cells, and lateral connections between the simple cells.

4.3.1 Top-down inhibition

The first inhibition scheme consists in connecting the complex cells back to the simple cells using plastic inhibitory connections. There are as many inhibitory connections as excitatory ones. When a complex cell spikes, it sends an inhibitory signal back to all the simple cells it is connected to, of which the strength is determined by the weight of the inhibitory synapse. The signal acts on the neurons by subtracting the value of the weight from the current membrane potential value. There is no delay in those connections, so the effects are immediate. To avoid extreme activity suppression, we cap the membrane potential of neurons to a minimum value, fixed at -80 mV. Biological neurons show similar capping mechanisms when submitted to intense inhibitory signals.

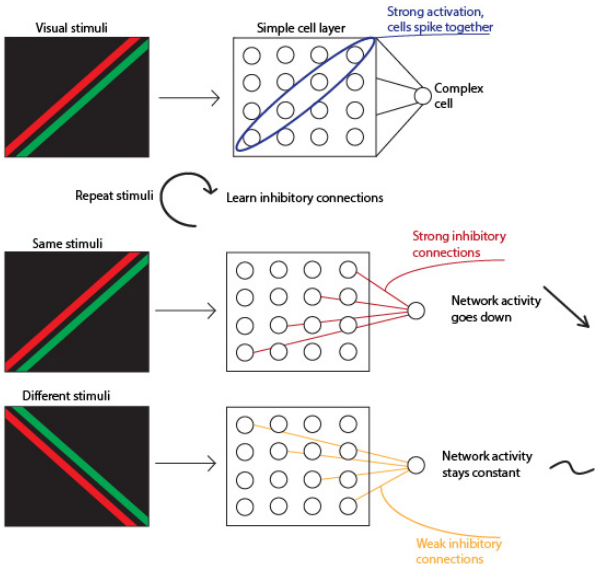


Figure 4.8: Top down inhibition

The learning is done similarly to the excitatory synapses. We use the STDP rule applied to the inhibitory connections. Simple cells receive the inhibitory spikes from the complex cells they are connected to and store them. Once a simple cell spikes, it updates the inhibitory connections associated weights using a similar STDP window as the one used for excitatory connections. We use the same value for τ_{LTP} and τ_{LTD}, but different learning rates, which are written as η_{ILTP} and η_{ILTD}. There is also a specific normalization factor for the inhibitory connections.

The primary effect of this inhibition mechanism is that cells that fire together on a specific visual pattern will simultaneously inhibit each other, and therefore reduce their cumulative spike rate. We illustrate this effect in Fig. 4.8. Take a moving edge, cells that are locally close together will be submitted to this pattern at the same time. This in turn drives their potential up and triggers the complex cells they are associated with. Those complex cells transmit immediate inhibition signals to the lower level of simple cells. Among those, the ones that continue spiking due to the moving edge simultaneously reinforce the inhibitory connection with the complex cells.

After presenting the same pattern enough times, the inhibitory connections become strong enough that they prevent the set of simple cells from spiking entirely, which reduces the average activity rate of the network.

4.3.2 Lateral inhibition

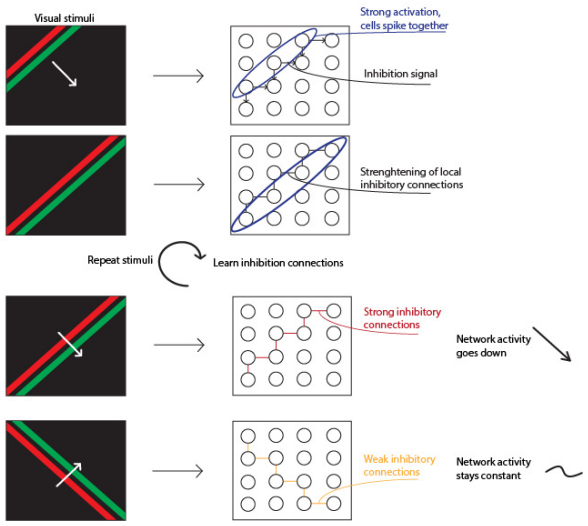


Figure 4.9: Lateral inhibition

The lateral inhibition scheme is similar to the top-down one, except it only acts from simple cells to other simple cells. The connections are designed so that one simple cell inhibits neighboring simple cells, contrary to static inhibition, which only works on cells of different neuronal maps sharing the same visual fields. Otherwise, we use the same general properties and learning rules as the top-down scheme, while simply reducing the learning rate and normalization factor since there will be many more spikes from simple cells than complex cells.

Intuitively, we can interpret this lateral inhibition mechanism as some form of predictive learning. When presented with a moving stimulus, like a moving edge, adjacent simple cells will activate very closely in time, in a very specific pattern. The moving edge first activates a set of simple cells, which in turn sends inhibitory signals to adjacent simple cells. Then, the edge moves and activates locally close simple cells, some of which just received the inhibitory signal from the previous simple cells. The inhibitory connections between those two are therefore reinforced. This phenomenon is illustrated in Fig. 4.9.

Repeat these visual patterns enough times, and at some point, those inhibitory connections will become very strong. Then, the simple cells become a predictor of the visual pattern. For instance, the moving edge, once it triggers the first set of simple cells, will strongly inhibit the simple cells associated with the next location of the moving edge. The pattern used to activate the simple cells, but due to the now strong inhibition, they remain silent. The total activity of the network for this visual pattern drops significantly.

If you show a different pattern, like a moving edge going in another direction, the network activity will be unchanged, since the first set of simple cells inhibits other simple cells that would not be triggered by this specific visual pattern. We effectively designed a prediction mechanism that can be used to learn to differentiate visual patterns and extract a useful reward from them.

4.3.3 Intrinsic reward from activity

The reward itself is directly generated from the simple cells' activity. To be precise, we compute a rolling average of the simple cell population activity every 1 ms as follows:

$$S_t = \alpha \overline{S_t} + (1 - \alpha) S_{t-1} \tag{4.8}$$

with S_t the activity rolling average, $\overline{S_t}$ the averaged sum of all simple cell spikes in that 1ms window, and α a discount factor that we set to 0.75.

Then, we compute the reward as:

$$R = \gamma \frac{(\beta - S_t)}{E_t} \tag{4.9}$$

with R the intrinsic reward and E_t the average event rate, which is computed similarly to the simple cell event rate. γ and β are simply scaling factors. We chose $\gamma = 5$ and $\beta = 90$ to obtain a reward close to the previously used extrinsic reward.

We divide the simple cell activity by the event rate to maintain a stable reward even when submitted to variable event rates in the scene. This will be useful in some of the reinforcement learning task since where events are generated by both camera and object movements. In some case, the relative motion of the object in the visual field will depend on the motor action compared to the object's direction. The speed of the camera is either added or subtracted from the object's speed. Since the number of events is heavily correlated to the speed of objects, each case will generate a different amount of events. This in turn will impact the spike rate of simple cells. By dividing by the event rate, we rectify this difference. We showed in Sec. 3.3.5 that the spike rate of cells is strongly correlated to the event rate. We can therefore safely normalize the simple cell activity by the event rate without introducing too much instability.

4.4 Application to tracking and visual field stabilization

We present in this section the validation of our framework on two different visual tasks. We tested the model with both an extrinsic and intrinsic reward and then compare the results.

4.4.1 Simulation of visual environment

In this thesis, we focus our attention on agents learning to solve a task in a simulated environment. We use CoppeliaSim to create diverse scenarios upon which the agent evolves and acts. It is a robotic simulator coupled with an efficient physics engine. In our case, we used Bullet 2.78 as the physics engine. We describe in more detail the simulation framework in the appendix (see Sec. C.2).

By default, CoppeliaSim is not capable of generating event-based camera output. to solve that issue, we capture frames at a very high rate using a small simulation time step of 1ms. This gives us a frame rate of 1000 images per second, which we then convert to event streams using an event-based camera emulator called PIX2NVS [148]. Even though this is not as precise as real event-based data, the emulator and frame rates are enough to produce accurate models of event streams. Since images are created every 1ms, we also jitter the events in time to more accurately simulate the output of a real event camera and avoid a cluster of events around frame timestamp generation.

We created 2 different environments to test our framework, each having specific properties and challenges.

4.4.2 Tracking task

The tracking task consists of one motorized camera with one axis of rotation. Figure 4.10 presents a screenshot of the environment. A textured ball is moving in a circle around the camera whose radius is 1.3 meters from the optical center of the camera. It is either moving clockwise or counter-clockwise. The goal of the task is simple, to be able to track the ball by keeping it in the center of the visual field. The visual field

For this task, we designed the following network architecture. The simple cell layer is composed of a thin strip of 30 neurons in width and 6 neurons in height. This gives a practical visual field of 300×60 pixels. We learn a common representation

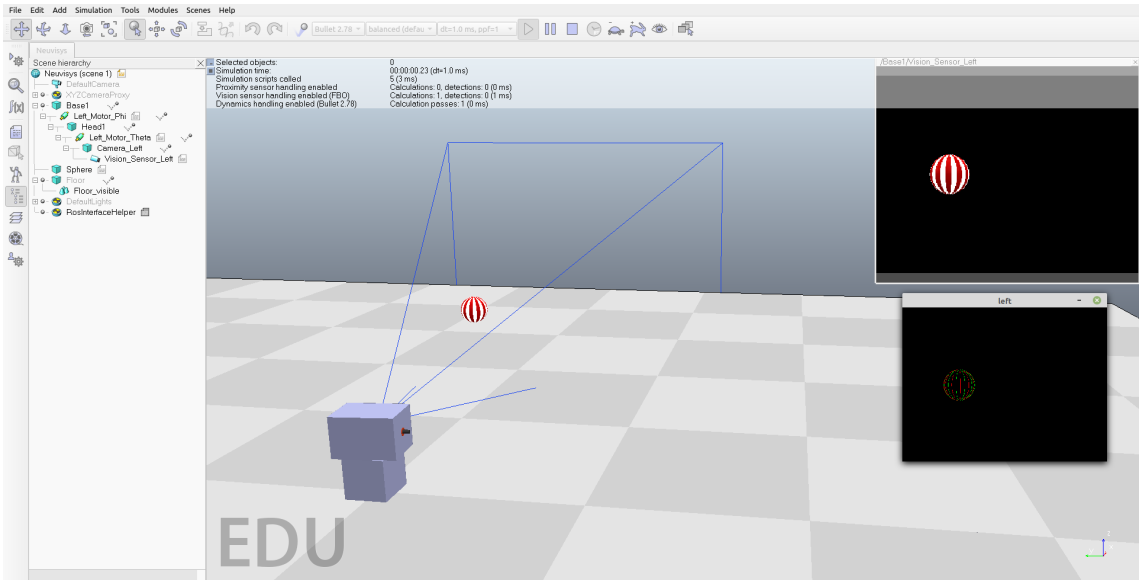


Figure 4.10: tracking environment, composed of one motorized agent (gray boxes) and one ball (with white and red textured stripes). The goal is to bring the ball to the center of the visual field, as seen in the top right of the image. Simulation images are sampled at high frame rates and then transformed into event streams, as seen under the visual field representation. The agent can either turn right or left in the horizontal plane.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	1	1	yes	30,6,64	10,10,2
Complex cells	static	none	1	1	no	10,2,16	3,3,64

Table 4.3: Network architectural parameters for the tracking task.

basis of 64 receptive fields. This corresponds to a total of $30 \times 6 \times 64 = 11520$ simple cells. Then, since each complex cell is connected to a 3 by 3 simple cell visual field, we cover the entire simple cell layer with 10 by 2 complex cells, plus there is 16 complex cell for each separate visual field. This gives a total of $10 \times 2 \times 16 = 320$ complex cells. This is detailed in Table 4.3.

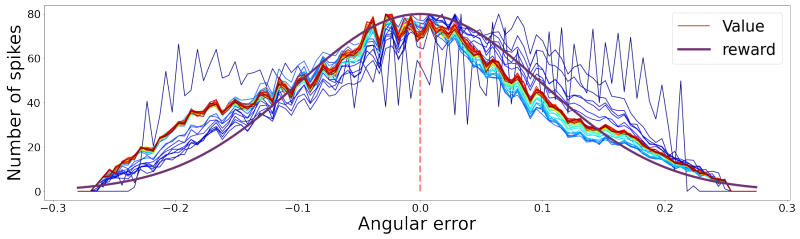
4.4.2.1 Learning with two actions

First of all, we are interested in seeing how the value and policy are developing over time. We focus on the simplest scenario, with only two possible actions, moving the camera to the right or the left. We used 100 critic neurons as well as 100 actor neurons, 50 for each action. Those cells are connected to both the simple and complex cell layers. In a first time, the reward is externally generated from the environment. When the ball is in the center, the reward is maximum and minimum when on the edges of the visual field.

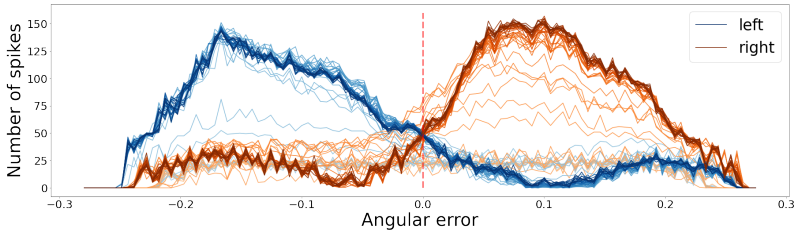
For this task, we first learned the efficient coding layer weights and fixed them to focus on the reinforcement learning framework. We learned a diverse basis similar to the one presented in Sec. 3.4.1.1. We start in full exploration mode, where every action is selected randomly. During that phase, only the critic neurons' weights are updated. Every 2 seconds of simulation time, we decrease both the exploration rate, critic and actor learning rate as well as the time between two action selections. We described the decay process in Sec. 4.2.5. As we continue decreasing the exploration rate, the actions selected are less and less random and the network focuses on fine-tuning its policy.

We recorded the weights of the network every 2 seconds in the simulation. Then, we observe the evolution of the network's performance in a simple validation task. The tracking environment consists in moving the ball back and forth once from the left to the right part of the visual field. For each location, we accumulate the spikes from both the critic and actor neurons to get a sense of the estimated value and policy. We submit the network to this validation for every recorded weight during training. We observe the variation of value and policy as the network learns the task in Fig. 4.11.

Fig. 4.11a shows the evolution for one representative run of the value function compared to the obtained reward over the course of the whole training. Evolution over time is represented by a shift in color from blue to red. In the beginning, the value function is flat and noisy, but very quickly, over a few simulation seconds, it learns to reflect the reward somewhat precisely. As the learning rate declines, the value stabilizes around the reward curve.



(a)



(b)

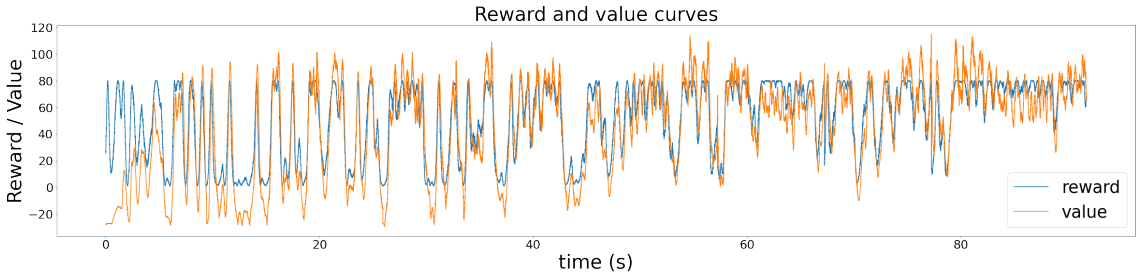
Figure 4.11: (a) Evolution of the value function during training for different ball positions. Evolution is represented from blue (early in the training) to red (late in the training). The reward is the curve in purple. (b) Evolution of agent policy during training for different ball positions. The policy is shown as the actor cell spike rates. Evolution is represented by the color intensity, from light tones to heavy tones. The left and right actions are respectively shown in blue and orange.

Fig. 4.11b shows the policy evolution during training. The blue and orange curves correspond respectively to the left and right motor action. If the ball is in the left part of the visual field, the agent must rotate the camera to the left to bring the ball back to the center. The opposite is true if the ball is in the right part of the visual field. The blue and orange curves show the actor's spike rates for the different ball positions in the visual field. As training progress, the network learns to select the right action depending on the position of the ball. The evolution of the policy over time during training is represented by the color intensity in the curves, from light to deep tones.

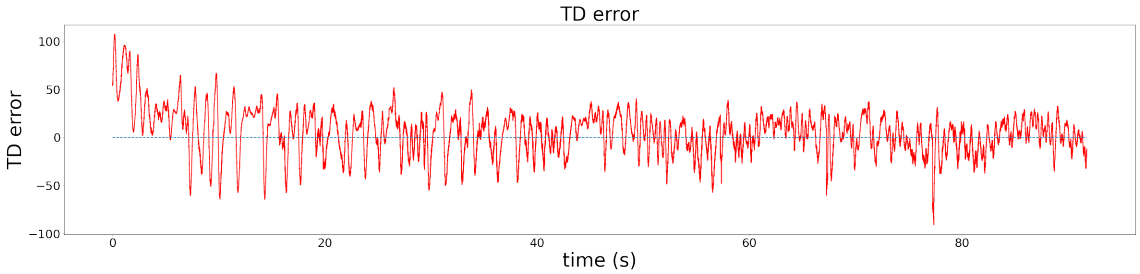
In this environment, the network was able to learn an effective policy in a very short time on this simple task. After only a minute of training, the network easily differentiates the different states of the ball and can select the correct action accordingly.

4.4.2.2 Adding a third action

To make the task a little more complicated, we added a third action, where the camera stops moving. The problem with adding that action is that if nothing is moving, we do not receive events anymore and the network stops working. It is not necessarily the case here since the ball is always moving. But without the speed of



(a)



(b)

Figure 4.12: (a) Reward and value function (blue and orange respectively) evolution during one training of around 100 seconds in simulation time. (b) TD error evolution during training.

the camera, the ball generates many times fewer events, which would impact the learning significantly. We solve that issue by adding constant jittering to the camera. This is similar to the eye microsaccades and ocular drift, which has been proven to increase visual acuity [149]. It moves left, right, up, and down in saccadic movements, following an Orsnstein-Uhlenbeck stochastic process that can be written as:

$$dx_t = \theta\,(\mu - x_t)\,dt + \sigma dW_t \tag{4.10}$$

where $\theta > 0$ and $\sigma > 0$ are parameters that controls respectively the attraction and drift component. μ is the central point to which the system goes back. W_t denotes a Wiener process, a stochastic value generator. The idea behind this jitter is that the camera will continuously drift from the central point μ due to the Wiener process while being attracted back to it over time.

With this jitter, we generate many events at all times, even when the camera stops rotating. The network can therefore work properly no matter the selected action. We train the network similar to the previous case. The only difference is the addition of the jitter and having 150 actor cells instead of 100 to take into account the new action.

Fig. 4.12a shows how the value function evolves during training for one example network over a span of 100 seconds in simulation time. We display the reward in blue and the value in orange. The reward follows a Gaussian distribution and is maximal when the ball is in the center. It is then equal to +80. When the ball is on

the sides, it is close to 0. At the beginning of training, the value starts at -30 and quickly rises as the critic neurons start receiving inputs from the simple and complex cells. Since it is far below the reward, the TD error is positive. This can be seen in Fig. 4.12b which displays the evolution of the TD error during training. This in turn drives up the critic neuron activity by increasing the weights through the R-STDP rule. In a few seconds, the critic neurons learn to associate a higher value for central stimuli, i.e. when the ball is in the center.

Once the value gets closer to the reward, the TD error decreases significantly and starts oscillating around 0. As the value efficiently predicts the reward, the oscillations reflect more and more the derivative part of equation (4.1). That is when the actor neurons start to learn to associate action with either a positive or negative appreciation. For instance, going left when the ball is on the left part of the visual field will bring it back to the center. This will drive the value function up and the TD error will be positive. The actor neurons learn to associate positive feedback between that specific ball state and action.

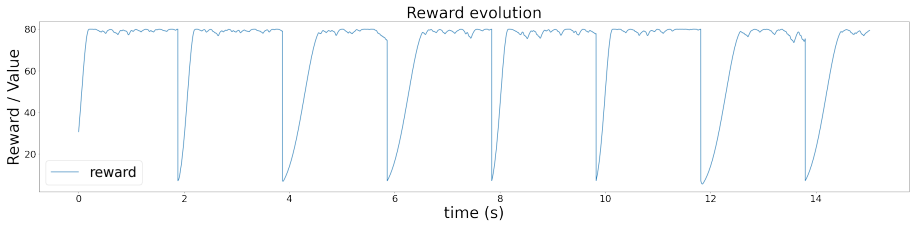
As training continues, the policy improves. Since the exploration rate decreases over time, we start to select the right action more and more, which in turn results in more effective behaviors. That is why the reward is closer and closer to the maximum at the end of training. The interval between two actions also decreases with time.

Since Fig. 4.12 can be difficult to read in detail due to the long training, we selected two smaller parts of the figure at both the very start and end of training. We present them in the appendix as Fig. A.4 and Fig. A.5.

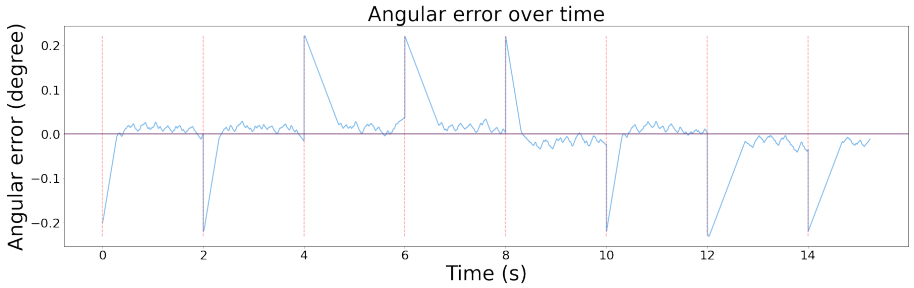
To estimate network performance, we tested the network's final policy in the simulated environment. The network selects actions every 10 ms, and every 1 second, we reset the ball to either the left or right border of the visual field. Fig. 4.13a presents the value and reward for this validation task during 15 seconds in simulation. Here it is clear the network has learned an efficient policy since it can get back quickly to the maximum reward every time the ball is reset.

The reward does not distinguish between the case where the ball is reset to the left or the right. To extend the validation results, we keep track of the angular error between the center of the visual field and the center of the ball. Fig. 4.13b shows the resulting error during 1 test. We observe that the network successfully brings back the ball to the center of the visual field every time the ball is reset, represented by a red dashed line. Then, the network keeps the ball in the center. We can note that the tracking is not perfectly stable, as there is some jittering around when trying to keep the ball in the center. We also observe a slight shift towards the left part of the visual field. But overall, the policy is effective.

We submitted the network to a similar validation scenario than when using two actions. The ball is going back and forth from left to right then right to left. We kept

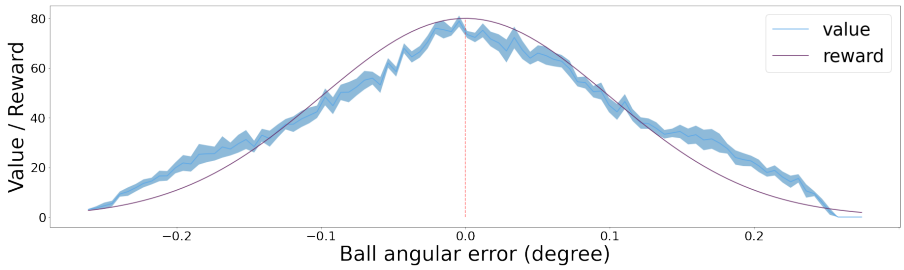


(a)

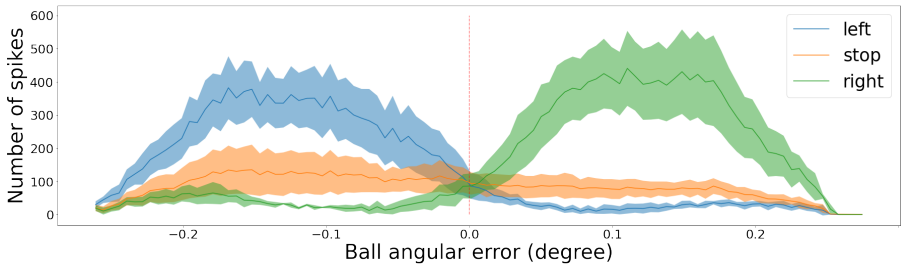


(b)

Figure 4.13: (a) Reward variation over time for the validation scenario on the tracking task (b) Angular error variation over time for the validation scenario on the tracking task with extrinsic reward. Red dashed bars represent the time at which the ball is reset to a random location.



(a)



(b)

Figure 4.14: Validation scenario for the tracking task with extrinsic reward. Mean and error bands of 1 standard deviation from 5 experiments with different starting seeds. (a) Reward and value function. (b) Action decision visualized from the activity of the actor neurons.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	1	1	yes	30,24,64	10,10,2
Complex cells	static	none	1	1	no	10,8,16	3,3,64

Table 4.4: Network architectural parameters for the 2D tracking task.

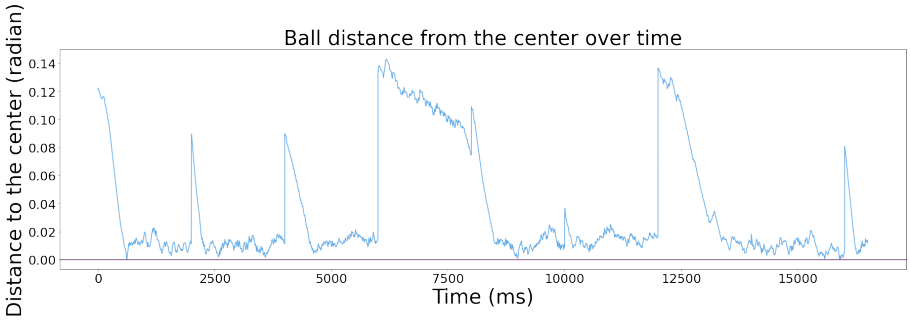


Figure 4.15: Euclidean distance between the center of the visual field and the center of the ball for the 2D tracking reinforcement learning task after learning on an exploitation test scenario. The ball is reset every 2 seconds to a random location in the visual field, represented by red dashed vertical lines.

track of the spike train of both the critic and actor neurons and created a histogram of activity based on the angular error from the ball to the center of the visual field. Fig. 4.14a presents the critic histogram in blue compared to the reward in purple. We can see that the activity of the critic neurons follows closely the external reward. Fig. 4.14b shows the activity of the 3 subgroup of actor neurons, 1 for each action. When the ball is on the left part of the visual field, the actor neurons associated with the turning left action to spike the most. Oppositely, the actor neurons linked to the turning right action spike more when the ball is on the right part of the visual field. Finally, there is more uncertainty in the middle. On average, the stop action is spiking more than the others, but not by much. This turned out to depends on the learning, with some scenario favoring the stop action in the middle while not in others. A solution to that could be to try different reward shapes in hope that it incites the network to decrease the left and right action even more when the ball is in the middle. Nevertheless, it shows that the learned policy is sensible and allows the network to efficiently track the ball in any situation.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static	none	1	1	yes	16,16,144	10,10,2
Complex cells	static	none	1	1	no	4,4,16	4,4,144

Table 4.5: Network architectural parameters for the stabilization task.

4.4.2.3 Tracking in 2D

We extend the tracking task by adding another dimension to the environment. The ball can now move up and down in addition to the previous left and right movements. Since the camera is fixed, the ball is tied to a sphere around the camera with a 1.3-meter radius. The ball is given a random 2D direction and speed that is changed every 2 seconds. The speed is comprised between 0 and 23 degrees per second. If the ball exits the visual field, it is reset on the other side of it. To solve that task, more actions are therefore necessary. We added the up and down camera action, as well as 100 more actor neurons (50 actor neurons per action for a total of 200). We limited the actions to one of the 4 main directions, without any combination between them. We also added some simple and complex cells to cover most of the visual field, since the ball can now move freely in both dimensions. This adds up to a total of $30 \times 24 \times 64 = 46080$ simple cells and $10 \times 8 \times 16 = 1280$ complex cells. Table 4.4 recaps the network architecture.

With a similar training procedure, we trained our network and tested it on a full exploitation scenario. Fig. 4.15 presents the Euclidean distance between the center of the ball and the center of the visual field during one of these tests. We observe that the network can bring the ball back to the center somewhat consistently, and then keep it there. The ball is reset every 2 seconds in a random location in the visual field. Only once the ball was not brought back in the center before the reset. This shows that it is possible to extend our work to more complicated environments with higher numbers of states and actions.

4.4.3 Stabilization task

The stabilization task also consists of one motorized camera. The environment is shown in Fig. 4.16. This time, the rotation is performed around the camera's optical axis. This task tries to simulate in a simplified way the process of a flying insect that would have to keep its flight horizontal using visual cues. We generate the

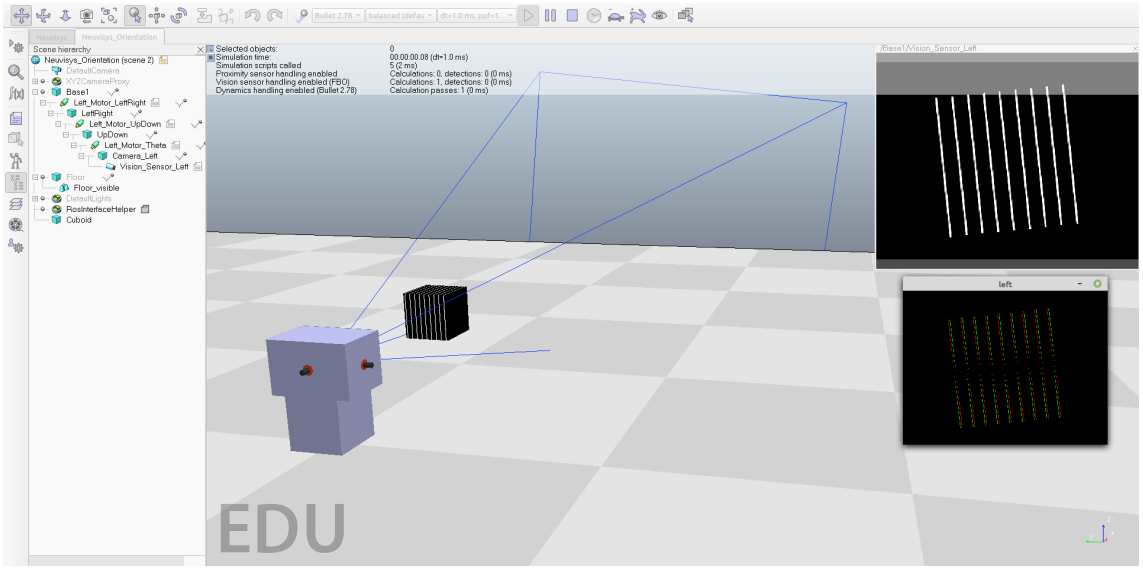
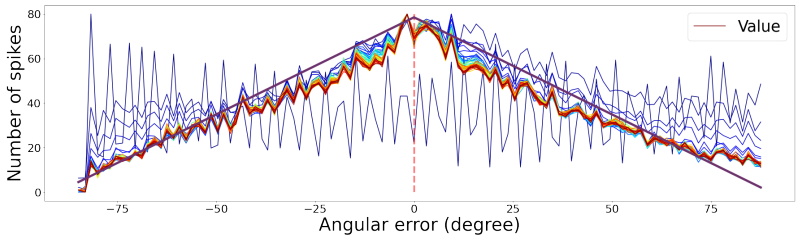
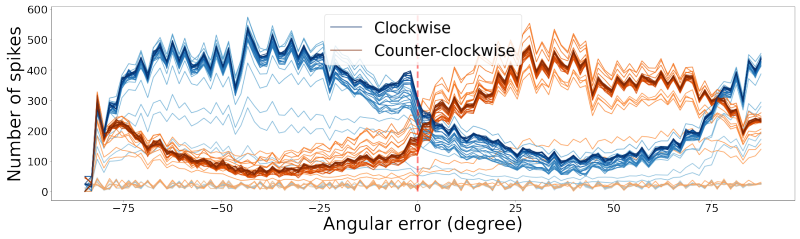


Figure 4.16: Stabilization environment, composed of one motorized agent (gray boxes) and a grating stimulus consisting of white bars on a black background. The goal is to bring the bars to a horizontal position by rotating the camera around its optical axis.



(a)



(b)

Figure 4.17: (a) Evolution of the value function during training for different bar orientations. Evolution is represented from blue (early in the training) to red (late in the training). The reward is the purple curve. (b) Evolution of agent policy during training for different bar orientations. The policy is shown as the actor cell spike rates. Evolution is represented by the color intensity, from light tones to heavy tones. The clockwise and counter-clockwise actions are shown respectively in blue and orange.

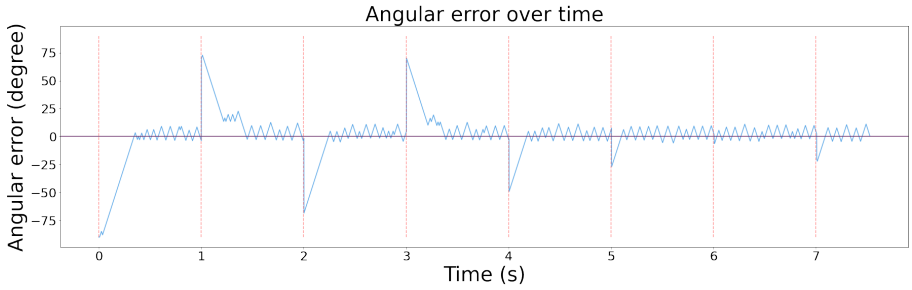


Figure 4.18: Angular error variation over time for the validation scenario on the stabilization task with extrinsic reward. Red dashed bars represent the time at which the grating is reset to a random orientation.

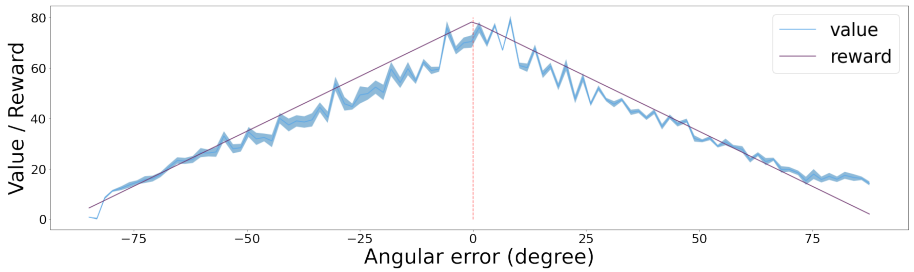
visual cues from a set of straight bars. A leveled flight corresponds to the bars being horizontal. We derive the reward from that fact, with horizontal bars giving the maximum reward, vertical bars a zero reward, and linear interpolation in between.

This task is harder than the previous one since the network must be able to process the bar orientation to effectively distinguish between states. We demonstrated in Sec. 3.3.7 that the network cells can efficiently differentiate between oriented stimuli, which makes them effective state representation for such a task. The actions are rotating clockwise or counter-clockwise to bring back the bars to the optimal state.

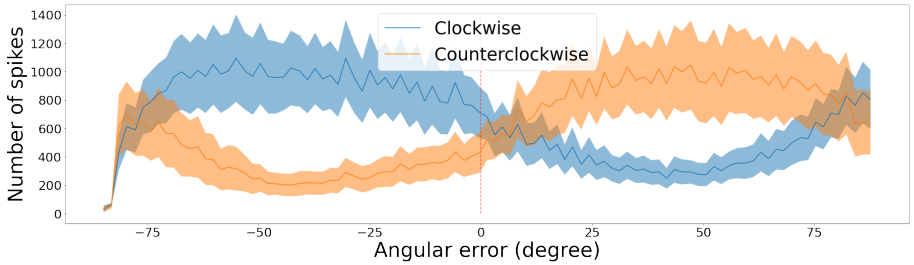
We changed the neurons' disposition for that task to be able to fit the entire visual stimulus in the visual field of the network. The stimulus is located in a square in the center of the visual field. We use $16 \times 16 \times 144 = 36864$ simple cells, which gives a visual field coverage of 160 by 160 pixels. Complex cells are connected to patches of 4 by 4 simple cells, corresponding to a layout of $4 \times 4 \times 16 = 256$ complex cells. There are 100 critic and actor neurons connected to both the simple and complex layers. This is detailed in Table 4.5.

As for the tracking task, we learned the efficient coding layer independently first. We also use a similar exploration and exploitation learning strategy. We saved the weights at regular intervals during training and present the results in Fig. 4.17. Fig. 4.17a demonstrates that we learn an efficient value function during training. Very quickly, the value raises to reflect the reward in black. Similarly, Fig. 4.17b shows that the actor neurons learn to separate the states as training progress.

We also recorded the angular error between the actual and optimal bar orientation (horizontal) during an exploitation test scenario on the fully trained network. Actions are selected every 10 ms. Fig. 4.18 presents those results. The bar orientation is reset every second at a random orientation. We can observe that the network can bring the bars to the right place no matter the initial orientation. Once in the right orientation, the network can keep the bars horizontal most of the time, even though it presents a small oscillating behavior.



(a)



(b)

Figure 4.19: Validation scenario for the stabilization task with extrinsic reward. Mean and error bands of 1 standard deviation from 5 experiments with different starting seeds. (a) Reward and value function. (b) Action decision visualized from the activity of the actor neurons.

Then, similarly to the tracking task, we designed another simple validation scenario to observe the value and policy of the network when changing the bar orientations. We start with bars at -90° orientation (vertical in our case), then rotate them to +90° and back again to -90°. That way, the network observes all possible orientations in both rotating directions. During that test, we deactivated the actions and recorded the spike trains of the critic and actor neurons.

Fig. 4.19a shows the activity histogram of the critic neurons compared to the extrinsic reward. We averaged the clockwise and counterclockwise rotations together for more accuracy. The graph demonstrates that the value function learned is very close to the reward. Fig. 4.19b presents a similar representation but for the actor neurons. We can see the switch in which actor neurons spike the most around the horizontal orientation (0°). We note that the decision boundary is slightly shifted to the right, which can also be observed in Fig. 4.18. This shows that our spiking reinforcement learning framework work well overall, but might lack resilience for learning a precise policy on more complicated visual tasks.

One possible improvement would be to add a third action that stops the rotation, similarly to the tracking task. But we run into some difficulties with the jittering and the number of events generated in this specific stabilization task, which made the

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static, lateral	1,1	1	1	yes	30,6,64	10,10,2
Complex cells	static, topdown	none	1	1	no	10,2,16	3,3,64

Table 4.6: Network architectural parameters for the tracking task with intrinsic reward.

learning difficult. We will focus instead on learning the same task while generating the reward intrinsically.

4.5 Intrinsic reward through inhibition

We demonstrated that our network can learn to solve a task in a simulated environment. However, we were using perfect extrinsic rewards until now to do so. To generate the reward intrinsically, we rely on the ability to tune cell activity variation with the use of inhibition. We will present a scheme that naturally reduces network activity for frequently observed stimuli. To demonstrate that, we learn with specific input statistics and the results show that the activity varies with the frequency of observed patterns.

4.5.1 Spatial inhibition

In the first experiment, the goal is to inhibit visual patterns appearing at the center of the visual field when using a diverse basis of oriented receptive fields similar to the one used for the tracking task described in Table 4.6. We fixed the excitatory weights and focused on learning the top-down and lateral inhibitory connections.

The AEC hypothesis implies it is possible to learn basic behaviors by shaping the efficient coding component to the input stimulus. For that reason, we want to reinforce efficient coding by encouraging reduced activity on frequently shown stimuli. This simple effect can lead to efficient behaviors well adapted to solving visual tasks, such as tracking. We are using the tracking simulation environment as a way of generating stimuli. We showed the network a repetition of a short recording that involves the ball moving back and forth around a specific part of the visual field. STDP works by an exposition through repetition. It is also true for inhibitory

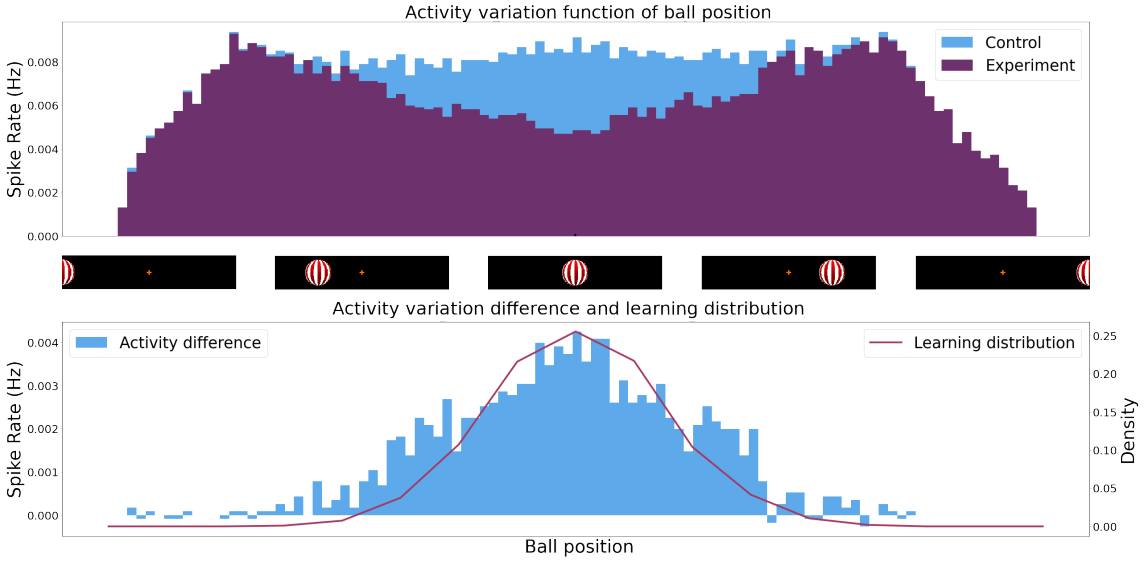


Figure 4.20: Activity variation of the network when presented to oriented gratings from 0 to 360°. In purple, the experiment network after learning the inhibition on a Gaussian distribution of oriented gratings centered around 0°. In blue, the control network is the experiment network with shuffled inhibition weights. The bottom graph presents the Gaussian distribution used for learning in red superimposed on the activity difference between the control and the experiment.

connections. Therefore, to inhibit central visual stimuli, we need to expose the network to a majority of visual patterns happening in the center of the visual field. For that purpose, the position of the ball in the visual field is selected according to a normal distribution whose mean is located at the visual center. That way, we make sure the network is mostly exposed to visual inputs that will excite the center cells, which in turn will drive up the inhibitory connections between those cells. With enough recordings, the network starts to learn to strongly inhibit input at the center of the visual field, while not so much on the borders of the visual field.

Fig. 4.20 presents the result of a single recording of the ball moving from the left to the right of the entire visual field. We recorded the total cell activity and present it as a histogram of activity variation over time. In blue, we have the control network, which is the network with shuffled inhibitory weights. In purple, the experiment network with inhibitory weights learned on the distribution mentioned above. The distribution can be seen in red in the bottom graph. The latter also presents the activity difference between the control and experiment network.

The activity of the experiment network drops significantly when the ball approaches the center of the visual field. More importantly, the drop is gradual. This means we can use the activity of the network as an effective intrinsic reward signal. Low activity means a high reward, whereas high activity means a low reward. The activity of the networks drops when the ball exits the visual field on the right or

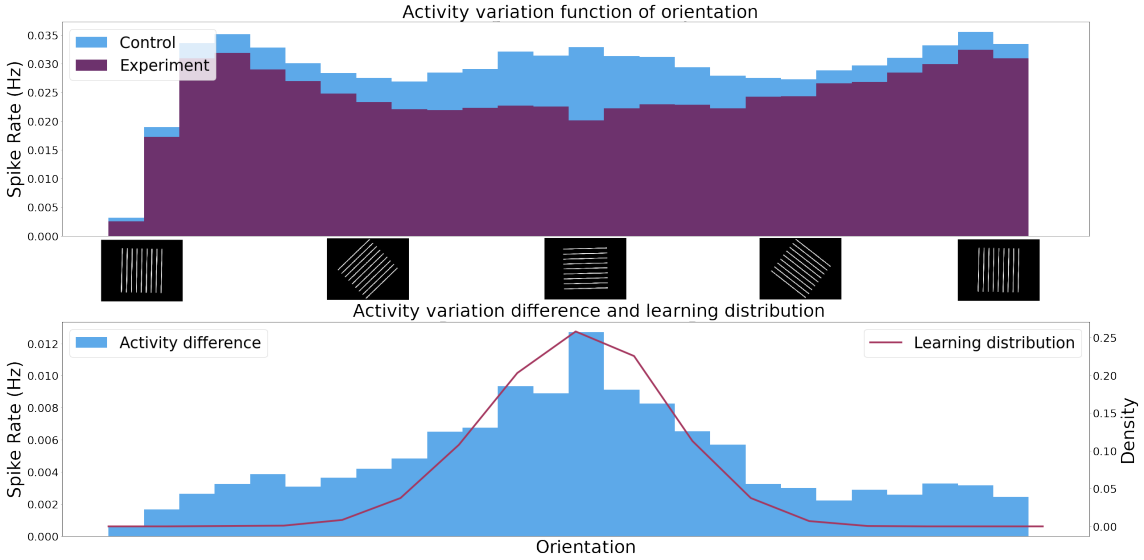


Figure 4.21: Activity variation of the control and experiment network when presented to a moving ball from the left to the right of the visual field. In purple, the experiment network after learning the inhibition on a normal distribution of ball recording centered on the middle of the visual field. In blue, the control network is the experiment network without inhibition weights. The bottom graph presents the normal distribution used for learning in red superimposed on the activity difference between the control and the experiment.

left since there are much fewer cells to excite in those regions. This does not pose a problem since we normalize the intrinsic reward by the number of events.

4.5.2 Inhibition on oriented patterns

For the second experiment, we focus on observing various ranges of oriented patterns and learning inhibitory connections on them. We use the stabilization task environment to generate the stimuli from rotating bars. We use the same network as the one for the stabilization task described in Table 4.7 while adding the lateral and top-down inhibition.

	Inhibition types	Lateral inhibition range	Nb synapses	Nb visual regions	Weight sharing	Nb cells per region	Size receptive fields
Simple cells	static, lateral	1,1	1	1	yes	16,16,144	10,10,2
Complex cells	static, topdown	none	1	1	no	4,4,16	4,4,144

Table 4.7: Network architectural parameters for the stabilization task with intrinsic reward.

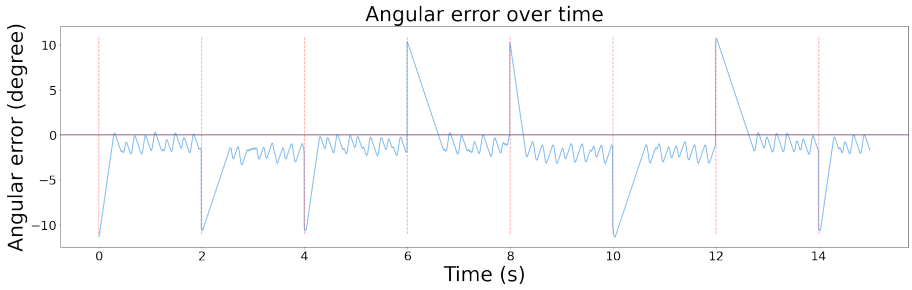


Figure 4.22: Angular error variation over time on the validation scenario for the tracking task with an intrinsic reward. Red dashed bars represent the time at which the ball is reset to a random location.

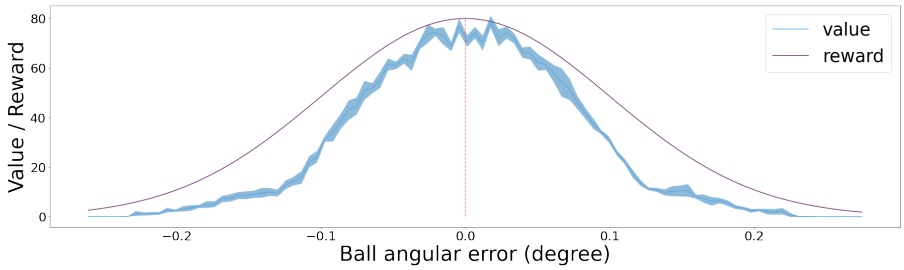
We present the network to a normal distribution of oriented gratings, centered around the horizontal orientation. The first graph in Fig. 4.21 shows the network activity as a function of grating orientation. The distribution of orientation is shown in red in the bottom graph, along with the activity difference between the experiment and control conditions.

We observe that the activity of the network decreases significantly close to the center of the distribution, that is for horizontal stimuli. This is represented as a bigger activity difference between the control and experiment network. Furthermore, the decrease in activity is gradual as we change the orientation. This means we can extract a smooth reward directly from the network activity and use it in our reinforcement learning. However, the control naturally presents less activity for oblique orientations, while the input event rate is mostly constant. This makes it difficult to generate a proper intrinsic reward, which is reflected by a almost flat activity near the horizontal orientation in the experiment.

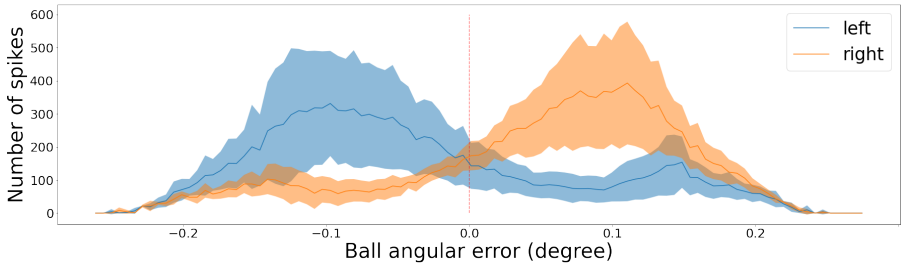
4.5.3 Tracking task with intrinsic reward

Now that we designed a network capable of generating an intrinsic reward, we train it on the tracking reinforcement learning task. The procedure is similar to the one in Sec. 4.4.2 with the two actions. We generate the intrinsic reward according to equation (4.9).

After learning, we test the network performance similarly to the extrinsic reward scenario. Fig. 4.23 presents the network after training. Compared to the task with extrinsic reward, we present here only the results after training. Fig. 4.23a and 4.23b show that the network has successfully learned to extract an effective value and differentiate between the two actions. Compared to the extrinsic reward, we can observe larger error bands, meaning that some learning will lead to slight inaccuracy when the ball is in the center. But on average, the network is still able to correctly separate the actions decisions properly. In Fig. 4.22, we can see that the network is



(a)



(b)

Figure 4.23: Validation scenario for the tracking task with intrinsic reward. Mean and error bands of 1 standard deviation from 5 experiments with different starting seeds. (a) Reward and value function. (b) Action decision visualized from the activity of the actor neurons.

still able to correctly keep the ball in the center, but this time with a small positive angular bias and generally more oscillations compared to the extrinsic reward case.

4.5.4 Stabilization task with intrinsic reward

Similarly to the tracking task, we learned the network with an intrinsically generated reward on the stabilization environment. Fig. 4.24 shows that the network can keep the level horizontal, but with a bigger error and more pronounced oscillating behavior than with the extrinsic reward case. This time, the value function flattens a bit when

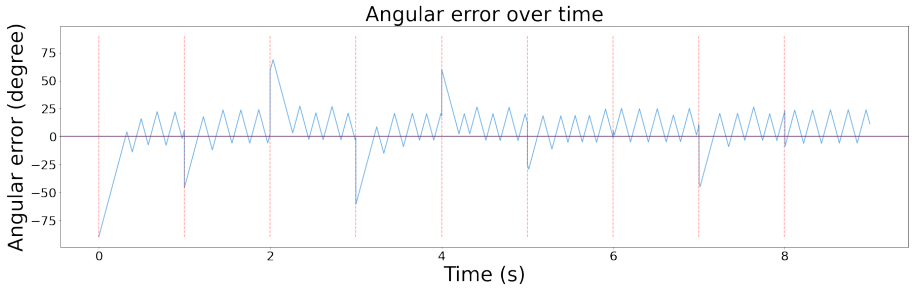
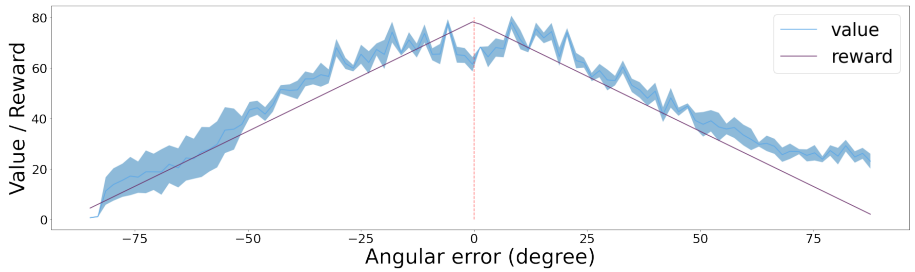
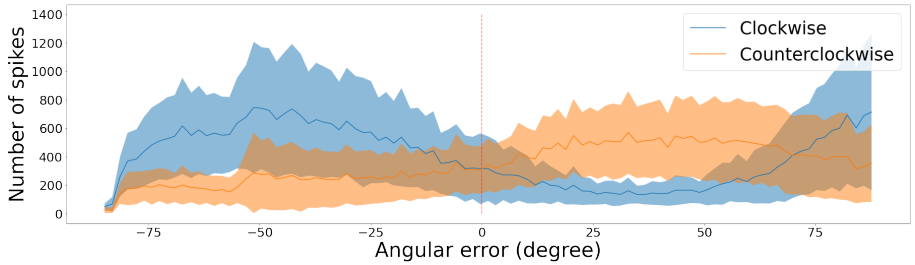


Figure 4.24: Angular error variation over time on the validation scenario for the stabilization task with an intrinsic reward. Red dashed bars represent the time at which the grating is reset to a random orientation.



(a)



(b)

Figure 4.25: Validation scenario for the stabilization task with intrinsic reward. Mean and error bands of 1 standard deviation from 5 experiments with different starting seeds. (a) Reward and value function. (b) Action decision visualized from the activity of the actor neurons.

presented with horizontal orientations as can be seen in Fig. 4.25a. This is most likely due to the intrinsic reward being also quite flat for horizontal orientations, as discussed in Sec. 4.5.2. This causes in turn indecision on the optimal policy around the horizontal orientation, which led to a shift in the action decision, as shown by Fig. 4.25b.

Conclusion

We addressed in this chapter how we were able to solve simple visual tasks using a fully spiking reinforcement framework derived from a more traditional discrete TD formulation. Reinforcement learning with SNN is still for the most part in its infancy. In traditional frame-based reinforcement learning, images are a dense representation that arrives at discrete time steps. Event streams on the other hand make it difficult to learn since they have a variable rate and are continuous in time.

We propose a framework that can overcome those challenges based on the activity of a specialized population of neurons, the critic, and actor cells. It is a versatile model that learns through the combination of unsupervised learning of representation

triggered by a third-factor reward signal. We presented the results on dense reward functions, but the network can theoretically also work with sparse rewards. We demonstrated that the framework can learn efficient policies when combined with our efficient coding model. The critic and actor neurons adapt their neural representation to associate visual states with both a value and policy.

We then addressed one of the main limitations of today's work in artificial intelligence, the reliance on some form of external supervision. Reinforcement learning frameworks are already a great step in that regard compared to supervised learning methods. But they still require a reward signal to operate, often based on human assumptions, which limits their self-sufficiency. We propose a model that self-generates a reward signal from the efficient coding layer. Based on a dual inhibition scheme, it learns to associate sparser code to frequent stimuli, which in turn drives the network's activity down.

We first presented our work using an extrinsic reward on two visual tasks, namely, object tracking and visual field stabilization. We demonstrated that the network can learn to distinguish between the visual states and select the correct action accordingly. Then, we swapped the extrinsic reward with the intrinsically generated one and compared the results. We showed that the drop in performance is small.

Future work will focus on learning the actor-critic neurons and the inhibitory connections for the intrinsic reward generation at the same time. This would offer a more realistic model of how the visual system could develop behaviors based on visual inputs fully autonomously.

Chapter 5

Discussions and Perspectives

5.1 Conclusions and discussions

We have presented a fully spiking framework based on the AEC model. Everything works in the spike domain, from visual sensing with event-based cameras to motor commands, using a combination of a SNN for the efficient coding part and a spiking reinforcement learner that is a continuous formulation of traditional discrete models. The network is self-sufficient; it does not require external supervision. It learns efficient visual representations in an unsupervised manner while tuning its behavior from an intrinsically generated reward based on a frequently observed stimulus.

Once put together, all the parts of our model can sense visual information in a scene, extract some of its essential components while sparsifying it simultaneously, and learn efficient policies for solving complex visual tasks. We demonstrated the ability of the network to track an object precisely or understand relative orientation information and stabilize its visual field accordingly.

Efficient coding of event-based visual inputs Raw visual data contains much information but is very unstructured and cannot be effortlessly processed by any

system without a pre-processing scheme. Event-based cameras remove much redundant information contained in illuminated scenes. They filter static components while extracting the edges of objects. They act similarly to the human retina by pre-processing much information, focusing on the scene's dynamic movements. However, due to their high temporal precision, they still output a heavy stream of information that needs further processing for optimal results.

We propose a simple dual layered SNN architecture inspired by simple and complex cells in the human visual system that can efficiently encode event-based data. It can extract features from the event stream, such as orientation, motion, or even disparity. It also reduces the amount of information by sparse coding the input. More importantly, this stage is capable of precisely tuning to the properties of the scene and adapting in case those statistics change over time. We demonstrated that on various test case scenarios, especially for driving sequences, presenting exciting challenges and widely different visual statistics. The real learning is done in an unsupervised manner using a modified STDP rule. Finally, the network is an effective tool for encoding visual information. We showed that it is possible to distinguish visual patterns by looking at the neuronal activation in the network. It is an ideal first stage for more advanced blocks, such as a classifier or a reinforcement learner.

Solving closed-loop visual motor tasks Based on our first contribution, we tackled learning motor controls for visual applications. Solving closed-loop visual tasks with a fully spiking environment is challenging. Most computer vision applications rely on well-established discrete reinforcement learning algorithms. Nevertheless, learning to control in the spiking domain is hard as time continuity challenges traditional frameworks. We demonstrated a spiking reinforcement learner able to tackle simple but tangible visual applications such as tracking or stabilization. We do not rely on substantial visual databases and minimize the need for external supervision. We presented scenarios with an external reward and a way to control by intrinsically generating the reward from the efficient coding layer.

It is a novel method that relies on learning to encode and sparsify frequent visual patterns more than infrequent ones. With time, the network activity goes down when exposed to those patterns as it has learned to inhibit them strongly. In turn, it reinforces the agent to learn a policy that converges towards those frequent visual patterns. The network consequently learns to solve the visual task by associating the policy with the final objective, such as keeping a ball in the center of the visual field or the horizon leveled.

5.2 Perspectives on improvements

Improving on the limits of biological inspiration Many parts of our model take inspiration from the human visual system, such as the simple and complex cells or the three-factor rule similar to dopamine reward signals. However, our model is only partially biologically plausible. We can note limitations in many areas where the need for simplicity exceeded the will to stay close to biological systems. For instance, synaptic connections always present a delay due to chemical and physical constraints in the brain. Instead, we use instantaneous synaptic transmission in most scenarios. Also, inhibition in a biological system is always performed by specialized cells, whereas we use inhibitory connections without complicated dynamics of their own.

Concerning the reinforcement learning part, the framework could be more realistic too. Neuronal systems for control are often much more complicated than our simple actor-critic model. In our case, action decisions are based on an external clock rather than a neural system. Moreover, many computations such as value estimation, kernel convolution, or spike counting are done directly on CPU rather than using neuronal models. We wanted to take inspiration from biological systems, especially the human visual system, not to create a perfect replica of it. Biological limitations are, therefore, to be expected due to computing and time constraints. Nevertheless, it is possible to reuse our model and update it to make it more realistic and in harmony with biology.

Joint learning of the different components Another limitation of our model is how coupled the learning is. In order to work progressively and keep the tasks manageable, we mostly separated the learning phases. We often learn the efficient coding representation first, then the reinforcement learning actions. It helped us correct errors in each component separately. However, the framework of the AEC stipulates that the visual representation and control policy are optimized jointly. While theoretically possible in our framework, it would require more work and testing to ensure that the joint learning happens appropriately and can converge to an effective solution.

5.3 Perspectives on future applications

5.3.1 Extension of our framework

Extension to stereo-vergence applications Future work will focus on autonomously learning the self-calibration of spike-based active binocular vision sys-

tems. We want to extend our approach to the simultaneous learning of disparity representations and vergence eye movements in a fully spiking implementation. For that, we will have to extend our work on learning efficient disparity representation in the efficient coding layer and use that as a basis for the reinforcement learner. By distinguishing between different disparities, the agent could understand how far the two eyes are from optimal vergence, producing zero disparity in the target visual input. By showing frequent stimuli with zero disparity, the network would learn to encode that pattern well and reduce its activity. In turn, the reward would increase for zero disparity inputs and encourage the reinforcement learner to learn a vergence policy.

Implementation on neuromorphic hardware At the moment, our framework works on a standard CPU. It is not an optimal solution as traditional computing hardware is not made for parallel computing, such as in a SNN. Each neuron is an independent unit that can be processed on its own. GPUs could be more adapted, as they possess many cores able to perform parallel operations. We thought about implementing our network on GPU, but this would have required a lot of time and effort. A better solution would be the use of neuromorphic hardware. They are perfectly adapted to work with SNN and offer impressive computing time and power performance. Implementing our model on a neuromorphic chip would speed up computation by a huge factor and allow us to learn to solve a visual task in real-time and extend it to real-world applications. It could also facilitate scaling up our approach to much larger network sizes.

5.3.2 Application to robotics

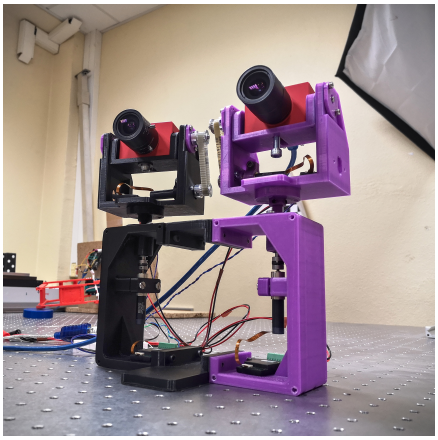


Figure 5.1: Pan-tilt robotic head unit.

We demonstrated the effective use of our model either on recorded event-based data or on a simulator. The next logical step would be to demonstrate our framework on real-world applications using robotic equipment. Both of our simulated visual tasks could be implemented on robotic hardware.

Tracking, stabilization, and vergence control We built a pan-tilt platform capable of holding two event-based cameras to apply our model to a real-world scenario. Similarly to the human eyes, it can move both cameras on two axes at the same time. The motors are very re-

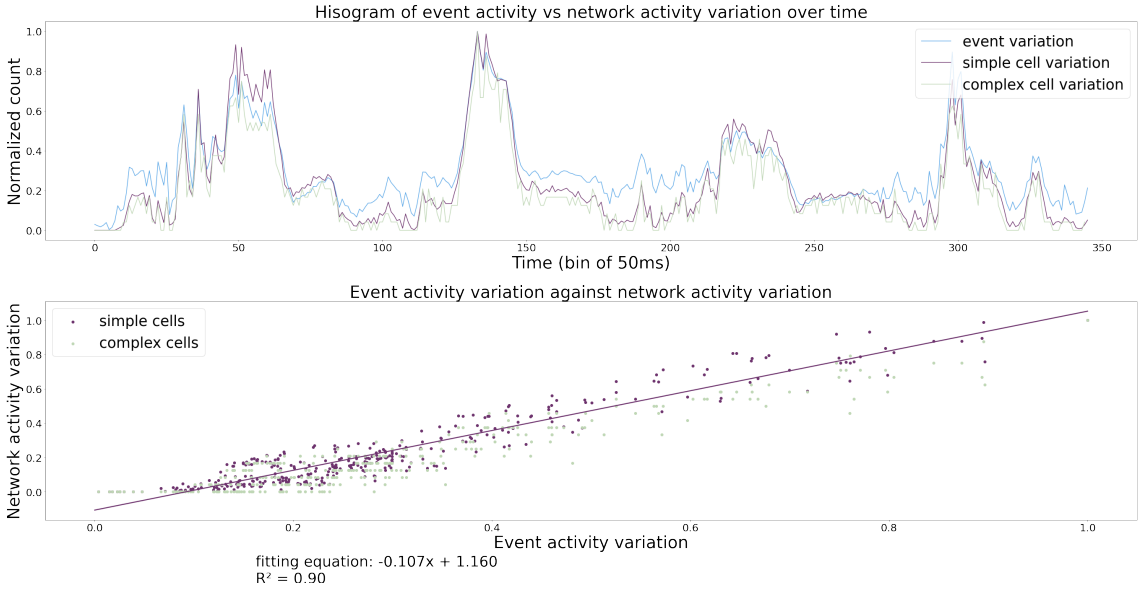
active, making them suitable for use with the fast response time of event-based cameras. Fig. 5.1 is a picture of the system with both event-based cameras. Using that platform, we intend to replicate the learning of the tracking task and extend our framework on the vergence control. However, the pan-tilt platform cannot be used for the stabilization task since it lacks a rotation on the yaw axis. We would have to create a different robotic platform for that scenario.

Application to driving scenarios Finally, we were interested in using the fast reaction time of the event-based camera to help in driving scenarios. Autonomous driving has emerged as one of the most critical research applications in recent years. There is a significant stake in creating efficient self-driving cars. Event-based cameras could be a prime candidate as a sensing alternative to frame-based cameras. They could detect static and dynamic obstacles much faster than traditional cameras. In that regard, we thought about demonstrating our sensing architecture for driving scenarios such as urgent braking. Fig. 3.17 shows a picture of our mobile robotic platform in a recreated urban environment. This simple platform could be ideal for demonstrating a proof of concept for that specific scenario.

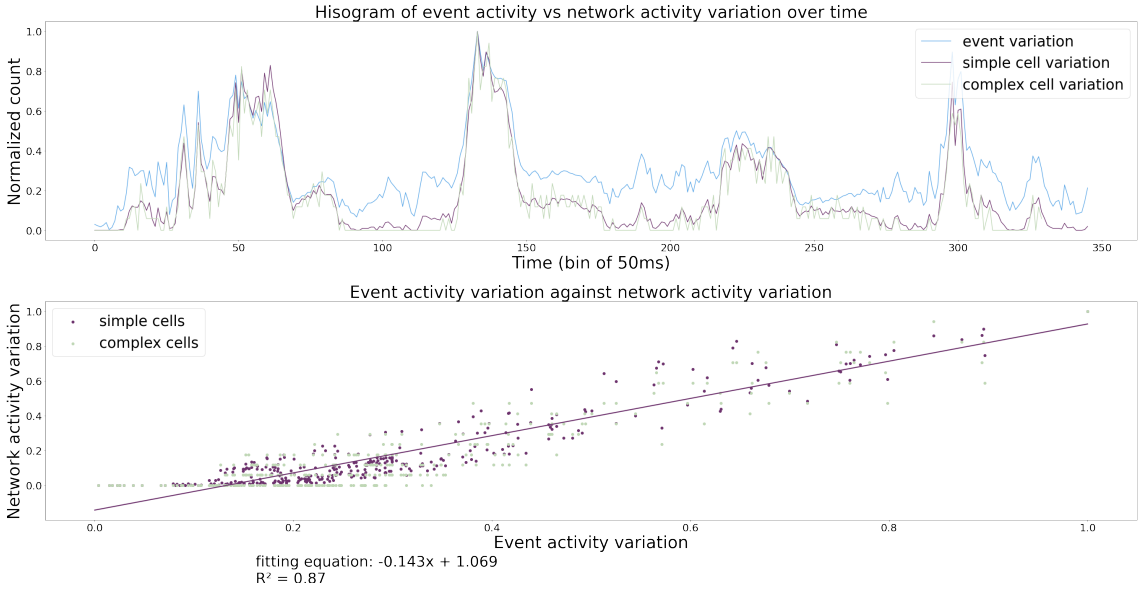
Appendices

Appendix A

Supplementary material

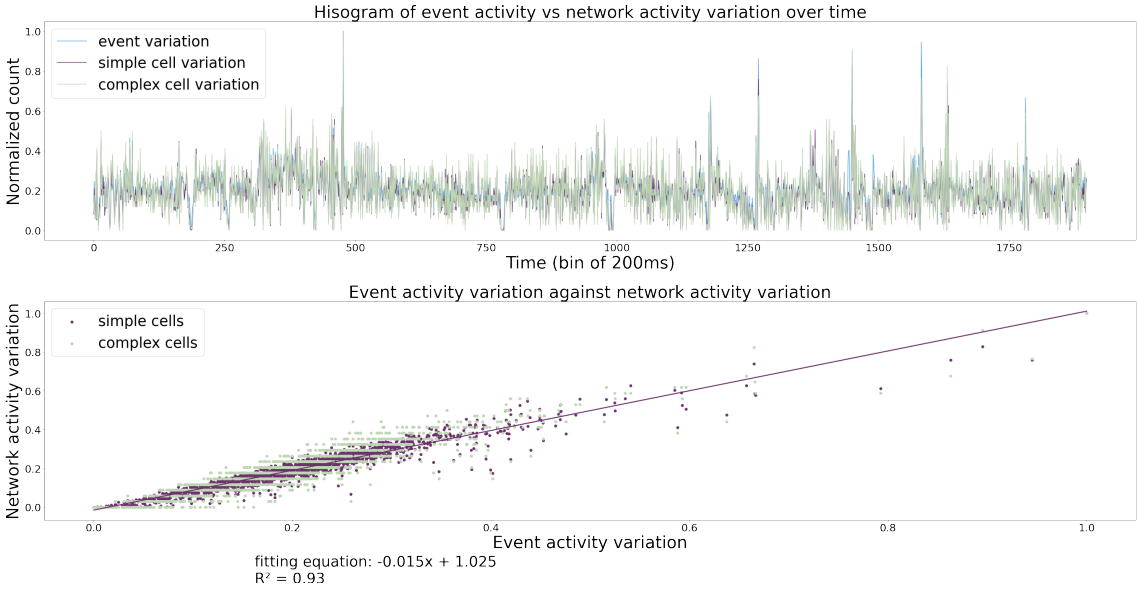


(a)

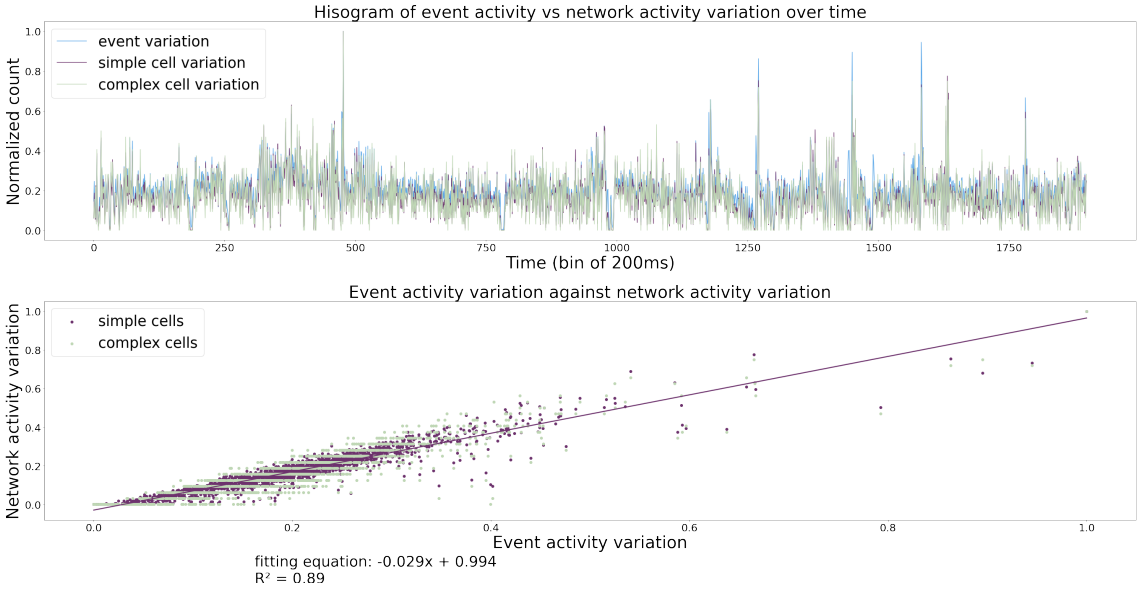


(b)

Figure A.1: Same on short recording of an office. (a) With learned weight. (b) With random weights.

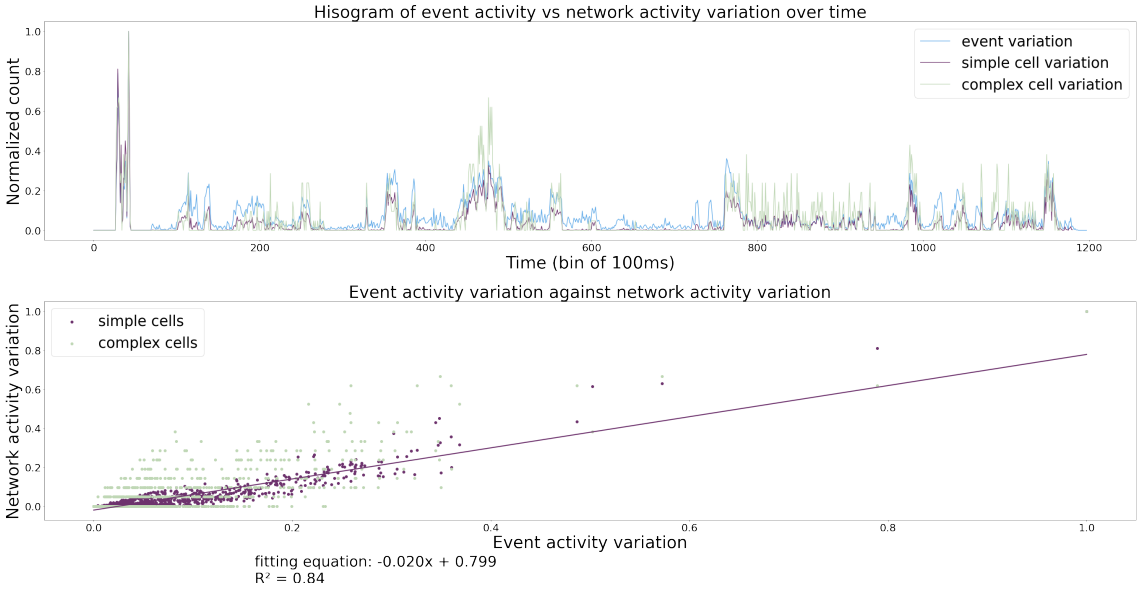


(a)

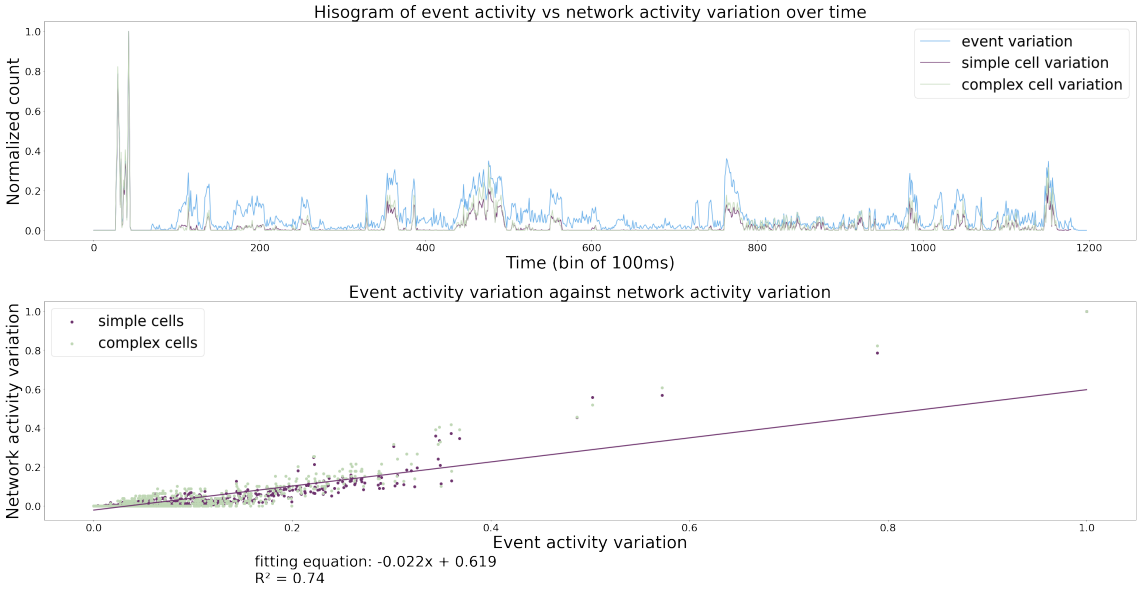


(b)

Figure A.2: Same on a recording of someone juggling with 3 balls. (a) With learned weight. (b) With random weights.

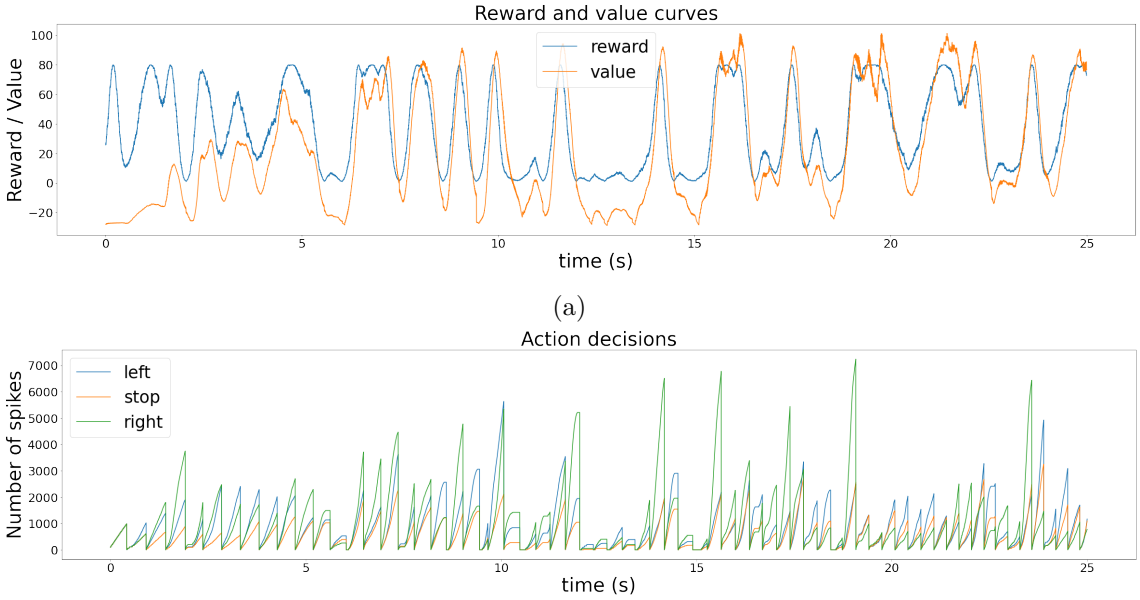


(a)



(b)

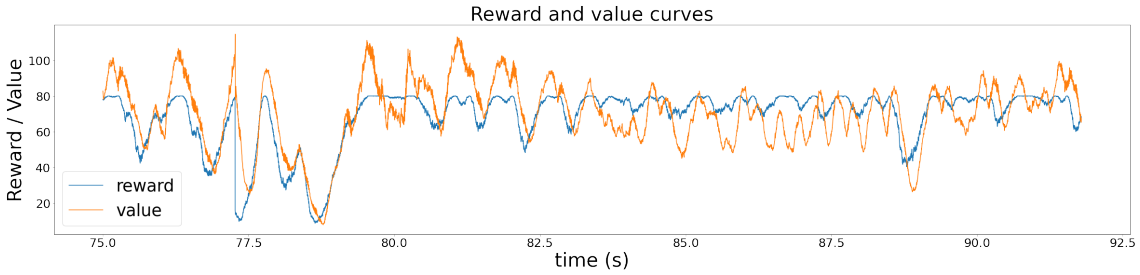
Figure A.3: Same on an outside recording of an urban environment with a mobile robotic platform. (a) With learned weight. (b) With random weights.



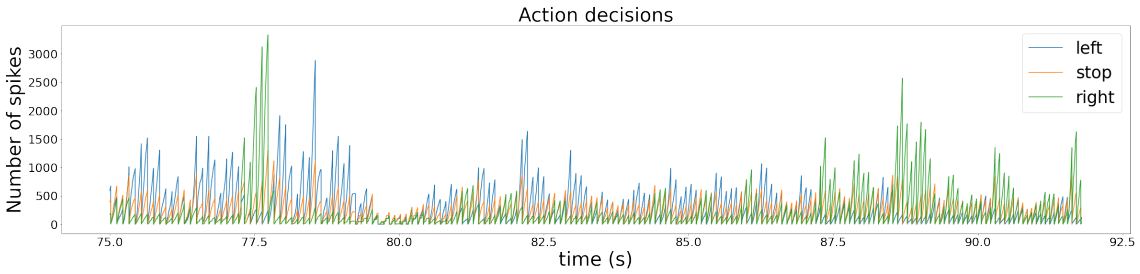
(a)

(b)

Figure A.4: (a) Reward and value function at the start of training (b) Actor neurons spike count evolution at the start of training for the 3 possible actions (left in blue, stop in orange, and right in green). Every time an action is selected, the 3 counts are reset to 0. The action with the highest amount of spikes is not necessarily the one that is selected, due to the exploration strategy.



(a)



(b)

Figure A.5: (a) Reward and value function at the end of training (b) Action decision at the end of training

Appendix B

Publications and communications

The contributions of this thesis have been published in two international conferences, and submitted to an international journals.

International Proceedings and Journals

Work related to the first contribution has been published as

> [1] T. Barbier, C. Teulière, and J. Triesch, "Unsupervised learning of spatio-temporal receptive fields from an event-based vision sensor," *ICANN Artificial Neural Networks and Machine Learning*, I. Farkaš, P. Masulli, and S. Wermter, Eds., pp. 622–633, 2020

and has been extended in a second version as

> [2] T. Barbier and J. Triesch, "Spike timing-based unsupervised learning of orientation, disparity, and motion representations in a spiking neural network," *CVPR 2021 Workshops*, vol. 25, pp. 1377–1386, 2021 **(accepted for video presentation and a question session)**

Work related to the second contribution will be submitted to a journal.

Proceedings & Workshops without act

Other communications of this work include a poster presentation as

> Thomas Barbier, "Autonomous learning in a neuromorphic vision system", *Journée Scientifique des Doctorants* (JSD ED-SPI), June, 2021.

Appendix C

Source code

All our code sources have been made publicly available on the lab's GitHub page, https://github.com/comsee-research, for reproducibility and broad accessibility and licensed under the GNU General Public License v3.0. It includes:

Neuvisys available at https://github.com/comsee-research/Neuvisys.

Neuvisys-simulator available at https://github.com/comsee-research/Neuvisys-simulator.

Neuvisys-python available at https://github.com/comsee-research/Neuvisys-analysis.

C.1 Neuvisys

The Neuvisys project stands for Neuromorphic Vision System. It is a library offering access to a Spiking Neural Network (SNN) with different possible kinds of neurons. The library is written in c++. It can be launched with command lines or via a Qt Graphical User Interface (GUI). There is also a possible connection with the Coppeliasim simulator, also known as V-REP, via a Robot Operating System (ROS) interface.

C.1.1 Requirements

Neuvisys

- Python
- OpenCV
- HDF5

Neuvisys uses libraries such as Eigen, a JSON parser, cnpy, and caer, all linked locally from src/dependencies.

OpenCV To install OpenCV:

```
sudo apt install libopencv-dev python3-OpenCV
```

HDF5 To install HDF5:

```
sudo apt-get install libhdf5-dev
```

The HDF5 format is used to store event files, that can then be used by the network. The format should be as follows:

- a group named "events"
- 5 dataset in that group: "t" for timestamps, "x" for pixel width axis, "y" for pixel height axis, "p" for polarities and "c" for camera (0 for left camera, 1 for right camera).

Event Based Cameras To connect to event-based cameras, and use the SNN live, you need to install caer: https://github.com/breznak/caer The libcaer can be downloaded from https://gitlab.com/inivation/dv/libcaer

Qt install QT 5 with the **Qt Charts** module:

```
sudo apt install qt5-default
sudo apt install libqt5charts5-dev
```

Coppeliasim / ROS Download and install the Coppeliasim framework:

https://www.coppeliarobotics.com/

There are three available versions, **player**, **edu** and **pro**. The player is enough to work with neuvisys, but the scene save function is disabled.

Install ROS Noetic (Other ROS distribution might work, but this is uncertain):

http://wiki.ros.org/noetic/Installation/Ubuntu

It is advised to use the **Desktop-Full Install**, though other lighter versions may also work.

C.1.2 Neuvisys libraries

By default, only the neuvisys library core is compiled. Four more libraries add functionality:

- Camera: allows the connection to event-based cameras thanks to caer. With it activated, we can use event-based cameras and feed the events directly to the SNN in real-time.
- Simulator: allows the connection to Coppeliasim. With it activated, we can create complex environments, simulate event-based camera outputs, and feed it to the SNN in real-time.
- Motor control: allows the connection with Faulhaber Brushless motors. With it activated, you can pilot the motors in real-time.
- GUI: allows the use of a graphical user interface.

C.1.3 Launch

To compile the Neuvisys library in the root folder, run:

```
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
```

If we want to use some of the abovementioned functionalities, we can compile them with the following:

```
cmake -DBUILD_CAMERA=ON -DBUILD_SIMULATOR=ON
-DBUILD_MOTOR_CONTROL=ON -DBUILD_GUI=ON -DCMAKE_BUILD_TYPE=Release ..
```

(put "OFF" on the functionalities you do not want to use and compile).

The core neuvisys library does not need any installation requirements except OPENCV. However, adding more functionalities means installing adequate libraries (see Requirements section).

If there are some errors, we may have to install the following python packages:

```
pip install empy
pip install catkin-pkg
```

Run:

```
make -j
```

to compile all targets

Or run:

```
make [target-name]
```

To compile only one target. Possible targets are:

- *neuvisys-exe* is the command line executable.
- *event-camera* is a module to connect an event-based camera (Davis by default).
- *neuvisys-simulator* is an executable that connects to Coppeliasim via ROS.
- *neuvisys-qt* is similar to neuvisys but with an added Qt interface.

Compiled targets are found in the "build/src" folder.

An example of use with the *neuvisys-exe* target:

```
./neuvisys-exe [m\_networkPath] [eventPath] [nbPass]
```

m_networkPath corresponds to the path of the network structure. It must link to the network config file, such as: "./network/configs/network_config.json".

eventPath is the relative path to an event file in the .npz format.

nbPass is the number of times the events will be presented to the network.

Create empty Network We can generate an empty spiking network ready to use from, run:

```
cd  build
./neuvisys-exe [networkDirectory]
```

The parameters will be set to their default values, but you can change them afterward using the GUI or directly via the JSON config files.

C.1.4 Configuration guide

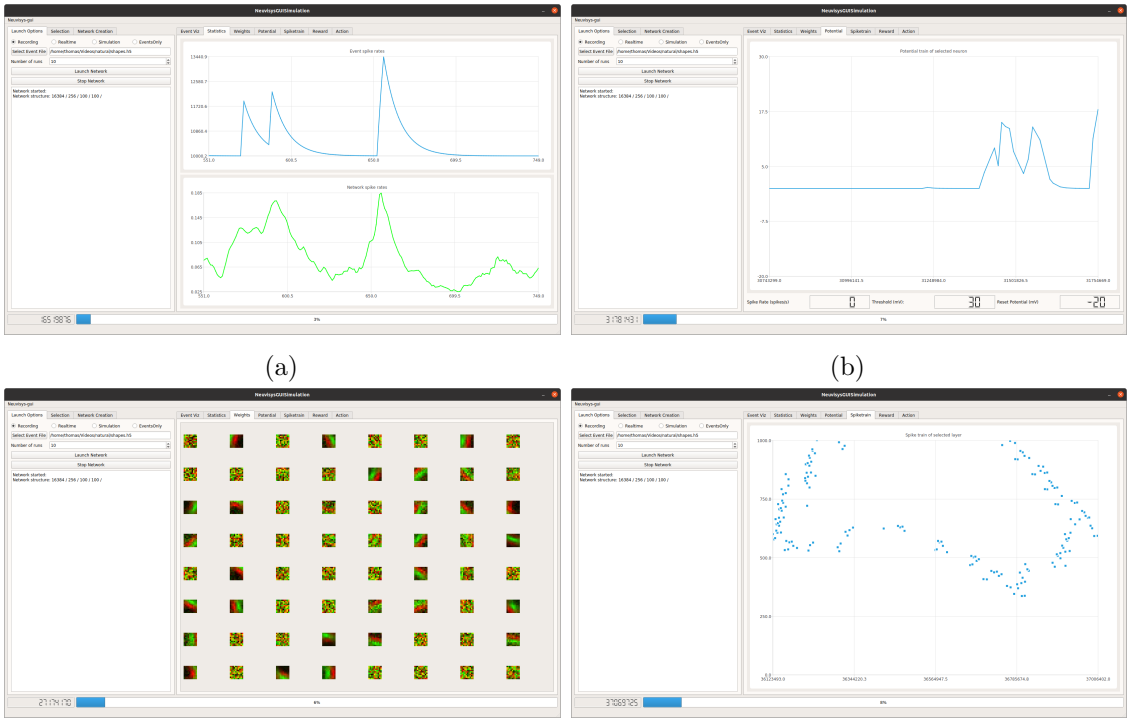
The network parameters are saved in JSON configuration files:

- network_config.json : describes the network architecture, number of layers, neurons, types of newStaticInhibitoryEvent...
- simple_cell_config.json : simple neuron parameters
- complex_cell_config.json : complex neuron parameters
- critic_cell_config.json : critic neuron parameters
- actor_cell_config.json : actor neuron parameters

C.1.5 Graphical User Interface

It is possible to observe the network's behavior in real-time using the graphical user interface developed specifically to work with it.

Fig. C.1 presents a few screenshots showing its use and features. The GUI can serve multiple purposes.



(a) (b)

(c) (d)

Figure C.1: Qt Graphical User Interface made for the Neuvisys spiking neural network code. (a) Events and Network spike rate information. (b) Cells' potentials. (c) Cells' receptive fields (d) Cells' spike trains.

Network management The GUI offers features to manipulate the networks before launching any training. It is possible to create new networks but also change the parameters of existing ones, as well as select the input files and training parameters.

Training supervision During training, the GUI can display real-time network information such as input and cells' spike rates, potentials, and spike trains, but also see the evolution of the cells' receptive fields.

Reinforcement learning supervision When using the GUI combined with the reinforcement learning framework (and possibly the CoppeliSim simulator), we can display information about the reinforcement learning procedure. The reward, value function, TD error, and action selection in simulation time are all available.

C.2 CoppeliaSim event-based simulation

CoppeliaSim is a robotic simulator made for creating test scenarios in which robots can operate. They propose multiple physics engines as well as tools to create various environments, as well as include different types of sensors.

Event-based cameras are not included by default. Instead, we use a frame-based sensor and convert the frames into event streams. For that purpose, we use small simulation timesteps to obtain high framerates to improve the event generation. We based the generation of events on the event-based camera emulator PIX2NVS [148].

We propose four different scenarios, three of which we actively used in this manuscript:

Tracking Environment based on a pan-tilt head. The goal is to track a ball moving in a circle around the camera on the horizontal plane. The ball can move left or right.

Tracking 2D Similar environment to the tracking one but with an added axis of rotation, and the ball can now move freely in a sphere around the camera.

Stabilization In this environment, the goal is to maintain straight bars in the desired orientation. The bars can also rotate to increase difficulty. There is one axis of rotation for the camera.

Stereo vergence This environment is based on two pan-tilt head. The goal is to verge the two eyes on objects at varying distances of the cameras.

For each scenario, a jittering mechanism based on an Ornstein-Uhlenbeck process can be added to the camera. It can help create events even when the camera is still and mimic the jittering and drift of the human eye.

C.2.1 ROS integration

To connect to the simulation, we use a ROS publisher and subscriber program written in c++. It allows the recovery of frames from the simulator and other information, such as the reward. Then, orders to the motors can be similarly sent from the program to the simulator.

CoppeliaSim usage Before launching CoppeliaSim, we need to activate ROS:

```
roscore
```

Then launch CoppeliaSim. If ROS is working, the "ROS interface was found." message should appear on the CoppeliaSim console at the bottom.

We can open the Neuvisys Coppelia scene with the following:

File/Open scene... Open src/Coppelia-scenes/Neuvisys.ttt

Then launch the neuvisys-simulator target with the following:

```
./build/src/neuvisys-simulator
```

C.3 Neuvisys-analysis

The Spiking Neural Network Analysis Library is a regrouping of python scripts, jupyter-notebooks, and python classes to analyze and display information about the neuvisys c++ SNN code.

C.3.1 Requirements

A python envrionment. Alternatively, the library is made to be used with poetry.

Installation with Poetry Install poetry:

```
curl -SSL https://install.python-poetry.org | python3 -
```

Install the dependencies from the library folder:

```
poetry install
```

We can either launch a script with poetry run python your_script.py or activate the virtual environment with poetry shell.

Python Packages Here is the list of packages and what they do:

- *driving_dataset*: a package to work with the DDD17 driving dataset.
- *events*: a package to create, modify, and convert event files.
- *frames*: a package to analyze traditional frames.
- *spiking_network*: a package to modify, visualize and launch the SNN of the neuvisys project.

C.3.2 Jupyter-Notebooks

Two jupyter-notebooks offer an interface to most of the packaged functions mentioned before:

Neuvisys.ipynb: Used to visualize various information from a spiking network.
Utils.ipynb: Used to launch and modify spiking networks.

Bibliography

[1] T. Barbier, C. Teulière, and J. Triesch, "Unsupervised learning of spatio-temporal receptive fields from an event-based vision sensor," *ICANN Artificial Neural Networks and Machine Learning*, I. Farkaš, P. Masulli, and S. Wermter, Eds., pp. 622–633, 2020.

[2] T. Barbier and J. Triesch, "Spike timing-based unsupervised learning of orientation, disparity, and motion representations in a spiking neural network," *CVPR 2021 Workshops*, vol. 25, pp. 1377–1386, 2021.

[3] D. A. Patterson, J. Gonzalez, Q. V. Le, *et al.*, "Carbon emissions and large neural network training," *ArXiv*, 2021.

[4] D. Richter, *Metabolism of the nervous system*. Pergamon Press, 1957.

[5] R. Guyonneau, R. VanRullen, and S. J. Thorpe, "Temporal codes and sparse representations: A key to understanding rapid processing in the visual system," *Journal of Physiology-Paris*, vol. 98, pp. 487–497, 4-6 Jul. 2004.

[6] Y. Zhao, C. A. Rothkopf, J. Triesch, and B. E. Shi, "A unified model of the joint development of disparity selectivity and vergence control," *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–6, Nov. 2012.

[7] L. Lonini, S. Forestier, C. Teulière, Y. Zhao, B. E. Shi, and J. Triesch, "Robust active binocular vision through intrinsically motivated learning," *Frontiers in Neurorobotics*, vol. 7, Nov. 2013.

[8] C. Teulière, S. Forestier, L. Lonini, *et al.*, "Self-calibrating smooth pursuit through active efficient coding," *Robotics and Autonomous Systems*, vol. 71, pp. 3–12, 2015.

[9] A. Lelais, J. Mahn, V. Narayan, C. Zhang, B. E. Shi, and J. Triesch, "Autonomous development of active binocular and motion vision through active efficient coding," *Frontiers in Neurorobotics*, vol. 13, 2019.

[10] S. Eckmann, L. Klimmasch, B. E. Shi, and J. Triesch, "Active efficient coding explains the development of binocular vision and its failure in amblyopia," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 117, 11 2020.

[11] L. Klimmasch, J. Schneider, A. Lelais, M. Fronius, B. E. Shi, and J. Triesch, "The development of active binocular vision under normal and alternate rearing conditions," *eLife*, vol. 10, 2021.

[12] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Computation*, vol. 19, pp. 1468–1502, 6 2007.

[13] S. S. Kety, "The circulation and energy metabolism of the brain," *Clinical neurosurgery*, vol. 9, pp. 56–66, 1963.

[14] H. B. Barlow, "Possible principles underlying the transformations of sensory messages," *Sensory Communication*, pp. 216–234, Oct. 2013.

[15] M. Chalk, O. Marre, and G. Tkačik, "Toward a unified theory of efficient, predictive, and sparse coding," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, pp. 186–191, 1 Jan. 2018.

[16] P. Seenivasan and R. Narayanan, "Efficient information coding and degeneracy in the nervous system," *Current Opinion in Neurobiology*, vol. 76, p. 102 620, Oct. 2022.

[17] J. J. Atick and A. N. Redlich, "What does the retina know about natural scenes?" *Neural Computation*, vol. 4, pp. 196–210, 2 Mar. 1992.

[18] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, pp. 3311–3325, 23 Dec. 1997.

[19] C. Zhang, Y. Zhao, J. Triesch, and B. E. Shi, "Intrinsically motivated learning of visual motion perception and smooth pursuit," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1902–1908, Sep. 2014.

[20] T. Parr and K. J. Friston, "Active inference, novelty and neglect," *Current topics in behavioral neurosciences*, vol. 41, pp. 115–128, 2019.

[21] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, J. O'Doherty, and G. Pezzulo, "Active inference and learning," *Neuroscience and Biobehavioral Reviews*, vol. 68, pp. 862–879, Sep. 2016.

[22] M. Traub, M. V. Butz, R. Legenstein, and S. Otte, "Dynamic action inference with recurrent spiking neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12895 LNCS, pp. 233–244, 2021.

[23] T. Isomura, H. Shimazaki, and K. J. Friston, "Canonical neural networks perform active inference," *Communications Biology 2022 5:1*, vol. 5, pp. 1–15, 1 Jan. 2022.

[24] G. Gallego, T. Delbruck, G. Orchard, *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, Jan. 2022.

[25] T.-j. Lee, "A review of bioinspired vision sensors and their applications," *Sensors and Materials*, vol. 27, pp. 447–463, 6 2015.

[26] S. C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*. 2014.

[27] M. A. M. Carver Mead, *Scientific American*, vol. 264, May 1991.

[28] G. Gallego, T. Delbrück, G. Orchard, *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[29] 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*, Nov. 2013.

[30] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic- procedural system," *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 845–848, 2007.

[31] T. Delbruck, "Frame-free dynamic digital vision," *Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pp. 21–26, 2008.

[32] A. Glover and C. Bartolozzi, "Event-driven ball detection and gaze fixation in clutter," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2203–2208, Nov. 2016.

[33] S. Schraml, A. N. Belbachir, N. Milosevic, and P. Schön, "Dynamic stereo vision system for real-time tracking," *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, pp. 1409–1412, 2010.

[34] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck, "A pencil balancing robot using a pair of aer dynamic vision sensors," *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 781–784, 2009.

[35] F. Rea, G. Metta, and C. Bartolozzi, "Event-driven visual attention for the humanoid robot icub," *Frontiers in Neuroscience*, 7 DEC 2013.

[36] A. Glover and C. Bartolozzi, "Robust visual tracking with a freely-moving event camera," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-September, pp. 3769–3776, Dec. 2017.

[37] X. Lagorce, C. Meyer, S. H. Ieng, D. Filliat, and R. Benosman, "Asynchronous event-based multikernel algorithm for high-speed visual features tracking," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 1710–1720, 8 Aug. 2015.

[38] Z. Ni, S. H. Ieng, C. Posch, S. Régnier, and R. Benosman, "Visual tracking using neuromorphic asynchronous event-based cameras," *Neural computation*, vol. 27, pp. 925–953, 4 Apr. 2015.

[39] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Régnier, "Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics," *IEEE Transactions on Robotics*, vol. 28, pp. 1081–1089, 5 2012.

[40] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Asynchronous, photometric feature tracking using events and frames," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11216 LNCS, pp. 766–781, 2018.

[41] H. Li and L. Shi, "Robust event-based object tracking combining correlation filter and cnn representation," *Frontiers in Neurorobotics*, vol. 13, p. 82, Oct. 2019.

[42] T.-J. Chin, S. Bagchi, A. Eriksson, and A. van Schaik, "Star tracking using an event camera," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1646–1655, 2019.

[43] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," *Robotics: Science and Systems*, Feb. 2018.

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

[45] R. Benosman, C. Clercq, X. Lagorce, S. H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 407–417, 2 Feb. 2014.

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

[47] Z. Li, P. Mazzoni, S. Song, and N. Qian, "Asynchronous event-based motion processing: From visual events to probabilistic sensory representation," *Neural Computation 31, 1114–1138*, vol. 1138, pp. 1114–1138, 2018.

[48] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 884–892, Dec. 2016.

[49] M. Monforte, A. Arriandiaga, A. Glover, and C. Bartolozzi, "Exploiting event cameras for spatio-temporal prediction of fast-changing trajectories," *Proceedings - 2020 IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2020*, pp. 108–112, 2020.

[50] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1346–1359, 7 Jul. 2017.

[51] J. Kogler, C. Sulzbachner, M. Humenberger, and F. Eibensteiner, "Address-event based stereo vision with bio-inspired silicon retina imagers," *Advances in Theory and Applications of Stereo Vision*, Jan. 2011.

[52] J. Lee, T. Delbruck, P. K. Park, *et al.*, "Live demonstration: Gesture-based remote control using stereo pair of dynamic vision sensors," *ISCAS 2012 - 2012 IEEE International Symposium on Circuits and Systems*, pp. 742–745, 2012.

[53] A. Grimaldi, V. Boutin, S.-H. Ieng, R. Benosman, and L. U. Perrinet, "A robust event-driven approach to always-on object recognition," *TechRxiv preprint*, Jan. 13, 2022.

[54] R. Benosman, S. H. Ieng, P. Rogister, and C. Posch, "Asynchronous event-based hebbian epipolar geometry," *IEEE Transactions on Neural Networks*, vol. 22, pp. 1723–1734, 11 2011.

[55] P. Rogister, R. Benosman, S. H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 347–353, 2 Feb. 2012.

[56] J. Carneiro, S. H. Ieng, C. Posch, and R. Benosman, "Event-based 3d reconstruction from neuromorphic retinas," *Neural Networks*, vol. 45, pp. 27–38, 2013.

[57] M. Firouzi and J. Conradt, "Asynchronous event-based cooperative stereo matching using neuromorphic silicon retinas," *Neural Processing Letters*, vol. 43, pp. 311–326, 2 2016.

[58] E. Piatkowska, J. Kogler, N. Belbachir, and M. Gelautz, "Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 370–377, 2017.

[59] Z. Deng, B. Zhou, Y. Xu, *et al.*, "Recent progresses of organic photonic synaptic transistors," *Flexible and Printed Electronics*, vol. 7, no. 2, p. 024 002, May 2022.

[60] 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, pp. 1470–1484, 10 Oct. 2014.

[61] C. Posch, "Bioinspired vision sensing," *Biologically inspired Computer Vision: Fundamentals and Applications*, pp. 11–28, Oct. 2015.

[62] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, pp. 2426–2429, 2010.

[63] R. H. Masland, "The Neuronal Organization of the Retina," *Neuron*, vol. 76, no. 2, pp. 266–280, 2012.

[64] P. Dayan and L. F. Abbott, *Theoretical neuroscience : computational and mathematical modeling of neural systems*. Massachusetts Institute of Technology Press, 2001.

[65] S. Fischer, R. Redondo, L. U. Perrinet, and G. Cristóbal, "Sparse approximation of images inspired from the functional architecture of the primary visual areas," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, pp. 090 727–122, 2007.

[66] M. A. Dichter and G. F. Ayala, "Cellular mechanisms of epilepsy: A status report," *Science (New York, N.Y.)*, vol. 237, pp. 157–164, 4811 1987.

[67] S. Zhou and Y. Yu, "Synaptic e-i balance underlies efficient neural coding," *Frontiers in Neuroscience*, vol. 12, p. 46, FEB Feb. 2018.

[68] A. M. Sillito, "The contribution of inhibitory mechanisms to the receptive field properties of neurones in the striate cortex of the cat.," *The Journal of Physiology*, vol. 250, p. 305, 2 Sep. 1975.

[69] L. U. Perrinet, "Role of homeostasis in learning sparse representations," *Neural Computation*, vol. 22, no. 7, pp. 1812–36, Jul. 17, 2010.

[70] L. U. Perrinet, "An adaptive homeostatic algorithm for the unsupervised learning of visual features," *Vision*, vol. 3, no. 3, p. 47, Sep. 1, 2019.

[71] L. N. Long and G. Fang, "A review of biologically plausible neuron models for spiking neural networks," *AIAA Infotech at Aerospace 2010*, 2010.

[72] K. Yamazaki, V. K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain Sciences*, vol. 12, 7 Jul. 2022.

[73] X. Wang, X. Lin, and X. Dang, "Supervised learning in spiking neural networks: A review of algorithms and evaluations," *Neural Networks*, vol. 125, pp. 258–280, 2020.

[74] M. Taylor, "The problem of stimulus structure in the behavioural theory of perception," *South African journal of psychology = Suid-Afrikaanse tydskrif vir sielkunde*, vol. 3, pp. 23–45, Jan. 1973.

[75] W. B. Levy and O. Steward, "Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus," *Neuroscience*, vol. 8, pp. 791–797, 4 Apr. 1983.

[76] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, pp. 213–215, 5297 Jan. 1997.

[77] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: A hebbian learning rule," *Annual Review of Neuroscience*, vol. 31, no. 1, pp. 25–46, 2008.

[78] W. G. Nicolas Frémeaux, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Frontiers in Neural Circuits*, vol. 9, p. 85, Jan. 2016.

[79] Y. Lian, D. B. Grayden, T. Kameneva, H. Meffin, and A. N. Burkitt, "Toward a biologically plausible model of lgn-v1 pathways based on efficient coding," *Frontiers in Neural Circuits*, vol. 13, p. 13, 13 Jan. 2019.

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

[81] J. A. Pérez-Carrasco, C. Serrano, B. Acha, T. Serrano-Gotarredona, and B. Linares-Barranco, "Spike-based convolutional network for real-time processing," *Proceedings - International Conference on Pattern Recognition*, pp. 3085–3088, 2010.

[82] M. Pfeiffer and T. Pfeil, "Deep Learning With Spiking Neurons: Opportunities and Challenges," *Frontiers in Neuroscience*, vol. 12, p. 774, 2018.

[83] K. Pozo and Y. Goda, "Unraveling mechanisms of homeostatic synaptic plasticity," *Neuron*, vol. 66, no. 3, pp. 337–351, May 2010.

[84] J.-A. Perez-Carrasco, C. Serrano, B. Acha, T. Serrano-Gotarredona, and B. Linares-Barranco, "Spike-based convolutional network for real-time processing," *ICPR*, Aug. 2010.

[85] E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data," *Frontiers in Neuroscience*, vol. 11, Jun. 2017.

[86] E. Chicca, P. Lichtsteiner, T. Delbruck, G. Indiveri, and R. Douglas, "Modeling orientation selectivity using a neuromorphic multi-chip system," *IEEE International Symposium on Circuits and Systems*, Jun. 2006.

[87] T. Masquelier and S. J. Thorpe, "Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity," *PLoS Computational Biology*, vol. 3, no. 2, K. J. Friston, Ed., p. 31, 2007.

[88] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018.

[89] 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, pp. 1–11, 1 Feb. 2015.

[90] G. Srinivasan, P. Panda, and K. Roy, "STDP-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 14, no. 4, pp. 1–12, 2018.

[91] L. Paulun, A. Wendt, and N. Kasabov, "A retinotopic spiking neural network system for accurate recognition of moving objects using NeuCube and dynamic vision sensors," *Frontiers in Computational Neuroscience*, vol. 12, Jun. 2018.

[92] H. Akolkar, S. Panzeri, and C. Bartolozzi, "Spike time based unsupervised learning of receptive fields for event-driven vision," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4258–4264, 2015.

[93] A. Tavanaei, T. Masquelier, and A. Maida, "Representation learning using event-based stdp," *Neural Networks*, vol. 105, pp. 294–303, Sep. 2018.

[94] T. N. Chandrapala and B. E. Shi, "Invariant feature extraction from event based stimuli," *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, vol. 2016-July, pp. 1–6, 2016.

[95] Q. Liu, G. Pan, H. Ruan, D. Xing, Q. Xu, and H. Tang, "Unsupervised AER Object Recognition Based on Multiscale Spatio-Temporal Features and Spiking Neurons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5300–5311, 2020.

[96] V. Boutin, A. Franciosini, F. Y. Chavane, and L. U. Perrinet, "Pooling in a predictive model of v1 explains functional and structural diversity across species," *PLoS Computational Biology*, Jul. 18, 2022.

[97] G. Orchard and R. Etienne-Cummings, "Bioinspired visual motion estimation," *Proceedings of the IEEE*, vol. 102, pp. 1520–1536, 10 2014.

[98] H. N. Stephan Tschechne Roman Sailer, "Bio-inspired optic flow from event-based neuromorphic sensor input," *Artificial Neural Networks in Pattern Recognition*, vol. 8774, N. El Gayar, F. Schwenker, and C. Suen, Eds., pp. 171–182, 2014.

[99] G. D'Angelo, E. Janotte, T. Schoepe, *et al.*, "Event-Based Eccentric Motion Detection Exploiting Time Difference Encoding," *Frontiers in Neuroscience*, vol. 14, p. 451, 2020.

[100] G. Haessig, A. Cassidy, R. Alvarez, R. Benosman, and G. Orchard, "Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 4, pp. 860–870, Aug. 2018.

[101] M. Hopkins, G. Pineda-Garcia, P. A. Bogdan, and S. B. Furber, "Spiking neural networks for computer vision," *Interface Focus*, vol. 8, no. 4, p. 20 180 007, Jun. 2018.

[102] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor, "A spiking neural network architecture for visual motion estimation," *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2013.

[103] F. Paredes-Valles, K. Y. W. Scheper, and G. C. H. E. D. Croon, "Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception," *PAMI*, Aug. 2018.

[104] A. Grimaldi and L. U. Perrinet, "Learning heterogeneous delays in a layer of spiking neurons for fast motion detection," *Submitted*, 2022.

[105] L. Steffen, D. Reichard, J. Weinland, J. Kaiser, A. Roennau, and R. Dillmann, "Neuromorphic stereo vision: A survey of bio-inspired sensors and algorithms," *Frontiers in Neurorobotics*, vol. 13, pp. 1–20, 2019.

[106] J. Kaiser, J. Weinland, P. Keller, *et al.*, "Microsaccades for neuromorphic stereo vision," *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds., pp. 244–252, 2018.

[107] G. Dikov, M. Firouzi, F. Rohrbein, J. Conradt, and C. Richter, "Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware," *Biomimetic and Biohybrid Systems*, pp. 119–137, 2017.

[108] M. Osswald, S. H. Ieng, R. Benosman, and G. Indiveri, "A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems," *Scientific Reports*, vol. 7, 2017.

[109] T. Chauhan, T. Masquelier, A. Montlibert, and B. R. Cottereau, "Emergence of binocular disparity selectivity through hebbian learning," *Journal of Neuroscience*, vol. 38, pp. 9563–9578, 44 Oct. 2018.

[110] T. Chauhan, T. Masquelier, and B. R. Cottereau, "Sub-optimality of the early visual system explained through biologically plausible plasticity," *Frontiers in Neuroscience*, vol. 15, p. 1203, Sep. 2021.

[111] G. Debat, T. Chauhan, B. R. Cottereau, T. Masquelier, M. Paindavoine, and R. Baures, "Event-based trajectory prediction using spiking neural networks," *Frontiers in Computational Neuroscience*, vol. 15, p. 47, May 2021.

[112] N.-W. Tien and D. Kerschensteiner, "Homeostatic plasticity in neural development," *Neural Development*, vol. 13, no. 1, Jun. 2018.

[113] J. Triesch, A. D. Vo, and A.-S. Hafner, "Competition for synaptic building blocks shapes synaptic plasticity," *Elife*, vol. 7, e37836, Sep. 2018.

[114] A. Borst, "Models of motion detection," *Nature Neuroscience*, vol. 3, no. 1, p. 1168, Jan. 2000.

[115] Y. Lian, A. Almasi, D. B. Grayden, T. Kameneva, A. N. Burkitt, and H. Meffin, "Learning receptive field properties of complex cells in v1," *bioRxiv*, 2020.

[116] K. P. Körding, C. Kayser, W. Einhäuser, and P. König, "How are complex cell properties adapted to the statistics of natural stimuli?" *Journal of Neurophysiology*, vol. 91, no. 1, pp. 206–212, 2004.

[117] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 130 dB 3 micros latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.

[118] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd17: End-to-end davis driving dataset," *Arxiv*, Nov. 2017.

[119] H. Rebecq, D. Gehrig, and D. Scaramuzza, "Esim: An open event camera simulator," *CoRL*, Oct. 2018.

[120] D. Attwell and S. B. Laughlin, "An energy budget for signaling in the grey matter of the brain," *Journal of Cerebral Blood Flow and Metabolism*, vol. 21, pp. 1133–1145, 10 Aug. 2016.

[121] I. M. Park, S. Seth, A. R. Paiva, L. Li, and J. C. Principe, "Kernel methods on spike train space for neuroscience: A tutorial," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 149–160, 2013.

[122] C. A. Rothkopf, T. H. Weisswange, and J. Triesch, "Learning independent causes in natural images explains the spacevariant oblique effect," *ICDL 2009. IEEE 8th International Conference on Development and Learning*, pp. 1–6, Jan. 2009.

[123] B. Li, M. R. Peterson, and R. D. Freeman, "Oblique effect: A neural basis in the visual cortex," *Journal of neurophysiology*, vol. 90, no. 1, pp. 204–217, 2003.

[124] M. Mazurek, M. Kager, and S. D. Van Hooser, "Robust quantification of orientation selectivity and direction selectivity," *Frontiers in Neural Circuits*, vol. 8, p. 92, 2014.

[125] E. C. Tolman, "Cognitive maps in rats and men," *Psychological Review*, vol. 55, pp. 189–208, 4 Jul. 1948.

[126] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers in Neurorobotics*, vol. 12, p. 35, 2019.

[127] Z. Jiang, Z. Bing, K. Huang, and A. Knoll, "Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot," *Frontiers in Neurorobotics*, vol. 13, 2019.

[128] A. Linares-Barranco, H. Liu, A. Rios-Navarro, F. Gomez-Rodriguez, D. P. Moeys, and T. Delbruck, "Approaching retinal ganglion cell modeling and fpga implementation for robotics," *Entropy*, vol. 20, p. 475, 6 Jun. 2018.

[129] V. Vasco, A. Glover, Y. Tirupachuri, F. Solari, M. Chessa, and C. Bartolozzi, "Vergence control with a neuromorphic icub," *IEEE-RAS International Conference on Humanoid Robots*, pp. 732–738, 2016.

[130] D. Patel, H. Hazan, D. J. Saunders, H. T. Siegelmann, and R. Kozma, "Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game," *Neural Networks*, vol. 120, pp. 108–115, Dec. 2019.

[131] G. Tang, N. Kumar, R. Yoo, and K. P. Michmizos, "Deep reinforcement learning with population-coded spiking neural network for continuous control," Oct. 2020.

[132] N. Frémaux, H. Sprekeler, and W. Gerstner, "Functional requirements for reward-modulated spike-timing-dependent plasticity," *Journal of Neuroscience*, vol. 30, pp. 13 326–13 337, 40 Oct. 2010.

[133] G. W, L. M, L. V, C. D, and B. J, "Eligibility traces and plasticity on behavioral time scales: Experimental support of neohebbian three-factor learning rules," *Frontiers in neural circuits*, vol. 12, Jul. 2018.

[134] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks," *Pattern Recognition*, vol. 94, pp. 87–95, Oct. 2019.

[135] H. Ghaemi, E. Mirzaei, M. Nouri, and S. R. Kheradpisheh, "Biolcnet: Reward-modulated locally connected spiking neural networks," *ArXiv*, Sep. 2021.

[136] M. Yuan, X. Wu, R. Yan, and H. Tang, "Reinforcement learning in spiking neural networks with stochastic and deterministic synapses," *Neural Computation*, vol. 31, pp. 2368–2389, 12 Dec. 2019.

[137] W. Potjans, A. Morrison, and M. Diesmann, "A spiking neural network model of an actor-critic learning agent," *Neural Computation*, vol. 21, pp. 301–339, 2 Feb. 2009.

[138] J. Jitsev, A. Morrison, and M. Tittgemeyer, "Learning from positive and negative rewards in a spiking neural network model of basal ganglia," *Proceedings of the International Joint Conference on Neural Networks*, 2012.

[139] E. Nichols, L. J. McDaid, and N. Siddique, "Biologically inspired snn for robot control," *IEEE Transactions on Cybernetics*, vol. 43, pp. 115–128, 1 Feb. 2013.

[140] N. Frémaux, H. Sprekeler, and W. Gerstner, "Reinforcement learning using a continuous time actor-critic framework with spiking neurons," *PLoS Computational Biology*, vol. 9, p. 1 003 024, 4 Apr. 2013.

[141] P. Weidel, R. Duarte, and A. Morrison, "Unsupervised learning and clustered connectivity enhance reinforcement learning in spiking neural networks," *Frontiers in Computational Neuroscience*, vol. 15, p. 18, Mar. 2021.

[142] H. Anwar, S. Caby, S. Dura-Bernal, *et al.*, "Training a spiking neuronal network model of visual-motor cortex to play a virtual racket-ball game using reinforcement learning," *PLoS ONE*, vol. 17, 5 May May 2022.

[143] A. Jaegle, V. Mehrpour, and N. Rust, "Visual novelty, curiosity, and intrinsic reward in machine learning and the brain," *Current Opinion in Neurobiology*, vol. 58, pp. 167–174, 2019.

[144] Y. Wang and B. E. Shi, "Improved binocular vergence control via a neural network that maximizes an internally defined reward," *IEEE Transactions on Autonomous Mental Development*, vol. 3, pp. 247–256, 3 Sep. 2011.

[145] A. Gibaldi, M. Chessa, A. Canessa, S. P. Sabatini, and F. Solari, "A cortical model for binocular vergence control without explicit calculation of disparity," *Neurocomputing*, vol. 73, pp. 1065–1073, 7-9 2010.

[146] A. Gibaldi, A. Canessa, and S. P. Sabatini, "Vergence control learning through real v1 disparity tuning curves," *International IEEE/EMBS Conference on Neural Engineering, NER*, vol. 2015-July, pp. 332–335, Jul. 2015.

[147] P. Chorley and A. K. Seth, "Dopamine-signaled reward predictions generated by competitive excitation and inhibition in a spiking neural network model," *Frontiers in Computational Neuroscience*, vol. 5, p. 21, May 2011.

[148] Y. Bi and Y. Andreopoulos, "Pix2nvs: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2017-September, pp. 1990–1994, Feb. 2018.

[149] J. Intoy and M. Rucci, "Finely tuned eye movements enhance visual acuity," *Nature Communications*, vol. 11, pp. 1–11, 1 2020.