



**HAL**  
open science

# Study and design of an energy efficient perception module combining event-based image sensors and spiking neural network with 3D integration technologies

Maxence Bouvier

## ► To cite this version:

Maxence Bouvier. Study and design of an energy efficient perception module combining event-based image sensors and spiking neural network with 3D integration technologies. Micro and nanotechnologies/Microelectronics. Université Grenoble Alpes [2020-..], 2021. English. NNT : 2021GRALT038 . tel-03405455

**HAL Id: tel-03405455**

**<https://theses.hal.science/tel-03405455v1>**

Submitted on 27 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## THÈSE

Pour obtenir le grade de

### DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : **NANO ELECTRONIQUE ET NANO TECHNOLOGIES**

Arrêté ministériel : 25 mai 2016

Présentée par

### Maxence BOUVIER

Thèse dirigée par **Gilles SICARD**, Ingénieur/Chercheur,  
Université Grenoble Alpes  
et codirigée par **Alexandre VALENTIAN**, Ingénieur de recherche,  
Université Grenoble Alpes

préparée au sein du **CEA LIST**  
dans l'**École Doctorale Electronique, Electrotechnique,  
Automatique, Traitement du Signal (EEATS)**

**Étude et conception d'une brique de perception  
efficace énergétiquement combinant capteurs  
d'images événementiels et réseau de neurones  
impulsionnels en intégration 3D**

**Study and design of an energy efficient perception  
module combining event-based image sensors  
and spiking neural networks using 3D integration  
technologies**

Thèse soutenue publiquement le **9 juin 2021**,  
devant le jury composé de :

**Madame Loréna Anghel**

PROFESSEUR, Grenoble INP Institut d'Ingénierie et de Management,  
Présidente du Jury

**Monsieur Michel Paindavoine**

PROFESSEUR, Université de Bourgogne Franche-Comté, Rapporteur

**Monsieur David Bol**

PROFESSEUR, Université Catholique de Louvain, Rapporteur

**Monsieur Gilles Sicard**

INGENIEUR HDR, CEA LETI, Directeur de Thèse

**Monsieur Alexandre Valentian**

INGENIEUR DOCTEUR, CEA LIST, Invité, Encadrant

**Madame Edith Beigné**

INGENIEUR HDR, Facebook, Invitée

**Monsieur Christoph Posch**

INGENIEUR DOCTEUR, Prophesee, Invité



“There is no real ending. It’s just the place where you stop the story.”

***Frank Herbert, Dune***

# Acknowledgements

---

First of all, I hugely thank the jury: Professor **Loréna Anghel** president of the Jury, Professor **Michel Paindavoine**, and Professor **David Bol** for having taken the time to read this manuscript, to evaluate my work. I also thank Ms. **Edith Beigné** and Dr. **Christoph Posch** for having accepted my invitation to the jury and for their fruitful comment during the defense of my work.

Because most of the people concerned are French speakers, most of this section is in Molière's dialect.

En premier lieu, je souhaite remercier chaleureusement mon chef de laboratoire, mais surtout encadrant, **Alexandre V.** qui a largement contribué à faire de cette thèse ce qu'elle est devenue. Il m'a accompagné dans mes moments de doute et d'espoir, et a su m'écouter quand j'en avais besoin. Tu as été un excellent encadrant, tu m'as donné les moyens de parvenir à mes fins, rien n'aurait été possible sans toi.

Ensuite, merci à **Gilles S.** - a.k.a. *The Director* - qui a repris le flambeau, a su me montrer le chemin quand j'étais égaré, et m'a permis d'assister à de belles rencontres scientifiques. Evidemment, merci à **Edith B.** sans qui je n'aurais pas écrit ces lignes. Elle est partie bien vite à l'aventure mais m'a tout de même aiguillé vers de belles contrées.

Je remercie aussi mes collègues et amis, nombreux et mémorables. **Roman G.** et **Valentin E.** qui ont su me sustenter de conseils, de bout de codes, de remontage de moral, et de soirées de qualité. **Alexis B.** qui a apporté ses compétences afin de faire avancer mon étude doctorale avec intérêt et sérieux. **Thomas M.** et **François R.** le duo qui n'a rien à envier à Laurel et Hardy. **Miguel S.** - prononcé *Mighel* - dont l'accent Argentin n'a d'égale que son talent de machine learner. **Stéphane B.** érudit parmi les sages. **David C.** et **Sota S.** sans qui l'escalade est bien plus difficile. **Ivan M.** sans qui mon accélérateur aurait été un ralentisseur. **Luis C.** un ami dans l'adversité, pointilleux mais respectable. **Manon D.** et **Antoine H.** que je n'aurais pas vu longtemps et qui, je le leur souhaite, sauront briller dans leur sujets. **Axel A.** et **Mathieu M.** qui ont assisté à beaucoup de choses, m'ont tendu la main quand j'allais mal, et m'ont apporté un savoir que seuls les sages peuvent se targuer d'avoir. **Stéphane C.**, **William G.**, **M. Reyboz**, **Johannes T.**, **Arnaud V.**, et tant d'autres, qui ont contribué de prêt ou de loin à mes travaux. Merci aussi à **Pascal V.**, qui m'a apporté beaucoup de soutien. Merci encore à Luis, Manon, Antoine, Thomas, Gilles et Alexandre pour leur relecture minutieuse de ce document, pourtant aussi épais qu'un étouffe-chrétien.

I must also thank **Neha M.**, **Daeyeon L.**, and **SangBum K.** who taught me what it means to be a researcher. It is them who triggered my desire to do a Ph. D. study.



Enfin, merci à ma famille.

Ma copine, ma conjointe, que dis-je, ma citoyenne solidaire, **Sido**, avec qui j'ai l'impression d'avoir déjà vécu plusieurs vies. Tu as su me faire chanter alors que je croyais ne plus avoir de musique en moi.

Mes frères. **Benj**, t'as fait les bêtises avant moi, m'a servi de bouclier, et as su m'apporter ta sensibilité quand il fallait, tu es un bel exemple. **Dim**, petit poisson dans un océan bien trop grand, tu sais pourtant parfaitement où aller, tu es chaque jour plus impressionnant, tu m'inspire. Je regrette du trop peu de temps passé ensemble.

Et enfin, à **mes parents**. Sans eux je ne serais pas là (première nouvelle). Même si vous ne comprenez pas totalement ce que je fais, ni pourquoi, vous m'avez accompagné et m'avez permis d'arriver où j'en suis.

Merci.

# Abstract

---

Bio-inspired vision sensors and processors have started to attract attention as, after several decades of research, they start being broadly used for industrial purposes. These sensors, also called event-based, generate sparse data that intrinsically present three characteristics that bring important advantages for many computer vision applications. Indeed, event-driven acquisition permits to generate sparse data, with high acquisition speed at the order of the microsecond, while conserving an exceptionally large dynamic range. Event-driven imagers are thus highly suited for deployment in situations where speed and robustness are important. However, event-based image sensors come with major drawbacks that render them nearly impracticable in embedded situations. They are noisy, poorly resolved and generate an incredible amount of data relatively to their resolution.

This Ph.D. study thus focuses on understanding how they can be used, and how their drawbacks can be alleviated. The work explores bio-inspired applications for tasks where frame-based methods are already successful but present robustness flaws because classical frame-based imagers cannot be intrinsically high speed and high dynamic range. This manuscript provides leads to understand and decide why some algorithms matches more than other to their novel data type. It also tries to touch upon the reasons these sensors cannot be used as they are, but how they could be efficiently integrated into classical frame-based algorithmic pipelines and systems by deploying motion compensation of the raw data.

In addition, a bio-inspired hardware-based solution to simultaneously reduce the output bandwidth and filter out noise, directly at the output of a grid of event-based pixels, is presented. It consists in the hardware implementation of a bio-inspired convolutional neural network accelerator - a *neuromorphic* processor – distributed near-sensor, which takes major advantages from being conceived toward a three-dimensional integration. This system was designed for minimizing its power budget, at the 28nm FDSOI node, and demonstrates a 2.86pJ per synaptic operation – or 93.0aJ per input event per pixel. On top of that, it is scalable for megapixel resolution sensors without induced overhead.

# Résumé

---

Les capteurs et processeurs de vision bio-inspirés ont commencé à attirer l'attention puisque, après plusieurs décennies de recherche, ils commencent à être largement utilisés à des fins industrielles. Ces capteurs, également appelés capteurs « événementiels » ou « à impulsions », génèrent des données éparses qui présentent intrinsèquement trois caractéristiques apportant des avantages notables pour de nombreuses applications de vision par ordinateur. En effet, l'acquisition événementielle permet de générer des données éparses, avec une vitesse d'acquisition élevée de l'ordre de la microseconde, tout en conservant une dynamique d'entrée exceptionnellement large. Les capteurs d'images événementiels sont donc parfaitement adaptés au déploiement dans des situations où la vitesse et la robustesse de l'application sont d'une grande importance. Cependant, les capteurs d'images basés sur les événements présentent des inconvénients majeurs qui les rendent pratiquement inutilisables pour un déploiement dans des systèmes embarqués. Ils génèrent beaucoup de bruits, sont mal résolus et émettent une énorme quantité de données par rapport à leur résolution.

Cette étude de doctorat vise donc à comprendre comment ils peuvent être utilisés et comment leurs inconvénients peuvent être atténués. Les travaux explorent des applications bio-inspirées pour des tâches où les méthodes classiques – basées sur des images pleines - de vision par ordinateur sont déjà efficaces mais présentent des défauts de robustesse qui découlent du fait que les imageurs standards ne peuvent pas acquérir de données à haute vitesse tout en conservant une grande dynamique d'entrée. Ce manuscrit fournit des pistes pour comprendre et décider de pourquoi certains algorithmes s'adaptent mieux que d'autres à leur nouveau type de données. Il aborde également les raisons pour lesquelles ces capteurs ne peuvent pas être utilisés tels quels, mais comment ils pourraient être intégrés efficacement dans des pipelines et des systèmes algorithmiques classiques basés sur des images en appliquant une compensation de mouvement des données brutes.

En outre, nous présentons une solution matérielle bio-inspirée permettant de réduire simultanément la bande passante de sortie et de filtrer le bruit, directement à la sortie d'une grille de pixels à impulsions. Cette solution consiste en l'implémentation matérielle d'un accélérateur de réseau de neurones convolutifs bio-inspiré - un processeur neuromorphe - distribué à proximité du capteur, qui tire parti d'une possible conception en technologies d'intégration tridimensionnelles. Ce système a été conçu pour minimiser son budget énergétique, au nœud FDSOI de 28 nm, et démontre une consommation de 2,86 pJ par opération synaptique - ou 93,0 aJ par impulsion d'entrée par pixel. De plus, il est évolutif et peut être dupliqué en mosaïque pour permettre la réalisation de capteurs événementiels à des résolutions de l'ordre du mégapixel sans induire de surcoût.

# Table of Contents

---

<b>I</b>	<b>Introduction</b>	<b>1</b>
I.A	The Digital World . . . . .	1
I.B	Advanced Machines and Usages . . . . .	2
I.C	System Considered . . . . .	2
<b>II</b>	<b>Building Blocks of Our System</b>	<b>4</b>
II.A	Introduction . . . . .	5
II.B	What are Frame-Based Imagers? . . . . .	6
II.B.1	Pixel Operation . . . . .	6
II.B.2	Readout Pipeline . . . . .	8
II.B.3	Imager Characteristics and Downstream Impacts . . . . .	11
II.C	Event-Based Vision Sensors . . . . .	16
II.C.1	The Biological Vision Pipeline . . . . .	16
II.C.2	Pioneer Event-Based Imagers . . . . .	17
II.C.3	Event-Based Pixel Grid Readout Working Principle . . . . .	19
II.C.4	Event-Based Data Encoding Format . . . . .	20
II.C.5	Current and Future Event Based Imagers . . . . .	22
II.C.6	Next Generation Event-Based Sensors . . . . .	26
II.D	Three-Dimensional Integration Technologies . . . . .	29
II.D.1	Integrated Circuits Manufacturing . . . . .	29
II.D.2	Interests of 3D Integration Technologies . . . . .	30
II.D.3	Interests Related to Image Sensors . . . . .	30
II.E	Neural Networks . . . . .	33
II.E.1	From Biological Brain to Spiking Neural Networks . . . . .	33
II.E.2	Neural Circuits and Neural Networks . . . . .	39
II.F	Conclusion . . . . .	46
<b>III</b>	<b>Analyzing the Environment with Event-Based Sensors and Algorithms</b>	<b>48</b>
III.A	Introduction to the Event-Based Paradigm . . . . .	49
III.A.1	Reminders About the Properties of Event-Based Imagers . . . . .	49
III.A.2	Limitations of Spiking Neural Networks . . . . .	50
III.A.3	The Search for an Application . . . . .	50
III.B	Object Detection with Events . . . . .	51
III.B.1	Clustering and Tracking . . . . .	51
III.B.2	General Multi View Computer Vision Principles . . . . .	53

---

III.B.3	Depth Extraction with Stereo Vision . . . . .	54
III.C	Stereo Vision with an Event-Based Camera . . . . .	55
III.C.1	Spiking Neural Networks for Event-Based Depth Extraction . . . . .	55
III.C.2	Event-Based Depth Extraction with Spike-to-Spike Matching . . . . .	58
III.C.3	Virtual Dataset Elaboration . . . . .	62
III.D	Generate Frame-Based Data from Events . . . . .	64
III.D.1	Introduction to the Principles of Visual Inertial Odometry . . . . .	65
III.D.2	Event-Based Visual Inertial Odometry . . . . .	67
III.D.3	Ego-Motion Compensated Event Frames and Usages . . . . .	73
III.E	Discussion . . . . .	75
III.E.1	The Main Issue of Event-Based Sensors . . . . .	75
III.E.2	Our View of a Smart Event-Based Imager . . . . .	76
III.F	Conclusion . . . . .	77
<b>IV</b>	<b>Spiking Neural Networks and 3D Integrated Event-Based Imagers</b>	<b>79</b>
IV.A	Introduction . . . . .	80
IV.B	Combining What is Best . . . . .	81
IV.B.1	From the World of Imagers . . . . .	81
IV.B.2	From Neuromorphic Circuits . . . . .	83
IV.B.3	Efficient Near-Sensor Filtering . . . . .	93
IV.C	Mapping a CSNN onto a 32x32 Macropixel . . . . .	94
IV.C.1	Definition of the Convolutional Spiking Neural Network . . . . .	94
IV.C.2	From Algorithm to Hardware . . . . .	95
IV.C.3	Number of Pixels and Target Performances . . . . .	100
IV.D	Proposed Architecture . . . . .	101
IV.D.1	Pixel Behavior Emulation . . . . .	101
IV.D.2	Arbiter . . . . .	101
IV.D.3	Transmitter . . . . .	103
IV.D.4	Computer . . . . .	103
IV.E	Results . . . . .	105
IV.E.1	Methodology . . . . .	105
IV.E.2	Input Rate, Chip Frequency, and Power Consumption . . . . .	106
IV.E.3	Discussion . . . . .	107
IV.F	Conclusion . . . . .	110
<b>V</b>	<b>Conclusion</b>	<b>111</b>
	<b>List of Contributions</b>	<b>114</b>
	<b>Bibliography</b>	<b>115</b>

---

<b>Appendix A The Biological Vision Pipeline</b>	<b>128</b>
A.1 The Eye’s Retina Function . . . . .	128
A.1.1 Rods and Cones . . . . .	129
A.1.2 Ganglion Cells . . . . .	130
<b>Appendix B Arbitration Power Consumption</b>	<b>132</b>
<b>Appendix C Details About Selected Recent Event-Based Sensors</b>	<b>134</b>
C.1 Samsung . . . . .	134
C.2 Prophesee . . . . .	134
C.3 CelePixel . . . . .	135
<b>Appendix D 3D Integration Technologies</b>	<b>136</b>
D.1 The End of More’s Law . . . . .	136
D.2 Standard 2D Integration Manufacturing Process . . . . .	137
D.3 3D Processing Working Principle . . . . .	138
D.3.1 Bumps and Through Silicon Vias . . . . .	138
D.3.2 Sequential Monolithic 3D Integration . . . . .	139
<b>Appendix E Training a Spiking Neural Network</b>	<b>141</b>
E.1 Conversion . . . . .	141
E.2 Unsupervised Learning . . . . .	142
E.3 Supervised Learning . . . . .	143
E.4 Considerations About Training a SNN . . . . .	144
E.4.1 The MNIST Standard . . . . .	144
E.4.2 On-Chip or Off-Chip Training? . . . . .	145
<b>Appendix F Methods for Energy Efficient Neuromorphic Hardware Design</b>	<b>146</b>
F.1 Techniques Applied for any NN Acceleration Platform . . . . .	146
F.1.1 Dynamic Voltage and Frequency Scaling . . . . .	146
F.1.2 Approximate Computing . . . . .	146
F.2 Techniques Exclusive to SNN Acceleration Platforms . . . . .	147
F.2.1 Data Skip . . . . .	147
F.2.2 Data Stream Processing . . . . .	148
F.3 Address Event Representation Protocol . . . . .	148
F.3.1 Drawbacks Exclusive to Spiking Neural Networks . . . . .	148
<b>Appendix G EFR-VIO Results Comparison with State-of-the-Art</b>	<b>149</b>

# Acronyms

---

- 3D IC technologies** three-dimensional integration technologies. 28–33, 46, 47, 80, 82, 83, 110–112, 134
- ADC** analog-to-digital converter. 6–11, 13–15, 31, 32, 83, 130
- AER** address event representation. 19–22, 28, 29, 39, 77, 83, 85, 92, 134, 135, 148
- AI** artificial intelligence. 1, 33, 34
- ANN** artificial neural network. 33–35, 39, 40, 44, 84, 89, 91, 92, 141–144, 148
- APS** active pixel sensor. 7, 23, 26
- ASIC** application-specific integrated circuit. 47, 87
- ATIS** asynchronous time-based image sensor. 17, 18, 21, 134
- AU** arbiter unit. 101, 102, 108
- BEOL** back end of line. 137–140
- BSI** back-side illumination. 33
- CD** contrast detection. 27, 134
- CMOS** Complementary MOS. 91
- CNN** convolutional neural network. iv, 41–44, 76, 81, 87
- CNTFET** carbon nanotube field-effect transistor. 136
- CPU** central processing unit. 136
- CSNN** convolutional spiking neural network. 3, 45, 75–77, 80, 84, 86, 87, 93–95, 98, 99, 110, 112
- CT** contrast temporal. 18, 62
- CUDA** compute unified device architecture. 88
- CV** computer vision. 12, 43, 52–56, 58, 64, 76, 78
- CVPR** computer vision and pattern recognition. 26
- DAVIS** dynamic and active pixel vision sensor. 22, 23
- DOD** die-on-die. 138
- DoF** degree of freedom. 65, 71, 73, 77
- DOW** die-on-wafer. 138
- DPS** digital pixel sensor. 11, 15–17, 26, 32, 46
- DRAM** dynamic random-access memory. 91, 92
- DSP** digital signal processor. 31
- DVFS** dynamic voltage and frequency scaling. 92, 146
- DVS** dynamic vision sensor. 17, 18, 23, 61, 62, 134
- EB** event-based. iv, 3, 5, 6, 15–28, 32, 33, 46–54, 56, 57, 60–62, 64, 65, 67, 70–72, 75–78, 80–84, 86, 89–91, 93–97, 101, 103, 105, 106, 108–112, 130, 134, 135, 144

---

**EBB** event-based behavior. 26, 46

**EFG** event frame generation. 67, 73–75, 77, 78, 149

**EFR** event frame reconstruction. 67, 71–74, 76, 112

**EMC** ego-motion compensation. 53, 68–74, 76, 77, 80

**ENOB** effective number of bits. 13

**FC** fully connected. 40, 41

**FD** floating diffusion. 7

**FEOL** front end of line. 137, 139, 140

**FIFO** first in first out. 103

**FinFET** fin field-effect transistor. 136, 140

**fps** frame per seconds. 11

**FSM** finite state machine. 103, 104

**GAAT** gate all around transistor. 136

**GALS** globally asynchronous locally synchronous. 90

**GPU** graphics processing unit. 136

**HAS** high acquisition speed. 3, 26, 28, 46, 49, 50, 77, 80, 81, 111

**HDR** high dynamic range. 3, 15, 26, 28, 46, 49, 50, 77, 80, 81, 111

**Hz** Hertz. 11

**IC** integrated circuit. 25, 29, 136, 137, 139

**IF** integrate and fire. 37, 44, 144

**IMU** inertial measurement unit. 23, 65–70, 112

**IOT** internet-of-things. 2

**IP** intellectual property. 105

**IQ** intellectual quotient. 34

**LIF** leaky integrate and fire. 37, 44, 46, 57

**LSB** least significant bit. 98, 106

**LTP** long-term potentiation. 38, 39

**LUT** look up table. 96, 98, 99

**MAC** multiply-accumulate. 91, 148

**MNIST** modified national institute of standards and technology. 90, 92, 93, 141, 144, 145, 147

**MOSFET** metal–oxide–semiconductor field-effect transistor. 136

**MPX** macropixel. 31, 32, 83, 93, 94, 102, 103, 109, 113

**MSB** most significant bit. 98

**MVSEC** multi vehicle stereo event camera. 48, 59–61

**NN** neural network. 33–36, 39–43, 84, 85, 91, 92, 95, 141, 144, 147

**NOC** network-on-chip. 85, 86



---

**NVM** non-volatile memory. 96, 110, 140

**PCB** printed circuit board. 30

**PCM** phase change memory. 140

**PE** processing element. 32, 85, 86, 89, 100, 101, 103–105, 109, 110

**PPD** pinned photodiode. 7, 8

**RAM** random-access memory. 19

**ReLU** rectified linear unit. 40, 41

**RF** receptive field. 17, 41, 84, 97, 100, 109

**RMSE** root-mean-square error. 68, 71, 73, 149

**ROI** region of interest. 33, 83, 135

**ROS** robot operating system. 65, 73, 74, 77, 149

**RRAM** resistive RAM. 96, 110, 140

**SAE** surface of active events. 59

**SIMD** single instruction multiple data. 83, 84, 136

**SLAM** simultaneous localization and mapping. 13, 50, 65, 67, 73, 74, 80

**SNN** spiking neural network. 3, 33, 34, 37, 39, 40, 43–46, 50, 51, 55, 56, 58, 64, 75, 78, 84, 87–92, 94, 95, 107, 108, 110–112, 141–145, 147, 148

**SNR** signal-to-noise ratio. 10, 12

**SOA** state-of-the-art. 5, 56, 67, 73, 74, 77, 81, 92, 107–109, 140, 141, 144, 148

**SOC** system-on-chip. 30, 136, 140

**SOP** synaptic operation. iv, 38, 81, 91, 92, 107, 108, 110, 112

**SRAM** static random-access memory. 85, 91, 92, 96, 100, 101, 103, 104, 106, 107, 110, 136

**SRP** smallest repeatable pattern. 99, 100, 102, 103, 108, 109

**ST** spatiotemporal. 21, 22, 24, 56, 57, 60, 76, 77, 90, 94, 95, 130

**STDP** spike timing dependent plasticity. 84, 97, 112, 142, 143

**STP** short-term potentiation. 38, 39, 44

**TPU** tensor processing unit. 13, 14, 88, 89

**TS** time surface. 59, 60

**TSMC** Taiwan Semiconductor Manufacturing Programming. 91

**TSV** through silicon via. 29, 138–140

$V_{th}$  threshold voltage. 37, 44, 96, 105

**VIO** visual inertial odometry. 23, 65, 67–69, 71–78, 80, 111, 112, 149

**VLSI** very large scale integrated. 29, 146

**VO** visual odometry. 65, 66, 77

**WOW** wafer-on-wafer. 138

**WTA** winner-take-all. 45, 96, 97



## Contents

I.A The Digital World . . . . .	1
I.B Advanced Machines and Usages . . . . .	2
I.C System Considered . . . . .	2

## I.A The Digital World

You, the reader, are probably aware that in current society, electronic devices are everywhere - otherwise something is wrong with the succession of events that led you to open this manuscript. The digitalization of our world brings many changes, with, as a simple chance

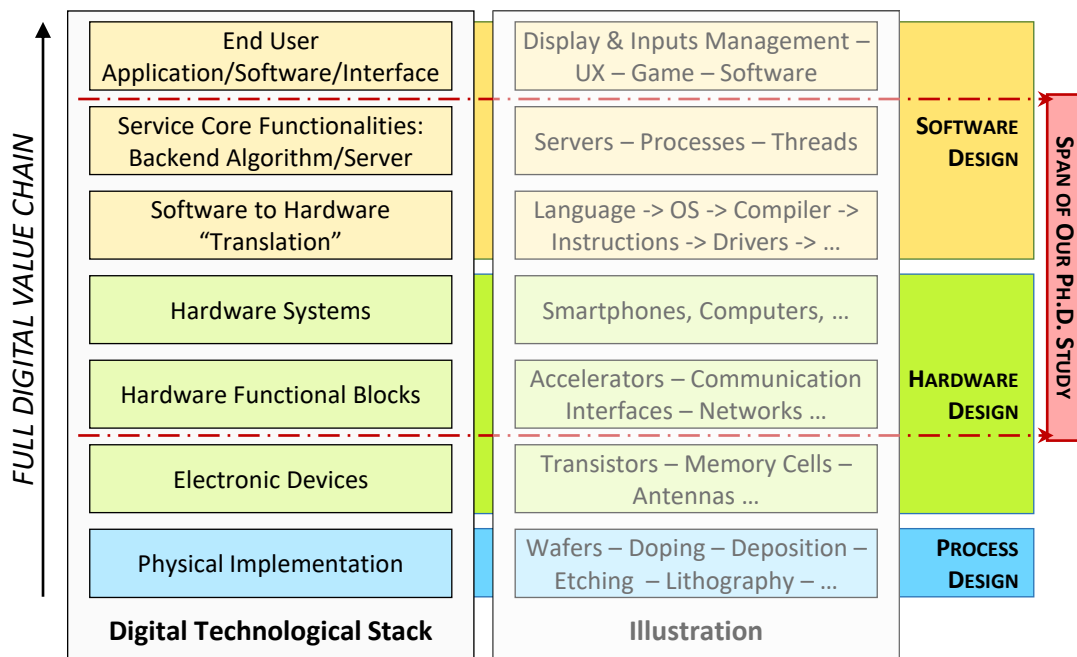


Figure I.1: Illustration of the full digital value chain.

example, the possibility to maintain a quasi-normal economic equilibrium while everyone stays home, better known as the lockdown. Digitalization of the society is happening and is happening fast. Many examples have shown that a few technological breakthroughs can change the world, recent notable ones being the rise of "artificial intelligence" for what concerns end users' usages, and the blockchain as far as finance is concerned. All of this relies on the full technological stack we tried to illustrate Figure I.1. One can distinguish several domains where this value chain applies.

- **Internet and Cloud**  
Computers, Compute Centers, etc.
- **Cyber Physical Systems**  
Cars, Robots, Drones, etc.
- The **Internet-of-things (IOT)** with:
  - Smart Homes  
Smart TVs, Personal Assistants, Connected Dishwasher, etc.
  - Embedded Devices  
Smartphones, Watches, etc.

In our Ph. D. study, we focus on the fields of cyber physical systems and embedded devices.

## I.B Advanced Machines and Usages

Autonomous vehicles, robots, and applications based on augmented and virtual (mixed) realities require the devices they rely on to localize themselves into space. For that we, as humans, integrate a few advanced systems that are the eyes, the inner ear, and proprioception capabilities. These systems can be categorized as sensors, whose data is processed and fused in our brain. They enable us to be aware of our position into space and physically interact with the external world. Even if we consider the operation of localizing ourselves in space natural and obvious, the Ph. D. study presented in this manuscript shows that artificial systems trying to realize the same operation are extremely complex, at both the hardware and algorithmic levels. Measuring the position of an agent into space relies on only a few

	GPS	IMU 6 axis	IMU 9 axis	OUR FOCUS Camera
<b>SPATIAL RESOLUTION</b>	Meter	Centimeter	Centimeter	Centi/Milli- Meters
<b>TIME RESOLUTION</b>	Infinite	Minutes	Minutes/Hours	Hours
<b>COMPLEXITY</b>	Requires Satellites	Small	Medium	Heavy

**Figure I.2:** Devices that permit to evaluate the movement and/or position into space of an agent, and associated advantages and drawbacks.

devices, listed in Figure I.2. In complex systems (as cars or phones), several modules are usually combined for increasing their robustness. Time resolution refers to the maximum duration range during which the algorithm stays accurate.

## I.C System Considered

Deploying cameras for self-localization of an agent in cyber physical systems and embedded devices is especially efficient as many other operations of robots or augmented/mixed realities, are based on visual data - what is called computer vision. However, in many situations, standard-classical-cameras-only solutions are not robust enough for the application at hand.

For example, self-localization at night outside - in the dark - using a standard camera may be a nightmare as they provide low contrast and blurred data.

As researchers marveling at the ability of nature to design incredible systems able to reconfigure themselves during online operation and to operate for incredibly extended periods of time, we decided to focus on bio-inspired methods for evaluating the position of an agent into space. We explore the use of event-based imagers for this task. Event-based imagers have the wonderful ability to provide simultaneously high acquisition speed (HAS), high dynamic range (HDR), and event-based (EB) properties, which would be very costly to obtain with a standard camera. However, they integrate much noise in their data and induce an important bandwidth which makes them hard to benefit from in low-power embedded systems.

We thus tried to find applications relevant for these sensors and proposed a silicon-based bio-inspired filtering module that should permit to solve these genuinely limiting drawbacks directly inside the sensor by exploiting 3D integration technologies. Figure I.3 illustrates the system finally envisioned in our study. The main goal is to reduce the amount of data sent to the costly and heavy algorithm "*Large Compute*", without losing information. The point of the filtering module - "*Small Compute*" - is not only to clear and to compress the raw data, but it also pre-processes and modifies the nature of the data by extracting relevant statistical features inside.

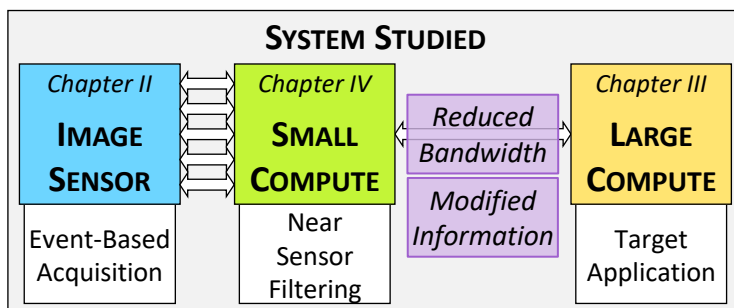
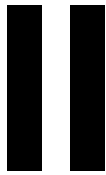


Figure I.3: System studied in our Ph.D.

In chapter II, we present classical imagers and the recent bio-inspired event-based image sensors. This chapter also introduce and discusses spiking neural networks, a bio-inspired version of machine learning algorithms naturally suited for dealing with event-based. Then, in chapter III, we give an overview of various algorithms adapted to event-based data. This chapter illustrates why such cameras are hard to deploy into embedded systems, because of the incredible complexity and amount of data to be managed by such systems. We also give hints as to why the system presented Figure I.3 adds value to event-based sensors. Finally, in chapter IV, we detail the filtering module, based on a convolutional spiking neural network which is made for being efficiently distributed and tiled behind an event-based sensor. This module permits to scale such imagers to megapixel (Mpx) resolutions and acts as a preprocessing layer for downstream algorithms.



# Building Blocks of Our System

---

## Contents

---

<b>II.A Introduction</b> . . . . .	<b>5</b>
<b>II.B What are Frame-Based Imagers?</b> . . . . .	<b>6</b>
II.B.1 Pixel Operation . . . . .	6
II.B.1.a Photo-Transduction and Charge Accumulation . . . . .	7
II.B.1.b CMOS Standard Pixel: the 4T Active Pixel Sensor . . . . .	7
II.B.2 Readout Pipeline . . . . .	8
II.B.2.a Charge Conversion . . . . .	8
II.B.2.b Readout Electronics . . . . .	8
II.B.2.c Acquisition and Readout . . . . .	9
II.B.2.d Analog-to-Digital Conversion . . . . .	11
II.B.3 Imager Characteristics and Downstream Impacts . . . . .	11
II.B.3.a Resolution and Output Bandwidth . . . . .	11
II.B.3.b Dynamic Range . . . . .	14
II.B.3.c Solution: The Digital Pixel Sensor . . . . .	15
<b>II.C Event-Based Vision Sensors</b> . . . . .	<b>16</b>
II.C.1 The Biological Vision Pipeline . . . . .	16
II.C.2 Pioneer Event-Based Imagers . . . . .	17
II.C.2.a Pixel Operation . . . . .	17
II.C.2.b Event-Based Pixel Noise . . . . .	18
II.C.3 Event-Based Pixel Grid Readout Working Principle . . . . .	19
II.C.3.a Arbiter and Handshake Operation . . . . .	19
II.C.3.b Asynchronous Circuit Principles . . . . .	19
II.C.4 Event-Based Data Encoding Format . . . . .	20
II.C.4.a The Address Event Representation Protocol . . . . .	20
II.C.4.b Details About Timestamp and Polarity . . . . .	21
II.C.4.c Spatiotemporal Representation of AER Data . . . . .	21
II.C.4.d Towards Augmented AER . . . . .	21
II.C.5 Current and Future Event Based Imagers . . . . .	22
II.C.5.a A History of the Dynamic Vision Sensor . . . . .	23
II.C.5.b Limitations of Current Event-Based Sensors and Possible Evolutions . . . . .	24
II.C.6 Next Generation Event-Based Sensors . . . . .	26
II.C.6.a Updates Brought to the Readout Pipeline . . . . .	26
II.C.6.b Emulation of Event-Based Sensors . . . . .	28

II.C.6.c	Technology Choice . . . . .	28
II.C.6.d	Other Notable Works . . . . .	28
<b>II.D</b>	<b>Three-Dimensional Integration Technologies . . . . .</b>	<b>29</b>
II.D.1	Integrated Circuits Manufacturing . . . . .	29
II.D.1.a	Traditional Integration Technologies . . . . .	29
II.D.1.b	Three-Dimensional Integrated Circuit Technologies . . . . .	29
II.D.2	Interests of 3D Integration Technologies . . . . .	30
II.D.2.a	Heterogeneous Integration . . . . .	30
II.D.2.b	Parallel Interconnection Capabilities . . . . .	30
II.D.3	Interests Related to Image Sensors . . . . .	30
II.D.3.a	Heterogeneous Integration . . . . .	30
II.D.3.b	Novel Readout Organization Possibilities . . . . .	31
II.D.3.c	Increased Fill Factor for Reduced Noise . . . . .	32
II.D.3.d	Realization Examples . . . . .	32
<b>II.E</b>	<b>Neural Networks . . . . .</b>	<b>33</b>
II.E.1	From Biological Brain to Spiking Neural Networks . . . . .	33
II.E.1.a	A History of Artificial Intelligence . . . . .	33
II.E.1.b	Biological Neurons . . . . .	35
II.E.2	Neural Circuits and Neural Networks . . . . .	39
II.E.2.a	Artificial Neural Networks Presentation . . . . .	39
II.E.2.b	Conceptualization of Neural Networks Operation . . . . .	42
II.E.2.c	Specificities of Spiking Neural Networks . . . . .	44
<b>II.F</b>	<b>Conclusion . . . . .</b>	<b>46</b>

---

## II.A Introduction

This thesis discusses image sensors aimed at realizing advanced visual data analysis, usually called smart imagers or vision sensors. Such devices can nowadays be brought to light and are physically implementable thanks to advances in a wide variety of technological fields, namely computer vision algorithms on the functional side, and imaging and integration technologies on the hardware side.

Relatively recently, thanks to major breakthroughs in the field of 3D integration technologies, stacking several silicon chips on top of one another and render the stack functional has opened the path to advanced designs and constructs. It permits to gain in functionalities and performances at relatively small production cost and promises continuous advances in the field of electronic chip performances.

This chapter gives an overview of the state-of-the-art of technologies regarding event-based sensors, which are at the heart of our thesis. It begins by introducing classical imager section II.B, and then switches to their event-based version section II.C. We then present 3D integration technologies without whom next generation imagers could not be

designed section II.D. To get a better understanding of issues related to imagers in general, classical imager sensors are briefly presented section II.B. Then, we detail the behavior and properties of a different types of imagers, the event-based (EB) bio-inspired image sensors section II.C, which are the focus of our work. We then deliver a few details about 3D integration technologies. We present the advantages it brings for imaging technologies and advanced processor design in section II.D. Finally, in section II.E, we introduce bio-inspired neural networks and discuss why full spiking networks cannot be realistically implemented inside an event-based imager and why it would not be profitable.

## II.B What are Frame-Based Imagers?

An imager can measure and memorize as a digital photograph the luminous information impinging onto its surface. Frame-based imagers - also called *classical* imagers - convert spatial luminance information into digital pictures, which are matrices of values between [0; 255] - encoded onto 8-bits - that represent light intensity at each pixel. We will refer to imagers integrating light to realize human-appreciable photographs of a scene, i.e., most of the image sensors present on the consumer market with smartphones or high-quality cameras, as classical or standard imagers.

More precisely, a classical imager is an array of cells - the pixels - that integrate luminous energy impinging on their photodetectors during a certain amount of time - called period of exposure. The information is extracted from the sensor under digital format, i.e., encoded as packet of binary values, after analog-to-digital conversion by an analog-to-digital converter (ADC).

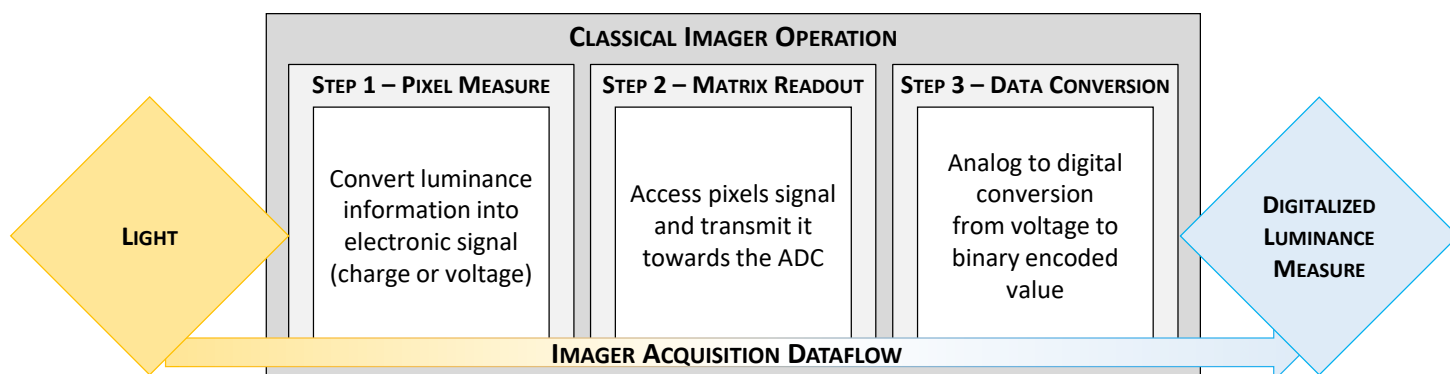
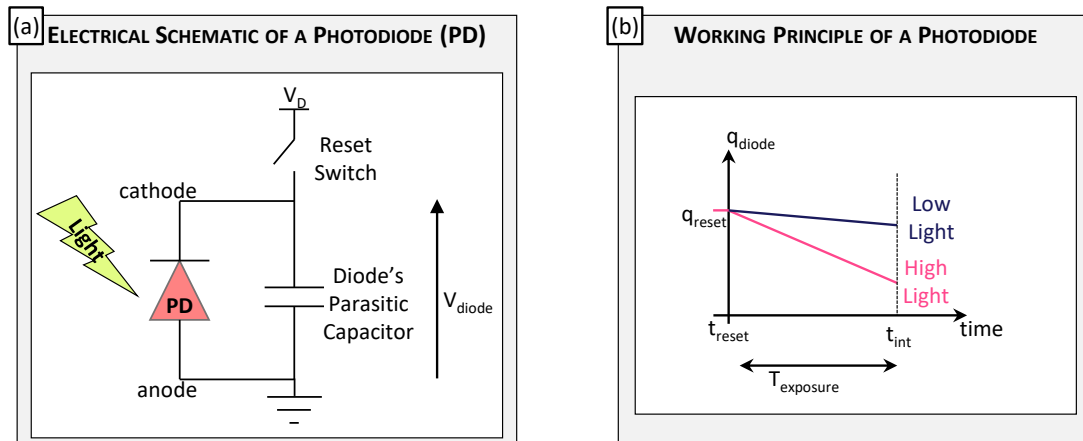


Figure II.1: Illustration of the full operation of a classical image sensor.

### II.B.1 Pixel Operation

A pixel is a light sensitive device composed at least of a photodetector and part of the access, readout and sometimes conversion electronics required for extracting the acquired data from the sensor [1]. We slightly detail the operation of the different pixel components below, by focusing on the more common pixel design, namely the 4T pixel.



**Figure II.2:** Working principle of a photodiode. (a) Electronic schematics analogy of a photodiode in floating mode. (b) Graph representing the charge accumulation inside the photodiode capacitance during the period of exposure for two different light intensities. Adapted from [3].

### II.B.1.a Photo-Transduction and Charge Accumulation

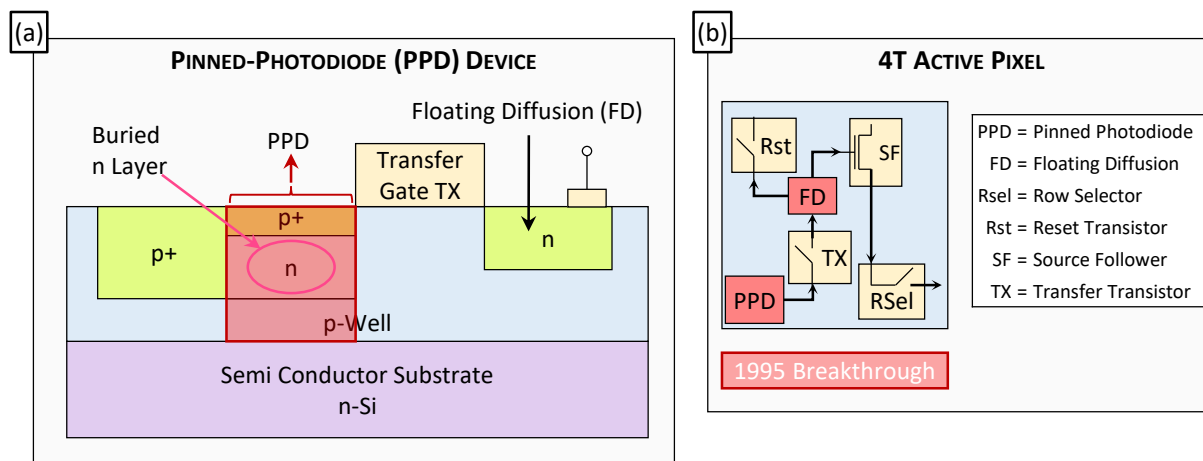
The photodetector is a device, usually a photodiode - that converts human-visible luminous energy into electronic carrier pairs - holes and electrons - resulting in a current flowing through the diode: the photocurrent. When in floating mode, as illustrated Figure II.2 (a), the photocurrent discharges the capacitor that was precharged by the reset switch prior to exposition. The charge accumulated is thus proportional to the time of integration - the exposure time - and the incoming light intensity (photons per seconds) as depicted depicted Figure II.2 (b). So, by setting the exposure time fix for every pixel of the image sensor, each pixel is measuring the light intensity. Which finally result, after readout of the full pixel grid, into a picture in levels of grey. Colors are obtained by applying optical filters onto the pixel devices following a Bayer filter strategy and post-processing operations. This point is not discussed here, and we refer the interested reader to the Wikipedia page concerning Bayer Filters [2] for a first approach of the method.

### II.B.1.b CMOS Standard Pixel: the 4T Active Pixel Sensor

The 4T active pixel sensor (APS) is the most used classical imager pixel design with external ADC. The denomination 4T comes from the fact that this pixel is composed of a pinned photodiode (PPD) - illustrated Figure II.3 (a) - and four transistors. It is an extension of the 3T APS who relied on a simple PN junction photodiode without floating diffusion (FD) - the sense capacitor. These four transistors each play a significant role in the pixel operation.

1. **TX** - Transfers the charge from the photosensitive diode to the floating diffusion well.
2. **SF** - Transfers the voltage of the sense node capacitance - the floating diffusion - to the pixel output.
3. **RSeI** - Allows the transfer of the pixel output value to the column node of the readout matrix.
4. **Rst** - Empty the charge contained in the floating diffusion by applying the reset via the reset transistor.





**Figure II.3:** (a) Schematics illustration of a pinned photodiode processing layers. (b) Associated 4T pixel organization diagram. The pixel is said *active* as it integrates a source follower (SF) transistor, which is actively - meaning that it brings energy into the system - contributing to the read operation.

In a nutshell, the advantages brought by using a pinned photodiode (4T) transistor with respect to a simple photodiode relates to the reduction of the read transfer noise, thermal noise, and a higher sensitivity, especially in the blue. The 4T pixel became the pixel of reference, sometimes even sharing part of its electronics with other pixels to reduce their size or improve their fill factor [4, 5]. More details about the specific characteristics of different pixel types are available in [1, 6].

## II.B.2 Readout Pipeline

Once pixels have integrated light, as explained section II.B.1.a, the information contained in each pixel must be extracted from the grid while keeping the spatial information.

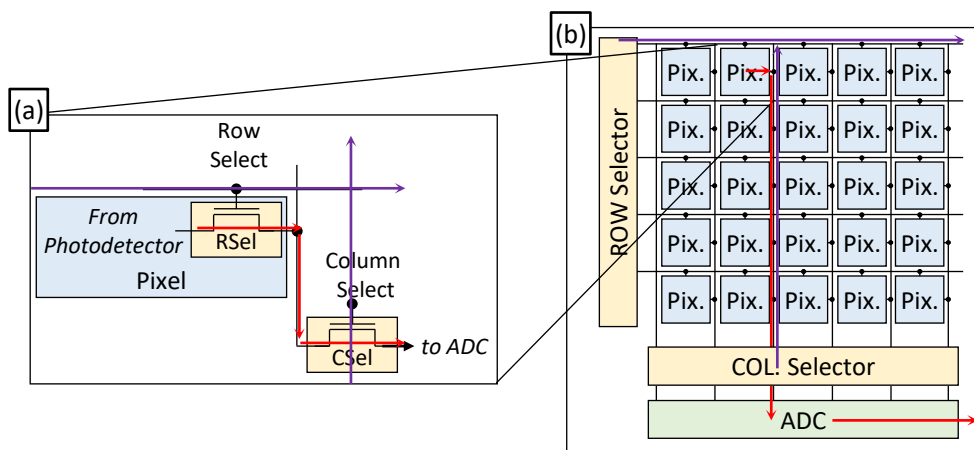
### II.B.2.a Charge Conversion

The integrated charge is converted to current or voltage for reading its value [1]. Reading the charge as a current inconveniently requires the current to cross the full interlines from the pixel to the ADC, passing by a sense amplifier, which induces an important power consumption. Hence, voltage conversion is more commonly used.

### II.B.2.b Readout Electronics

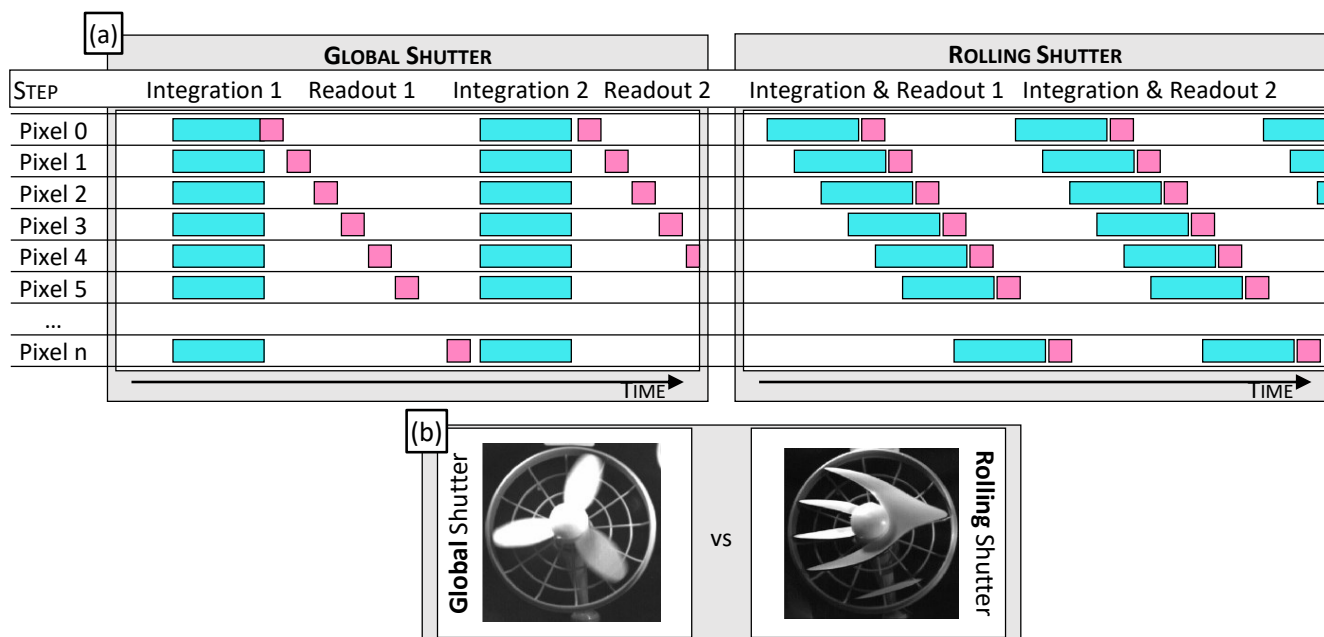
The electronics permitting to access each individual pixel follows a random-access scheme. It consists in having row and column selection transistors. First the row selection line is set high. The  $R_{Se1}$  transistor (see Figure II.4) of every pixel present on this row are thus ON, and the data becomes accessible on the column bus. Then, each column line is scanned, with the corresponding  $C_{Se1}$  transistor set to high, which gives an exclusive access to the specified pixel.

With this electronics circuitry, and because the ADC is shared between several pixels, the full matrix cannot be read in a single instant. So, to have an identical integration time between each pixel, sequential exposition and read operation must be organized throughout the whole pixel grid. Note that in real sensors, the full row is usually converted simultaneously because



**Figure II.4:** Readout mechanism of pixel grid illustration. **(a)** Electrical schematic representation of an imager voltage readout mechanism with row and column transistors selector for random access. **(b)** Schematic illustration of a pixel grid with the command selection modules and the ADC. It clearly depicts the dataflow from the pixel to the ADC (red arrows) with the row and column lines activation (purple arrows) for selection. Note that there are actually two column rails per line, 1 for command, the other for data transmission.

amplifiers, denoisers and ADCs are distributed by column - as depicted Figure II.7 (a.2). But a single encoder outputs each converted pixel signal sequentially column by column.



**Figure II.5:** **(a)** Rolling and global shutters working respective temporal synchrony at matrix level. **(b)** Illustration of the drawback of rolling shutter, the "Jell-O effect" on a picture of a running fan. Adapted from [7] ©2005 IEEE.

### II.B.2.c Acquisition and Readout

Synchronizing temporal sampling of the full pixel grid can be done in many ways, but two are preferred, namely the rolling and global shutters. Each pixel undergoes the following acquisition procedure, in the right order:

1. **Reset:** charge the pixel capacitor at a reference reset voltage..
2. **Exposure:** let the photodetector integrate light information.

3. **Read:** extract the information contained in the pixel and bring it to the ADC, as described section II.B.2.b.

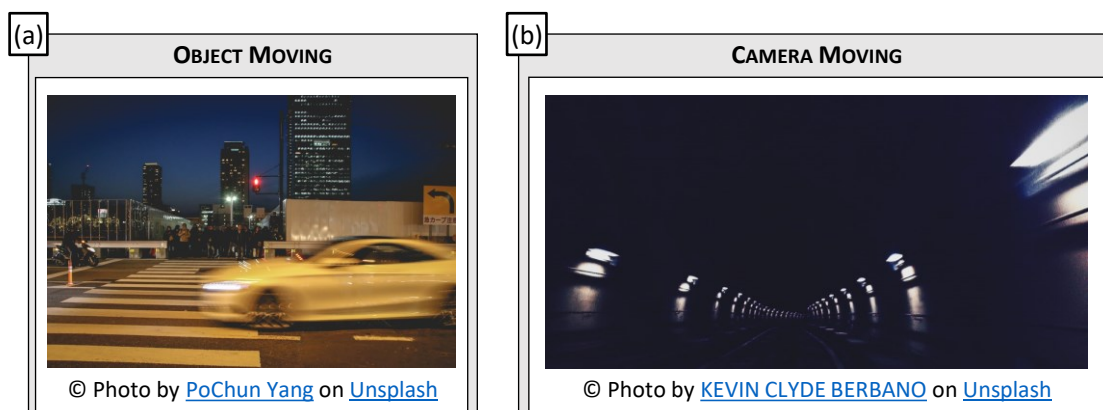
The rolling shutter consists in applying the acquisition procedure to every pixel sequentially, by applying the procedure to the pixel grid row by row. The global shutter consists in applying the reset and exposure to every pixel at the same time and reading the pixels sequentially by scanning the pixel grid. For clarification, an illustration of the temporal organization of the acquisition procedure can be seen Figure II.5 (a).

#### II.B.2.c.i Rolling vs Global Shutters

While rolling shutter is simpler to implement and control, it permits to use simpler - less complex - pixels but induces the "Jell-O-effect" distortion when acquiring moving objects [8], as illustrated Figure II.5 (b). Global shutter solves this problem. However, to apply global shutter, all pixels must be hidden from the scene simultaneously and should be able to conserve the charge accumulated for the duration of full the readout scan.

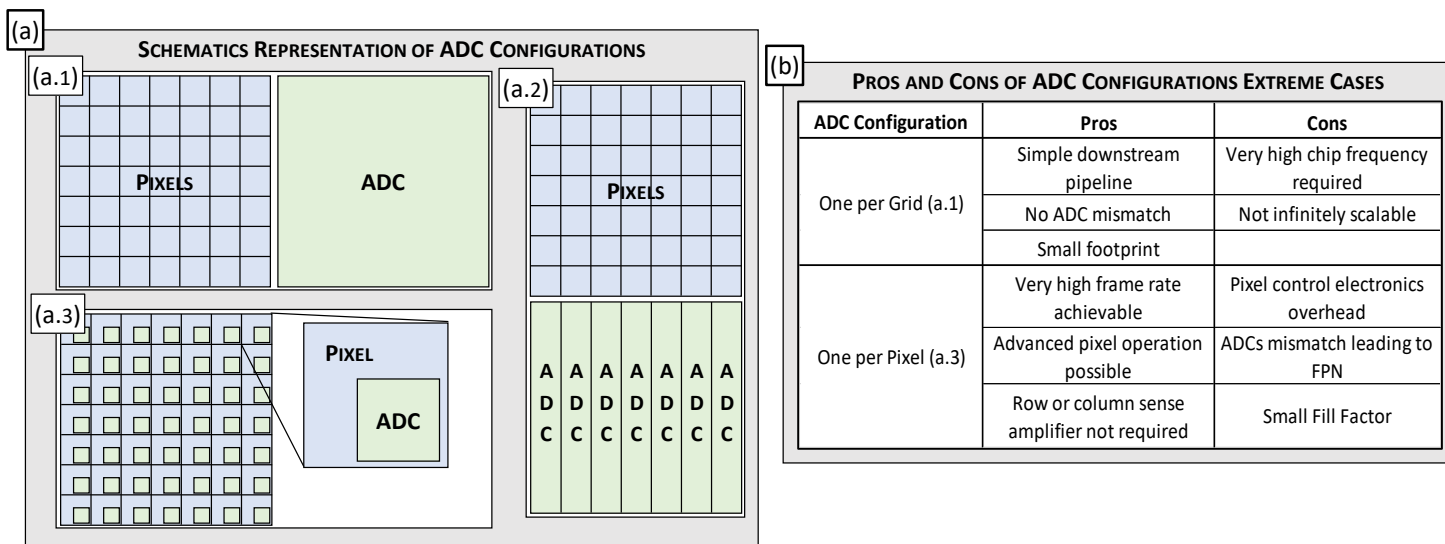
#### II.B.2.c.ii Motion Blur

Because exposure time is not instantaneous - a clearer image is obtained when several photons are integrated -; if the camera is moving, motion blur accumulates onto the picture, as well as objects moving relative to the sensor. This is an important trouble for applications relying on object detection or categorization, especially in embedded autonomous machine situations, where the camera is nearly always moving. Figure II.6 shows pictures acquired with motion blur.



**Figure II.6:** Illustration of motion blur due to: (a) Motion of an object in front the sensor. (b) Motion of the sensor itself - i.e., ego-motion of the camera.

Solving this relies either on complex algorithms, or sensors able to quickly acquire luminous information. However, for fast acquisition (in the order of hundreds or thousands of frames per second), pixel must be of extremely high quality to obtain an important enough signal-to-noise ratio (SNR) and illumination must also be relatively high. So, most classical imagers cannot be used in dark environments, especially sensors with small pixels.



**Figure II.7:** (a) Illustration of different ADC configurations. (a.1) One ADC for the whole pixel grid. (a.2) One ADC per column of pixels. (a.3) One ADC per pixel. (b) Pros and Cons of extreme ADC configurations.

### II.B.2.d Analog-to-Digital Conversion

The digitization occurs through what is called the analog-to-digital converter (ADC). It is a module that converts the analog and continuous current or voltage read from a pixel into a binary word of bitlength  $L_{pix}$  encoding the measured value.

Different configurations regarding the position and number of ADCs are possible, namely a single ADC for the full matrix, one ADC by column, or one by pixel [6]. The latter corresponds to digital pixel sensors (DPSs). Figure II.7 (b) summarizes the characteristics of the grid and per pixel ADC configurations. The column configuration is an intermediate solution that permits to tradeoff and mitigate the advantages and drawbacks of these two configurations. It is currently the most used in 2D image sensors.

## II.B.3 Imager Characteristics and Downstream Impacts

Because of the joint hardware-software orientation of this thesis, we detail here two hardware-related characteristics of image sensor that may impact the performance and complexity of downstream computer vision algorithms. These two parameters are the output bandwidth and the dynamic range of acquisition.

### II.B.3.a Resolution and Output Bandwidth

Classical imagers continuously output full frames at a certain frequency. This rate is regularly referred to as frame rate, expressed in frame per seconds (fps), or sometimes Hertz Hz. The bandwidth - the number of output bit per second (in Gbit/s) - is directly proportional to the frame rate, the number of pixels per frame, and the number of bits encoding a single pixel data. We give an insight of the resolution standard consumer market image formats in the table of Figure II.8 (a).

*II.B.3.a.i Frame Rate Requirements*

The required frame rate of acquisition depends on the final application. We distinguish two types of possible end-usage, human user related purpose and others. The former relates to movies, photograph, etc., where in most cases the required frame rate does not overcome the tens of frames per second. The latter are all other image-related application such as computer vision (CV) etc.

In the case of computer vision applications, the target frame rate depends on the application considered, the backend algorithm and other parameters, with notably the sensor resolution. For example, a quantitative study led by Handa et al. [9] has shown that data acquired at higher frame rate (up to 800Hz) usually implies a better accuracy for the task of depth extraction. This statement holds as long as the high frame rate is obtained without impacting other parameters like pixel' SNR. For that study, they were using relatively medium resolution (VGA scale at 0.31Mpx) cameras.

*II.B.3.a.ii Resolution Requirements*

On top of that, the resolution - expressed in number of megapixels, i.e.,  $10^6$  pixels -, can also drastically impact the performance of algorithms. E.g., reference [10] have shown that high resolution dataset (at 24Mpx) usually permits better accuracy than low resolution dataset (0.35Mpx) using the same 3D mapping algorithm on an identical scene acquired with two sensors - independently of any frame rate requirements. This comes from the fact that each pixel evaluates a non-infinitesimal discrete portion of space (few squared micrometers) and averages all luminous information coming over its photosensitive surface. The space discretisation phenomenon is better understood when looking at Figure II.9. This Figure illustrates that it is more difficult to perceive at low objects and information contained in the picture when resolution is too low.

(a)

Format Name	VGA	720p	1080p	4K	8K
Pixel Grid Dimensions (X x Y)	640x480	1280x720	1920x1080	4096x2160	8192x4320
Resolution (Mpx)	0.31	0.92	2.07	8.85	35.39

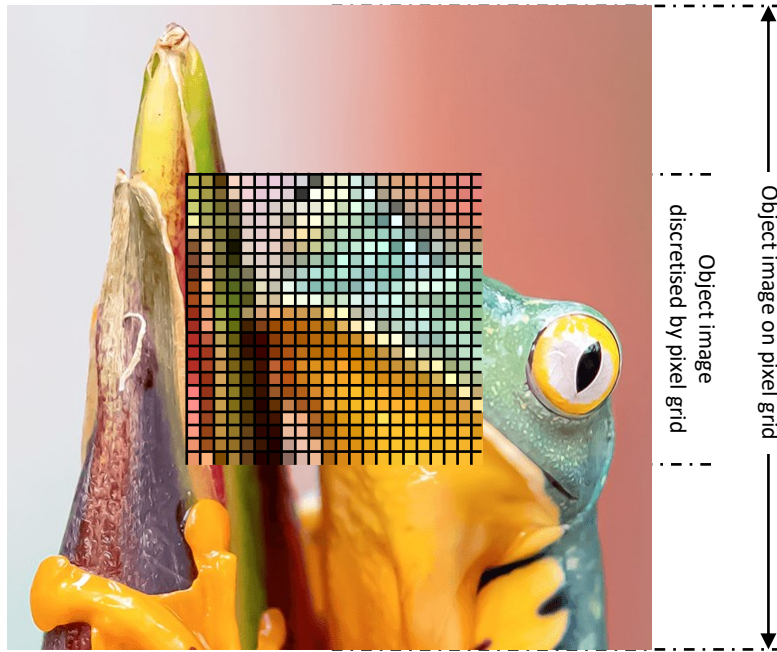
  

(b)

ADC Precision		8b				10b			
Frame Rate (Hz)		30	60	120	240	30	60	120	240
Resolution (Mpx)	0.31	0.07	0.15	0.29	0.59	0.09	0.18	0.37	0.74
	0.92	0.22	0.44	0.88	1.77	0.28	0.55	1.11	2.21
	2.07	0.50	1.00	1.99	3.98	0.62	1.24	2.49	4.98
	8.85	2.12	4.25	8.49	16.99	2.65	5.31	10.62	21.23
	35.39	8.49	16.99	33.97	67.95	10.62	21.23	42.47	84.93

(Coloured cells indicate the bandwidth - in Gbit/s).

**Figure II.8:** (a) Consumer market image format names with associated dimensions and resolution in megapixel values. (b) Evolution of the output bandwidth (in Gbit/s) of classical imagers with the resolution (in Mpx) and acquisition frequency (in Hz).



© Photo by [Stephanie LeBlanc](#) on [Unsplash](#)

**Figure II.9:** Illustration of space discretisation of pixel grids.

### II.B.3.a.iii Bandwidth

One could sum it up as follows: with identical scene and algorithm, the more the resolution  $N_{pix}$  and the frame rate  $r_{img}$ , the higher the accuracy, but the higher the amount of data generated. The bandwidth of a classical imager can be expressed as:

$$BW_{classical} = N_{pix} \times r_{img} \times L_{pix} \quad (II.1)$$

Where  $L_{pix}$  is the bitlength of a single pixel - also called effective number of bits (ENOB). It is the number of bits of the binary words onto which is encoded the pixel value, typically 8 bits in standard image formats like .png. The tables of Figure II.8 (b) shows a few samples of bandwidth values for different parameter settings considering final file encoding with  $L_{pix}$  set to 8 or 10 bits. Bandwidths reaching 10Gbit/s start to be extremely power consuming and thus hard to manage on embedded devices.

### II.B.3.a.iv Data Rate Implications on Frequency and Computation Complexity

At high resolution (above 2Mpx) , the attained bandwidths imply rates on the ADC that are simply not attainable for a single ADC per matrix [6]. That is why the by-column ADC configuration (see Figure II.7 (a.2)) has been preferred over the by-grid (a.3)ADC configuration. It permits to parallelize the data digitization operation and is easy to implement as the ADC is realized on individual pixels and does not require memorization nor neighborhood operations. However, the flexible and programmable operators required for various computer vision tasks, for example simultaneous localization and mapping (SLAM), must be adaptable and cannot easily be distributed by column. So high-bandwidth sensor is a trouble for efficient downstream processing.

To give a more precise idea of the implications of data rates reaching tens of Gbit/s, one can have a look at the tensor processing unit (TPU) of Google [11]. It is application specific



processor optimized for operations on 8b tensors - whom pictures correspond to. In a datacenter environment, the TPU can manage and process a peak input bandwidth of 272Gbit/s while consuming 40W, hence resulting in an energy consumed per bit ( $E_{bit}$ ) of 147pJ/b. Extrapolating this figure to a hypothetical embedded device under the standard 8b, 60Hz, 1080p - considering that datacenter operators are optimized for speed and low power because "a small fraction of a big number can nonetheless be relatively large" [11] - results in a power consumption of 146.3mW. Such an energy efficiency would permit an implementation into an embedded device. However, the TPU achieve this performance at maximum efficiency, i.e., at highest tolerated input data rate, and this number would scale proportionally if this limit would increase.

### II.B.3.b Dynamic Range

The dynamic range of a sensor also plays a significant role in algorithmic performances. It relates to the possibility of an algorithm to be functional under many illumination and movement speed scenarios.

#### II.B.3.b.i Definition of Dynamic Range

The dynamic range of an image sensor is traditionally defined as the ratio between [3]:

1. The largest detectable signal. It is directly linked with the full well capacity of the photodetector, i.e., the maximum number of electrons that can be stored inside the photodiode for a given exposure time. It also depends on the ability of the ADC to convert this full range.
2. The smallest detectable signal. This signal depends on the diverse sources of noises, which are related to the exposure conditions, ADC precision, and the in-pixel non idealities (dark current, reset noise, shot noise, fixed pattern noise, etc.) [3].

On top of that, with this definition, one can identify two types of dynamic ranges, namely inter and intra frame.

#### II.B.3.b.ii Inter and Intra Frame Dynamic Ranges

The *intra frame* dynamic range of an image sensor is the ability to have large variation of intensity measured on the same single picture. It is involved in situations where large variations of illumination occur on the same scene, e.g., a camera exiting a tunnel or an object suddenly reflecting the sun. The *inter-frame* dynamic range of an image sensor is its ability to capture images under global illumination changes. It qualifies the ability of an image sensor to take quality pictures in bright daylight as well as dark night illumination conditions.

The intra frame dynamic range is fixed by the hardware and depends on the pixel's photodetector size, process implementation quality, ADC precision, etc. It can be artificially increased algorithmically but generally requires acquiring several pictures at various exposure times. It can thus not be done at high acquisition speed [12].

### II.B.3.b.iii Inter-Frame Dynamic Range Modulation Possibilities

The inter-frame dynamic range can also be modulated by managing the exposure time of the sensor. However, at night, far less photons impinge on the photodetector for the same duration. Hence, the time required to gather enough light to actually "see" something is larger and can reach up to several seconds. But the longer a sensor integrates lights, the longer it also integrates noise and motion blur. So, pixel noise and motion situation may not enable to use this technique, on top of that longer exposure time means smaller frame rate.

Another - more hardware specific - possibility consists in scaling the ADC gain as a function of average illumination condition. Increasing the ADC gain permits to emulate a longer exposure time by artificially amplifying the accumulated charge. However, noise and dark current are also amplified, so, again, the noise level of a pixel may limit this method. Nevertheless, this method presents two major advantages. First, it does not impact the acquisition frequency, and can even be combined with exposure time adaptation. Secondly, it can be used for small full well capacity diodes and is thus uncorrelated with the intra frame dynamic range of the sensor. So, relatively poor-quality photodiodes can still be used for high inter-frame dynamic range. Furthermore, with reduced full wells the ADC precision can be reduced, thus enabling higher acquisition frequencies.

Mid-range consumer market image sensors usually have limited dynamic ranges and cannot be used under large intra frame illumination contrasts. Neither can they be used in the dark night situations either, without a volley of image processing behind. For applications like autonomous navigation, such inabilities make classical image sensors a source of lack of robustness in computer vision applications. For the rest of this manuscript, we employ the term high dynamic range (HDR) sensors that exhibit large *inter-frame* dynamic range abilities, no matter the method used to obtain it.

### II.B.3.c Solution: The Digital Pixel Sensor

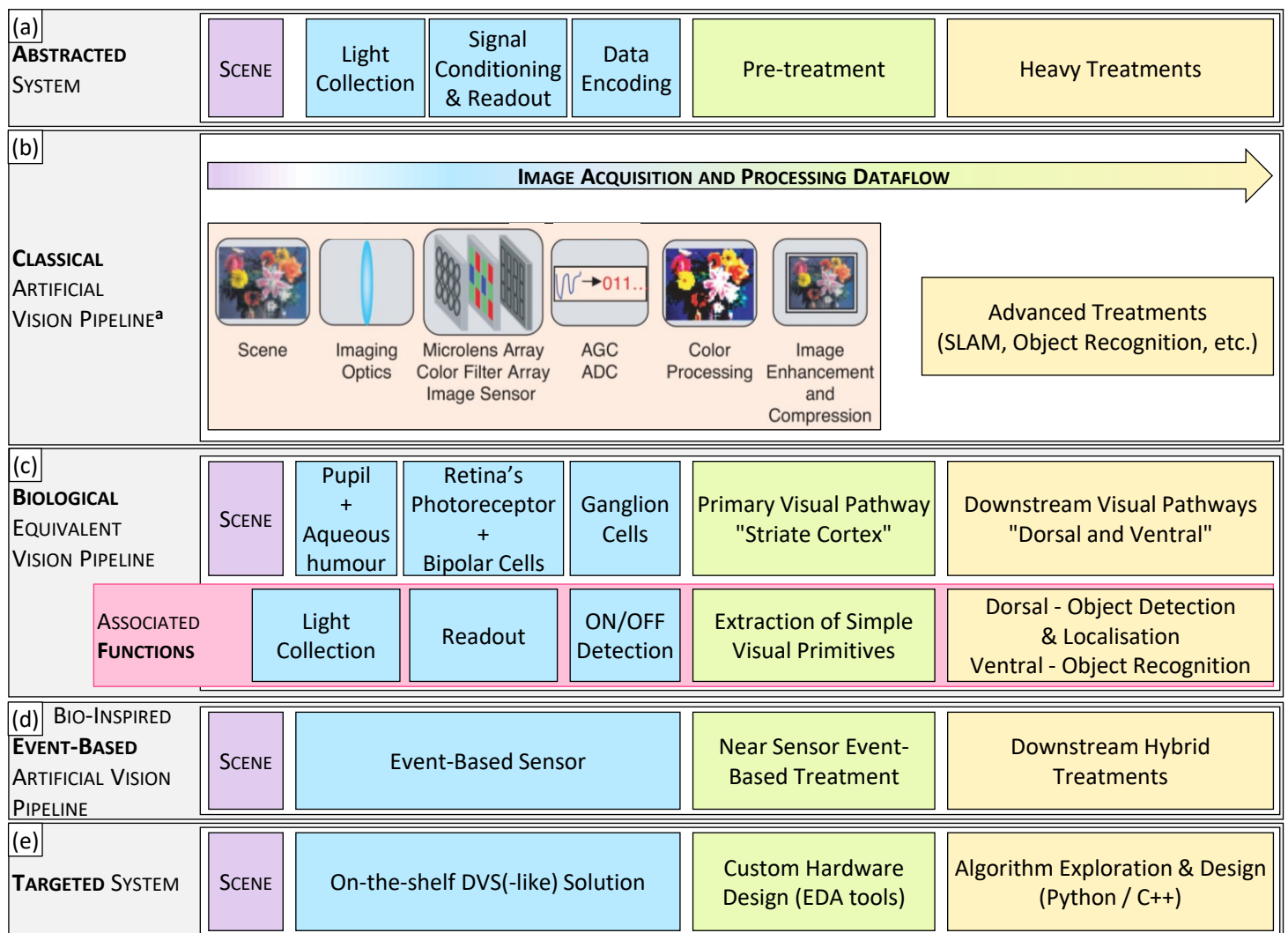
Digital pixel sensor brings forth major advantages for both readout and dynamic range of acquisition. In a digital pixel sensor, every pixel integrates its own ADC, as depicted Figure II.7 (a.3). The required chip clock frequency for sampling is quadratically reduced with respect to the single ADC by pixel grid configuration.

Now, if every pixel also integrates their own memory, the readout of a full matrix can be operated in total de-correlation of the acquisition operation. This enables to reach inter-frames delay of acquisition below the millisecond scale [13]. In addition, with reduced chip clock frequency requirements and increased proximity to the photodetector, inter-frame dynamic range can also be improved because exposure time can be increased while readout noise is reduced. Our thesis focuses on a specific type of DPS, the event-based imagers.



## II.C Event-Based Vision Sensors

**AN INTRODUCTION TO THE EVENT-BASED VISION PARADIGM** Event-based sensors are born from bio-inspiration. Apart from the DPS structure inherited from classical imagers' legacy; the differences between classical imagers and event-based sensors are relatively important. The readout mechanism is totally different in an event-based sensor: it is asynchronous pixel by pixel, not cadenced nor patterned. The data delivered by an event-based sensor does not represent the integration of photon flux onto its photodetectors surface. It is composed of events that indicate when the derivative of the luminance observed by any pixel has overcome a threshold.



**Figure II.10:** The vision pipeline seen from different perspectives. **(a)** Abstract formulation. **(b)** Artificial vision pipeline with classical image sensors. <sup>a</sup>Adapted from [1] ©2005 IEEE. **(c)** Biological mammal equivalent. **(d)** Bio-inspired artificial equivalent. **(e)** System dealt with for this thesis.

### II.C.1 The Biological Vision Pipeline

The whole organization of the human vision processing task is similar to the dataflow of a full artificial vision system, as illustrated Figure II.10 (b) and (c). The genuine biological system begins in the eye, with luminous information collection and conversion to an electrical

potential in the retina. This potential is encoded into a train of electric pulses - a.k.a. spikes - and output from the retina to be propagated downstream to neurons of the striate cortex.

The complete vision system is organized as a succession of neurons specialized to accomplish specific functions based on spatial and temporal information. The neurons of the striate cortex conserve spatial organization of the image projected onto the retina thanks to their organization in pillars of successive layer of neurons, structure called cortical columns [14] - or pyramidal organization. The information inside the cortical column flows mostly one-way. Every neuron in each layer collects information from a small portion/window of the previous layer of neurons, called receptive field (RF)[14], resulting in a final complex function. We direct the reader to the Appendix A for a detailed summary of the working principles of the human eye.

## **II.C.2 Pioneer Event-Based Imagers**

Event-based imagers are built on the idea of realizing an artificial retina up to the ganglion cells [15]. They are artificial bio-inspired imagers that deliver an asynchronous train of spikes at the output, where each pixel applies a temporal filtering on its field of view.

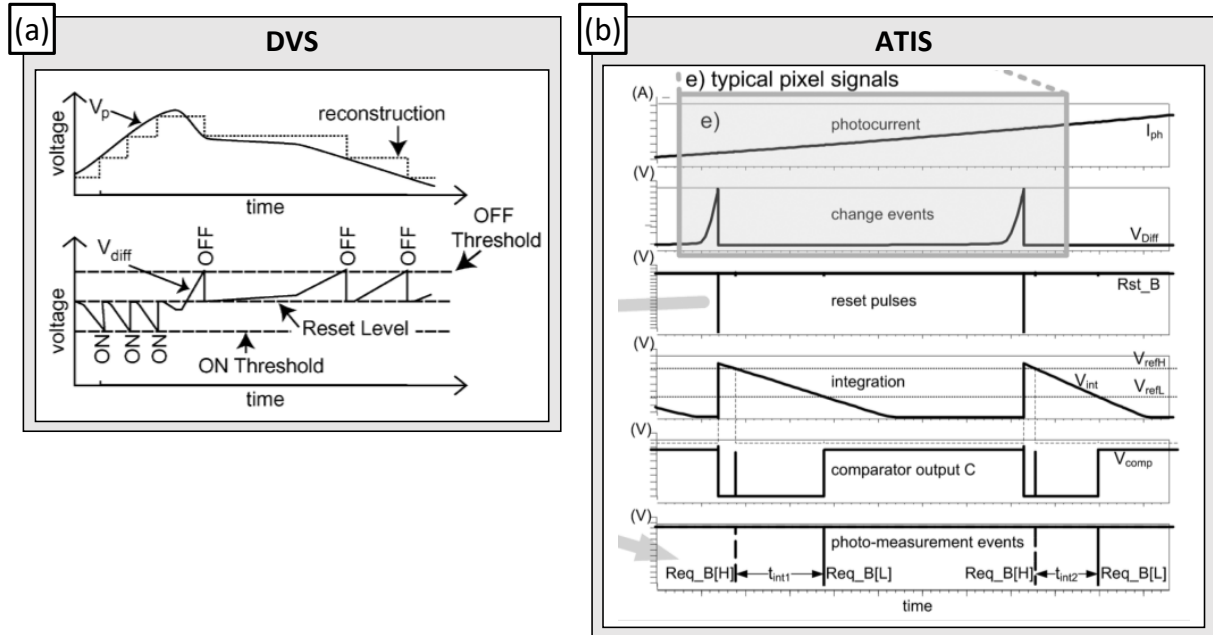
### **II.C.2.a Pixel Operation**

Event-based pixels are asynchronous DPS. Asynchronous because the full readout operation, described section II.B.2.c, is not controlled externally at a certain frequency, but triggered internally by each individual pixel without any clock constraint. They are called event-based because they continuously look for a change in illumination intensity with respect to a previously memorized value and send a signal only when the condition they monitor is verified. Bio-inspired because their operation and output format can be analogized to the human eye and data type, namely electric pulses/spikes.

The difference between event-based pixels and actual biological columns of cells inside the retina lies the spatial filtering operation. Each pixel observes the temporal evolution of impinging luminance, but it does not continuously spike under steady illumination conditions (as is the case with ganglion cells). As depicted Figure II.11, which illustrate the temporal behavior of different EB pixels, the behavior of the ON and OFF ganglion cells is not fully replicated. It is approximated to an all-or-nothing emission that depends on condition verification, instead of the biological spike rate modulation behavior. There are two famous types of event-based pixels, the dynamic vision sensor (DVS) [16] and the asynchronous time-based image sensor (ATIS) [17]. The main difference between the two is the output format.

#### *II.C.2.a.i DVS*

A DVS pixel outputs a single spike for every event indicating that the light intensity has changed, without "counting" the time it took for the integration threshold to be overcome. It also gives the direction of illumination change, called the "polarity". This polarity is usually



**Figure II.11:** Illustration of the electrical behavior of phototransduction operation of two types of event-based pixel: **(a)** the DVS pixel [16] and **(b)** the ATIS pixel [17]. ©2008 IEEE and ©2011 IEEE.

assimilated to the ON/OFF behavior of ganglion cells, but it actually does not contain spatial information, as explained in Appendix A. DVS pixels are sometimes also referred to as detecting temporal contrasts (CT).

### II.C.2.a.ii ATIS

On the other hand, the ATIS pixel gives both the direction of change of as well as the brightness level. At each event, it sends a spike to indicate that a change has been detected and two consecutive spikes whose delay in-between is a direct measure of the integration time. The ATIS pixel is thus composed of two modules: a change detector, operating relatively like the DVS pixel, and an exposure measurement that evaluates the light intensity at the time of spike. Note that for the rest of this manuscript, the distinction spikes and events are considered as being the same thing, and no distinction will be made.

### II.C.2.b Event-Based Pixel Noise

An critical issue with EB pixels, is the noise generated by low quality photo-detection, especially at low average luminance - i.e., in the dark. Indeed, an event-based pixel is applying a logarithmic differentiation on its diode photocurrent to determine if the spiking condition is verified:

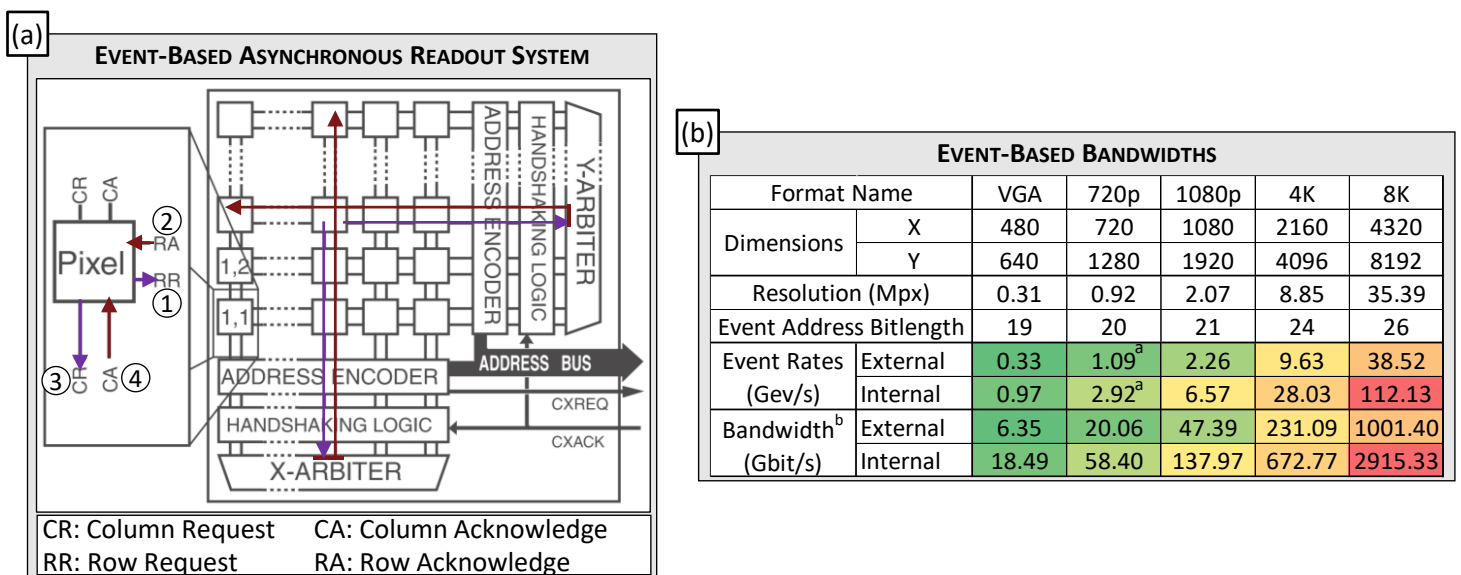
$$(\ln(I(t)) - \ln(I_{ref}))/\ln(I_{ref}) > C_{th} \quad (II.2)$$

Where  $C_{th}$  is the contrast threshold, usually set at around 10%. In low ambient light conditions, the photocurrent is generated with very few photons impinging onto the photodetector and because of the quantized aspect of the photon flux, it evolves by steps. Thus, a single photon could trigger an EB pixel to spike, which may generate an incredible amount of noise.

### II.C.3 Event-Based Pixel Grid Readout Working Principle

As classical imagers, the readout module of an EB sensor is also based on a RAM 2D array. However, the pixels work individually and asynchronously from each other, so they directly "ask" for the permission to output data when it is ready.

A problem arises when two pixels simultaneously send a request signal to the readout module. The decision of which to access first must be made. This is the job of the arbiter module. For a complete study of different possible arbitration processes, we direct the reader to the second chapter of Liu et al. [15]. The two-dimensional handshake protocol for reading out an event-based pixel is illustrated Figure II.12 (a) and described below.



<sup>a</sup>Based on values measured on actual sensor chip in an extreme full pixel grid flickering conditions [18]. <sup>b</sup>Bandwidth required when encoding a spike with only its address - the timestamp and polarity of the spike is not included, meaning that the bitlength is the minimum required when encoding spikes individually (without row or column compression).

**Figure II.12:** (a) Illustration of an arbiter and handshake protocol with a pixel. Adapted from [15] ©2015 John Wiley & Sons, Ltd. (b) Frame versus event frequencies and associated output bandwidths.

#### II.C.3.a Arbiter and Handshake Operation

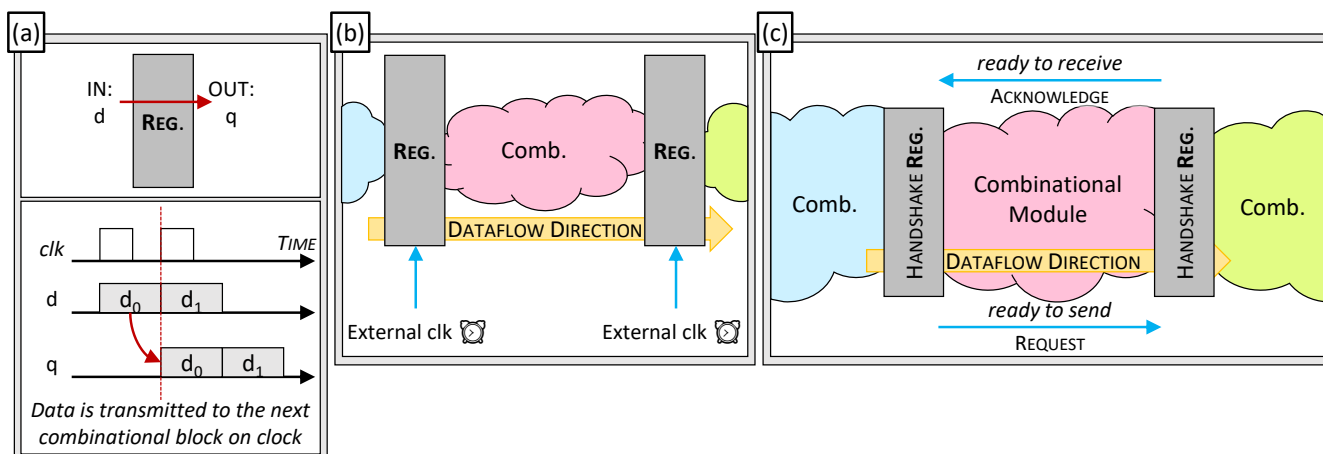
When a pixel spikes, it sends a request to the row handshaking logic, which, when the Y-arbiter authorizes it, sends back an acknowledge signal (RA) to the corresponding row and the address encoder which concurrently encodes the address of the selected row. Once the row is selected, the same process occurs for the column selection. After both the column and the row are selected and acknowledged, an event is emitted on the output bus under the address event representation (AER) communication protocol, see section II.C.4.a This request-acknowledge communication protocol is called handshaking [19].

#### II.C.3.b Asynchronous Circuit Principles

Two-dimensional arbiters are usually designed with asynchronous circuitry protocols [15]. In a nutshell, asynchronous circuit design is different from synchronous counterpart because every combinational block synchronizes themselves using only handshake operation. On the

other hand, synchronous electronics relies on a clock that switches the registers - the Flip-Flops - permitting the data to transit between combinational blocks. In between clock edges, the registers serve as memory that stabilizes the state of the inputs of every combinational blocks. Figure II.13 illustrates a register operation and abstracted views of synchronous and asynchronous circuits.

The biggest advantage of asynchronous circuits is the absence of clock, which permits to drastically reduce the power consumption of the circuit by avoiding uselessly switching registers, as any transistors consumes more energy when switched. However, due to handshake protocol and complex watchdogs' instantiation for guaranteeing quasi delay insensitive (QDI) behavior, an asynchronous circuit takes up around two times more space than its synchronous homologue [19]. So, designing fully asynchronous EB pixels and readout electronics induces a large silicon footprint overhead, and reduces the speed capabilities of large-scale sensors.



**Figure II.13:** (a) Digital electronics register working principle and associated digital timing diagram. (b) Standard synchronous electronic circuit principle. Combinational blocks are operated in between registers that store the states of the variables in input and output of the combination block. The registers take a snapshot of these variables on the edges of clock shared between all the registers of the same time domain. (c) Standard asynchronous electronic circuit principle. It behaves as a synchronous circuit, but the registers update their states based on a completed handshaking protocol between adjacent registers. There is no global clock, registers are synchronized from one to the next.

## II.C.4 Event-Based Data Encoding Format

### II.C.4.a The Address Event Representation Protocol

Spikes are encoded following what is called the address event representation (AER) format[15, Chapter 12]. Each event is represented by a binary word encoding for the address, the timestamp, and the polarity of the spike.

$$(address, timestamp, polarity) \quad (II.3)$$

The address corresponds to the X and Y coordinates of the pixel that emitted the spike. The timestamp is the machine time at which the event was emitted. The polarity is the

sign of the gradient of luminosity change that led the pixel to spike. This encoding format is particularly useful to communicate spikes in between modules. It has been used as a standard in neuromorphic circuits. Note that AER format is also suitable for the three pulses per event of the ATIS pixels.

### II.C.4.b Details About Timestamp and Polarity

#### II.C.4.b.i Timestamp

A timestamp is a time date usually encoded in fixed-point binary representation. It is characterized by the maximum representable duration  $t_{s_{max}}$ , and its unitary precision  $\delta t_s$ . These two parameters are related to the bitlength  $L_{ts}$  of the timestamp with  $t_{s_{max}} = \delta t_s \times (2^{L_{ts}} - 1)$   $\delta t_s$  is arbitrarily set by the chip designer and can be linked to the internal clock frequency of the chip [20].

#### II.C.4.b.ii Polarity

The polarity is encoded as a single bit. If the luminosity variation leading the pixel to spike follows an increase, the event is said to be positive and the value representing the polarity is +1, encoded either as 1'b0 or 1'b1 depending on the chip's convention. And inversely for a decreasing luminance variation leading to an event.

### II.C.4.c Spatiotemporal Representation of AER Data

When considering a group of spikes, it can be represented as a discrete spatiotemporal volume of binary points, as depicted Figure II.14 (a). This Figure is a three-dimensional spatiotemporal representation of classical images and spike data. It illustrates the fine-grained unpredictable sparsity of EB acquisition, depicted as a "cloud" of points. The red and blue colors indicate the polarity of the spikes, positive and negative, respectively.

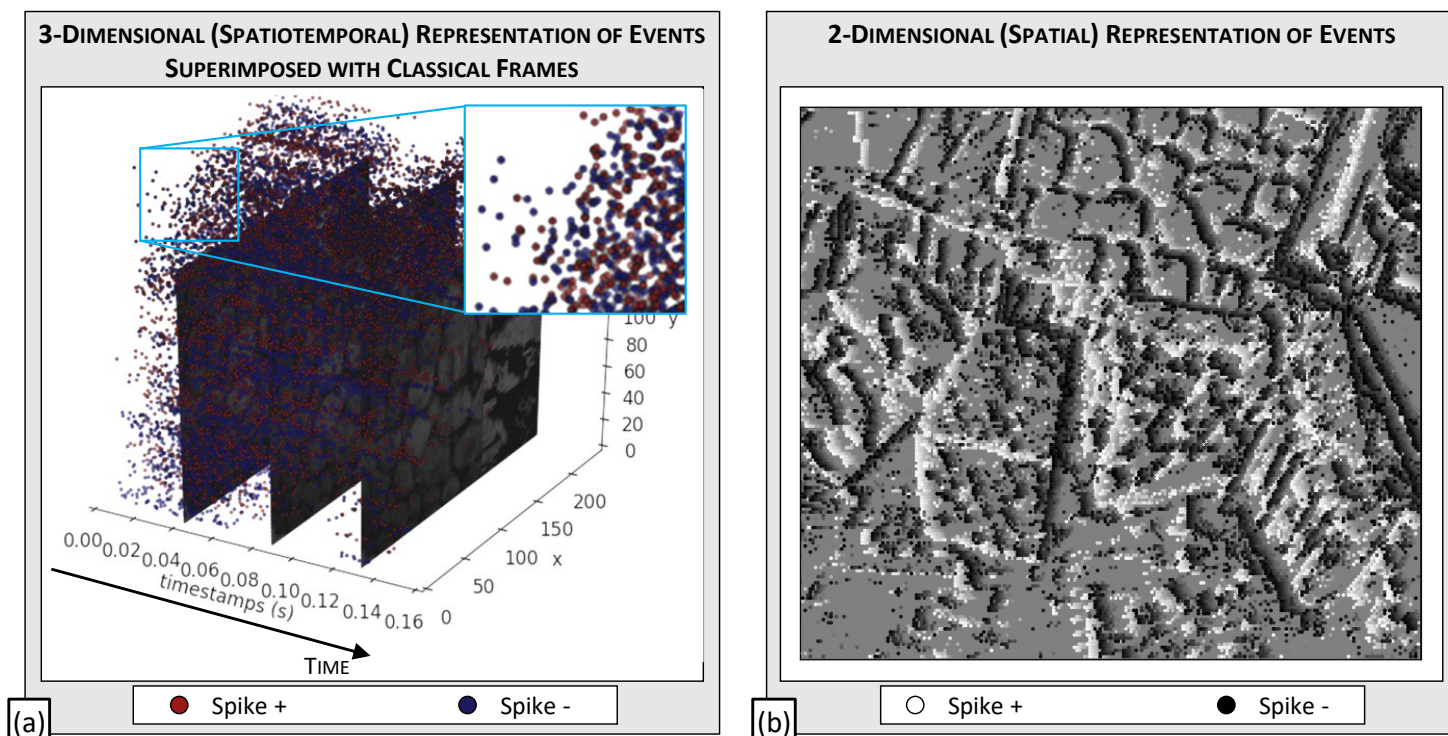
Even if each pixel operates asynchronously, time continuity is not preserved. Time is represented as a discrete value and measured with timestamps, whose precision  $\delta t_s$  is usually in the order of the microsecond [18, 21–23]. This is similar to spatial luminous information discretisation, as discussed section II.B.3.a.ii. In fact, any digital information is represented with discrete values, only full analog electronic systems permit to avoid discrete sampling. So, sensors that interface at their output with digital electronic devices always timestamp their data with discrete timestamps, which is the case of event-based sensors.

### II.C.4.d Towards Augmented AER

The AER protocol as described above can be extended to a broader format. The timestamp of the spike and the address of the emitting pixel - or "neuron" for any other non-imaging neuromorphic system, discussed section II.E - are conserved.

The third element, however, is not always the polarity. It can represent something else, like the luminance level of the pixel at the time of the spike [23]. In non-imaging neuromorphic chips, it can be replaced with a value coding for, for example, a neuron's kernel [25]. Such composition can be called *augmented* (or *extended*) AER.





**Figure II.14:** Illustration of two methods for representing EB data. Data from [21]. **(a)** Three-dimensional spatiotemporal representation of EB data along with associated classical frames acquired from the same scene with a DAVIS240 imager [24]. **(b)** Associated two-dimensional representation of EB data. Spikes are projected onto a 240x128 canvas filled with grey. Every positively (negatively) polarized pixel is represented in white (black) and directly put on the frame based on their (X,Y) pixel coordinates.

The notion of address event representation could be extended indefinitely, for example by adding even more data into a single spike. Under this consideration, the data output by a frame-based imager could be represented under an extended address event representation format. For example, with the address being the address of the imager in a system of sensors, the timestamp the time of the frame, and the third element the matrix representing the picture. In this sense, frame-based data and EB data are close from each other, the difference being the time density of the data. A stream of events could be seen as a very fine-grained frame-based flux, with the significant difference that a frame is a single spike. Nevertheless, for the purpose of this thesis report, we call *AER* the traditional format - the one that includes the polarity - and *augmented AER* a similar but not too complex format that includes a scalar value instead.

### II.C.5 Current and Future Event Based Imagers

Several actors are manufacturing EB sensors, each of them with their own specifications. In general, there is an increasing tendency to give up bio-inspiration for better efficiency or advanced functionalities. Below, we present an overview of the different EB players.

### II.C.5.a A History of the Dynamic Vision Sensor

The DVS was first designed in a lab of the Swiss Federal Institute of Technology in Zurich by T. Delbruck and C. Posch [16], following the path paved by C. Mead [26]. It was then industrialized through a startup called *iniVation* [27]. It is the precursor of EB sensors. They then developed other versions, notably with the intention of providing hybrid imagers that paved the way for modern event-based computer vision, as with the dynamic and active pixel vision sensor (DAVIS) [24]. The latter combines the pixels of a DVS with APS capabilities. It is thus able to acquire and deliver EB data and classical image data simultaneously. Figure II.14 (a) illustrates both types of data obtained with a DAVIS sensor. The interest of such a sensor is dual.

#### II.C.5.a.i Mono Camera System for Mono Camera Applications

First it permits to develop computer vision applications and compare the performances of traditional and novel EB algorithms on the same scene. Rapidly, the DAVIS was adopted as a reference for EB computer vision researchers. Having both type of imaging capabilities within the same pixel grid permits to avoid complex multicamera systems that require precise calibration to evaluate the respective performances of standard and EB algorithms. Indeed, there is no need to position pictures frames in space relatively to each other as a usual Multiview system demands [28].

#### II.C.5.a.ii Correct Acquisition Defects

Secondly, it permits to apply the standardized calibration pipeline to EB data. When acquiring data with a camera, light passes through lens that are circular and not perfect. This generates artifacts, for example:

- **Vignetting** - Pixels at the edge of the sensor acquire less light than pixels at the center resulting in a darkening halo at the edge of the picture.
- **Distortion** - As light rays are refracted by the lens of the collection optics, and because the sensor is rectangular while lenses are circular, images projected onto the sensor are distorted.

If not compensated for, these artifacts introduce errors that accumulate throughout the algorithmic pipeline. It can lead to large errors and even render an algorithm not functional. In traditional computer vision, correcting these defects have been worked on for years and correction tools are openly available and efficient, like the Kalibr suite [29]. But for event-based cameras, which are present on the market for less than 10 years, few solutions exist which does help in designing high-performance EB algorithms. Being able to rely on classical images to compensate for these problems is extremely helpful, hence the quickly acquired reputation of the DAVIS hybrid image sensor. Note that on top of that, the DAVIS camera system also includes an inertial measurement unit (IMU), permitting to work on a wide variety of computer vision applications such as visual inertial odometry (VIO).



### II.C.5.b Limitations of Current Event-Based Sensors and Possible Evolutions

Several advanced functionalities have already been proposed, with for example the integration of color acquisition. Color EB sensor demonstration, specifications, and a sample dataset have been proposed and published in references [30, 31].

IniVation also discussed integrating internal data processing at the pixel grid level for reducing the output bandwidth. They aim at removing noise and dead pixels' outputs, as well as eliminating redundant spikes [32]. For that, they implement spike counters by groups of 2x2 pixels. Binning four pixels together and applying spatial filtering on a sliding time window brings several advantages for cleaning the output data. The idea is that if something occurs in the scene, and if the object is big enough, then more than a single pixel should detect it. It should thus bring more than a single pixel of a 4-pixel neighboring to spike. So, if a single of the four pixels spikes, this event is considered noisy, and no spike is output by the filter. If 2 to 3 pixels spike, a single spike is output by the filter. This eliminates flicker and other unwanted light change conditions that brings an EB readout module to be saturated.

EB imagers can abstractedly be seen as sensors performing differentiation and comparison of luminance through time at the scale of individual pixels. Where every pixel works independently from each other's. Under the same abstraction of thoughts, their classical counterparts are sensors that realize luminance integration at the scale of the pixel grid, where each pixel is readout following a globally organized scheme at the grid level. Staying close to human-inspired function does not mean that the internal hardware operation must stay bio-inspired. The bandwidth and sampling limitations of EB sensors are pushing designers to consider innovative solutions to realize the same function but detach themselves from the imperatives of bio-inspired models, as is discussed further down.

#### II.C.5.b.i Event-Based Sensors and Required Data Bandwidth

The bottlenecks that limit the implementation of high spatiotemporal (ST) resolution event-based sensors are the pixel size, and the speed of operation of the arbiter along with the consequent number of spikes to be output from the imager. Indeed, as explained section II.C.3, a traditional two-dimensional arbiter asynchronously authorizes spikes to be extracted and encoded from pixels sequentially [15]. The arbitration and readout mechanisms along with all downstream operations of an event-based sensor can consume up to 62.2% [33] and 64.8% [24] of its total power consumption. Details on the calculation for obtaining the power consumption of the readout module independently of the power consumption of the pixel grid, as proposed in reference [32], can be found in Appendix B.

Moreover, as illustrated Figure II.12 (b), the required output bandwidth for a spike, when encoding its address onto a bitlength of  $L_{addr} = \text{floor}(\text{Log}_2(\text{resolution}))$ , without timestamp, polarity, or any other data, reaches values quite large with respect to classical imagers (see Figure II.8 (b)). Indeed, a high-performance standard imager with an 8K resolution (35.4Mpx), running at 240fps with 10bit per pixel requires a bandwidth of 85Gbit/s. On the other hand, a

1080p (2.1Mpx) EB sensor, with an average external event rate per pixel of 1.09keV/s/pix - value based on the maximum observed internal pixel frequency of last generation EB sensor [18] - demands an output bandwidth of 47Gbit/s. The resulting bandwidth is 2x smaller for a resolution 17x smaller. With an equivalent 240eV/s/pix this number falls to 10.38Gbit/s, thus a bandwidth 8.5x smaller for a resolution 17x smaller. Going to an 8K resolution EB sensor results in a minimum required bandwidth of 221.1Gbit/s, 2.6x larger than the same resolution and same per pixel frequency than a classical imager, whereas no timestamp nor data is encoded with the spikes.

#### *II.C.5.b.ii The Bottleneck of the Arbiter*

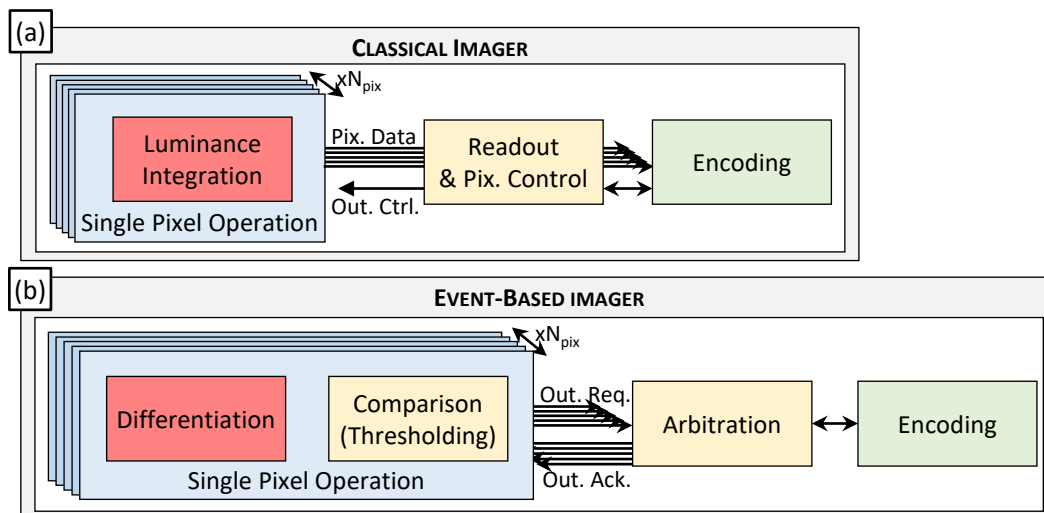
So, data rates demanded by EB sensor are extremely large with respect to classical imagers. And this is only a *side effect* of large scale EB sensors. The genuine limit lies in the sampling rate imposed on the two-dimensional arbiter. At an output event rate of 1Gev/s, a 2D arbiter must work at least 1GHz to avoid saturating the pixel grid - i.e., so that it has enough time to sample a pixel before another "wishes" to spike. Reaching such operating frequencies brings forth two key issues with integrated circuits: size and power consumption. The general relationship between both these metrics and a circuit clock frequency at identical technological node is direct proportionality (see equation F.1). So, the more the frequency, the more the dynamic power consumption and chip area. See section IV.B.2.c.iii, for more details on this topic. On top of that, as EB pixels include a considerable number of transistors - more than four -, EB pixels are wide. The smallest pixel is 5 micrometers wide in the Sony 90nm Back-Illuminated Imaging technology [18].

With this pixel pitch the distance between the leftmost and rightmost pixels nears 5mm for a 1080p sensor. Information moving at nearly the speed of light inside metallic conductors, one can consider that it travels at 0.3mm/ps. Thus, the time required for transmitting the information from a side-most pixel to an arbiter on the opposite side would be 15ps. Which would limit the maximum physically attainable frequency at 60GHz. And this frequency goes down to 1.5GHz for a 4K imager. Designing a 4K event-based sensor with two-dimensional arbitration is thus far from being trivial.

#### *II.C.5.b.iii Toward Less Bio-Inspired Sensors*

To alleviate the bandwidth and arbiter sampling limitations, EB sensors are moving away from retina mimicking objective. The current panel of actors and associated sensor particularities can be summed up as follows.

1. *Prophesee* [34] designed a sensor with asynchronous readout of the pixel grid by row of 1280 simultaneous pixels [18]. They were thus able to realize a 720p (0.92Mpx) sensor outputting 1Mev/s.
2. *Samsung* are designing sensors with synchronous readout by column [35]. While implementing synchronous readout; they bind pixels together to quickly detect spikes along a column. They thus keep an event-based behavior at the system level. It resulted



**Figure II.15:** Abstracted view of the internal function at the system level of (a) a classical imager, and (b) an EB imager. An event-based pixel being a sort of digital pixel sensor, part of it integrates more electronics than a classical 4T active pixel sensor.

in a 1.23Mpx EB sensor presented at the 2019 IEEE Conference on computer vision and pattern recognition (CVPR) [36].

3. *CelePixel Technology* [37] introduced a 1080p (2.07Mpx) sensors in 2019 [23]. They deploy an adapted rolling shutter by row readout strategy.
4. Even iniVation, initially close to bio-inspiration, are now implementing synchronous readout [32]. In addition, they question the interest of a  $1\mu\text{s}$  timestamp precision with respect to a  $200\mu\text{s}$  one in a white paper [38]. They argue that for many applications, low precision timestamping may not reduce the resulting accuracy of the algorithm.

These works clearly illustrates that industrializing an EB sensor requires to re-consider the historical bio-inspiration argument in favor of EB sensors. An exhaustive list of recent event-based sensors characteristics is presented Table II.1, and we discuss a selection of them in more details in Appendix C.

## II.C.6 Next Generation Event-Based Sensors

The examples of CelePixel, Prophesee, and Samsung show that the three intrinsic simultaneous properties HDR, HAS, and event-based behavior - in the sense no data produced when nothing occurs in the scene - are not straightforwardly attainable when trying to reach high resolutions.

### II.C.6.a Updates Brought to the Readout Pipeline

HDR is usually conserved, but the HAS and EB aspect are adapted with synchronous readout strategies. Industrials are ready to tradeoff the inter-spikes temporal precision to obtain instead these three characteristics simultaneously. Sequentially reading out several pixels requesting output at the same time introduces a latency that does not permit to precisely keep track of the time at which a spike is generated. This even appears in published figures, as, for example, the row readout strategy of Prophesee. They announce a  $200\mu\text{s}$  delay [18]

**Table II.1:** An overview of the state-of-the-art event-based imagers.

Manufacturer	Samsung				Prophesee		CelePixel	iniVation	
Sensor Name	DVS Gen4	DVS Gen3	DVS Gen2	DVS Gen1	- (CD)	Gen3 CD <sup>a</sup>	CeleX V	DVXplorer	DVS132
Data From	[36]		[35, 36]	[36]	[18]	[33]	[23]	[39]	[32]
Date	2019	2018	2016	2014	2020	2017	2018	2020	2019
X Res	1280	640	640	640	1280	640	1280	640	132
Y Res	960	480	480	480	720	480	800	480	104
Resolution (mpx)	<b>1.23</b>	0.31	0.31	0.31	0.92	0.31	1.02	0.31	0.01
Die Size X (mm)	8.37	8	8	9.7	6.2 / 10.1	9.6	14.3	-	-
Die Size Y (mm)	7.64	5.8	5.8	8	3.5 / 6.7	7.2	11.6	-	-
Pixel Size (um)	4.95	9	9	9	<b>4.86</b>	15	9.8	9	5
Fill Factor (%)	31	12	20	18	<b>77</b>	25	9	-	20
Dynamic Range (dB)	90	90	90	66	124	120	120	90	-
Lux Min	3	3	3	5	0.04	-	-	-	-
Lux Max	1.00E+05	1.00E+05	1.00E+05	1.00E+04	1.00E+05	-	-	-	-
Min. Detectable Contrast (%)	27.4	27.5	19	19	15.7	12	10	13	-
Acquisition Type	Scan by Col.	Scan by Col.	G-AER by Col.	2D AER	AER by Row	-	2D Scan	Scan by Col.	S-AER
Readout Sync.	Sync.	Sync.	Async.	Async.	Async.	-	Sync.	Sync.	Sync.
Interdata Delay (μs)	1.00E+03	5.00E+02	3.33E-03	1.54E-01	1.00E-03	1.52E-02	-	-	1.00E+03
Eq. Freq. (Hz)	1.00E+03	2.00E+03	3.00E+08	6.50E+06	1.00E+09	6.60E+07	-	-	1.00E+03
Temporal Res. (μs) <sup>b</sup>	1000*	500*	1	-	1	-	1	200	-
#Bit per Event	-	-	7	-	1.6	23	23	-	-
Clk Freq. (MHz)	-	-	50	-	100	-	-	-	50
Max. Out. (Mev/s)	-	-	300	-	<b>1066</b>	66	140	165	180
Power Max. (mW)	140	65	80	15	84	-	470	-	4.9
Power Min. (mW)	140	65	80	15	32	-	390	-	0.25
Imaging Tech. Node	Samsung 90nm BSI CIS				Sony 90nm BSI	180nm 1P6M CIS	65nm CIS	90nm BSI CIS	65nm 1P9M CMOS logic
Logic Tech. Node	?				Sony 40nm Logic	-	-	-	-
Product	N	Y	N	N	N	Y	Y	Y	N
3D Tech.	<b>Y</b>	N	N	N	<b>Y</b>	N	N	N	N
3D Type	Wafer Bonding	x	x	x	Cu-Cu Wafer Bonding	x	x	x	x
APS	Y	N	N	N	N	N	Y	N	N
Color	Y	N	N	N	N	N	N	N	N
Miscellaneous	Group Acces 4pix.	Global Hold	Group Acces 8pix.	Original AER	Event Rate Control	IMU (1kHz)	Optical Flow	IMU (8kHz)	Group Acces 4pix.
	De-noise	Global Reset	Encoding by Col. + Group	-	Dynamic Bit Compress.	-	Intensity	-	-
	Anti-flicker	In Pixel Event Storage	-	-	Events Encoded By Row	-	In-Pixel Timestamp	-	-

<sup>a</sup>CD stands for contrast detection.

<sup>b</sup>Precision of the timestamps.

Stars \* mean that the value shown is hypothesized based on other data.

with spikes timestamped at the microsecond precision while the internal activity of the sensor reaches 2.92Gev/s, corresponding to an average 342ps inter-spike delay.

### II.C.6.b Emulation of Event-Based Sensors

Changing the readout format from asynchronous to synchronous brings forth a major question "*why to do traditional EB sensors?*", in the sense where each pixel realizes a temporal filtering of the luminance. As said by iniVation [39], and as we experienced by using the CeleX cameras for acquiring data, at high event loads the output stream of events follows a grid scan pattern; by row for CelePixel and by column for iniVation. Thus, EB sensor behavior could be emulated with a traditional imager able to generate event-based data from high-speed standard frame format. It could be achieved following this procedure:

- Reproduce EB data format with HAS property by:
  1. Realizing fast acquisition of the order of the millisecond, as is already done for recent EB sensors [18, 23, 36, 39].
  2. Differentiate the acquired intensity frames with one another adjacent in time.
  3. Threshold the difference to obtain a frame of spike.
- Manage its inter-frame dynamic range of acquisition as discussed section II.B.3.b to conserve the intrinsic HDR characteristics of EB sensors.
- Output the matrix under an AER format - possibly compressed as in references [18, 35] - to conserve the event-based nature of the data and avoid outputting anything if the event frame is empty.

Several actors not yet visible on the EB sensor market, nor in the academic research domain, have already patented vision systems that rely on emulated EB sensors [40–42].

### II.C.6.c Technology Choice

On another hand, with the goal of conserving EB pixels as there are, Prophesee and Samsung have passed the step of 3D integration [18, 35]. They have shown that it permits to integrate complex functionalities as well as facilitating the readout operation of the full pixel grid, thus making megapixel resolution sensors achievable. The main advantage - for both classical and EB imagers - of 3D IC technologies is that it enables direct parallel access to pixels. Hence, interconnection organization between the pixels and the readout circuitry can be entirely reconsidered. With 3D IC technologies, accessing many pixels in parallel is easier than before, and thus, combined with synchronous event-based pixel readout, permits to easily realize high resolution event-based sensors [18]. This technology is further discussed section II.D.

### II.C.6.d Other Notable Works

Another type of EB sensor has also been considered by A. Verdant et al. [43]. They present an image sensor operating in the temporal domain, where each pixel would encode the duration it takes to reach a certain voltage threshold. The thresholding operation of every pixel would thus be directly done, but the sensor would run synchronously. They also

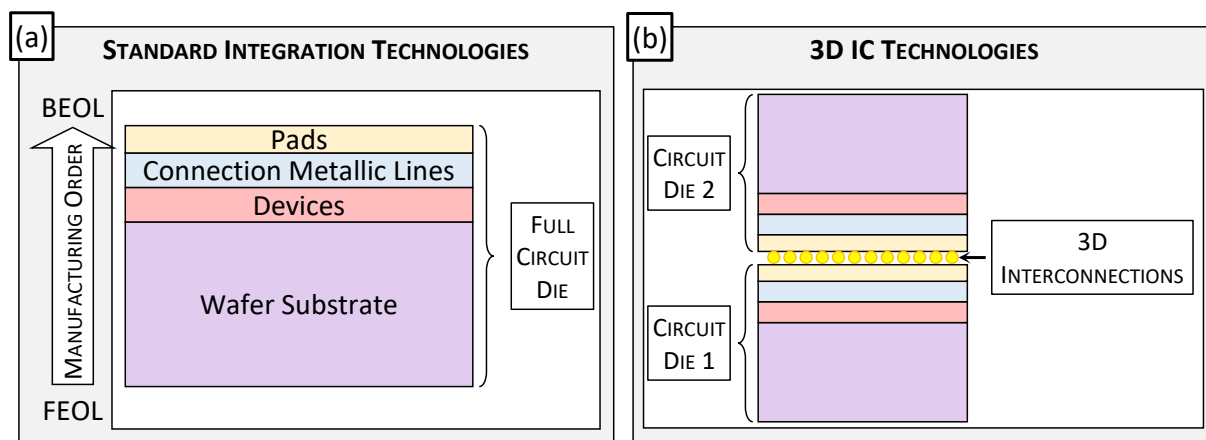
question the need for AER format output and propose a compression method that avoids extracting sparse and mostly empty data frames. Finally, for what concerns frame-based imager, more and more work demonstrate in-sensor compute capabilities[44], as will be discussed later on.

## II.D Three-Dimensional Integration Technologies

### II.D.1 Integrated Circuits Manufacturing

#### II.D.1.a Traditional Integration Technologies

Traditional very large scale integrated (VLSI) circuit technology consists in physically implementing devices - namely transistors, diodes, etc. - onto a solid substrate called a wafer at the micro- or nano- metric scales. These electronic devices are arranged in such a way that at the system level - multi-devices scale - logic functions are operated. The devices communicate together via integrated metallic lines that serves as interconnections. The resulting systems are called integrated circuits, are cut into chips, and composes most of our everyday-life electronics systems. They consist in operational 2D surfaces, with functional devices at the wafer's surface and metallic lines above. 3D integration consists in stacking two integrated circuits on top of each other to realizes system of chips with increased functionalities, as visible Figure II.16.



**Figure II.16:** Illustration of the manufactured layers of: (a) a standard integrated circuit (b) a 3D integrated circuit. Note that the three-dimensional integration technologies require the development of dense interconnection in-between the stacked circuit chips.

#### II.D.1.b Three-Dimensional Integrated Circuit Technologies

For stacking two circuits on top of each other, the main method consists in realizing two circuits on two separated wafers. The uppermost metallic layer present pads that will permit to interconnect the two chips. These pads are then bonded together with metallic bonding using for example copper (Cu) as binder [45]. This technology is usually referred to as Cu-Cu bonding. Other methods exist, such as monolithic 3D integration [46] or through silicon via (TSV) [47]. Detailed information about 3D IC processing is proposed in Appendix D.

## **II.D.2 Interests of 3D Integration Technologies**

### **II.D.2.a Heterogeneous Integration**

Current electronics systems are always designed as the combination of several modules, most of whom are dedicated to a single functionality, for example a mobile phone is the combination of cameras, antennas, battery, processors, etc. Associating several modules onto a printed circuit board (PCB) is more costly in terms of size, weight, and power consumption than 3D stacked modules, as has been demonstrated with the broad development of 2.5D system-on-chips. Co-integrating dedicated systems by minimizing performance degradation because of technological constraints when associating them is one of the main advantages of 3D IC technologies [48]. Note, however, that depending on the application, the interposer may not be silicon-based but may consist in an advanced PCB module. Latest demonstration of circuit chip deploying this strategy is the M1 processor [49], demonstrating highest ever energy efficiency levels with a multi-chip module system-in-package. On top of that, stacking several chips together permits to co-integrate wafers processed in different technologies, for example having an imaging chip and a processing chip, each of them realized with different technological nodes and specificities most suited for their own application. This is called heterogeneous integration, and is also thoroughly used with imagers, which we discuss below.

### **II.D.2.b Parallel Interconnection Capabilities**

Along with heterogeneous integration, 3D IC technologies permit to connect two separate chips with the highest level of interconnection density conceivable. This permits to maximize the number of connections in between modules of chips and avoid designing a single interconnection interface for communicating data from the different modules of the different chips. This opens up new possibilities, with, for example, permitting direct parallel access of a processor to a memory, or distributed access to pixels of an imager, facilitating the parallelization of the readout electronics [50].

## **II.D.3 Interests Related to Image Sensors**

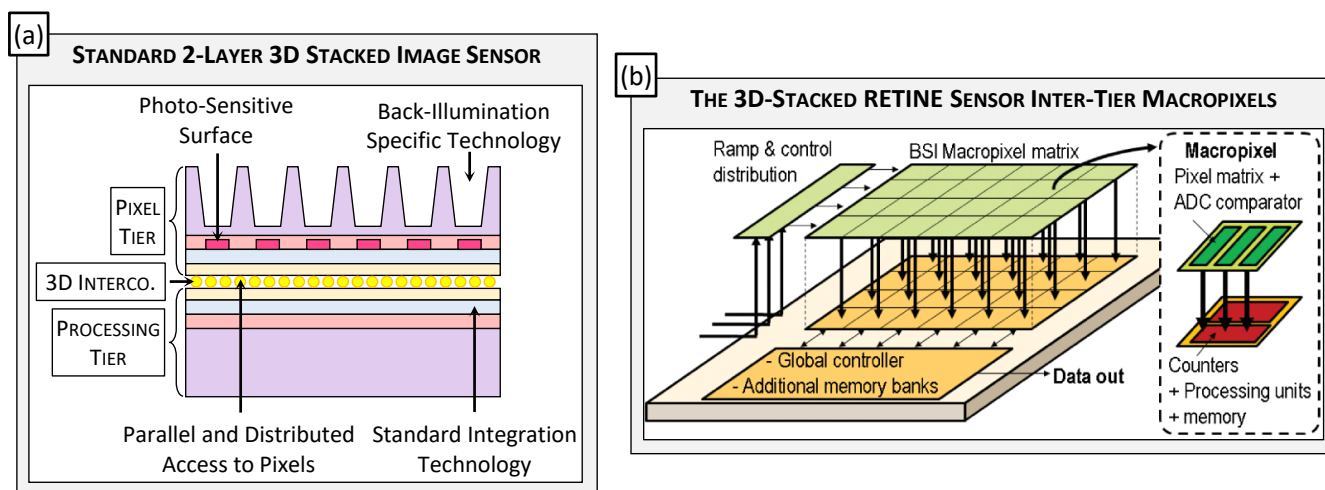
The 3D integration technology enables an enormous range of advantageous possibilities for sensor development. Distributed access to pixels and heterogeneous integration permit to solve most of the sensible points of imager design. Many examples of 3D stacked classical imagers have already been demonstrated. It generally comes with the idea of bringing more image processing capabilities directly inside the sensor chip for better image quality [51, 52], or faster acquisition rate [53], or more recently bring advanced compute capabilities directly inside the sensor [50, 54, 55].

### **II.D.3.a Heterogeneous Integration**

As previously discussed, section II.B, an imager is composed of pixels distributed onto a two-dimensional grid, along with readout electronics and potentially many other modules. It



embeds ADCs, readout control logic, potentially several (photo)-diodes per pixel, digital signal processors (DSPs), color management modules, an input/output manager, interconnections for communication with external hardware, etc. Each of these components takes space onto the chip's floorplan, so the footprint of the sensing part of the chip is far smaller than the full imaging chip area. With 3D IC technologies, any module not directly related to light sensing can be put on a chip separated from the pixel matrix chip. This chip is usually referred to as the digital or processing tier. The rest, all the analog electronic fully dedicated to quality light sensing, sometimes along with the ADC(s), can be kept on a single tier, called the pixel tier. The later can be processed in specific imaging technologies for guaranteeing high quality diode manufacturing, while the digital tier can be processed in advanced (smaller) nodes more adapted to compute electronics. Industrial actors are already using 3D stacked image sensors, with constructs such as the one depicted Figure II.17 (a), and many are even considering developing 3-layer stacked image sensors [56].



**Figure II.17:** Illustration of 3D-stacking for heterogeneous integration of imagers. (a) Schematic representation of the bonding of two different chips. (b) Illustration of the distributed 3D interconnection organization of the RETINE image sensor, adapted from [50] ©2019 IEEE. Its organization by block - called macropixel (MPX) - permits to share ADC between a local neighborhood of pixel and communicate the digitalized data to a local processor efficiently, reducing the latency of the full operation. The macropixel organization can be replicated and scaled at will, resulting in a high-resolution sensor able to realize fast acquisition that embeds compute capabilities.

### II.D.3.b Novel Readout Organization Possibilities

Column-wise readout was already used with classical 2D integrated imagers to reduce problems related to readout bandwidth, as discussed section II.B.3.a. However, as visible Figure II.7 (a.2), this impacts the footprint of the final imaging chip, and does not facilitate the realization of small sensors. With 3D IC technologies, modules not dedicated to light sensing can be "folded" below the imaging chip to gain in footprint [55]. On top of that, thanks to parallel interconnection possibilities, it is possible to access pixels locally. One limitation of that, however, is the inter ADC variability, which may for example result in fixed noise patterns.



### II.D.3.c Increased Fill Factor for Reduced Noise

3D design of imagers also permits to drastically increase the pixels fill factor. Most of the readout electronics can be put on the bottom tier, which lets more space inside the pixels for the photodiode, as was realized in [57]. For example, Prophesee demonstrated an increase in fill factor of more than 3x passing from  $FF=25\%$  on pixels of size  $15\mu\text{m}$  (image node  $180\text{nm}$ ) [33] to  $FF>77\%$  with pixel pitch of  $4.86\text{nm}$  (image node  $90\text{nm}$ ) [18]. Increasing the fill factor also intrinsically reduces the size of a pixel which permits to reduce application limitations due to space discretisation (see section II.B.3.a.ii).

### II.D.3.d Realization Examples

Millet et al. [50] have shown that deploying access to pixel with four pixels per ADC permits to achieve remarkably high acquisition speed for a relatively small energy cost. They readout the matrix of pixel by "Macropixel" (MPX), i.e., by square blocks of 16 by 16 pixels in a sensor called RETINE, represented Figure II.17 (b). Below each macropixel, they integrate a custom stack-based processor with 16 processing elements (PEs) working in parallel for fast and energy efficient largely programmable functionalities. Final data output is thus drastically reduced if needed as the sensor can locally process its data and images does not always have to be output.

Another functional demonstration was realized by Takahashi et al. [57]. They designed a pixel parallel ADC array to readout pixel by blocks of 10 by 16 cells. In this work, the ADCs are not instantiated on the pixel grid - which was the case in the RETINE sensor [50] - but on the second tier below, with Sony's hybrid bonding technology for interconnecting the two tiers. Thanks to heterogeneous integration the pixels are thus fully dedicated to quality light sensing and integration, with large fill factor pixels, etc.

So, the position and parallelism of the ADCs may vary depending on the target sensor configuration, with e.g., ADCs on top [50], ADCs on the compute tier [57], column [55] or block [50, 57] readout strategies. On top of fact, for an ever-better efficiency, Nose et al. [55] designed variable precision ADC to enable on-line adaptation of acquisition precision. They developed by-column access ADCs that enable either quality imaging or reduced precision (4b only) conversion for compute centric purposes. As downstream processing is realized in 4b logic, 10 to 12b ADCs conversions are not necessary. So, the same imager can either operate as an energy-efficient vision chip or a quality imaging chip.

Finally, the first ever 3D integrated event-based sensor has shown many advantages for this type of imaging technologies. Because EB sensors are digital pixel sensors, each pixel directly digitalizes its output data. So, being able to directly access each pixel in parallel with 3D IC technologies, the data can be directly treated, without need for further conversion. As such, Prophesee demonstrated in 2020 a 720p EB sensor with smallest ever pixel pitch of  $4.8\mu\text{m}$ . The readout module - working with a 1D asynchronous arbiter row-wise - takes advantage of the 3D interconnection to readout full rows of 1280 pixels simultaneously. On

top of that, they employ on-chip ROI-based filtering by groups of 40x23 pixels to reduce the activity of the imager where it is not useful application-wise. For example, continuously spiking because of clouds in the sky is not considered as a useful information for autonomous vehicles. Finally, they compress the output spike data by row of, enabling them to reduce data size down to 1bit per output spike when running at maximum event load. They used Sony's heterogeneous 3D integration process with the imaging tier realized with a 90nm back-side illumination (BSI) technology, and a 40nm node for the processing layer. This sensor is the demonstration of the immeasurable advantages of 3D IC technologies for advanced imaging solutions.

## II.E Neural Networks

As EB imagers are bio-inspired sensors an idea jumping out at anyone is to deploy bio-inspired processing on these data. For that, neural networks (NNs) are the most renowned bio-inspired algorithms. Their complexity has progressively increased, and they are now referred to as deep networks (for deep learning) due to substantial number of layers composing them. We distinguish two types of NNs: the classical ones, referred to as artificial neural networks (ANNs); and the event-driven ones - more bio-inspired -, called spiking neural networks (SNNs). The former has been studied since the beginning of machine learning, the latter are more recent, the first reports referring to spiking neural networks being published around 20 years ago [58]. Such processing should be most suited for the data delivered by event-based sensors.

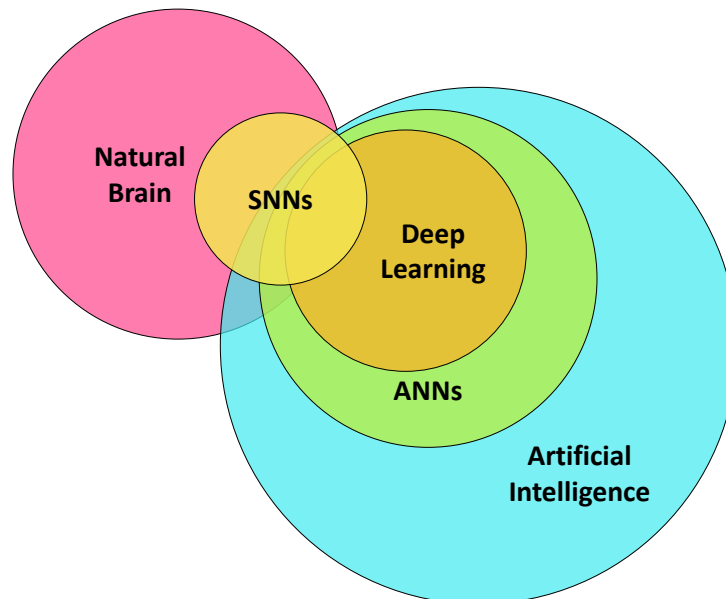
Because of the increase of the computational density of these algorithms, their evaluation on a traditional computer architecture, known as Von Neumann, now requires an extreme amount of time and power [59]. As a result, numerous hardware accelerators dedicated to the efficient evaluation of such algorithms have emerged and are now even available in our smartphones. NNs are based on the biological neural computation scheme occurring in the brain, it is thus no surprise that processors dedicated to their evaluation are also inspired by the brain structure. Such circuits are therefore referred to as *neuromorphic* [26].

This section gives an overview of the mechanisms and cognitive processes occurring in mammal and human brains, mostly from the point of view of a system architect. We then present the operations and relationships of biological neurons along with their artificial counterpart - the spiking neural networks - and discuss their differences with the more abstracted artificial neural networks. From there, we dive into the current issues related to large scale deployment of spiking neural networks.

### II.E.1 From Biological Brain to Spiking Neural Networks

#### II.E.1.a A History of Artificial Intelligence

Fully deterministic programming is not very efficient when it comes to developing general artificial intelligence (AI). The problem lies in the very definition of what *intelligence* is. Human



**Figure II.18:** Schematic representation of the different imbrication of biological, spiking, and artificial neural networks, and position regarding the global scope of artificial intelligence.

intelligence has been studied for years. Intelligence quantification and definition started with the psychometric approach and notably the work of A. Binet and T. Simon in 1905 [60]. They tried to quantify intelligence to quickly find students that may encounter scholar difficulties. Their work led to the elaboration of the standardized and well-known intellectual quotient (IQ) and associated test by D. Weschler in 1939 [61]. Half a century later, psychologists agree that intellectual quotient is not sufficient to define human intelligence. However, they do not agree on a general definition of intelligence [62].

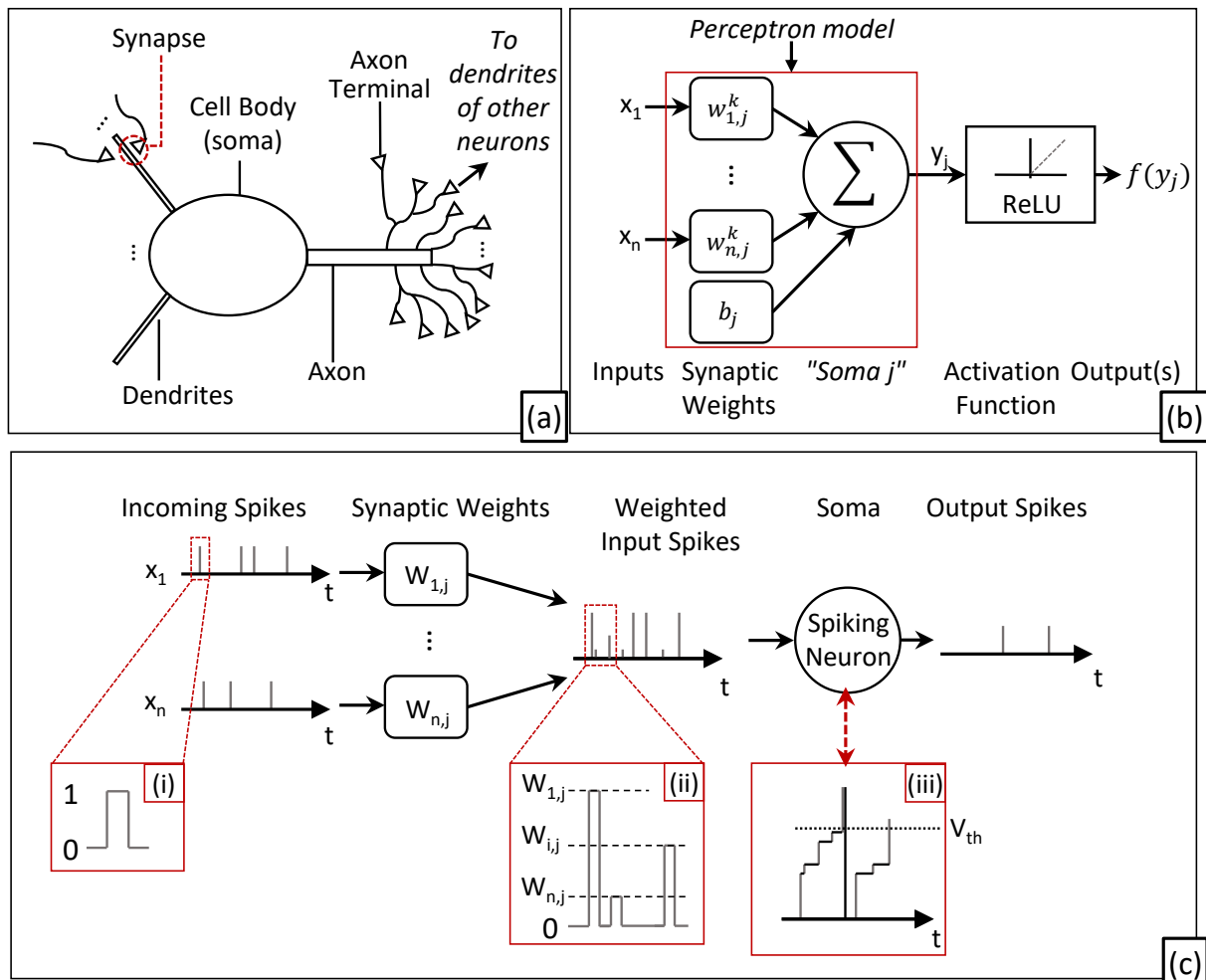
So lies the question "how to define and reproduce something we do not understand clearly?". There exist several strategies for understanding and mimicking the operation of the brain - called cognitive modelling. Connectionism is an extremely popular one because of the numerous algorithmic achievements within the last ten years. It consists in associating "small" computing units into large scale networks to obtain advanced functionalities [63, 64]. It directly comes from the observation that the brain is composed of neurons interconnected together via synaptic connections [14]. From that are born artificial neural networks (ANNs), an abstracted version of biological neural circuits. Spiking neural networks are similar to ANNs in their construction, but they aim at respecting better the brain operation principles.

ANNs have been very much in the news recently as they allowed major breakthroughs, with, as a non-exhaustive list of examples:

- **Question answering** at the Jeopardy! television game in 2011 with DeepQA [65].
- **Object classification** at the ImageNet Large Scale Visual Recognition Challenge in 2012 with AlexNet [66].
- **Go game play** with the world leader Go player beaten in 2016 by AlphaGo which realized moves that were qualified as "non-human" [67].

- **Protein structure prediction** on the Critical Assessment of Techniques for Protein Structure Prediction 14 in 2020 with AlphaFold [68, 69].

### II.E.1.b Biological Neurons



**Figure II.19:** Schematic representation of diverse types of NNs. **(a)** Biological ones. **(b)** Artificial ones. It represents the computational diagram of a non-linear perceptron, as described by Equation II.7, on top of which is added a non-linear activation, such as the represented Rectified Linear Unit (ReLU). The nonlinear perceptron is the building block of most ANN. **(c)** Spiking ones. Incoming binary spikes, received through the synapses, are weighted and integrated. The neuron integrates them and emit in turn binary spikes to downstream units. Inlet (i) represents a binary spike. Inlet (ii) illustrates the synaptic weighting of the spikes. Inlet (iii) shows the integration process on the membrane potential of a simple Integrate and Fire (IF) neuron model.

#### II.E.1.b.i What are Neurons?

A neuron is a specialized cell, mostly present in the brain, which receives information under the form of chemical or electrical potentials, treats it, and - depending on the result - sends some information to other neurons. See Figure II.19 (a) for an illustration of the different elements that compose a neuron. More precisely, a neuron receives inputs from upstream neurons via its dendrites. The cell body - also called soma - integrates those inputs onto their membrane voltage and transmits its output to an axon. Axon terminals are in charge of transmitting the information among many downstream neurons via the synapses - the

terminations of the axons. Under a system architect perspective, a neuron is the smallest computational element of the brain, a sort of biological computing unit.

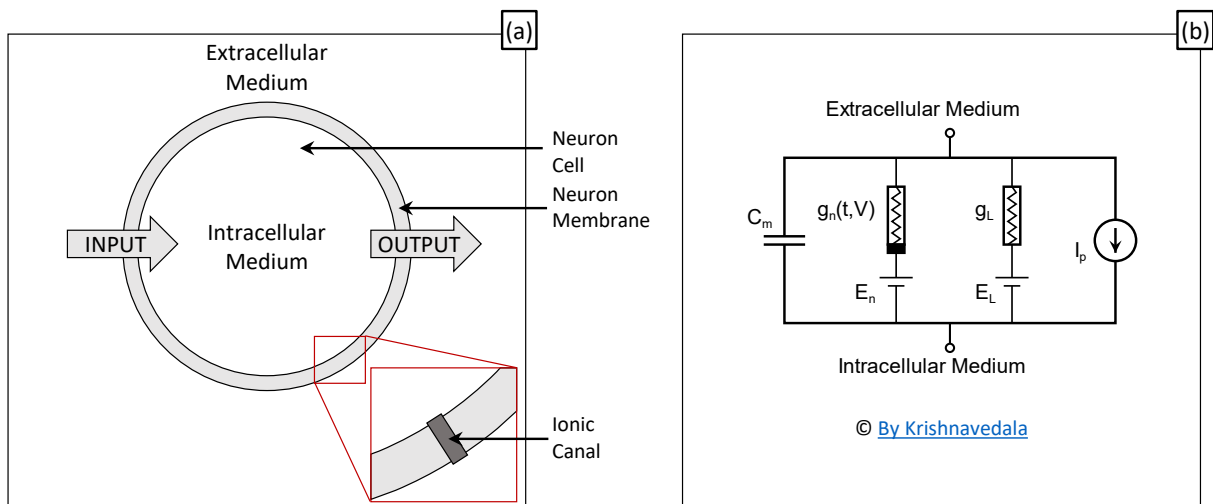
*II.E.1.b.ii Behavioral Models of a Neuron*

Under a connectionism approach, modelling neurons is necessary for emulating the behavior of the brain. Thus, there exist many models that describe the operation of the neuron's soma by defining the relationship(s) between its inputs and outputs.

**HODGKIN AND HUXLEY ELECTRICAL MODEL** According to Nobel prized Hodgkin and Huxley [70], a neuron can be electrically modelled based on the input and output voltages around its membrane. It is called the Hodgkin-Huxley (HH) model based on the authors' names. The membrane is represented as a simple electrical capacitance  $C_m$  as depicted Figure II.20 (a). The information flow inside the neuron depends on the various input and output ionic fluxes. The membrane equation can thus be expressed with a system of differential equations that characterizes the different sodium (Na) and potassium (K) ionic fluxes  $I_i$  through the cellular membrane, as:

$$C_m \frac{V(t)}{dt} = - \sum_i I_i(t, V) \tag{II.4}$$

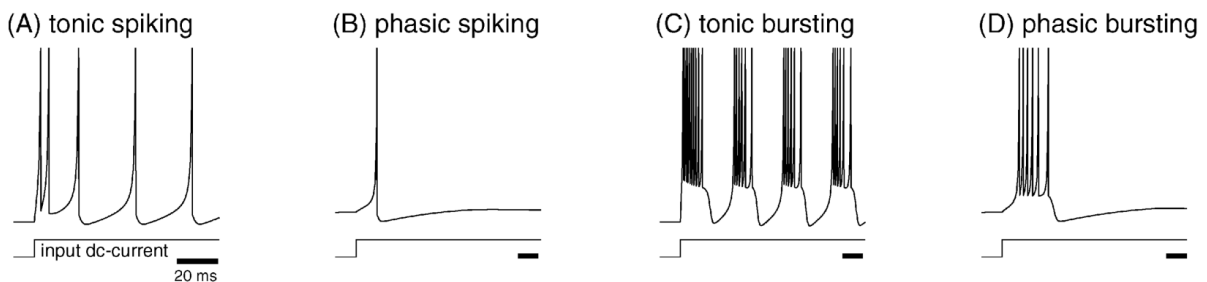
The corresponding schematic representation of the HH electrical model of a neuron is depicted Figure II.20 (b). For more details about the different currents and their expressions, we refer the reader to the original article [70]. The essential point of the HH model, which is useful for understanding both artificial and spiking neural networks, is the current summation principle: A soma continuously operates as a current integrator of its inputs. This is usually taken as true for any neuron model, be it either an artificial or an event-driven one.



**Figure II.20:** (a) Schematic representation of the cell body of a neuron. (b) Associated Hodgkin and Huxley electrical model.

**IZHIKEVICH MODELS** The Hodgkin and Huxley model staying general, many works have discussed and analyzed the computational abilities of various - more specific - models. The contribution of E. M. Izhikevich [71] is relatively famous; for it quantifies the compute

abilities of twenty models of spiking neurons. Any model considered can also be represented by the HH model. A sample of these spiking behaviors in response to a step current function as input are depicted Figure II.21. As can be seen, the behaviors are complex, and mathematically representing them is an arduous computational task. Indeed, these models are multi-input modules that evolve through time. Simulating them thus requires an important computational power. So, with the aim of reducing simulation complexity, abstracted models have appeared and have become mostly used in SNNs for deep learning purposes.



**Figure II.21:** Illustration of different quantified output spiking behaviors for a single current step as input.

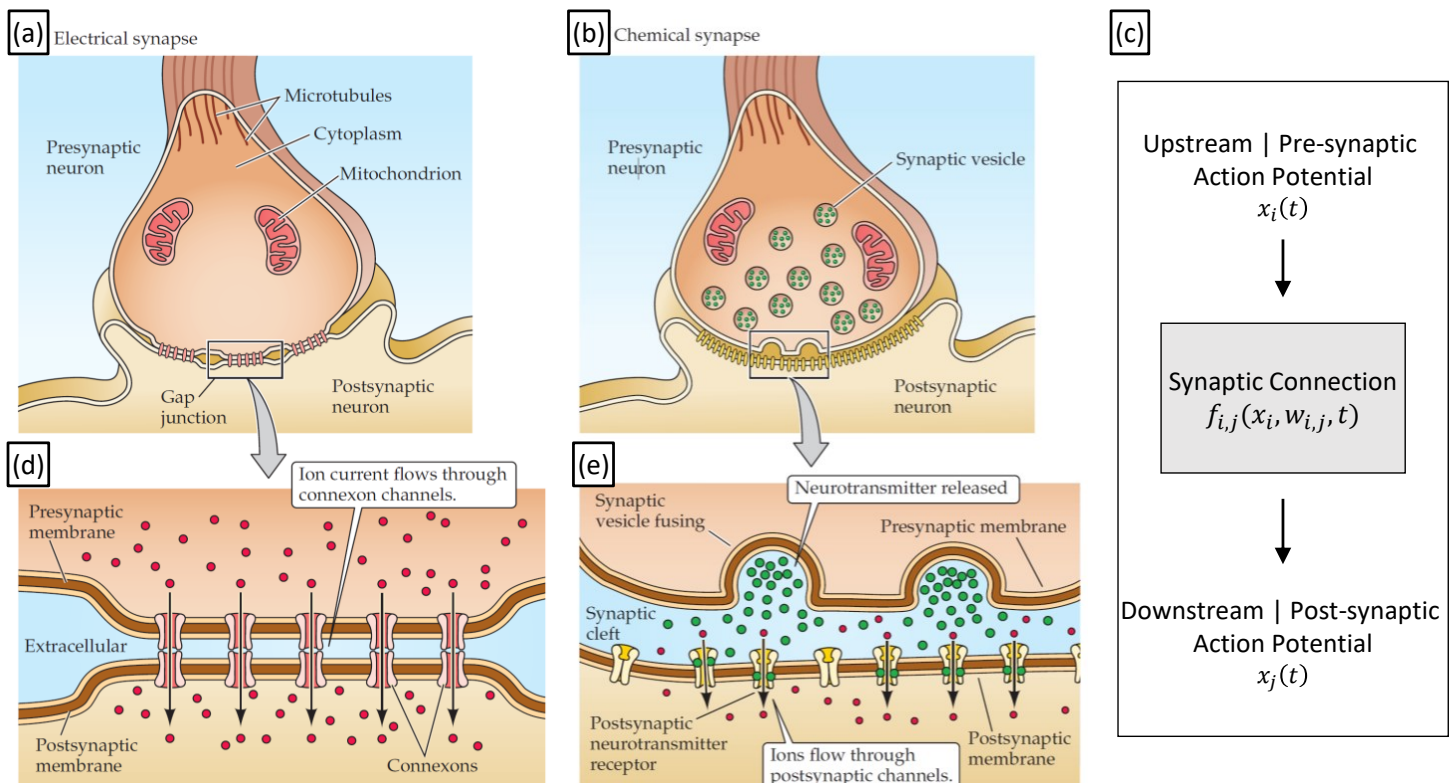
**THE INTEGRATE AND FIRE MODEL** Going to an elevated level of abstraction, a spiking neuron can be seen as a module that simply integrates its inputs over time. A spike is seen as a potential value that arrives at a certain time, and the potential value is directly added onto the membrane potential of the neuron. The integration continues throughout time and once the membrane potential of the neuron reaches a certain value, called the threshold voltage, the neuron sends a spike to downstream neurons in return, it is said to *fire*. This model is called the integrate and fire (IF) model and is broadly used in spiking neural networks hardware implementations. It does not account for the advanced spiking behaviors described above (bursting, phasic slow down, etc.). However, it is a simple model, as it does not take into account complex neuron dynamics and temporalities, and is thus preferred for simulation and deep learning purposes. As is the slightly evolved leaky integrate and fire (LIF) model, presented below in section II.E.2.c.i.

### II.E.1.b.iii Communication Between Neurons

The connection between several neurons is done via the synapses. A biological synapse is the place where an axon terminal of an upstream neuron rejoins the dendrite of a downstream one. The notions of upstream and downstream corresponds to the direction of the information flow within a neural circuit. They are also often called pre- and post-synaptic neurons.

**SYNAPTIC CONNECTIONS IN BIOLOGICAL NEURAL CIRCUITS** There are two types of synapses, chemical and electrical synapses. In a chemical synapse, the axon terminal transmits a post-synaptic potential - the action potential of the upstream neuron - to the downstream neuron by releasing chemical neurotransmitters into the extracellular medium. An electrical synapse communicates with the post synaptic dendrite with ion current flow through direct connexon channels.





**Figure II.22:** Illustration of the different types of synapses, electrical (a) and chemical (b) synapses, and their mathematical analogy (c). Insets (d) and (e) illustrate more clearly the information transmission mechanisms between the axon terminal and the post-synaptic dendrite of both synapse types. Adapted from [14] ©2018 Oxford University Press.

**MATHEMATICAL MODEL OF SYNAPSES** Both synapse types can be represented by a simple mathematical analogy, whose computational diagram is represented Figure II.22 (e). The synaptic operation is characterized by a modification of the pre-synaptic potential  $x_i(t)$  by the synaptic function called the synaptic weight  $w_{i,j}(x_i, t)$ . The weight characterizes the "strength" of the synaptic connection. If the post-synaptic potential is higher than the pre-synaptic one, the synaptic connection is strong, meaning that the pre-synaptic neuron has an important influence for the post-synaptic one. And inversely for a weak connection.

**TEMPORAL EVOLUTION OF THE SYNAPTIC POTENTIAL** The connection weight/strength is characterized by two major processes, the long-term potentiation (LTP) and short-term potentiation (STP) [72]. LTP is responsible for setting the weight on a long timescale. It is involved in long-term memory and hardwired functions through life-long learning. If the strength of the synaptic connection is important, it means that the LTP weight value is large.

On the other hand, STP characterizes synaptic adaptation phenomenon on a small timescale. When the upstream neuron gets overly excited, it may emit an important number of spikes per second. To avoid saturating downstream neurons, synapses can temporarily decrease the weight of the connection [72]. The post-synaptic potential is thus modulated according to the recent activity of the upstream neuron [73]. This mechanism permits homeostasis, as it either inhibits over-excited neurons, or amplifies the inputs from seldom spiking ones.

**SYNAPTIC CONNECTIONS IN ARTIFICIAL NEURAL NETWORKS** In artificial neural networks the synaptic weight  $w_{i,j}$  is taken as a scalar value, constant through time during the inference phase, and independent of the pre-synaptic potential. The pre- and post-synaptic potentials,  $x_i$ ,  $x_j$  are also scalar values that do not depend on the time. The synaptic function  $f_{i,j}(x_i, w_{i,j}, t)$  is a simple multiplication, and the resulting post-synaptic potential is expressed as:

$$x_j = x_i \times w_{i,j} \quad (II.5)$$

The synapse's weight of artificial neural networks represents only the long-term potentiation. Figure II.19 (b) illustrate the computational diagram of an ANN.

**SYNAPTIC CONNECTIONS IN SPIKING NEURAL NETWORKS** In spiking neural networks, a pre-synaptic potential takes the form of a single binary value: either it exists  $x_i(t) = 1$ , or it does not  $x_i(t) = 0$ . The information representation with binary spikes allows these systems to be faithfully implemented using analog or digital hardware, where a spike is either an actual impulsion or a packet of data encoded under the AER format. As with ANNs, the synaptic function is also a simple multiplication, and is expressed as:

$$x_j(t) = x_i(t) \times w_{i,j}(t) \quad (II.6)$$

The synaptic function in SNNs is thus similar to the one of ANNs. The difference is the time dependency of any variable of the equation. Indeed, the weight function  $w_{i,j}(t)$  can endorse complex LTP or STP behaviors, depending on the algorithm and/or the hardware deployed. For example, the Neurogrid [74] and ROLLS [75] neuromorphic accelerators implement temporally evolving synaptic weights with behaviors. On another hand, others do not, as TrueNorth [76], which and simply consider  $w_{i,j}$  as a constant value through time.

So, as for neurons, spiking synaptic models are more faithful to biological synapses than ANN ones, notably because they communicate with Dirac delta functions that represent electrical pulses. They are nonetheless abstracted at several levels to better respond to simulation or hardware implementation constraints.

## II.E.2 Neural Circuits and Neural Networks

Even though ANNs and SNNs employ different models of neurons and synapses, the organization and interconnections of neurons inside a neural network are based on the same construct. Thus, for a matter of clarity and to avoid phrasing redundancy, the term *neural network* in this section refers to both Artificial/Classical/Deep NNs and Spiking neural networks.

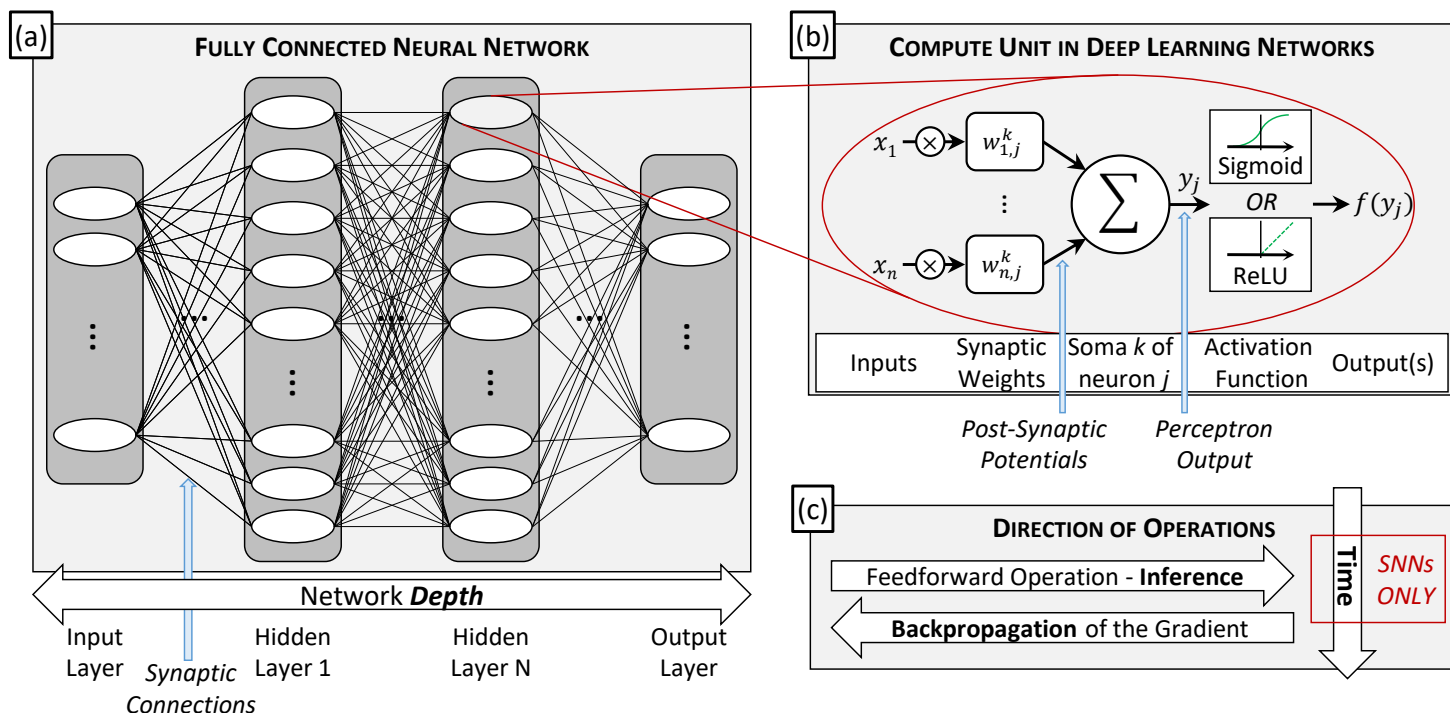
### II.E.2.a Artificial Neural Networks Presentation

#### II.E.2.a.i The dimensionalities of Neural Networks

A neural network consists in the arrangement of neurons into layers, where each layer is successively interconnected with adjacent ones [77]. A single layer of a NN is composed



of several neurons that all receive inputs from the same previous layer. Figure II.23 (a) illustrates the computational diagram of a fully connected (FC) NN. We discuss other connection patterns in section II.E.2.a.iii.



**Figure II.23:** Illustration of the computational diagram of an ANN. **(a)** Large scale structure of a neural network, organized in blocks of compute units - called *layers*. A fully connected network is represented, where each neuron of layer  $L$  receives as pre-synaptic activations the outputs of every neuron of the previous layer  $L - 1$ . **(b)** Illustration of the complete computational diagram of a classical neuron with the sigmoid or rectified linear unit (ReLU) activation function [66]. **(c)** Illustration of the dataflow directions during *inference* and *backpropagation*. Note that a spiking neural network also has a third dimension, the time. At each algorithmic time step, a full inference (depth-wise dimension) is realized.

### II.E.2.a.ii From Linear Perceptron to Nonlinear Compute Units

The linear perceptron is a simple classical neuron model, where a post-synaptic neuron output  $y_j$  is the result of the multiplication and summation of the pre-synaptic incoming activation values  $x_i$  with their corresponding synaptic weights  $w_{i,j}$ , plus an added bias term  $b_j$  [77, 78]:

$$y_j = \sum_i w_{i,j} \times x_i + b_j \quad (\text{II.7})$$

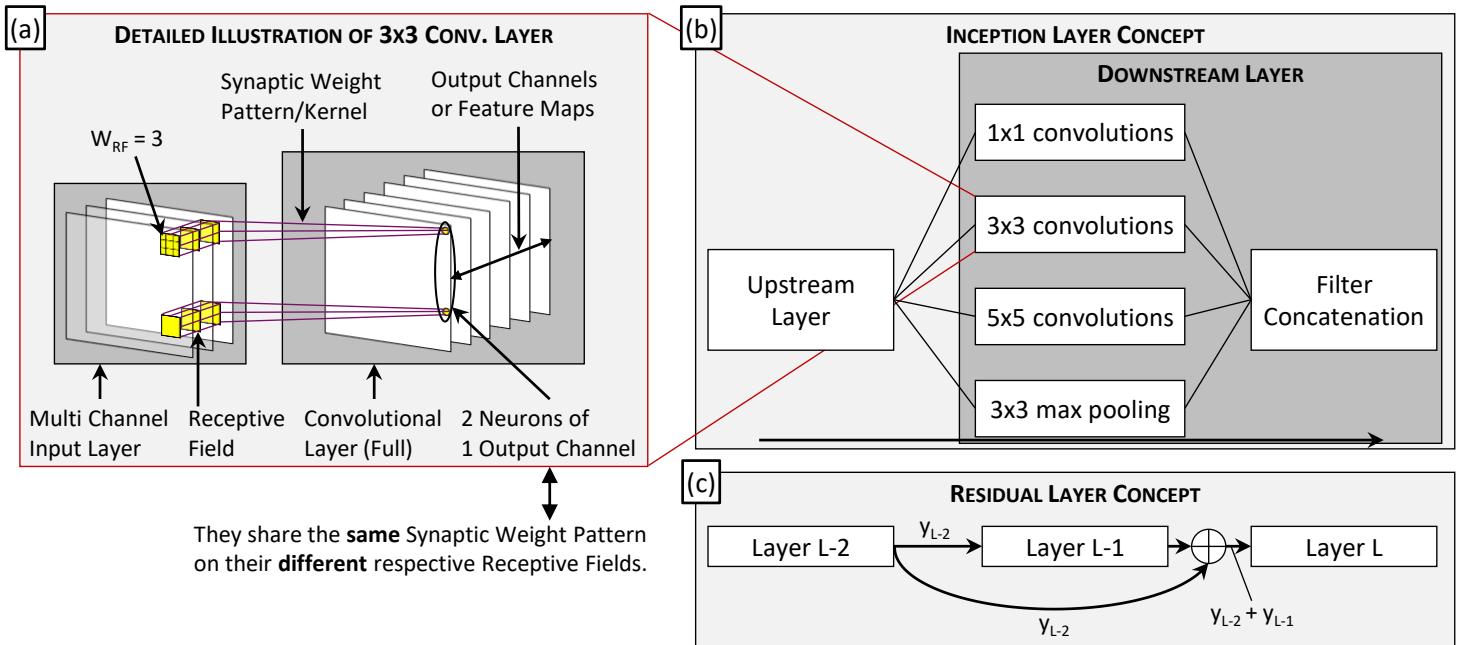
At that point, perceptrons are linear, whereas nonlinearity plays a crucial role for the universal mathematical approximation capabilities of neural networks [77–79]. Thus, current networks apply a nonlinear activation function  $f$  on  $y_j$ , as depicted Figure II.19 (b). At the end, keeping the previous notation, a pre-synaptic incoming activation value  $x^l$  of a neuron in layer  $l$  is equal to  $x^l = f(y^{l-1})$ .

There exist many different activation functions [78, 80]. The sigmoid function [81] has been broadly used for three reasons. It is nonlinear; differentiable, so backpropagation of the gradient can be operated on sigmoid based networks; and it faithfully represents the

usual relative response of biological systems to a specific stimulus [81]. Other activation functions, such as the rectified linear unit (ReLU), have been designed for reducing training and inference complexity. Note also that the weights of a network are usually referred to as its *parameters*.

### II.E.2.a.iii Varied Connection Schemes

There is an incredible and growing amount of connection schemes between neural layers employed in neural networks. The two main connection patterns are:



**Figure II.24:** Illustration of usuals neural network layer interconnections schemes. **(a)** CNN connection scheme illustration [82]. It shows the notion of kernels and output channels. **(b)** The inception layer concept [83]. **(c)** Residual layer concept [84].

- **Fully connected (FC).** As previously introduced, in a FC net, every neuron of each layer receives an input from every neuron of the previous neighboring layer. In this connection pattern, every neuron is usually evaluating a single kernel - i.e., each synaptic connection is represented by a single weight. FC layers demand many different weights as it is equal to the number of synaptic connections between layers is equal to  $N_L \times N_{L+1}$ , where  $N_i$  is the number of neurons contained in layer  $i$ .
- **Convolutional** [66, 82]. In a convolutional neural network (CNN), a neuron of layer  $L + 1$  receives in input several neurons of layer  $L$ . The "window" observed by a neuron - called the receptive field (RF) - is a square, usually of dimensions 3x3, 5x5, 7x7, 11x11 [85], as depicted Figure II.24 (a). This connection pattern is inspired by the cortical columns' organization - with pyramidal connections - of the visual cortex in mammal's brains [86]. In convolutional neural networks, each synaptic connection actually represents several weight values. The different resulting patterns are usually called "kernels" or "channels" or "feature maps". Every neuron of a layer shares the same kernel patterns, which extract specific spatial information from its input layer. This is mathematically represented as

$w_{i,j}^k = cst$  when the downstream neuron index  $j$  varies;  $k$  being the kernel index, and  $i$  being the upstream neuron index in the receptive field space. A neuron is thus defined by its connection pattern to neighboring layers, not by its weight values.

Others, like the **Inception** [83] and **Residual** [84] layers, are also broadly used. In inception networks, two layers are interconnected with several convolutional blocks in parallel, as illustrated Figure II.24 (b). It results in multi-scale CNNs that look for features/kernels of different scales at each layer, thus increasing the robustness of the topology. Residual networks introduce skip connections. These are layers taking in input the output of the adjacent previous layer onto which is added the output of a sup-previous layer, as depicted Figure II.24 (c). The interest of residual blocks is that it permits to conserve information through the depth of a neural network by avoiding the vanishing gradient phenomenon. The interested reader can learn more in various lectures or books [77, 78, 80].

### II.E.2.b Conceptualization of Neural Networks Operation

Neural networks are universal approximators [79], hence they present a huge interest for researchers and industrials. They can be conceptualized as a mathematical function  $y = f(x)$  trying to fit an ideal model function. The ideal model associates a list of inputs  $x_i$  - also called stimuli - with corresponding associated output  $y_i$  - usually called the labels or ground truth.

For example, let us consider the object recognition task as the ideal model function. Inputs are digital images, and their associated outputs are a one-dimensional vector of values whose index correspond to a specific object category - also called *class* [82, 87, 88]. These values indicate the probability that the input image corresponds to the associated indexed object category.

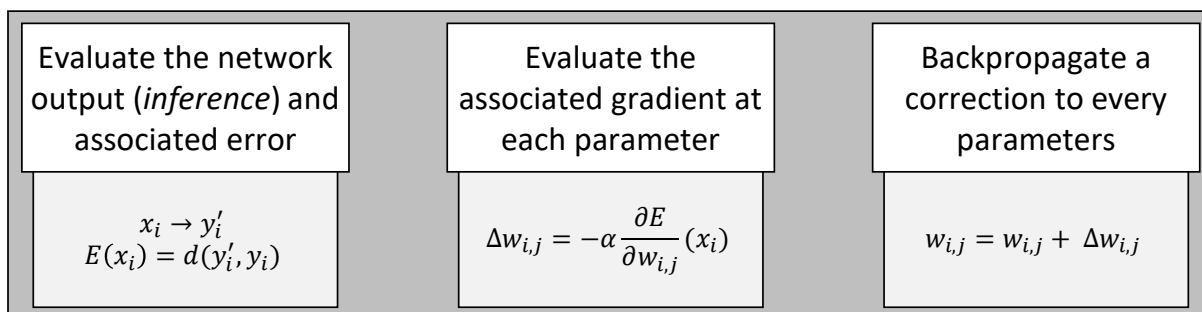
#### II.E.2.b.i Training

The function that associates every input with its corresponding label is not known a priori and can thus not be expressed formally. Hence, neural networks  $f_{net}$  must be "*trained*". Training a neural network consists in iteratively modifying its parameters - namely the synaptic weights  $w_{i,j}^l$ , neuron thresholds or biases  $b_j^l$ , etc. - so that it fits a given dataset [89]. A dataset is the list of inputs and associated outputs, which is taken as the absolute truth. It should ideally represent faithfully the model function  $y_i = f_{model}(x_i)$ .

There are many constraints and techniques for training a neural network which are thoroughly discussed in [78]. The most famous one is called backpropagation of the gradient. It consists in forcing the network to get closer to the best solution for every input-output couple  $x_i, y_i$  by *orienting* its parameters - weights and biases - so that the network delivers a minimal error  $E(x_i) = d(f_{net}x_i, f_{model}(x_i))$ . This minimum can be found by following the local gradient around each estimated point.

First, one gives an input to the network to be trained  $x_i$ . The network thus delivers a possible solution vector  $y'_i$ . The difference between the actual label  $y_i$  and the approximated solution is evaluated through an error function expressed as a distance  $d(y'_i, y_i, \cdot)$ . This error function - also called loss function or score - must be differentiable. Then, the gradient of this function is evaluated at the current local point  $\frac{\partial E}{\partial x}(x_i)$ . This gradient is then backpropagated through the entire network by evaluating the associated gradients at each compute unit's synapse and bias thanks to the chain rule. and updating these parameters accordingly. Fine tuning the backpropagation of the gradient requires some knowledge and a large amount of trial and errors run.

Note that training an SNN, which is mostly composed of nondifferentiable functions, through the network depth and time (see Figure II.23 (c)) adds some complexity. We discuss training an SNN further in Appendix E.



**Figure II.25:** Illustration of main steps of the backpropagation algorithm. (Order: from left to right.)

### II.E.2.b.ii Combining Networks for More Functionalities and Better Performances

Neural networks are now usually deployed as a combination of several smaller networks [90, 91]. Each individual small network is a complete neural network realizing a specific function  $f_k(x_k)$ . This strategy for building a neural network topology presents many advantages, such as:

- Accelerating training with transfer learning [80]. A fully trained NN is taken from open-source models, available online, and fine tuning is realized to adapt the weights of the final layers to its specific task [78].
- Accelerating the inference. The function of dedicated blocks being known, they can for example be evaluated separately in parallel on different hardware platforms.
- Guaranteeing functionalities by blocks, which is essential for industrial certifications.
- Modifying the internal behavior of some blocks to reduce the complexity of the complete operation. For example, in computer vision (CV), understanding its environment usually relies on first extracting a set of identifiable visual points [92]. Then, the identified points - usually referred to as *features* - are tracked through time to evaluate several things, e.g., the 3D position of the observer relative to these points. The task of identifying relevant points has been done with causal methods for years, but recently many actors have switched to deploying CNNs for this task instead as the first layers of CNNs can be seen

as encoders and are thus well suited for this task [91, 93]. Nevertheless, CNNs are heavy to evaluate, so being able to reduce the complexity of networks by blocks - because their action/operation is known and fixed - can drastically improve computational performance of the full chain [94].

It is thus possible to evaluate only part of a full network that has been trained on a full application. We build on this for the realization of the work presented in chapter IV.

### II.E.2.c Specificities of Spiking Neural Networks

Building SNNs follows the bottom-up strategy of associating compute units together. The differences with ANNs lie in the specific mechanisms employed by the compute units - the neuron models - as well as the method for training them.

#### II.E.2.c.i Specific Compute Mechanisms

**MEMBRANE POTENTIAL** The notion of membrane potential differs from the notion of an artificial activation in the sense that it is not output as it is by a spiking neuron. As explained in section II.E.2.a.ii, perceptron models receive as inputs the values integrated by the neurons of previous layers. Spiking neurons, however, take in input binary spikes as pre-synaptic potentials. With the integrate and fire model (see section II.E.1.b.ii), a neuron integrates post-synaptic potentials onto its membrane potential  $V_m$ . This potential is not directly transmitted, it is compared to a threshold  $V_{th}$  and if  $V_m$  is greater than  $V_{th}$  then the neuron fires a single binary spike. The IF model is broadly employed in neuromorphic systems - algorithms and hardware - as it is computationally simple. However, because it does not provide any temporal mechanism apart from time continuous integration of the inputs, the leaky integrate and fire is also a model of choice.

**LEAKAGE AND REFRACTORY PERIOD** LIF neurons present the specificity of *keeping track* of the time that passes. A stark difference between classical and spiking neural networks is that the latter naturally integrate time into their computational diagram. The usual mechanisms for computing with time are the leakage and the refractory period, which are discussed below. An illustration of these mechanisms is depicted Figure II.26 (a), and Rueckauer et al. [95] discuss the effect of both on final network accuracy.

A traditional IF neuron fires as soon as  $V_{th}$  is reached. With the refractory period mechanism instantiated, if the refractory time  $T_R$  is not overcome, i.e., the amount of time since the last output spike is smaller than  $T_R$ , the neuron cannot fire. The refractory period operates as a firing frequency limiting techniques, avoiding a neuron to fire too frequently when it is over-stimulated. It is a short-term potentiation mechanism, as was discussed for synapses in section II.E.1.b.iii.

A leaky IF neuron behaves exactly as its IF homologue, but it also instantiates the leakage mechanism. With leakage  $\lambda$ , the membrane potential decreases continuously over time. The leakage can be linearly dependent on the time and defined as  $V_m(t) = V_m(t-h)(1-\alpha h)$ .

Or it can also be exponential as  $V_m(t) = V_m(t - h)exp(-\frac{h}{\tau})$ . The parameters  $\tau$  and  $\alpha$  are usually called the leakage parameters.

In neuromorphic systems, an efficient way of applying the leakage is by removing a *leak value* after each neuron update, called leak-upon-load [96]. In this case, at time  $t$ , the membrane potential of neuron  $j$  of layer  $l$  is expressed as:

$$Vm_j^l(t) = Vm_j^l(t - h) + \sum_i w_{i,j} \times x_i^{l-1}(t) - \lambda(h) \quad (II.8)$$

or depending on the leak function expression:

$$Vm_j^l(t) = Vm_j^l(t - h)\lambda(h) + \sum_i w_{i,j} \times x_i^{l-1}(t) \quad (II.9)$$

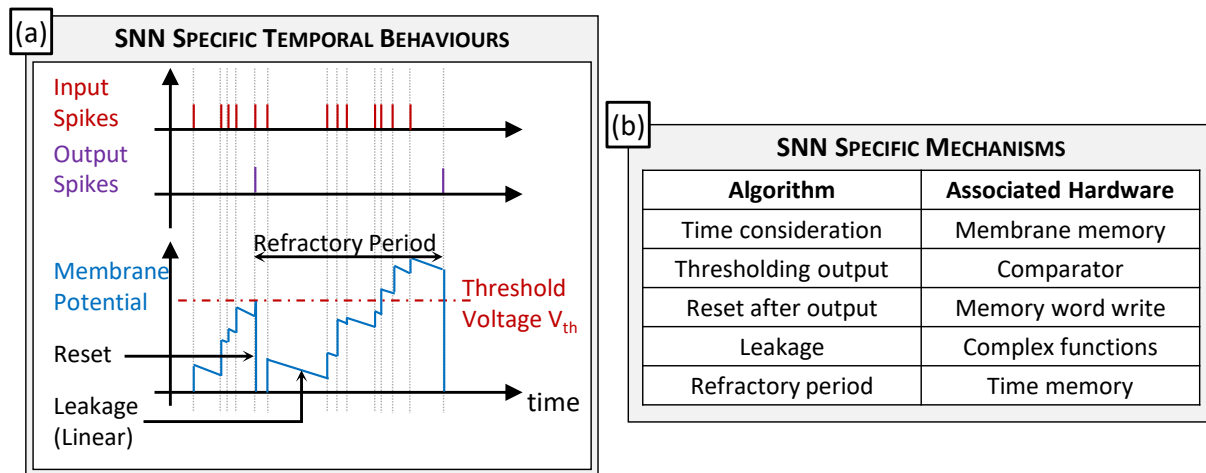
In these equations,  $\lambda$  is the leak function whose value depends on the duration between neuron updates  $h$ , and  $t$  is the algorithmic time step.  $h$  is the discrete time increment, also called algorithmic tick duration. Further details about the leakage operation and leak-upon-load can be found in chapter IV of this thesis, where the hardware design of a spiking neural network accelerator is presented in depth.

**WINNER TAKE ALL** Winner-take-all (WTA) is another brain-inspired mechanism implemented in many inhibitory SNNs [15, 97]. It consists in having the units of a single layer compete, where the one receiving the strongest input spikes first and inhibits all the others. WTA has been argued as a crucial factor in decision making [98, 99], and because functions like SoftMax - to discriminate the strongest voting neuron - are not available in SNNs [95], WTA is usually deployed as decision discrimination mechanism. Convolutional spiking neural networks (CSNN)s can also employ WTA by making all kernels of a single neuron compute between each other.

**TIME MANAGEMENT** As an SNN relies on time in its computational diagram, simulating or evaluating an SNN onto dedicated hardware requires to manage time evolution. There are two main strategies for that. One is to evaluate the state of every neuron at each algorithmic time step. This is adapted for massively parallel architectures where all neurons are implemented on a different core. Another, more suited for many hardware implementations, is to update a neuron only when it receives a spike in input.

In both cases, time is discretized. Which means that there is a quantum of time that defines the minimum time increment observed during simulation (or emulation) of the SNN topology. The duration of the time step usually varies between the nano to the millisecond depending on the algorithm or hardware platform [25]. In traditional hardware, the time does not encode itself, it must be artificially managed with ticks and synchronization throughout the whole network. This results in the fact that membrane values  $V_m$  must be stored, usually along with the last timestamps of reception  $t_{in}$  or emission  $t_{out}$  of input or output spikes respectively, between every algorithmic time step. And as will be discussed further down in section IV.B.2, memory transfer costs an incredible amount of energy and time.





**Figure II.26:** Illustration of the typical temporal evolution of the membrane potential of a leaky integrate and fire neuron. Several spikes are received asynchronously, the reset sets back the potential to zero and the refractory period forbids the neuron to spike until enough time has passed between two spikes.

## II.F Conclusion

Imaging solutions are numerous and new types of sensors are emerging quickly. Classical image sensors deliver dense and redundant frame-based data. With such imagers, simultaneously obtaining the two highly interesting properties for advanced computer vision tasks - high dynamic range (HDR) and high acquisition speed (HAS) - is relatively difficult and demands the use of advanced technologies.

The rise of 3D IC technologies permits to developed sensors with dense parallel interconnection to pixels and heterogeneous integration. These two characteristics permit to provide high quality image acquisition. But combining HDR with HAS still requires advanced data processing post-acquisition. A novel type of sensor, however, intrinsically present these two properties simultaneously, along with event-based behavior, which should guarantee non-redundant data acquisition. These are the EB sensors. They consist in bio-inspired digital pixel sensors that asynchronously realize continuous luminance differentiation through time. Recent industrial event-based sensors are moving away from the traditional bio-inspired model for more energy efficient and higher resolution solutions. However, what should be done with EB sensors is not clear yet, and with 3D IC technologies, many novelties could be brought to these devices.

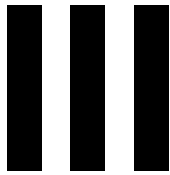
Full spiking neural networks could have been a fantastic opportunity, being bio-inspired and naturally adapted to the data type delivered by EB sensors. However, SNNs lack in precision - accuracy - with respect to their artificial counterpart because of the difficulty of to train them. Thus, even if they permit to exploit temporal mechanisms, the leakage and refractory period, which are naturally suited to extract temporal information contained in the data acquired by an event-based sensor, they are not yet ready to realize full applications. On top of that, they are complex algorithms that consume much power, and may not be adapted for implementing into an embedded event-based pipeline.

We thus ask, what should be done with EB imagers? What application best suits these sensors? What, then, is enabled by 3D IC technologies and how to exploit it?

To answer these questions, one must understand what advantages it can bring to computer vision applications that relies on event-based data. Several types of algorithm taking in input event-based data have already been developed [100]. The main theme of such algorithms are the ones aiming at obtaining the optical flow, the ego-motion of the sensor, or object detection and classification. These various algorithms employ several techniques, some bio-inspired - like Spiking Neural Networks -, some more classical, like depth extraction from stereo event-based imagers. Nevertheless, algorithms do not converge towards a standard way of operating and are usually not as efficient as classical imagers.

In the rest of our thesis, we discuss event-based computer vision before presenting an application-specific integrated circuit for it is aimed at being integrated with 3D IC technologies directly behind an EB pixel grid. Chapter III discusses other types of algorithms targeting detection and localization. Chapter IV discusses the hardware implementation of a near-sensor spiking convolutional neural networks that takes advantage of an envisaged 3D implementation for maximizing efficiency.





# Analyzing the Environment with Event-Based Sensors and Algorithms

---

## Contents

---

<b>III.A Introduction to the Event-Based Paradigm</b>	<b>49</b>
III.A.1 Reminders About the Properties of Event-Based Imagers	49
III.A.2 Limitations of Spiking Neural Networks	50
III.A.2.a Object Classification	50
III.A.2.b Other Tasks with Event-Based Data	50
III.A.3 The Search for an Application	50
III.A.3.a Bio Inspiration	50
<b>III.B Object Detection with Events</b>	<b>51</b>
III.B.1 Clustering and Tracking	51
III.B.1.a Motionless Agent Situation	51
III.B.1.b Moving Agent Situation	52
III.B.1.c Possible Solutions	53
III.B.2 General Multi View Computer Vision Principles	53
III.B.2.a Features with event-based Data	53
III.B.3 Depth Extraction with Stereo Vision	54
III.B.3.a The Geometry of a Stereo System: The Notion of Disparity	54
III.B.3.b Standard Operation Flow for Stereovision	55
III.B.3.c Resolution, Stereo System Tilt, and Disparity	55
<b>III.C Stereo Vision with an Event-Based Camera</b>	<b>55</b>
III.C.1 Spiking Neural Networks for Event-Based Depth Extraction	55
III.C.1.a Depth Extraction with a Trained Model	56
III.C.1.b Depth Extraction with a Hardwired Model	56
III.C.1.c Conclusion on Bio-Inspired Approach	57
III.C.2 Event-Based Depth Extraction with Spike-to-Spike Matching	58
III.C.2.a Algorithm Principle	58
III.C.2.b Event Matching Criteria	58
III.C.2.c The <i>Time Surface</i> - an Event Descriptor	59
III.C.2.d Event Matching Methodology	60
III.C.2.e Results Obtained on the MVSEC Dataset	60
III.C.3 Virtual Dataset Elaboration	62
III.C.3.a The Motivations	62
III.C.3.b Methodology	62
III.C.3.c Depth Extraction by Event Matching on Simulated Data: Results	63

<b>III.D Generate Frame-Based Data from Events</b> . . . . .	<b>64</b>
III.D.1 Introduction to the Principles of Visual Inertial Odometry . . . . .	65
III.D.1.a Technical Domain . . . . .	65
III.D.1.b Geometrical and Algorithmic Concepts . . . . .	65
III.D.2 Event-Based Visual Inertial Odometry . . . . .	67
III.D.2.a Working Principle . . . . .	67
III.D.2.b Ego-Motion Compensation and Frame Reconstruction . . . . .	68
III.D.3 Ego-Motion Compensated Event Frames and Usages . . . . .	73
III.D.3.a Deployment in a VIO Loop . . . . .	73
III.D.3.b Object Detection with Motion Blur . . . . .	74
<b>III.E Discussion</b> . . . . .	<b>75</b>
III.E.1 The Main Issue of Event-Based Sensors . . . . .	75
III.E.2 Our View of a Smart Event-Based Imager . . . . .	76
III.E.2.a Patent Deposited . . . . .	76
III.E.2.b Why it Works . . . . .	76
III.E.2.c Future Work . . . . .	76
<b>III.F Conclusion</b> . . . . .	<b>77</b>

---

## III.A Introduction to the Event-Based Paradigm

### III.A.1 Reminders About the Properties of Event-Based Imagers

The so-called event-based paradigm comes from the intrinsic properties of the EB image sensors. These bio-inspired sensors are naturally high dynamic range (HDR) on an inter-scene basis, meaning that in between the scenes they are able to observe and quantify an important range of illuminations. They acquire data at a high acquisition speed (HAS) thanks to their pixel-wise asynchronous readout module. It permits to detect extremely fast or small movements. It also provides an intra-scene HDR capability because the time between two "frames" - events in this case - is small, of the order of a millisecond. Finally, they provide data in an event-based (EB) manner. So, the data delivered is devoid of redundancy<sup>1</sup> and every pixel is readout independently from each other.

On the other hand, EB sensors come with important drawbacks. Namely, high resolution pixel grids would lead to gigantic amount of data generated - called the bandwidth and expressed in events per second (ev/s) - and algorithms able to efficiently exploit the new type of data are not well defined yet. On top of that, the data acquired is extremely noisy as event-based pixels quantify illumination by discrete relative logarithmic steps. They are thus extremely sensitive to quantization and thermal (dark) noises. We inform the reader that an overview of working principles of event-based image sensors are provided in section II.C of this document.

---

<sup>1</sup>The lack of redundancy is true only for non-moving sensors. On the contrary, when the sensor is moving it provides many redundant spikes.

## III.A.2 Limitations of Spiking Neural Networks

### III.A.2.a Object Classification

As discussed in chapter II, spiking neural networks are a specific type of bio-inspired machine learning algorithm that can naturally take as input event-based data. However, regarding their algorithmic performances, in terms of accuracy, they hardly reach the levels of their artificial counterparts. The interested reader can find more details about this point in section E.3 which discusses SNNs training.

### III.A.2.b Other Tasks with Event-Based Data

The lack of algorithmic accuracy is a truth about applications that employ SNNs for object classification/recognition. But what about algorithms performing other tasks? Could event-based properties - namely their inter-frame HDR, HAS, and EB acquisition capabilities - be taken advantage of for other tasks? These are the questions we try to answer in this chapter. To that aim, we tested, analyzed and re-designed event-based algorithms for clustering, depth extraction, and simultaneous localization and mapping (SLAM), to understand the advantages and drawbacks related to this type of data.

## III.A.3 The Search for an Application

As there is already a substantial number of algorithms that exploit EB data [100], we took inspiration from the brain operation to restrict the scope of our research.

### III.A.3.a Bio Inspiration

#### III.A.3.a.i *The Two-Streams Hypothesis*

Considering that an event-based sensor is close to an artificial retina - in the sense that the measurement realized, and the data transmitted are analogized from it [15] - it seems obvious to analyze the visual processing pipeline of the brain for finding a suitable event-based application. For what concerns the neural processing related to vision, there seems to be a consensus regarding the theory of the two processing pathways [101, 102]. These two processing pipelines are called the ventral and dorsal pathways. The former is usually referred to as the "what" path and the latter as the "where" path because they are responsible for the visual recognition of objects and their spatial localization, respectively. An important particularity of this hypothesis is that the two pathways seem to function at different speed, the "what" being slower than the "where" [103].

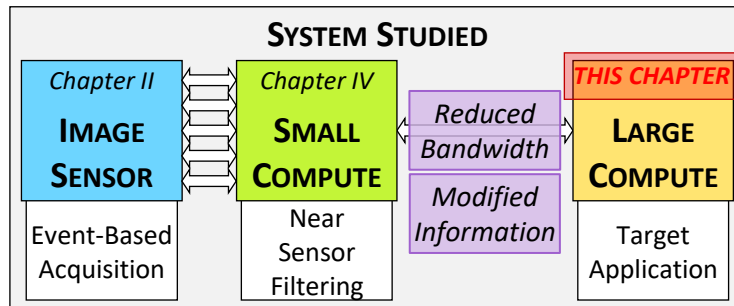
#### III.A.3.a.ii *Color and Object Recognition*

In addition, studies [104] argue that the agreement of color information in the task of recognizing objects plays a significant role in various object recognition tasks. For example, Rossion et al. have shown that adding color to pictures permits to reduce the response time of human subjects who were asked to name objects they were shown. Note that in cognitive science, the response that a human subject takes to accomplish a certain task is considered as a resilient way of measuring its cognitive ability to realize the task. The results

of Rossion et al. would thus motivate the idea that object recognition is better done with color information.

### III.A.3.a.iii Consequence: Event-Based Data for Movement Related Tasks

We make the link between the role of color in object recognition and the two-stream hypothesis. We thus orient ourselves towards considering that colorless event-based data would be of better used for tasks that are based on movement such as object detection - realized in the "where" path - especially thanks to their event-based nature.



**Figure III.1:** System studied in our Ph.D. This chapter focuses on discussing applications and algorithms that are best suited for taking advantages of event-based data properties.

In the next sections, we discuss the use of event-based cameras for applications where movement is at the center of the data, with for example detection of moving objects. We mostly studied scenarios where the observing camera is moving, and the pros and cons of event-based data in these situations. In this chapter, we first present the use of EB data for object detection in section III.B This leads to discussing depth extraction with a stereo event-based system in section III.C. Finally, section III.D presents a solution for EB systems that would be embedded onto a moving agent.

## III.B Object Detection with Events

Given that training a spiking neural network is an open problem, we look at non-machine learning based solutions.

### III.B.1 Clustering and Tracking

A naive method for detecting objects consists in considering we should be able to cluster spikes emitted by the same object into an ensemble that characterizes the object.

#### III.B.1.a Motionless Agent Situation

Litzenberger et al. [105] have shown that clustering objects and tracking the cluster can be successfully realized on event-based data. The dataset used for the demonstration was obtained with an EB camera looking at a multi-lane highway from a bridge onto which vehicles are passing vertically on the field of view, from the horizon to the bottom of the screen [106]. They demonstrate that they can reliably detect vehicles on the different lanes and track them up to the point when the vehicle gets out of field.

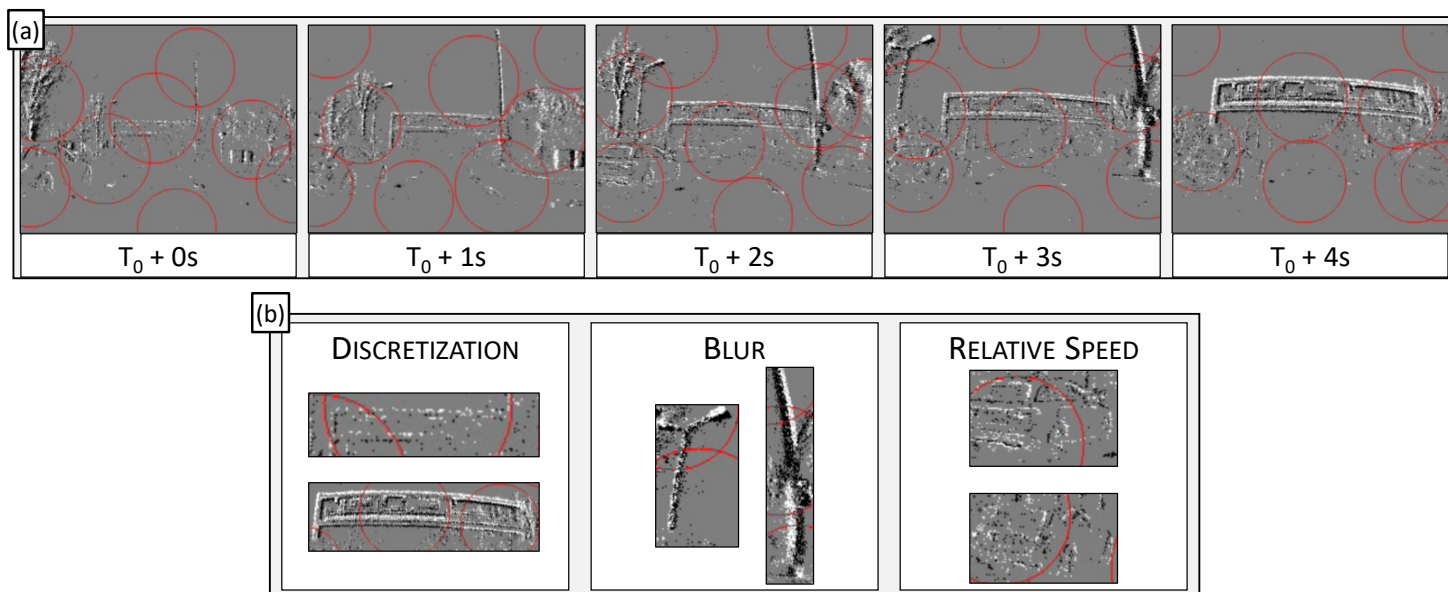
It is a nice demonstration of the capability of EB cameras, given that at the time, realizing the same operation with a classical camera was relatively hard and computationally expensive. However, the situation depicted here is ideal for the task because the camera is motionless.

### III.B.1.b Moving Agent Situation

In an embedded system, the agent - the camera - is subject to various motions. Ego-motion on an event camera comes with two problems:

1. When an event-camera is moving, objects static with respect to the world frame also emit spikes. In this case, it is thus difficult to correctly cluster data with only 2D event coordinates. So, the basic spike clustering algorithm does not work on situations where camera is moving, has can be seen on Figure III.2.
2. Objects moving in the same direction as the agent do not appear clearly in the data stream because the object speed relative to the camera is small. So, using event-based cameras without any other sensor may be dangerous for autonomous vehicle applications.

These issues come from the projection onto the image sensor. A camera can be seen as a device that projects the 3D structure of the light emitted by any object of the world onto a the 2D surface of the image sensor. This mathematical transformation can be characterized, for example, by the pinhole camera model [107].



**Figure III.2:** (a) Trials of clustering events of a camera put onto the windshield of a car on running on a highway. Each picture is one second apart in time from the precedent. This data sample depicts the malfunctioning operation of the cluster algorithm on the scene. (b) Illustration of problems of using an event-based camera for computer vision. *Discretisation* (see section II.B.3.a.ii) makes the far objects nearly invisible. The less an object moves *relative* to the camera, the less clearly it appears. Integrating several spikes from a unique object onto an event frame makes it appear *blurry* on the resulting picture. Source data from the Davis Driving Dataset 17 [108]. Cluster algorithm from [105]. Illustrated samples are from our own implementation.

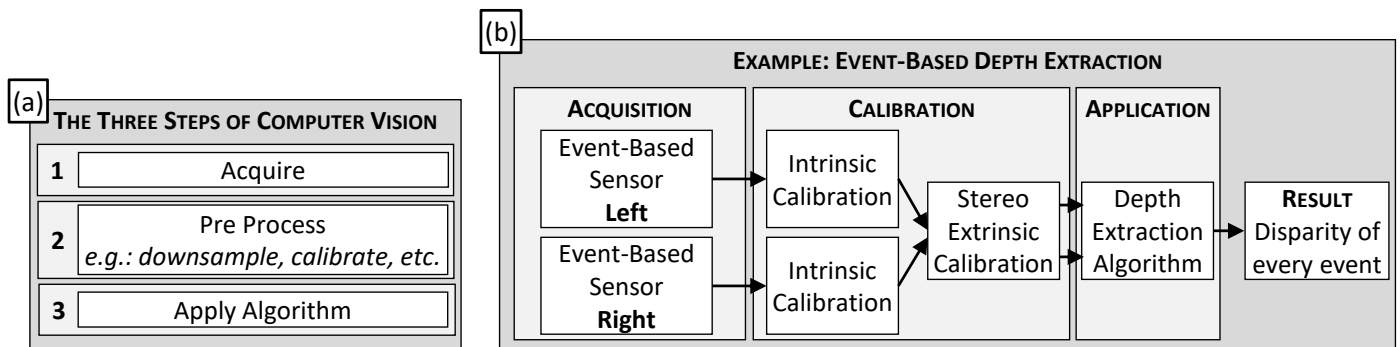
### III.B.1.c Possible Solutions

During the 3D to 2D transformation, a large amount of information is lost. Indeed, final spikes are characterized only by discretized two-dimensional pixel coordinates. No depth information - the depth being along the normal axis (axis Z) relative to the image sensor - is conserved. On top of that, motion blur is also a problem, as depicted Figure III.2 (b).

To solve these difficulties, several solutions exist. The former, lack of depth information, can be solved with stereo vision, which is discussed section III.B.2. The latter with ego-motion compensation (EMC), which is discussed section III.D.

### III.B.2 General Multi View Computer Vision Principles

To obtain a third coordinate corresponding to the depth while using only cameras, one possibility is to use two sensors and exploit stereo vision. As with many multi view computer



**Figure III.3:** (a) Diagram illustrating the 3 main steps of any computer vision pipeline. (b) Associated example the computational flow of any event-based stereo vision algorithm for depth extraction applications using only event-based cameras.

vision algorithms, realizing depth extraction with stereo vision systems is based on identifying specific points in both left and right pictures, and being able to match these points together with as few false positives as possible [109].

To identify features that do not depend on the acquisition system, it is required to calibrate the system before any operation. Any camera acquires data with errors that accumulate all along the acquisition pipeline (optics, light sensing, discretisation of data and space, etc.) which results in visible defects like distortion, chromatic aberration, and others. Correcting these defects is called intrinsic calibration [28, 110].

The feature extraction usually relies on standardized detectors and descriptors such as the Harris corner detector [111], Shi-Tomasi features, Speeded-Up Robust Features (SURF) [112], and many others. But these descriptors work on frame-based data and have not been conceived and calibrated for event-based data.

#### III.B.2.a Features with event-based Data

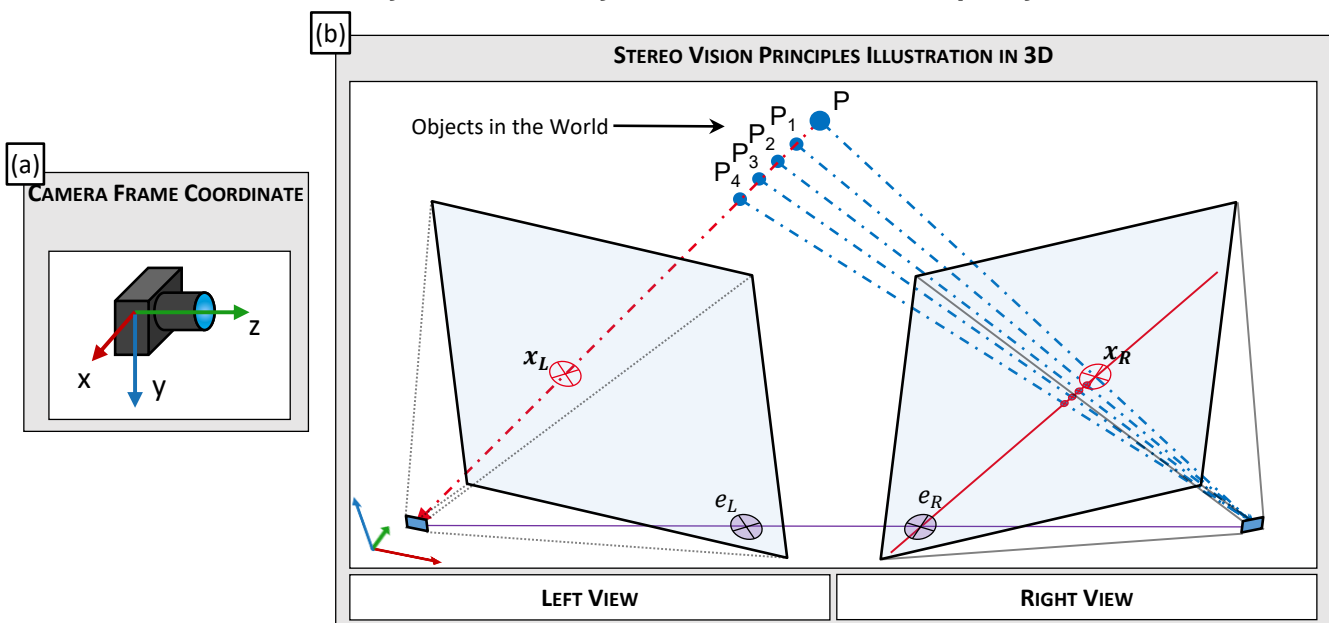
A question thus lies with event-based data, namely, how to detect and describe features from a stream of spikes? There exist several strategies for doing so, that are illustrated and discussed in the following sections:

- Use spiking neurons as detectors or descriptors (section III.C.1).
- Characterize each spike individually with a set of descriptors (section III.C.2).
- Reconstruct frame-based data from a set of events and work on these frames as in classical computer vision (section III.D).

The interested reader can find many more examples and details about event-based computer vision in the survey of Gallego et al. [100].

### III.B.3 Depth Extraction with Stereo Vision

#### III.B.3.a The Geometry of a Stereo System: The Notion of Disparity



**Figure III.4:** (a) Illustration of the conventional camera three-dimensional frame coordinate system  $x, y, z$ . The  $X$  and  $Y$  axis are the standard two-dimensional coordinates for representing a picture on the image sensor focal plane. The  $Z$  axis is normal to the sensor plane and corresponds to the apparent depth of object on the picture. It actually corresponds to the optical axis in classical optics. (b) Illustration of the disparity principle. The object in the real world is depicted by the point  $P$ . If one keeps the projection of  $P$  onto the left image sensor constant (i.e.,  $x_L$  kept unique), there is still one degree of freedom on the right sensor corresponding to the object  $P$ . The line of possibilities of projections on the sensor at the right is called the epipolar line. The difference of  $x$  coordinates on this line is directly dependent on the depth of  $P$  relative to the stereo system. Note that as it is, the line is not aligned with the  $x$  axis of the image right image sensor. Thus, extrinsic calibration must be realized to align this line with the  $x$  axes of both sensors. Note also that an ideal configuration to ease this calibration process is to have both sensor planes perfectly parallel with the origin points confounded. More information about stereo vision can be found in [109].

Depth extraction with stereo vision consists in having a set of two cameras aligned, i.e., with the  $x/y$  frame coordinate parallel, next two each other a few centimeters apart. Under this condition, they will share a field of view relatively identical, with the only difference that the  $x$  coordinates of acquired data will be different for any objects observed.

After distortion compensation, and calibration of intrinsic and extrinsic camera parameters, the difference in  $x$  coordinate between the left and right frames is called the disparity  $d$  and is inversely proportional to the actual depth of the point of the object observed [109]. The

proportionality coefficient  $G_z$  depends on the camera sensor and optics parameters (focal length, pixel size, etc.).

$$x_L - x_R = \text{disparity} = G_z * \frac{1}{\text{depth}} \quad (\text{III.1})$$

The depth being the distance along the Z axis of the camera frame, as depicted Figure III.4 (a). So, matching points in space and computing the difference in observed  $x$  coordinates of the left and right points (the disparity  $d$ ) permits to obtain the depth of the object.

### III.B.3.b Standard Operation Flow for Stereovision

As with any computer vision algorithms, stereo vision requires to apply intrinsic calibration first. On top of that, because a stereo vision system integrates two different cameras, calibration of extrinsic parameters is also required. Extrinsic calibration consists in realigning the images of the two sensors as if they would be perfectly parallel. An illustration of this step is depicted Figure III.3 (a). Note that the only necessary corrections anterior to the stereo vision pipeline are the intrinsic and extrinsic parameters calibration - for minimizing errors related to the geometry of the scene -, other calibrations may be done later.

### III.B.3.c Resolution, Stereo System Tilt, and Disparity

The geometrical parameters of a stereo system that have an important impact on the depth resolution of the system are the following:

- The baseline.
- The angle between the camera system.
- The size of the pixels.

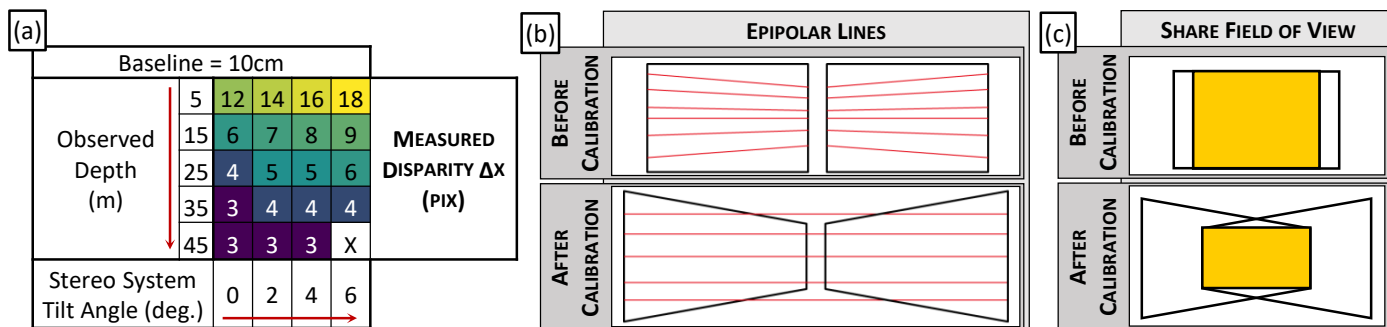
Following the realization of a virtual dataset (discussed section III.C.3), we were able to evaluate the impacts of the tilt angle of the stereo system and the depth of observed objects on the measured disparity. A sample of these measurements for a system of two VGA (640x480) cameras at a baseline of 10cm is given in Figure III.5 (a). The variation of the observed disparity with the tilt angle of the stereo system can be explained by the stereo rectification step, which has a drastic effect on the field of view shared by both cameras, as illustrated Figure III.5 (b). On top of that, the more the cameras are tilted - as when a human is squinting - the more the epipolar lines are inclined. Tilting cameras has the advantage of increasing the depth resolution of a system, however, it makes the computation requirements more important, as the maximum disparity  $d_{max}$  observable increases and it reduces the field of view.

## III.C Stereo Vision with an Event-Based Camera

### III.C.1 Spiking Neural Networks for Event-Based Depth Extraction

Networks of spiking neurons can be deployed for being used as depth estimators. Two methods can be used. The standard machine learning one, that is training a spiking neural network onto a dataset for depth estimation. And another one, that is considering spiking





**Figure III.5:** (a) Measured values of disparity for several object depths and stereo system tilt angles. Measurements realized on a virtual dataset. Details about the realization of this dataset are available section III.C.3. (b) Impacts of stereo rectification on epipolar lines (goal of stereo rectification). (c) Impact of stereo rectification on shared field of view. Close objects may require an important maximum observable disparity  $d_{max}$  and far some objects may disappear from field of view.

neurons as spatiotemporal filters and deploying them following a man-made interconnection pattern.

### III.C.1.a Depth Extraction with a Trained Model

Training an SNN for estimating depth and other elements, such as ego-motion, has been successfully demonstrated by Zhu et al. [91]. They trained a specific model of convolutional spiking neural network with unsupervised learning. However, the results obtained are far from the state-of-the-art with classical cameras. That mostly comes from the fact that training a SNN is still an open question, as discussed in Appendix E.

### III.C.1.b Depth Extraction with a Hardwired Model

#### III.C.1.b.i Spiking Neurons as spatiotemporal Filters

Using neural networks for event-based computer vision applications can also be done by viewing spiking neurons as spatiotemporal filters. Indeed, using a time related mechanism such as leakage - see section II.E.2.c.i for more details - in a spiking neuron permits to restrain the time window of observation of the data of this specific neuron. Then, deploying such neurons with known synaptic pattern, for example a convolutional connection, permits to restrain the field of view of each neuron and thus render them as local spatial filters.

#### III.C.1.b.ii Connections for Depth Extraction

To evaluate depth from a hardwired model, Osswald et al. [113] designed a network of spiking neurons consisting in only two layers. The first layer serves as a coincidence detector between each pixel pair of the left and right picture. Each neuron of this layer spikes if two pixels, one from the left and the other from the right camera, spike in a short amount of time one from the other.

The neurons of the second layer serve as disparity filters. Only the possibly observed disparities are allowed to spike when coincidence detectors spikes. The possibility of a disparity observed depends on the pixel size (discretisation of space), inclination of the cameras and depth of object.

The neuron deployed are modeled as exponential leaky integrate and fire neurons. The time constants of the exponential decrease are not given in the original paper [113]. After many trials, we set them up at 1ms and 500 $\mu$ s for the coincidence and disparity selector neurons, respectively.

### III.C.1.b.iii Limitations of the Method

Using such an algorithm is limiting in two ways. First, the algorithm designer sets the maximum observable disparities and the duration of the time windows of the neurons. However, these parameters directly limit the possible detectable depth (disparities allowed) and object speeds (duration of time window). So, the range of observable object is directly limited by these choices.

Secondly, the computational complexity is large. The number of calculations required for the evaluation of the two layers is proportional to  $n^4$ , where  $n$  is the number of pixels of an image sensor. Indeed, the first layer contains a number of neurons equal to  $N_{L_1} = res_x * res_y * \Delta y * d_{max}$ , where  $\Delta y$  is the tolerated error along the y dimension - height of the image - after extrinsic calibration of the stereo camera system, and  $d_{max}$  is the maximum disparity allowed. These parameters are arbitrarily set by the algorithm designer. The second layer has  $N_{L_2} = res_x * res_y * res_z / stride^3$  neurons, where  $res_z$  is the number of different depths observable, and  $stride$  is the step (in number of pixels) between the centers of the receptive fields of adjacent neurons.

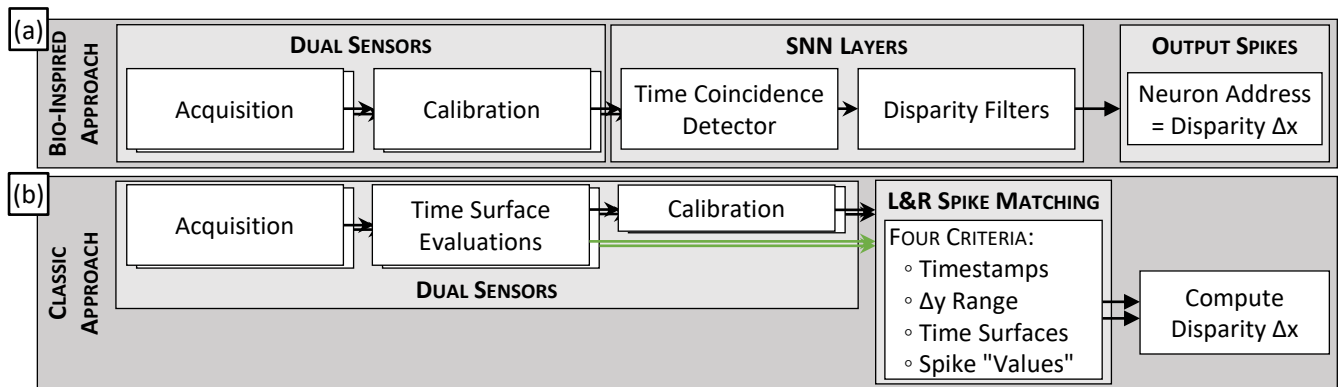
Furthermore, the problem with event-based data is that the data is continuously evolving, so the more the time required to acquire the full data of a scene, the more computations are required. So, the number of computations realized is directly proportional to the square of the bandwidth of a single imager, expressed in event per sec (ev/s). As discussed section II.B.3.a, the bandwidth is directly proportional to the resolution of an image sensor. Therefore, the computational complexity of this algorithm is  $O(n^{24}) = O(n^8)$  through time.

### III.C.1.c Conclusion on Bio-Inspired Approach

Using neurons as hardwired spatiotemporal filters for obtaining the depth of each event is not as efficient as it seems. First, the computational complexity of the algorithm makes it hardly scalable to high-definition event-based image sensors. However, having a larger resolution permits to have a larger field of view. And combining large resolution with small pixel sizes permits to have an improved depth resolution. The depth resolution can be improved by tilting cameras of the stereo system, but it reduces the field of view.

In addition, the algorithm resolution and precision rely on a set of parameters, namely the neuron sensibility (associated with the threshold voltage) and the neuron time characteristic, associated with the leakage time. The choice of parameters will impact spatial and temporal resolution of the application, which may limit the range of depth and object speed observable. An illustration of the limit of this algorithm can be observed Figure III.8 (a). This is a sample

of obtained results where the spikes characterizing the disparity of an object is depicted in the color range varying between the blue (far) and green (close). It clearly depicts the lack of specificity of results delivered by the algorithm, as a single object (in this case the car moving from left to right) is evaluated at a disparity between -6 (close) and -1 (far) pixel at the same time. So, for such fine-grained task, deploying a non-reconfigurable and hardwired model of spiking neural network is poorly efficient.



**Figure III.6:** (a) Computational diagram of the depth extraction algorithm by Osswald et al. [113]. This algorithm uses two layers of spiking neurons to obtain the depth of objects. Illustration of results obtained by deploying this algorithm can be found Figure III.8 (a). (b) Computational diagram of the depth extraction algorithm by [114].

## III.C.2 Event-Based Depth Extraction with Spike-to-Spike Matching

### III.C.2.a Algorithm Principle

Another method following a more classical computer vision oriented strategy has been proposed by [114]. It is based on matching features, where each spike is considered as a detection event. Every spike must thus be to be described by descriptor. In classical computer vision, descriptors are mathematical tools that permit to characterize the geometry of a picture around a detected feature, by considering that a digital image is a geometrical surface. So, for example, local gradients can be computed, and as well as other descriptors.

### III.C.2.b Event Matching Criteria

There exist few descriptors that can be used for matching two stream of events spike by spike. According to leng et al. [114], these criteria amount to the number of four.

- Their **geometrical position**, i.e., the x and y pixel coordinates after stereo calibration - as for any depth extraction algorithm. This criterion permits to restrain the search window spatially to a few hundreds of pixels only.
- Their **timestamps**, the two events that one tries to match should not be emitted too far apart in time from one another. This criterion permits to restrain in time the search window to a few milliseconds only.
- The **light intensity** at the time of spike, if available, otherwise the **polarity** of the spike. This criterion permits to reduce the set of searched events inside the search window to events that are more potentially to be alike.

- Their **time surface**. This criterion is considered by many as a relatively strong event descriptor.

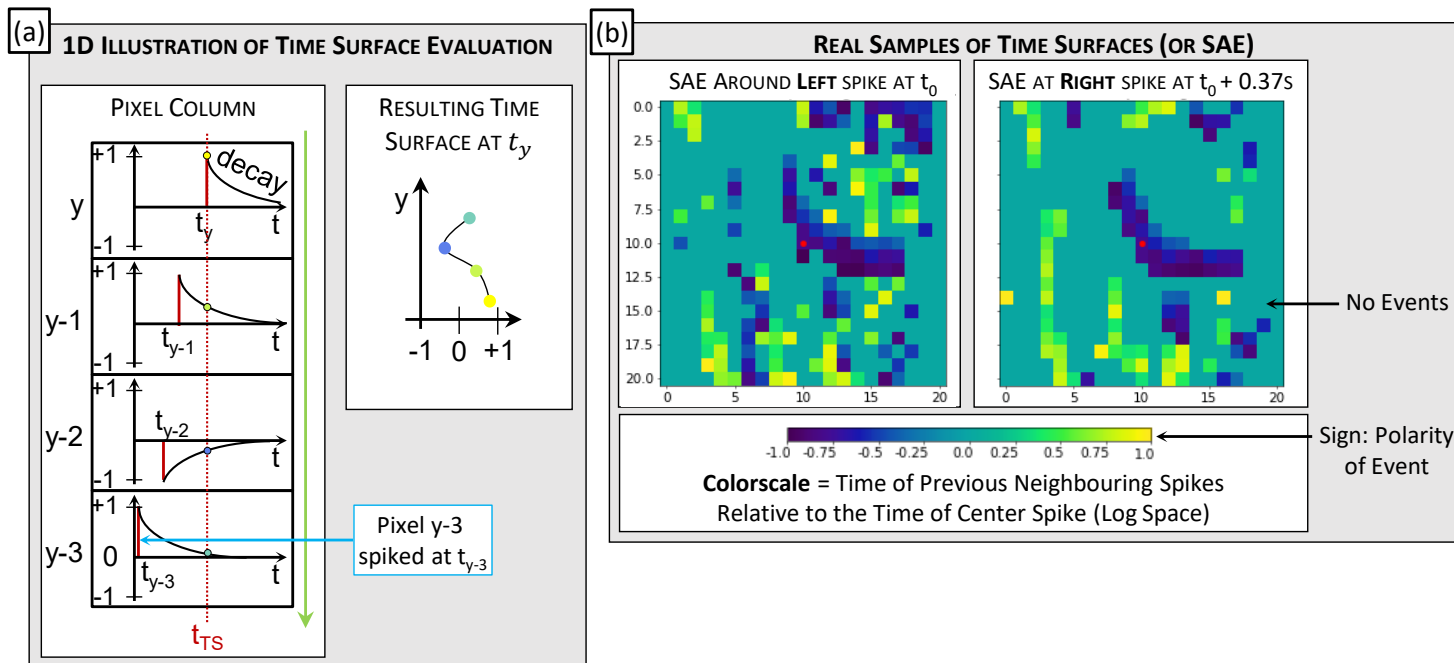
### III.C.2.c The Time Surface - an Event Descriptor

The time surface - also called surface of active events (SAE) - has been developed especially for the task of matching events together for tracking or depth extraction purposes [114, 115]. The idea behind the time surface is that behind each meaningful event lies an object - excepted if the event is noise-generated. This object is likely to have been responsible for the emission of other events in close proximity to the considered event. So, by keeping track of the time of each last event for each pixel, one should be able to represent and see objects on the screen.

The idea is thus to maintain a grid of values, where each cell of this grid represents the time of last event emitted by the associated camera pixel. Each pixel value corresponds to:

$$pol(t_i) \times \exp\left(\frac{t_{TS} - t_i}{\tau_{decay}}\right) \quad (III.2)$$

Where  $t_{TS}$  is the time at which is computed the time surface, which corresponds to the time of the last spike received.  $t_i$  is the time of the pixel  $i$ .  $pol(t_i)$  is the polarity of the event received at the pixel  $i$  at the time  $t_i$ . And  $\tau_{decay}$  is a time factor characterizing the exponential decay of each pixel value of the time surface. Examples of time surfaces are depicted Figure III.7 (b). As the reader can see, one can appreciate the similarity between the two distinct time surfaces at sight. With this descriptor at hand, it is thus possible to try and match together two different streams of events.



**Figure III.7:** Time surface illustration. **(a)** Illustration of the computational flow undergone at each pixel for evaluating the time surface. Only four pixels are depicted. **(b)** Sample of time surfaces obtained when we re-implemented the algorithm proposed in Reference [114]. The data onto which we applied the algorithm is taken from the MVSEC dataset [22].

### III.C.2.d Event Matching Methodology

With the objective of minimizing the computational requirement of the event stream matching algorithm, we re-implemented, in Python, the algorithm proposed by Leng et al. [114] following the method described here. For each event of the right camera stream - called master -, we first filter out any event from the left camera stream that does not have the same polarity as the master. This is a single bit comparison and is thus minimal. Note that if the polarity is not available, but the intensity is (i.e., the camera used is not based on an ATIS[17] pixel), one can filter out events based on their relative intensity instead. After that, any event that does not meet the time and geometrical criteria - i.e., every event that is not inside the tolerated spatiotemporal window around the master event - is also filtered out. At that point, for each event of the master stream, one has a reduced set of potentially matching event from the left stream. For each of these left events, the time surface is then evaluated. Following the different event selection steps in this order avoids evaluating too many time surfaces and thus drastically reduces the number of computations required.

Now, the four criteria discussed above are used to give a matching score to every remaining left event. This score is the sum of the distance - or error - of each criterion between the left and right event. For example, if the timestamp of the master event is 13.7ms, a left event having a timestamp of 13.9ms will have a timestamp distance/error greater than an event with a timestamp of 13.75ms. So, the lower the matching score, the higher the probability the two events are matched. At that point, every left event is assigned a score. Finally, the last operation simply consists in considering that the left event that matches the master event is the one with the smallest score.

### III.C.2.e Results Obtained on the MVSEC Dataset

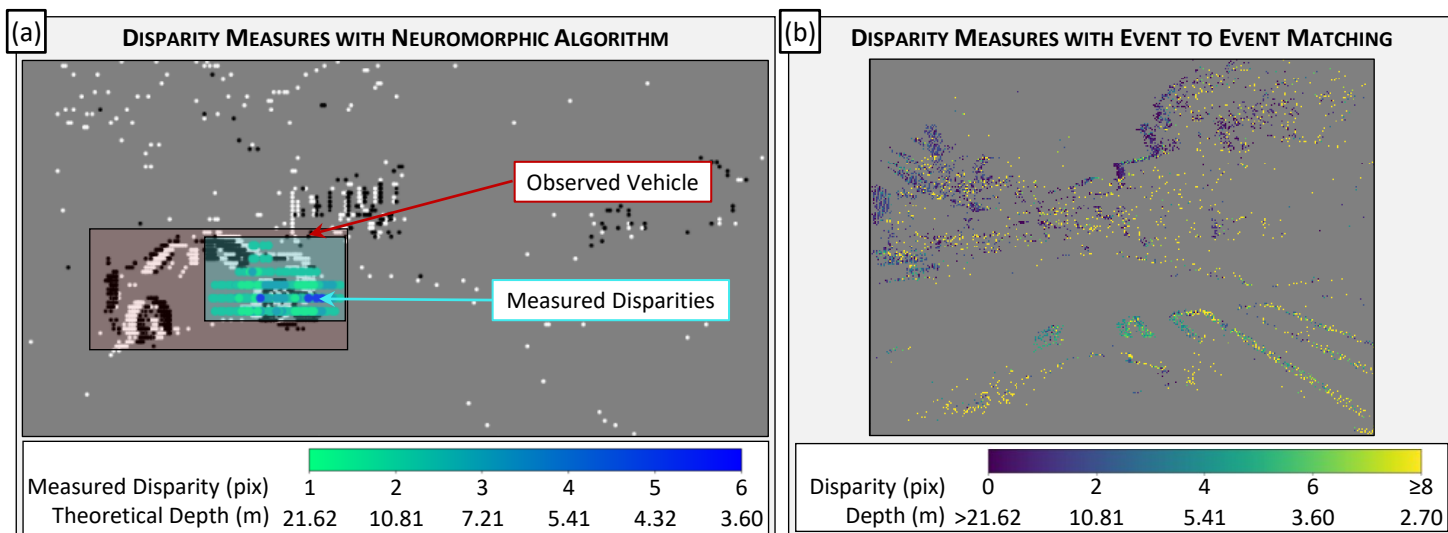
We tested this algorithm on several samples of scenes extracted from the multi vehicle stereo event camera (MVSEC) dataset [22]. After many trials, it clearly appeared that the algorithm is not as simple to apply as it seems. Both the algorithm and the dataset are involved in the lack of accuracy obtained. We ended up categorizing these limitations in the following way.

- In a stereo system, the proportionality coefficient between the disparity (in pixel) and the depth (in meters) of the objects observed can be expressed as  $G_z = \frac{f \times b}{p}$  [116]. So, the dataset sorely lacks in precision due to the configuration of the stereo system. The EB camera used are the DAVIS346 that contains only 346x240 pixels with pitches of  $p=18.5\mu\text{m}$  [24]. The focal length and the baseline are set at  $f=4\text{mm}$  and  $b=10\text{cm}$  [22]. In this configuration objects further than 11 meters appear with a disparity value of 1. And objects further than 21 meters with a disparity of 0. So, is not suited for trying to detect objects that far from the camera system.
- Moreover, a small acquisition imprecision can result in a large error through the algorithm. We list here the sources of acquisition error that impact the considered algorithm.

- The synchronization material between the two cameras of the stereo system into timestamp imprecision.
- Errors generated during the calibration procedure lead to important geometrical errors.
- Mistakes in pixel light intensity evaluations due to mismatch between the photodiodes of the image sensor lead to intensity discretisation error. Note that spike intensity measurements are not available with DAVIS346 sensors, as they are DVS [16] pixels.
- Time surfaces are extremely sensitive to noisy spikes. So, the sensor noise induces mistakes during spike-to-spike matching.

Once combined, all these fault factors result in large algorithmic errors, which adds up to the stereo system’s poor precision discussed above.

The numerous tests we carried out led us to two conclusions. First, we remind the reader that  $G_z$  characterizes the smoothness of the space discretisation along the z dimension (see equation III.1). It directly depends on the pixel pitch  $p$  and the baseline  $b$  of the stereo system [116]. Thus, depth estimation is better done with high resolution sensors and large stereo system baselines. The latter is easier to achieve with large vehicles, but more difficult with portable devices such as mixed reality helmets and mobile phones. Secondly, noisy data generated by naturally EB sensors lead to an increase in computational requirements and algorithmic errors. So, to test more precisely the algorithm, we decided to reduce the uncertainties at the maximum and elaborated a virtual dataset onto which we could have more control.



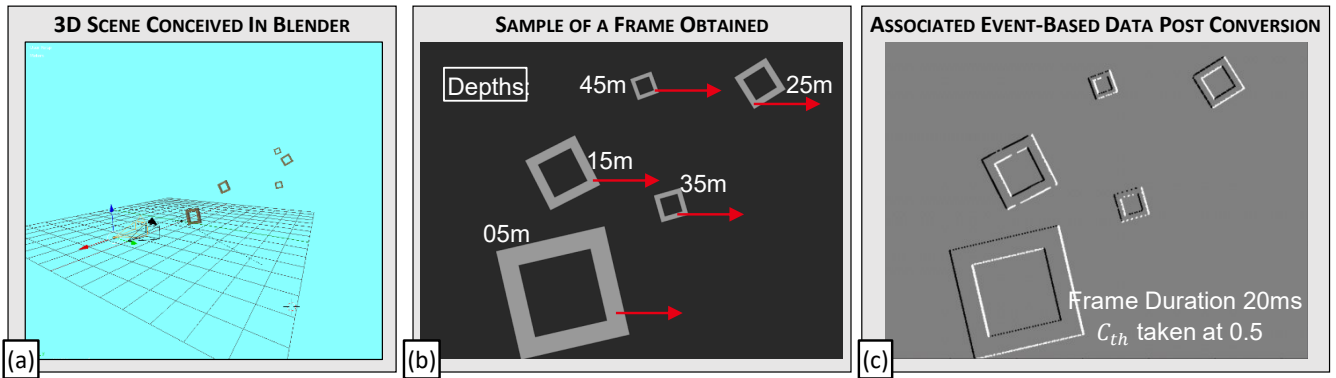
**Figure III.8:** Samples of depth extraction results obtained on the MVSEC dataset [22]. **(a)** By deploying the bio-inspired algorithm for depth extraction by Osswald et al. [113]. **(a)** By deploying the event-to-event matching algorithm for depth extraction by leng et al. [114].



### III.C.3 Virtual Dataset Elaboration

#### III.C.3.a The Motivations

To compare the accuracy of the method presented above with respect to a standard block matching algorithm deployed on classic frame-based data, with as few biases as possible, we generated our own synthetic dataset. There are many advantages to using a virtual dataset. First, the amount of control over the observation parameters. One can very easily change the baseline and tilt of the stereo system, as well as the distance between the objects and the sensors. In addition, the data obtained is noise-free, which permits to characterize only the algorithm and not accumulate errors that arises from measurements artifacts. Also, it becomes easy to compare a frame-based method with an event-based method as the fields of view and calibration techniques are perfectly identical for both type of data.



**Figure III.9:** (a) Virtual three-dimensional scene realized under Blender™. (b) Grey-level frame acquired by the left camera of the scene. The red arrows depict the direction of the movement of the objects. Note that every square has the same size but appears smaller when they are far from the sensors. (c) Associated event-based data obtained after conversion of the frame-based data.

#### III.C.3.b Methodology

##### III.C.3.b.i Simulating Event-Based Data

We detail here the method we conceived for realizing a virtual EB dataset. For obtaining event-based data from a three-dimensional virtual scene, the first step is to realize scene a rendering software. We used Blender™, and the scene realized is visible Figure III.9 (a). The second step is to render the scene as grey level pictures at high frame rate. Then, the third and ultimate step is the conversion from pictures to events. It is realized by emulating the temporal behavior of an event-driven contrast temporal sensor. This is realized sequentially on every temporally adjacent image. For every pixel, the operation realized by pixel  $i$  has been taken exactly equal to:

$$out(i, t) = \begin{cases} 1 & \text{if } \frac{I(i, t) - I(i, t_{last})}{I(i, t_{last})} \geq C_{th} \\ -1 & \text{if } \frac{I(i, t) - I(i, t_{last})}{I(i, t_{last})} \leq -C_{th} \\ 0 & \text{otherwise.} \end{cases} \quad (III.3)$$

We thus obtain the polarity of the spike of any pixel that verifies the spiking condition - based on a DVS behavior [16] - at time  $t$ .

III.C.3.b.ii Details About our Choices

**REGARDING THE RENDERING TOOL** We decided to use Blender™ to realize the dataset for several reasons. It is a freely available open-source software that is every bit as good as other charge for tools. It can easily be scripted under the python language, which permits to automatize generation of datasets with various parameters sets (baseline and tilting). It is relatively easy to handle, and for specific point the online community is huge and available for help. So, as beginners in the field, it was the perfect tool.

**REGARDING THE SCENE** We realized the simplest scene to avoid any artifact generation during the frame to event conversion. It consists in perfectly flat square frames of the same size, all parallel together, positioned at different depths from the image sensors. They slide from the left to the right of the field of view of two cameras placed at a specific baseline and tilt angle. The resolution of the virtual cameras was set to 640x480, and several measurements have been realized to characterize the performances of the algorithm proposed by leng et al. [114]. This scene permits to reduce all possible artifacts that may arise from illumination variation as the squares are parallel and monochromatic. Also, we discovered that is it essential to render the images with Gaussian anti-aliasing [117]. Indeed, as it smooths the borders of the objects rendered, it avoids generating noisy events during the conversion.

DEPTH EXTRACTION ERROR PERCENTAGE FOR SEVERAL STEREO RIG CONFIGURATIONS AND OBJECT DEPTHS																									
		Baseline=10cm				Baseline=25cm				Baseline=50cm				Baseline=75cm				Baseline=100cm							
Object Depth (m)	5	1.0	0.4	3.1	1.4	5	3.0	0.6	0.1	0.2	5	0.2	0.7	0.1	3.3	5	0.7	0.2	0.3	10	5	0.6	0.5	3.6	7.6
	15	1.0	0.4	3.1	1.4	15	0.4	0.6	0.1	0.2	15	0.2	0.7	0.1	1.2	15	0.7	0.8	0.3	1.7	15	0.6	0.3	2.3	0.5
	25	1.0	7.0	3.4	1.4	25	4.2	2.5	2.6	X	25	1.6	2.3	2.5	X	25	0.7	2.2	0.3	X	25	0.6	0.5	1.0	X
	35	1.0	13	3.1	1.4	35	0.4	0.6	5.1	0.2	35	0.2	0.7	0.1	1.2	35	0.7	0.8	2.0	13	35	1.2	1.2	0.3	19
	45	19	7.0	3.4	X	45	6.3	7.3	5.1	X	45	2	0.8	4.9	X	45	0.4	3.2	1.2	X	45	0.3	0.3	2.4	X
Stereo System Tilt Angle		0°	2°	4°	6°	0°	2°	4°	6°	0°	2°	4°	6°	0°	2°	4°	6°	0°	2°	4°	6°	0°	2°	4°	6°
Color Scale:		0	1	2	3	4	5	6	7	8	9	≥10													

The displayed error is evaluated as  $E_{\%} = \frac{n_F}{n_T + n_F} \times 100$  with  $\begin{cases} n_T = \text{number of correctly matched spikes} \\ n_F = \text{number of not correctly matched spikes} \end{cases}$  for each object.

**Figure III.10:** Relative depth estimation error (%) as a function of title angle (deg.), stereo system baseline (cm) and object depth (m). Cells are left blank (X) when no disparity point could be measured (object out of frame). The inlet gives the equation for computing the percentage, as proposed in the original paper [114].

III.C.3.c Depth Extraction by Event Matching on Simulated Data: Results

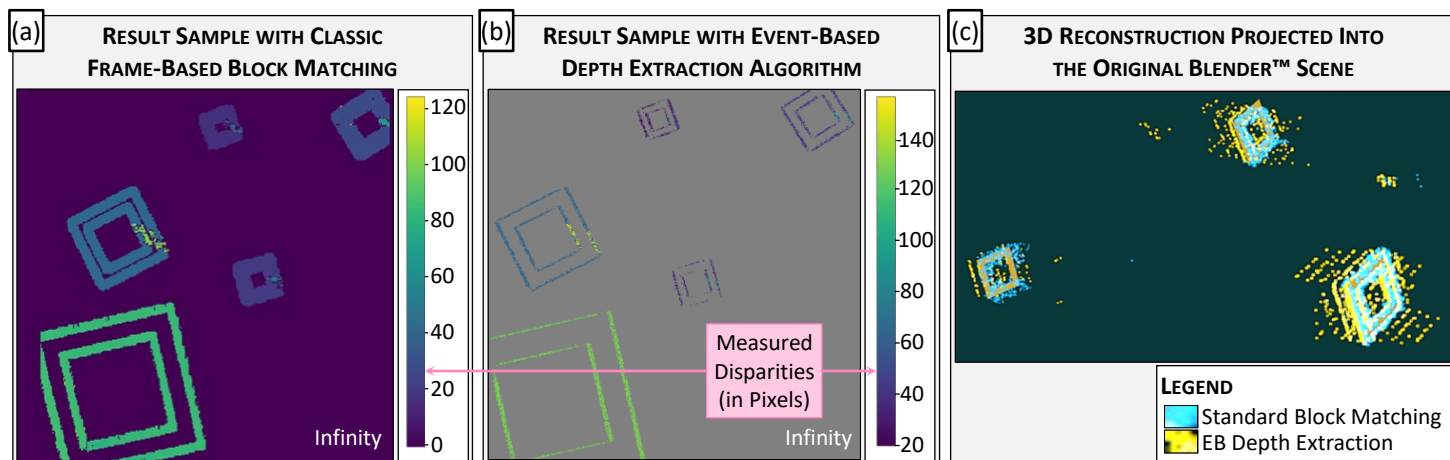
We applied our re-implementation of the algorithm proposed by leng et al. [114] on a large set of baseline and tilt configurations. An exhaustive summary of the evaluation error for different object depth and the various virtual stereo rig configurations tested is given Figure III.10. In the worst case, 19.2% of the left and right spikes are wrongly matched. Note that the error percentage is statistically influenced by the number of matched spikes, and far



objects sometimes generate very few events when the tilt angle is important. So, in this case, the errors are thus not incredibly significant, but for an actual application deployment it represents an important and dangerous noise. An illustration of the best results we were able to obtain is depicted Figure III.11 (c).

To get a better understanding of the meaning of these data, we reconstructed the depth extracted points into the original Blender™, illustrated Figure III.11 (c). When comparing the results obtained with the EB algorithm to measurements realized with a standard block matching algorithm [118] - Figure III.11 (b) and (a) respectively -, it clearly appears that the performances of the event-based algorithm are far behind the ones of the block matching. This is due to many factors, and notably the fact that sub-pixel precision disparity estimation can be realized with block matching and cannot with the event-based pipeline.

Moreover, as for the 2-layer SNN depth extractor (section III.C.1), the number of computations and memory required is relatively important and are proportional to the bandwidth of the event-based camera. And, as shown previously, high resolution sensors are necessary for a large depth resolution capability of a stereo system, because the depth is inversely proportional to the disparity. So, depth resolution decreases with object distance to the sensors and this is the major nemesis of computer vision: resolution and precision.



**Figure III.11:** (a) Sample of the disparity map obtained on the frame-based data with the standard block matching algorithm available in the OpenCV library [119] which was proposed by K. Konolige [118]. (b) Sample of the best case of disparity map obtained on the home-made virtual dataset with the event-based algorithm. The points are projected back onto the left camera frame. (c) The inset shows the reconstruction of the measured data in the original 3D blender scene. Yellow dots are reconstructed from on the disparity measurements resulting from the event-based algorithm. The blue dots are reconstructed from the disparity measured by a standard block matching algorithm on the associated frame-based data.

### III.D Generate Frame-Based Data from Events

**FUSING INFORMATION RELATIVE TO MOVEMENT INTO AN EB DATA STREAM** Another way of dealing with event-based data is to recompose classical greyscale frames from it. The trouble when recomposing event frame is plural: how long should one integrate event-based

data onto per frame, how to recompose greyscale data from discretized polar data, how to remove motion blur from the data, etc. All these questions have been answered in a relatively large scope by H. Rebecq et al. [120]. They first remove motion blur by motion compensating the event-based data using motion information obtained with visual inertial odometry (VIO). Then, they integrate the spikes onto the frame by using a bi-dimensional gaussian voting system to associate a value to each pixel neighbors of every motion compensated event.

**OUR CONTRIBUTIONS** In this section, we present and discuss their method and our own implementation of their algorithm under the robot operating system (ROS) [121] with the open-source version of VINS-Mono [122] as the backend VIO pipeline. The work presented here led to the deposition of three patents in 2020 and 2021.

### **III.D.1 Introduction to the Principles of Visual Inertial Odometry**

#### **III.D.1.a Technical Domain**

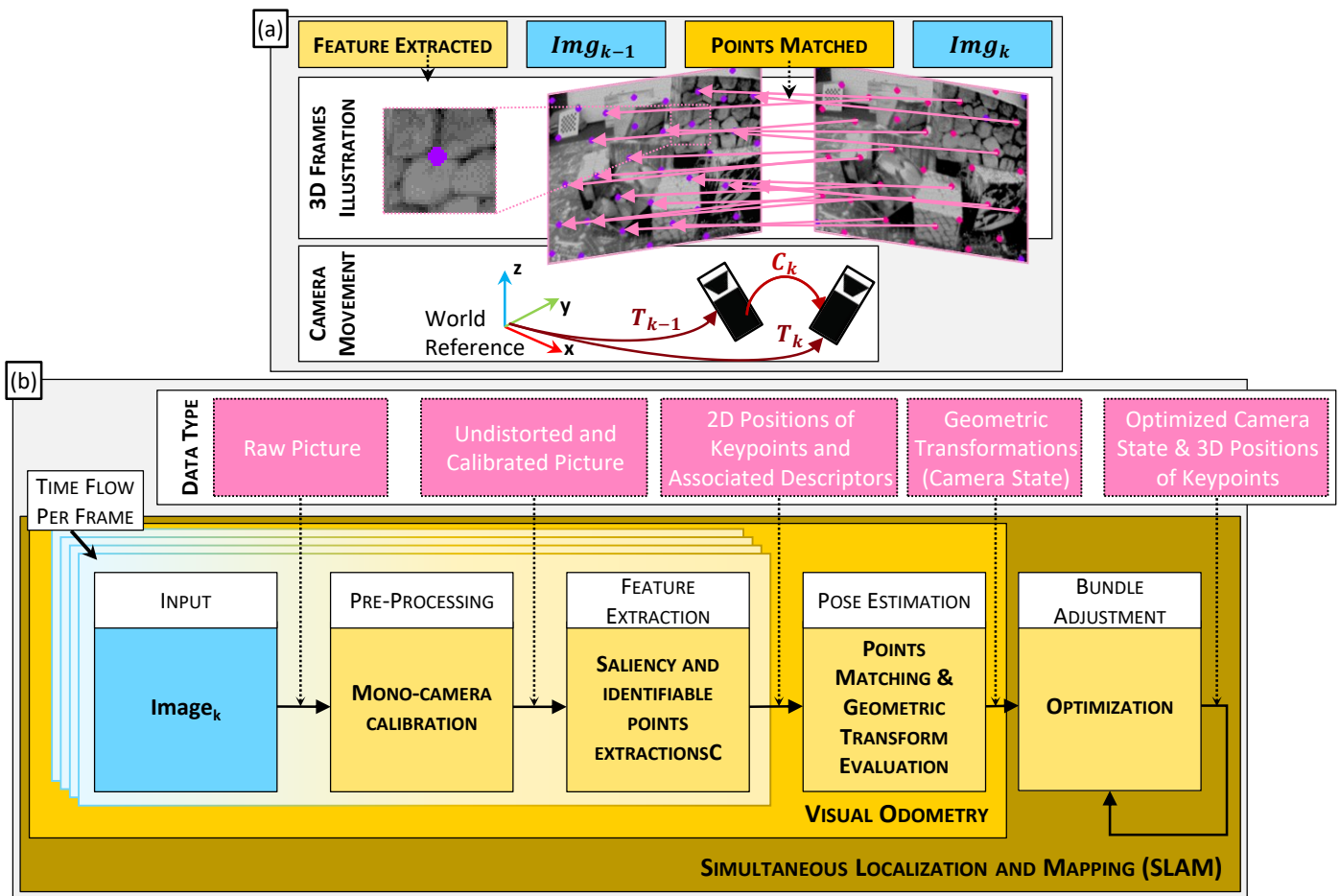
Several current and future applications require solutions for precisely evaluating the position of an agent in space, for example autonomous machines (drones, robots, and vehicles), mixed realities (AR/VR), outdoor motion capture, etc. In these applications, several characteristics of the positioning system are required:

- Centimeter or even millimeter precision.
- Independence on illumination conditions - at night, during the day, directly under the sun, etc.
- Fast position evaluation.
- Fast moving object and potentially dangerous situation detection.

To have a system - sensors and software - that meets these criteria, the best solution for now is to use sensor fusion. It consists in using several sensors, acquiring different type of information, and combining all their data together to fully describe the position of the agent that embeds all these sensors. The spatial localization/position of an agent can be fully described with respect to an origin point and axes (called coordinate system) with only 6 coordinates: 3 of translation ( $x, y, z$ ) and 3 of rotation ( $q_0, q_1, q_2$ ) - called quaternion. As a group, these six values are called the 6 degrees of freedom (6DoF), or camera state.

#### **III.D.1.b Geometrical and Algorithmic Concepts**

To obtain these values, several methods exist. The most famous consists in integrating the data obtained from an inertial measurement unit (IMU) - the combination of an accelerometer, a gyroscope, and, sometimes, a magnetometer - and/or to realize visual odometry (VO). Combining/fusing both solutions in the same algorithm is the basis principle of visual inertial odometry. On top of VIO, adding mathematical optimization upon all the positions evaluated through time is called simultaneous localization and mapping (SLAM). SLAM gives the opportunity to maximize the precision of a VIO algorithm, while characterizing the environment (establishing a "map").



**Figure III.12:** (a) Geometrical representation of the visual odometry principle. A single camera acquires several pictures on a scene with shared field of view between the frames. Features are extracted onto these frames and compared between the two frames for matching them. Once the features are matched, thanks to their pixel coordinates, the mathematical transformation between the frames can be computed. Knowing the geometrical transformation between the frames permits to find back the three-dimensional displacement followed by the image sensor during between the times of acquisition of the frames. (b) Illustration of the visual odometry algorithm pipeline.

Concisely, VO algorithms are based on the idea that a picture obtained by a camera "exists" in the world three-dimensional space and can be localized inside of it. The movement between two pictures can thus be expressed mathematically with a geometric transformation in a real three-dimensional Euclidean space. Technically, the two pictures must have some overlapping in their field of view, allowing to match points of interests that permit to evaluate the geometric transformation between them, as depicted in Figure III.12. These algorithms are usually deployed on group of sensors larger than the simple {camera; IMU} pair. It can include cameras, IMUs, radars (such as LiDAR or laser sensors) [123], and others.

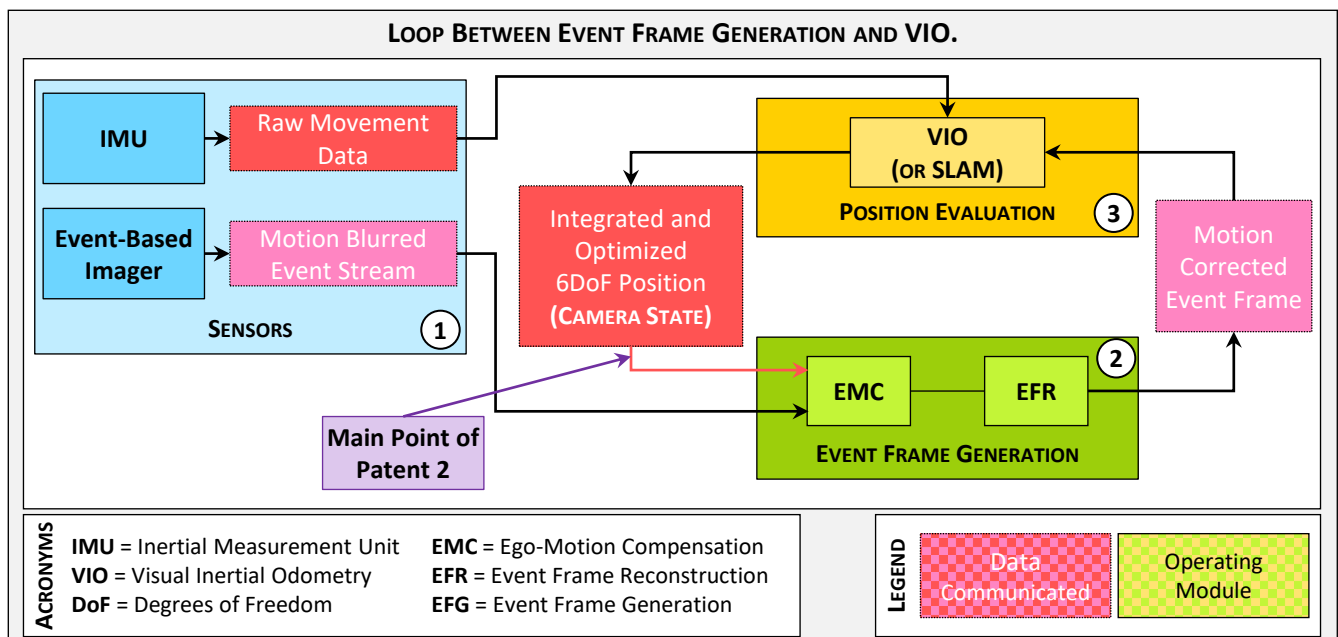
However, none of these sensors simultaneously meet the 3 properties described above. Standard cameras enable centimeter precisions but cannot be both independent to light conditions and acquire data quickly. IMUs do not enable centimeter precision, at least not for more than a few minutes since the accumulated acquisition error - called the *drift* - has a

cubic relationship to the position [124]. Nor do radars. That is why we focus on event-based (EB) image sensors that enable measurements up to the micrometer precision [125].

### III.D.2 Event-Based Visual Inertial Odometry

#### III.D.2.a Working Principle

As for event-based depth extraction, fully event-driven oriented pipelines for self-localization exist but are generally less efficient than standard ones [126–129]. A solution that aroused our interest was thus to adapt the event-based data so that it can fit in a standard state-of-the-art frame-based pipeline. That because frame-based VIO and SLAM algorithms are known to be very efficient, and as discussed previously, event-based data present many advantages for these tasks. Reconstructing frames from a stream of spike is a problem studied by many actors [130–132], and we focus on the solution proposed by H. Rebecq et al. [120]. The main idea is to generate classical frames from event data acquired by an event-based camera after having first compensated the motion of the camera. The motion compensation is realized based on IMU measurements, and the frames are then used by a VIO algorithm to evaluate more precisely the position of the camera. Introducing motion-compensation



**Figure III.13:** Illustration of the basic EFG-VIO loop allowing the generation of ego-motion compensated event frames. This system has been developed by [120]. The EFG-VIO loop takes in input EB data and IMU. From that the EFG module can generate an EB frame that is given to the VIO module for position evaluation. The camera state extracted is then given to the EFG module to generate the next frame.

in the event frame reconstruction (EFR) pipeline permits to reduce the global error of the VIO algorithm and opens up possibilities for further downstream processing. The working principle of the system is depicted Figure III.13.

### III.D.2.b Ego-Motion Compensation and Frame Reconstruction

#### III.D.2.b.i Summary of the Full Process

To compensate the motion included in the event data due to the movement of the camera - the *ego*-motion -, events are first grouped into sets. The duration of a set is of the order of tens of milliseconds. The motion of the camera during that time is estimated from IMU measurements, which are integrated into poses. These poses are then used to compensate the position of each spikes included in the set, as if they would have been emitted at the time zero of the event set (corresponding to the time of the first event of the set). The set is then integrated onto a frame with bilinear spatial interpolation of the data.

#### III.D.2.b.ii From Event-Based Data to Event Sets

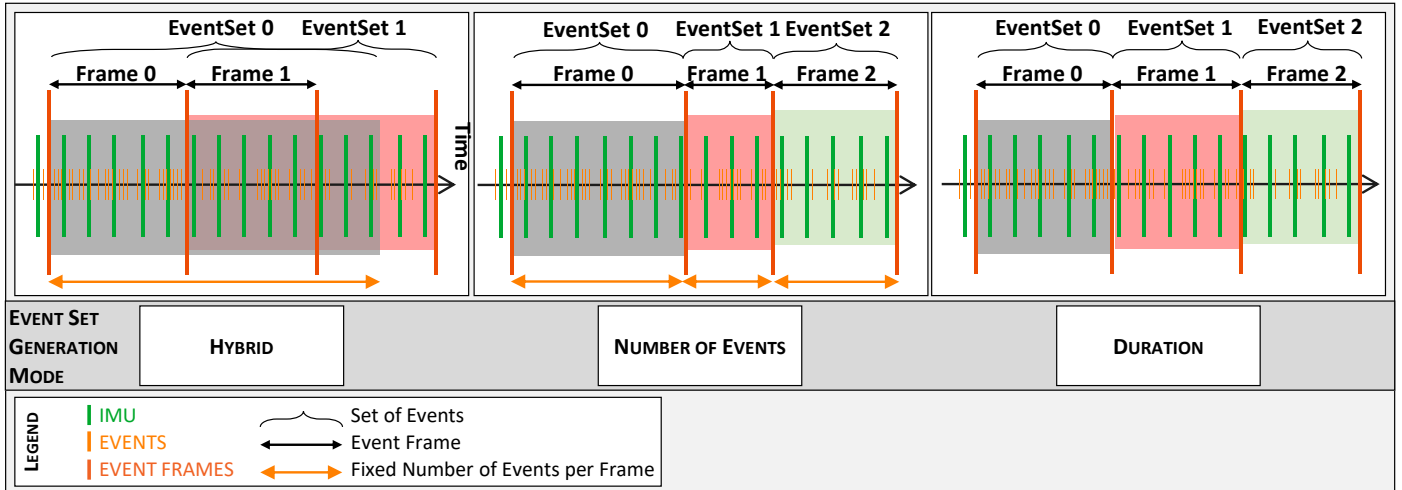
Several methods can be deployed for grouping events into selected sets of events, called event set generation modes.

1. **By number of events.** Starting from the event  $k$  at  $t_k$ , one takes the next  $N_e$  events. The following event set then starts from the  $k + N_e$  event and also takes the next  $N_e$  events. And so on and so forth.
2. **By duration.** Starting from the event  $k$  at  $t_k$ , one takes all events that arise between  $t_k$  and  $t_k + T_F$ , where  $T_F$  is the duration of an event set. The next set starts with the event that comes right after  $t_k + T_F$ . The problem of this solution is when nearly nothing occurs (the camera stays still and the scene does not change) as no events are generated, the sets are nearly empty of events.
3. Finally a **hybrid** solution consists in grouping events by number of events at an imposed rate. The idea is to select the start time of each set independently from the events contained in the sets and force each set to regroup a fixed number of events. This leads to event sets whose duration vary with the camera movement speed. It permits to avoid generating too many event sets when the camera moves quickly, and to avoid having event sets too far apart in time as may arise in the number of event set generation mode.

These different strategies are better illustrated Figure III.14. While it appears as a simple formality, the mode for generating the event sets is actually particularly important regarding the results obtained by a VIO algorithm on frames generated using different modes. Indeed, as depicted Figure III.17 (a), the root-mean-square error (RMSE) - calculated as proposed in [133] - obtained drastically depends on the event set generation mode and parameters. It is also a method for trading off accuracy with computational complexity, as the number of frames generated directly depends on this parameter.

#### III.D.2.b.iii Ego-Motion Compensation of Events

The event sets, along with the associated IMU measurements, and the last known camera pose obtained from VIO evaluation at the time  $t_{vio}$ , are then passed to the module that manages the ego-motion compensation operation. Through ego-motion compensation, every event coming from an identical non-moving object in the 3D world should ideally



**Figure III.14:** Schematic representation of the contents and durations of event sets for the three different generation modes considered. The *frame i* corresponds to the *event set i*.

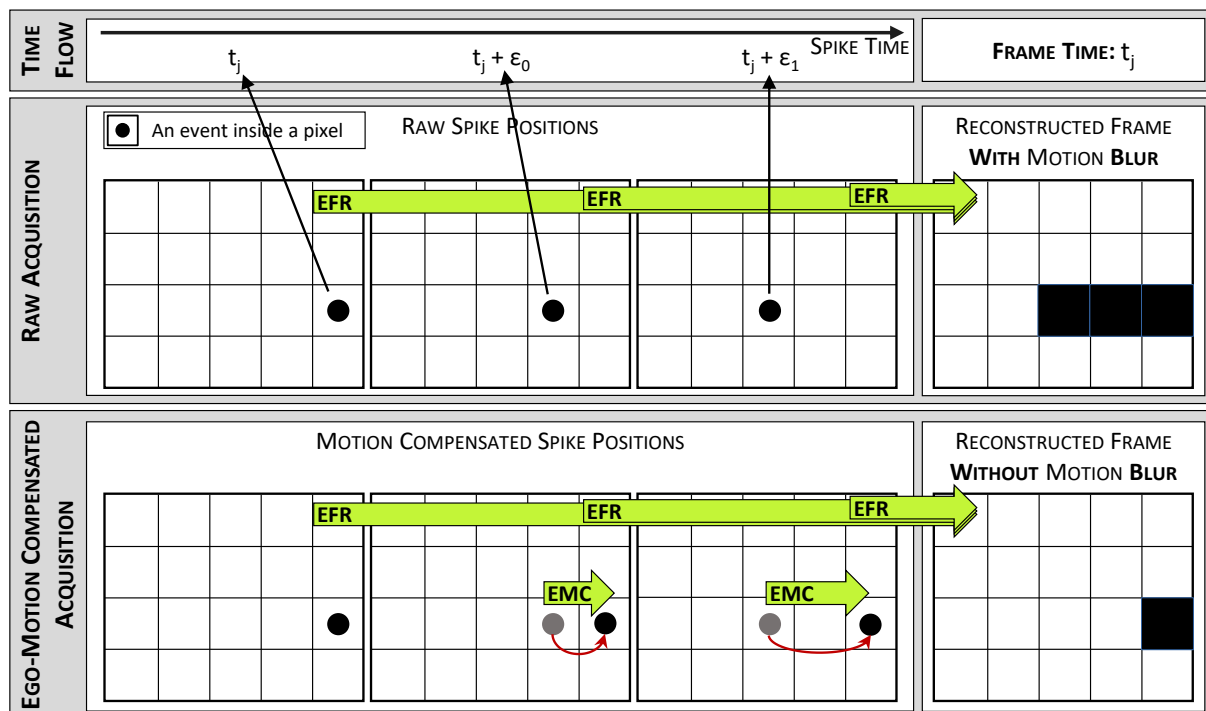
by projected back at the same identical 2D position on the frame after correction. This is illustrated Figure III.15, which schematically depicts the operation of the EMC module on three distinct spikes emitted by three distinct pixels that have "seen" the same spike source. To obtain the required geometrical transformation - the correction - the protocol described next is followed.

Basically, the IMU measurements are integrated following an IMU model and Euler integration, so that the fine-grained poses  $p(t_{imu})$  and orientations  $R(t_{imu})$  of the camera can be obtained at each IMU samples times  $t_{imu}$ . Then, for each event  $k$ , the associated camera state at  $t_k$  is interpolated from the camera states (IMU integrated samples) adjacent in time. Note that the initial camera state also contains several other information such as approximated depths  $Z(x, y, t_{vio})$  for several  $\{x, y\}$  pixel coordinates, estimated camera velocity vector  $v(t_{vio})$ , etc., all obtained from the VIO algorithm, which permits to compensate the position of each event more accurately.

Then, the EMC process simply consists in computing a corrected position for each event by applying the following equation (III.4). What it does is, in the right order:

1. Project the 2D pixel position  $x_k$  into the homogeneous space using intrinsic projection matrix. The homogeneous space is a mathematical 3D Euclidean space where points coordinates are expressed independently of the camera parameters (pixel dimensions, optics characteristics, etc.), permitting to treat data coming from any sensor uniformly. Each frame taken by a camera "exists" and can be represented by a plane in this space.
2. Project the normalized coordinates into the 3D camera reference frame using the depth  $Z_k$  of the event estimated with the last known depth at this pixel position obtained with the last VIO evaluation at  $t_{vio}$ .
3. Interpolate the geometrical transformation  $T_{t_{set_0}, t_k}$  between the timestamp of the event  $t_k$ , and the first timestamp of the event set  $t_{set_0}$ , relative to the movement of the camera. The reference transformations used for interpolation are obtained with the integrated IMU

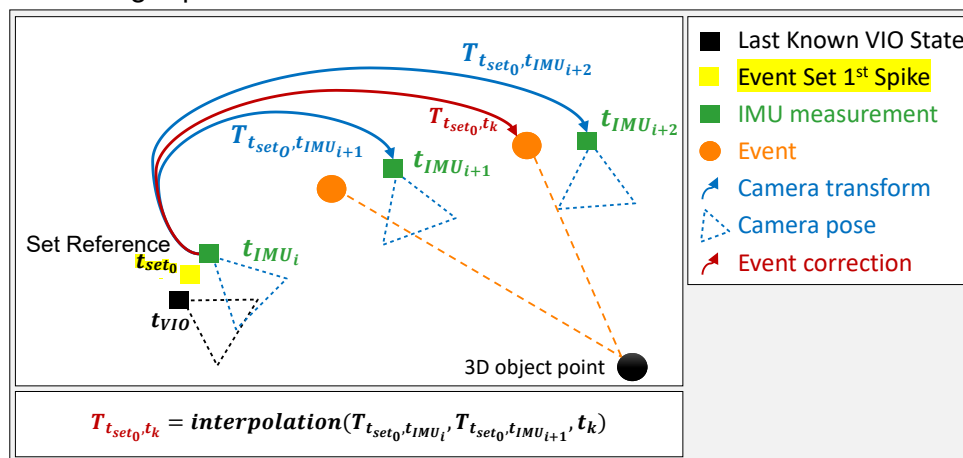




**Figure III.15:** Schematic representation of event-based motion blur and ego-motion compensation.

samples in between two event sets. With an IMU sampling at around 1kHz, and event sets grouped with tens of milliseconds duration, there are several IMU measurements available for each event sets, as illustrated Figure III.16.

4. Correct the 3D event position with the operation  $\mathbf{X}'_k = T_{t_{set0}, t_k} \mathbf{X}_k$ , where  $\mathbf{X}_k$  is the homogenous coordinates vector of the event at  $t_k$ , and  $\mathbf{X}'_k$  the ego-motion compensated one in the homogeneous space.
5. Project back the event into pixel coordinates with  $\mathbf{x}'_k = \pi(\mathbf{X}'_k)$ , where  $\pi$  is the intrinsic camera calibration matrix. At that time, the final ego-motion compensated pixel coordinates  $\mathbf{x}'_j$  are non-integer (represented with floating point format) - i.e., they are not aligned on integer pixel coordinates.



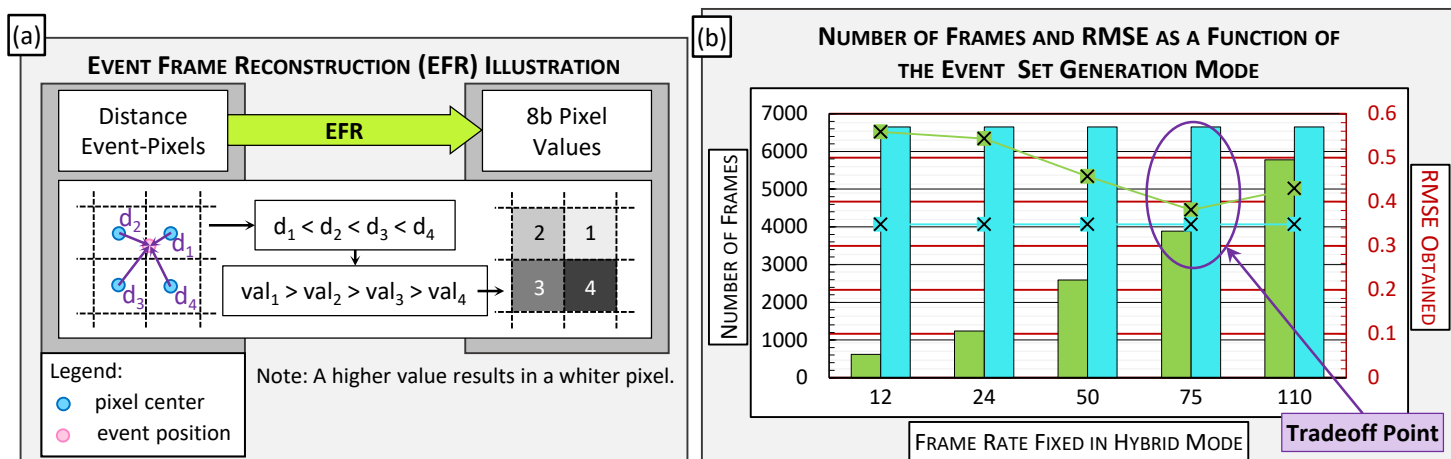
**Figure III.16:** Illustration of the mathematical transformations of the EMC. It shows the transformation interpolation (red) in-between two camera transforms (blue) obtained for inertial measurement samples (green) integration for EMC of two distinct events (orange).

To sum up, the full operation for compensating the motion - EMC - of each event of an event stream finally relies on a matrix multiply operation between the camera state at the time of spike and its pixel coordinates [120]:

$$\mathbf{x}'_k = \pi \left( T_{t_{set_0}, t_k} \left( Z_k \right) \pi^{-1} \left( \mathbf{x}_k \right) \right) \quad (\text{III.4})$$

### III.D.2.b.iv Event Frame Reconstruction from Motion Compensated Events

Then, each event polarity and their new coordinates  $\mathbf{x}'_k$  is used to reconstruct the event-based frame  $F_{set}$  using a Gaussian distribution, based on the work of [134]. The method to synthesize an image from a list of motion compensated events with floating points coordinates (i.e., not aligned with the pixel grid integer coordinates) is the following. For each motion compensated event, its polarity is used to update the accumulated polarities of the four nearest pixel locations around the event. Four weights characterizing the proximity between the event and its neighboring pixels are computed using the distance to the four integer pixel locations. The farther is the pixel to an event the smaller the associated weight added to the pixel. A Gaussian distribution is used in the weight computation to smooth the reconstructed image and improve robustness against noise. By adding all the weights of all events to every corresponding pixel, we obtain a pixel grid  $I_{set}$  with floating-point intensity values (a standard greyscale images with floating point values). The process of associating a weight to each pixel is called bilinear interpolation [134]. Finally, a standard 8-bits greyscale image is recomposed with a floating-point to integer conversion. The operation of the event frame reconstruction module (the block 2 of Figure III.13) is detailed in the following pseudo code.



**Figure III.17:** (a) Illustration of the EFR working principle. An ego motion compensated event does not have integer pixel coordinates. Hence, it affects four neighbor pixels, not an individual one. This impact, i.e., the value added to the pixel, depends on the distance between the event and the pixel. The closer the pixel, the higher the value added. (b) Tradeoff illustration between the two event set grouping modes. Both the *event number* and the *hybrid* event set generation modes are parameterized for regrouping 20k events per set - and thus, per frame. Under this condition, with hybrid grouping at 75fps, the RMSE obtained is only 8.7% worse (higher) than with *event number* mode. However, the computation required represents only 58.4% of the one required with the *event number* mode as there are only 3884 frames generated against the 6653. These results have been obtained on the *boxes\_6DoF* data sample from the dataset of reference [21] using our own EB VIO pipeline without pose graph optimization.



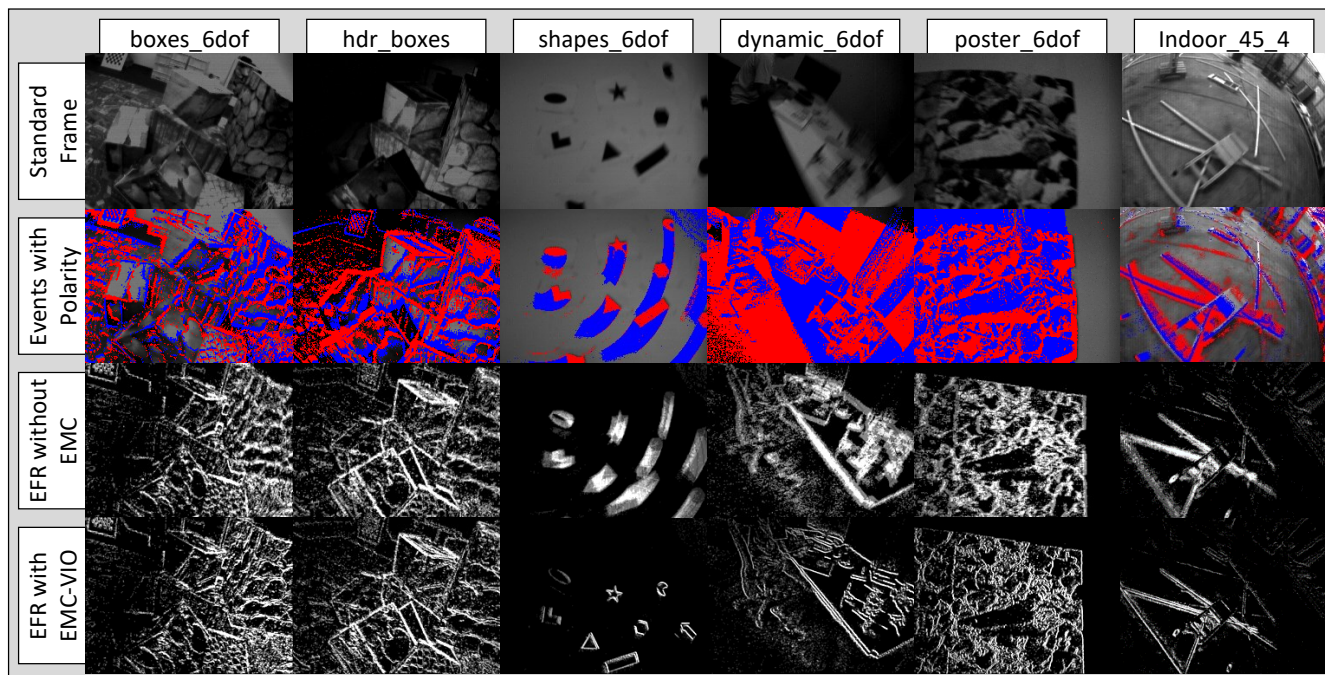
Note that the input events have modified pixel coordinates (ego-motion compensated) that are not aligned on the pixel grid, i.e., they have non-integer coordinates.

- For each ego-motion compensated event of an event set:
  - Find the 4 closest pixels.
  - Evaluate the distance between the event and these 4 pixels.
  - Add a value between 0 and 1 to each of these 4 pixels based on their distance to the event. The further the event is from the pixel, the smaller the added value.
- Once all events of the event set are treated:
  - Get the maximum pixel value of the whole pixel matrix.
  - Normalize the whole matrix by this value.
  - Multiply the whole matrix by 255 to project it back to the range [0; 255].
  - Discretize the matrix to 8b integer values.

At that point, one finally gets an 8b greyscale matrix, i.e., a standard picture, that we call an event frame. See Figure III.17 (a) for an illustration of the EFR module working principle.

#### III.D.2.b.v Illustration of Motion Compensation on Event Data

Samples of obtained frames using the full pipeline, with the VINS-Mono [122] VIO algorithm as backend can be seen in the Figure III.18 below.

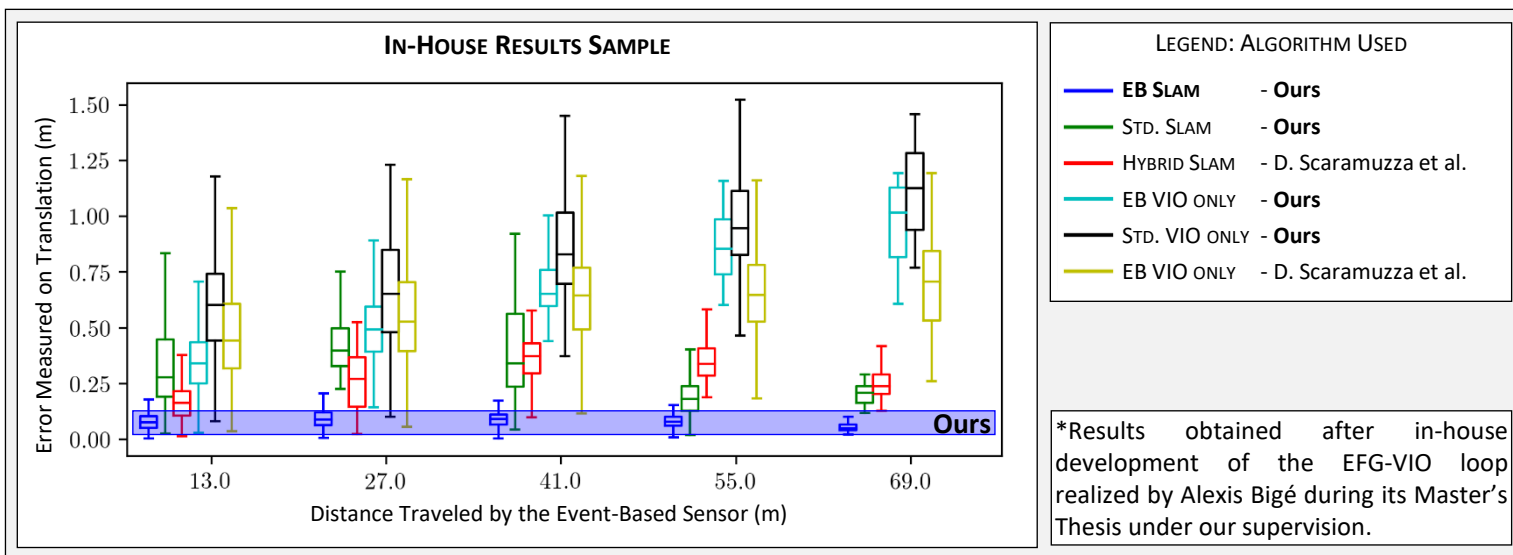


**Figure III.18:** Samples of standard frames, associated events (with polarity), associated EB frame without motion compensation, and associated event-based motion compensated frame for several dataset samples. Each row corresponds to a type of frame, with, from top to bottom: standard greyscale illumination picture; events, i.e., the superposition of 20k events on the standard picture without processing (blue for positive polarity spike, red for negative polarity spike); EFR without EMC is a reconstructed greyscale event frame without EMC applied; and EFR with EMC-VIO is a reconstructed event frame with EMC based on camera states obtained by the VIO module. Each column corresponds to different data samples from the datasets [21, 135].

### III.D.3 Ego-Motion Compensated Event Frames and Usages

#### III.D.3.a Deployment in a VIO Loop

The resulting event frames can then be used to estimate a new pose with a VIO/SLAM module. The accuracy of the position estimation depends on the VIO backend and parameters, but it also depends on the parameters of the event frame generation module - ego-motion compensation (EMC) and reconstruction (EFR). For example, one can change the event set generation mode or the precision of the camera state used for movement compensation. Through experimentation, we have shown that by modifying the full pipeline, namely by using the VINS-Mono [122] VIO backend instead of the original OKVis [136] proposed by [126], it is possible to achieve better results than the state-of-the-art demonstrations [126, 137] on the dataset proposed in reference [21].



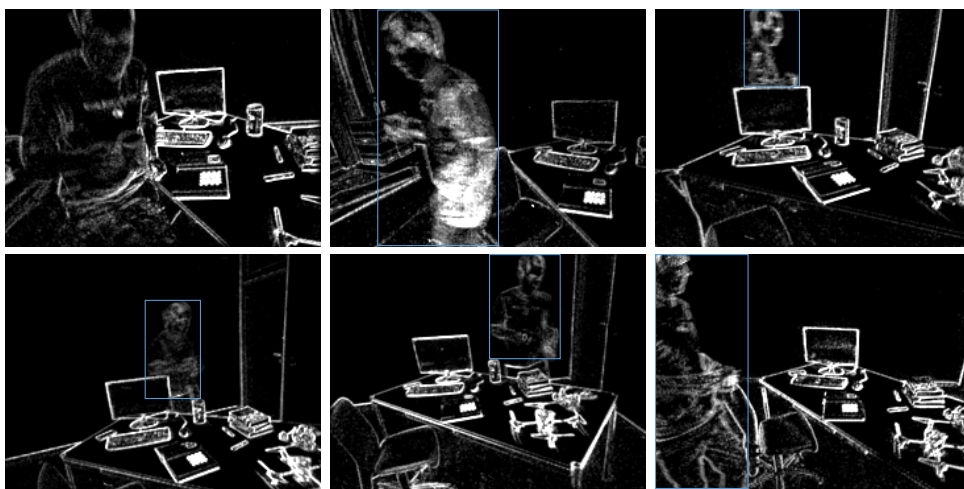
**Figure III.19:** Samples of results obtained with our implementation of the EFG-VIO algorithm on the *Boxes\_6DoF* sample of the dataset of [21]. The different boxes depict the average, the standard deviation, the minimum, and the maximum distances from the groundtruth (in meters). Different version of the algorithm are tested (with and without pose graph optimization). The results titled *Hybrid SLAM EB VIO Only* are directly provided by [137] and [126]. All other results are obtained with our ROS based implementation. On this dataset sample, we reach a translation RMSE of 0.062 with the *EB Slam* version. In comparison, the Ultimate SLAM [137] reaches a RMSE of 0.199. A more exhaustive comparison of the performances of the different implementations (ours and the ones of [126, 137]) can be found in Appendix G.

These algorithms have been tested on all the samples of data illustrated Figure III.18. It clearly appears that using SLAM (i.e., VIO along with pose graph optimization) usually permits to obtain better results. That is true for both our implementation and the one of the team of D. Scaramuzza [137]. A crucial point to highlight is that our implementation of the EFG-VIO loop do not use standard frames for evaluating the pose of the imager. Yet, we still obtain better results than the implementation proposed by [137], called "Ultimate SLAM", that does use both standard frames and event data in the pipeline. A sample of these results is depicted Figure III.19. More results are available in Appendix G.

This study has led to the deposition of a patent - referred to as patent 2 - that has been sent for deposition. This patent discusses the possibility of modifying the precision of the camera state information sent to the EFG module during operation. It also presents a method that permits to initialize the EFG-VIO loop without requiring to modify the VIO backend algorithm. Our ROS implementation can thus integrate any VIO/SLAM module proposed by the state-of-the-art without inducing internal modifications.

### III.D.3.b Object Detection with Motion Blur

Once Ego Motion Compensation is realized, objects that do not move in the world (desk, trees, etc.) appear - ideally perfectly - clear and neat. However, objects that are moving with respect to the world appear relatively blurry, as visible Figure III.20.

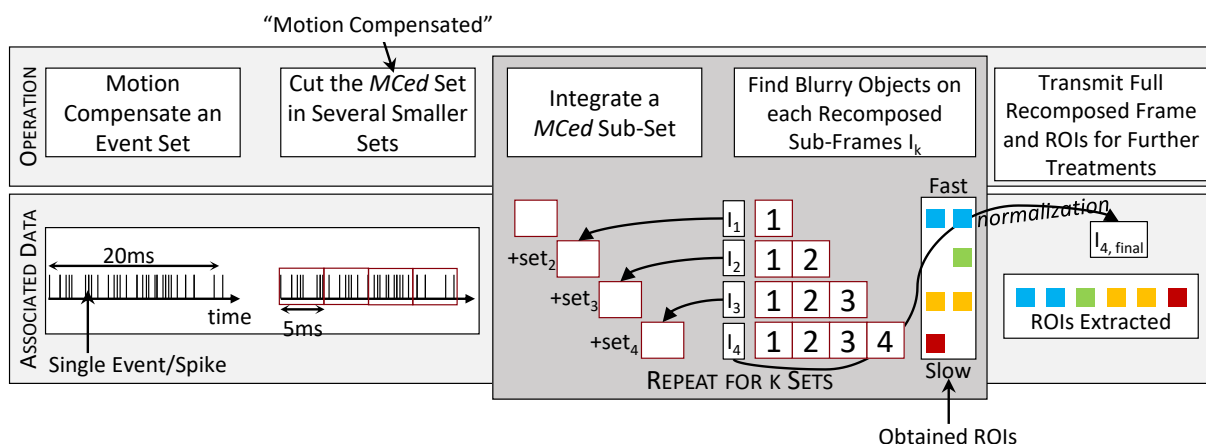


**Figure III.20:** Samples of motion compensated frames based on the VIO mode of our EMC module. The integration time of a single frame is 20ms. The object in movement (human) appears more blurry than other objects (chair, desk, etc.). Note that the black pixels in these frames correspond to an absence of spikes. It illustrates that the generated frames are extremely sparse pictures (all zero-valued pixels can be considered empty).

The apparent blurriness of an object depends on two factors: 1) the projected speed of the object with respect to the camera, 2) the time of integration of the recomposed frame. Some configurations are thus not ideal for detecting objects using this method. The projected speed of the object depends on the direction of its movement with respect to the observing sensor, as well as its speed with respect to the world reference frame. So, detecting an object that may collide with the camera - whose movement is not due to the moving agent/camera - should be easy because this object will very probably appear blurry.

In addition, the time of integration of the recomposed frame is a parameter that may be varied online. Because the user (or program) can control it, several different object speeds can be observed. We explain that point here, and it is illustrated Figure III.21. Once the EMC is realized, the parameters for recomposing the frames (EFR) can be varied without requiring starting again the EMC operation, as these two blocks are independent. Furthermore, recomposing a frame from 0ms to 20ms can be done incrementally by recomposing four intermediate frames of respectively 5, 10, 15 and 20ms, without starting from scratch every

time. So, with this method, recomposing several frames of different duration does not cost more. Now, in the frame of duration 5ms, the objects that move relatively slowly with respect to the image sensor will appear neater than objects that move quickly (like a vehicle crossing the picture from one side to the other). And in the frame lasting 20ms, the object moving slowly will appear blurrier than in the frame lasting 5ms. As a result, diverse types of object can be detected using this technique. Moreover, detecting an object from an EB data stream does not cost much in terms supplementary computations with respect to the full EFG pipeline. It simply a measure of local contrast - as for example with the method proposed in [134] - throughout the generated event frames. And as these frames are extremely sparse, much of the computation can easily be bypassed. The method proposed here has been patented and sent for deposition in 2021. In this manuscript, we refer to it as patent 3.



**Figure III.21:** Computational diagram of the motion blur obtainment algorithm, along with associated data types. This figure also depicts the method for maximizing the efficiency of multi time-scale analysis by incrementally recomposing the event-based frame to extract objects of different speeds.

## III.E Discussion

### III.E.1 The Main Issue of Event-Based Sensors

Throughout this chapter, we have presented and discussed results obtained with four different algorithms: clustering, depth extraction with a shallow SNN, depth extraction with spike-by-spike matching, and EB visual inertial odometry. Every time, we have regularly noted that the complexity of the algorithm is relatively large, and in any case, is always proportional to the number of spikes managed. The number of spikes depends on the camera throughput, which increases exponentially with the sensor resolution (see chapter II). One guiding thread of our thesis is thus to filter out the data emitted by an EB imager directly at the output of the sensor. This in order to avoid having to send and manage too many spikes in downstream processing steps - *large computing*. Going further with this thought led us to consider a system where a convolutional spiking neural network (CSNN) could be used as a filter module for reducing the throughput of event-based sensors.



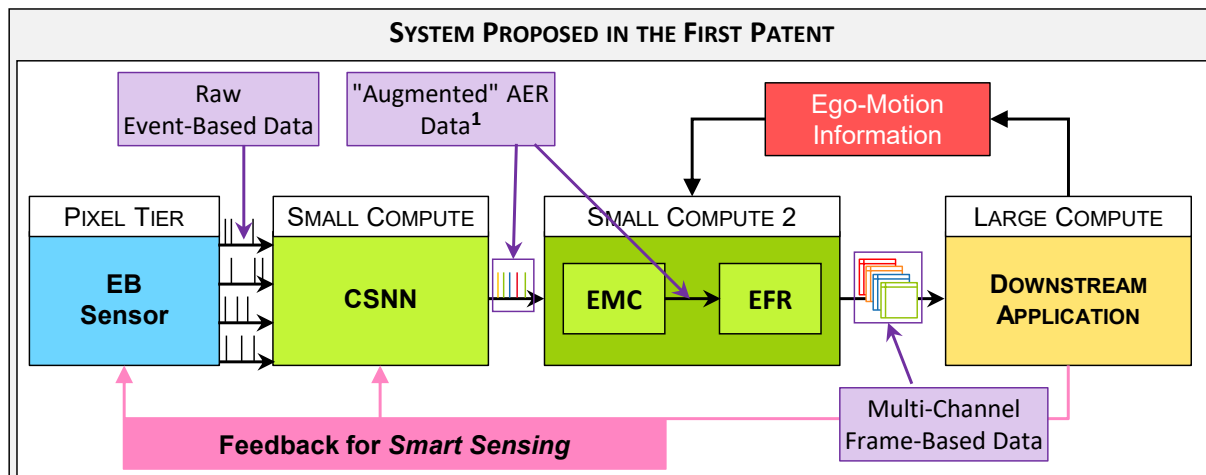


Figure III.22: Illustration of the system presented and patented of patent 1.

## III.E.2 Our View of a Smart Event-Based Imager

### III.E.2.a Patent Deposited

We studied the possibility of applying a CSNN as layer for filtering the data at the output of an image sensor? This point is discussed in depth in chapter IV. This idea led to the deposition of a patent - that we call patent 1. The main claim of this patent is illustrated Figure III.22. It consists in a system where an event-based sensor output is fed into a CSNN module that output augmented AER (multi-channel events), which are then motion compensated and accumulated onto multi-channel frames AER with the EMC and EFR method presented above.

### III.E.2.b Why it Works

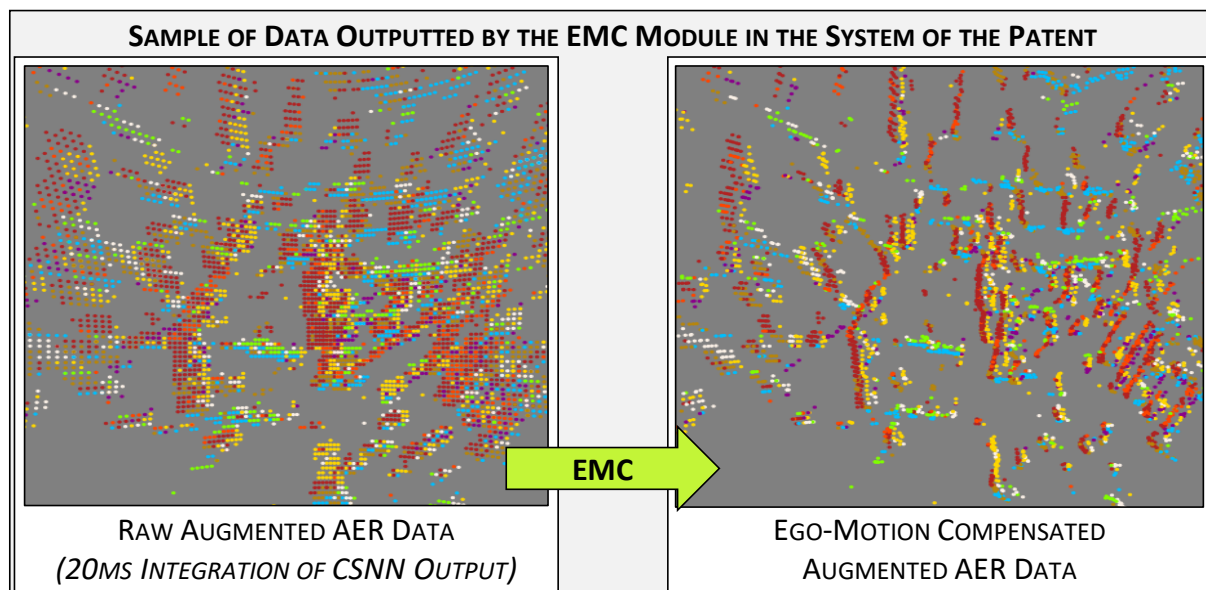
Before patenting, we tested the idea in two ways. First, we simulated a convolutional spiking neural network module onto data acquired by an EB sensor. We have shown that under certain conditions, it can easily permit to reduce the amount of data generated by an EB sensor by a factor 10x. Again, this point is discussed further chapter IV.

After that, the remaining point was to verify that the filtered event stream could still be ego-motion compensated. And as the reader can appreciate Figure III.23, it is indeed possible to apply EMC on data output by a CSNN layer.

### III.E.2.c Future Work

It is still required to test if the augmented AER data could be used inside the EB VIO loop to correctly evaluate the position of the sensor. That point is not resolved yet. However, we are aware that features can be extracted with CNNs on classical data and then used as descriptors inside computer vision pipelines [138]. So, applying a CSNN directly at the output of a sensor should not make downstream algorithms fail. But it is necessary to develop feature descriptors based on the data type output by the CSNN module - which, incidentally, correspond to spatiotemporal oriented edges, more about that in chapter IV.

Moreover, as we end up reconstructing event frames at low frequency with respect to the initial timestamping precision of the EB sensor (respectively millisecond and microsecond



**Figure III.23:** Motion compensating output data of a convolutional spiking neural network. Raw data from the *boxes\_6DoF* sample from the dataset of reference [21]. First our convolutional spiking neural network algorithm is applied onto it. Then, spikes - augmented AER data - are regrouped to compose pictures of 20ms (left). After what we apply motion compensation with our implementation of the EMC module (right). The resulting pictures represent the 3D to 2D projection of these spatiotemporal volume of spikes of 20ms along the time dimension. The reader can find more information about the color of the dots in the next chapter, chapter IV.

time scales); one may wonder why should EB data be generated with as much temporal precision. We have no answer to that. However, we assume that the EFG-VIO loop could be deployed on data converted from frame-based imagers. If the frames are acquired at a relatively high frequency (in the kHz range), using a frame to events conversion pipeline as proposed in section III.C.3.b.i, then the method presented here would stay functional. That idea has also been patented in the patent 3.

### III.F Conclusion

In this chapter, we discussed the possibility of exploiting EB data for computer vision tasks such as object detection, depth extraction and visual odometry. Our original idea was based on the observation that visual processing inside the brain occur through two parallel neural circuits, called the ventral and dorsal pathways[101, 102]. We argue that event-based image sensors should be devoted to uses where traditional camera are not efficient and not robust enough. As a reminder, they present interesting simultaneous HDR, HAS and event-based acquisition properties. These properties are highly attractive for robust - meaning resilient variations of environment and illumination conditions - object detection and movement evaluation. We thus started to test if we could detect objects by clustering events and ended up implementing a full EFG-VIO pipeline under the robot operating system to convert an event stream into ego-motion compensated event frames. Our implementation permitted us to achieve better results than the ones obtained by the state-of-the-art reference [137].

The developments presented here led us to deposit three patents. Patent 2 relates to details that concern the EFG-VIO loop implementation. Patent 3 discusses the possibility of detecting objects thanks to their appearing blurriness on ego-motion compensated event frames. Finally, patent 1 propose a solution for reducing the amount of data generated by EB sensors.

Most of the sensors available on the market deliver inadequate quality data, due to their rough pixel discretisation, noise sensitivity, and low resolution. We thus argue that next generation sensors should focus on providing better quality data, with namely on-chip noise reduction capabilities. This point has been proposed by [18, 32] before us, as well as high resolution and smaller pixel sizes. The best current realization is the one of Prophesee [18] that proposes sensors with on-chip programmable filtering capabilities and the smallest event-based pixel pitch ever of 4.8 $\mu$ m. However, what has not been tested is to implement an SNN directly behind the pixel grid of an event-based sensor. We argue that filtering data near-sensor using such a module could be highly beneficial for many EB algorithm, may they be CV oriented or machine learned. This system has been patented in patent 1 and is discussed in detail in chapter IV.



# IV

## Spiking Neural Networks and 3D Integrated Event-Based Imagers

---

### Contents

---

<b>IV.A Introduction</b>	<b>80</b>
<b>IV.B Combining What is Best</b>	<b>81</b>
IV.B.1 From the World of Imagers	81
IV.B.1.a Event-Based Imagers	81
IV.B.1.b 3D Integrated Image Sensors	82
IV.B.2 From Neuromorphic Circuits	83
IV.B.2.a Bio-Inspired Near-Sensor Filtering	83
IV.B.2.b General Hardware Implementation Strategies	84
IV.B.2.c Three Scopes of Neuromorphic Circuit Design	89
IV.B.3 Efficient Near-Sensor Filtering	93
<b>IV.C Mapping a CSNN onto a 32x32 Macropixel</b>	<b>94</b>
IV.C.1 Definition of the Convolutional Spiking Neural Network	94
IV.C.1.a Neuron Connections and Mechanisms Implemented	94
IV.C.2 From Algorithm to Hardware	95
IV.C.2.a Neuron States	96
IV.C.2.b Memory Area and Number of Pixels	96
IV.C.2.c Algorithmic Parameter Definition	96
IV.C.2.d Standard Hardware Optimizations	97
IV.C.2.e 3D-Enabled Optimization: The Smallest Repeatable Pattern	99
IV.C.3 Number of Pixels and Target Performances	100
IV.C.3.a Minimal Area Limit	100
IV.C.3.b Maximum Frequency Allowed	100
<b>IV.D Proposed Architecture</b>	<b>101</b>
IV.D.1 Pixel Behavior Emulation	101
IV.D.2 Arbiter	101
IV.D.3 Transmitter	103
IV.D.3.a Buffer	103
IV.D.3.b Mapper	103
IV.D.4 Computer	103
IV.D.4.a Neuron Memory	104
IV.D.4.b Processing Element	104
<b>IV.E Results</b>	<b>105</b>
IV.E.1 Methodology	105

IV.E.1.a Design and Tools . . . . .	105
IV.E.1.b Validation . . . . .	105
IV.E.1.c Equivalent 2D Model . . . . .	106
IV.E.2 Input Rate, Chip Frequency, and Power Consumption . . . . .	106
IV.E.2.a Target 400MHz . . . . .	107
IV.E.2.b Target 12.5MHz . . . . .	107
IV.E.3 Discussion . . . . .	107
IV.E.3.a Comparison with Other Works . . . . .	107
IV.E.3.b What is Good . . . . .	108
IV.E.3.c What Could be Improved . . . . .	109
<b>IV.F Conclusion . . . . .</b>	<b>110</b>

## IV.A Introduction

Event-based vision sensors present several interesting properties for computer vision applications like depth-extraction and visual inertial odometry - namely high dynamic range (HDR), high acquisition speed (HAS) and event-based (EB) acquisition. However, they come with several notable drawbacks that drastically impact their efficiency for the realization of such algorithms. They output an exceptionally large bandwidth and acquire data with a relatively large amount of noise. On top of that, EB imagers are limited to small resolutions because their readout module consumes much power as they may need to operate at extremely high speed, as discussed in section II.C.3 and detailed in the Appendix B. Nevertheless, even with poorly resolved sensors, realizing accurate simultaneous localization and mapping operation is highly effective, and deploying ego-motion compensation of the acquired data may enable new types of algorithm and motion blur correction at low cost [100], as introduced in chapter III. To compensate for the drawbacks of EB imagers, namely their noisy acquisition and large output bandwidth, we designed a pre-processing hardware accelerator that is aimed at near-sensor operations.

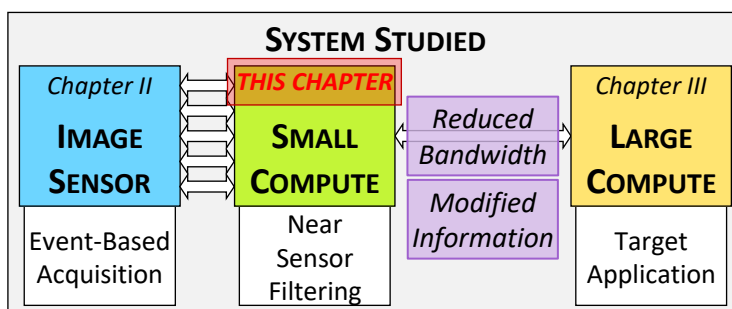


Figure IV.1: System studied in our Ph.D. This chapter focuses on near-sensor filtering.

This chapter presents a neural processing unit evaluating a convolutional spiking neural network (CSNN) conceived to be easily distributed behind an EB pixel grid by employing 3D IC technologies [139, 140]. It is a modified version of our paper accepted at the 58<sup>th</sup> DAC conference to be held in December 2021, which has thus not been published yet

[141]. We present the hardware modules in more details, while the algorithmic section is reduced as neural networks have already been thoroughly introduced chapter II. The neural core presented in this chapter evaluates 256 convolutional spiking neurons and consumes only 2.86pJ per synaptic operation (SOP). It takes advantage of the envisaged 3D parallel interconnection with a block of 32x32 pixels by integrating the pixel readout module which encodes the spikes' address in a custom format that allows for optimized synaptic mapping information - i.e., weights and associated neuron addresses - storage requiring only 300 bits. The timing constraints on the arbiter is also drastically reduced by a factor 900 with respect to an equivalent 720p image sensor, as only 1024 pixels are read by our core. Moreover, the neurocore presented has been designed so that it can be tiled and distributed onto a larger plan to manage high resolution sensors without inducing overhead and the readout, as a macropixel of the RETINE sensor [50]. This requires constraining the allowed core area to the one of the pixels above it, whose pitch is set at 5 $\mu$ m targeting the state-of-the-art 720p EB imager [18] characteristics, resulting in only 0.026mm<sup>2</sup> available for a whole neurocore. The various ideas behind the realization of the circuit are introduced section IV.B. This section also presents the neuromorphic computing paradigm. An efficient way to map a convolutional neural network, along with considerations about how the network parameters have been set, are presented section IV.C. Finally, the micro-architecture of the accelerator is presented section IV.D, and a few characteristic figures are discussed section IV.E.

## IV.B Combining What is Best

### IV.B.1 From the World of Imagers

#### IV.B.1.a Event-Based Imagers

Event-based imagers are bio-inspired sensors based on the asynchronous readout of individual pixels. As was discussed in chapter II, event-based pixels behave as time differentiators of the illumination that impinges onto their photo-sensitive surface [16, 17]. From there, they simultaneously deliver HAS, HDR and EB acquisition properties, without any need for trading them off. In chapter III we illustrated the fact that combining the high acquisition speed and event-based properties enables to efficiently evaluate the position of the sensor relative to the environment [120]. On this HDR adds up and provides the capability of the system to work efficiently both in bright and dark illumination conditions. They are thus relatively different from standard frame-based image sensors, as illustrated by the tables of Figure IV.2.

They are based on models of the rod cells that play the role of photodetectors in the human retina [14]. Each pixel works asynchronously from its neighbors and delivers a spike when a change in the illumination measured overcomes a threshold [16, 17]. The problem with this type of operation is that they logarithmically trigger on the variations of luminance observed, which makes them noisy sensors. Indeed, when the global luminance is low, a small number

(a) EVENT-BASED IMAGERS		(b) CLASSICAL IMAGERS	
PROS	CONS	PROS	CONS
High Dynamic Range (inter & intra "frame")	Large Bandwidth	Dense Information	No Simultaneous HDR and High Frame Rate
High Acquisition Speed	Noisy	Scalable	Redundant (temporally)
Event-Based Acquisition	Important Power Consumption		
	Hardly Scalable		

**Figure IV.2:** Acquisition characteristics of standard and event-based imagers for computer vision applications. **(a)** Properties of event-based sensors. **(b)** Properties of standard frame-based imagers.

of photons received can make any pixel trigger. On top of that, because of the asynchronous and individualized nature of the pixels, a grid of such devices requires complex readout circuitry. Indeed, the electronics permitting to randomly output any cell at any time must be able to receive output requests and respond by deciding which pixel to read first, and then sending a signal to the pixel confirming it has been read. This circuit is called an arbiter, and many different versions of it has been proposed in the literature [15, 142]. Finally, as illustrated Figure II.12 of chapter II, scaling such imagers to high resolution is nearly impossible without modifications of the readout protocol, as the required operation speed would be in the GHz scale. All of this results in a large power consumption, the pet peeve of embedded devices.

So, to compensate for these drawbacks that render EB imagers unsuitable for deployment in power efficient systems as they are, several solutions are being proposed. The two main improvement axes are: adding functionalities inside the pixel grid to eliminate noise before outputting the data from the sensor or modifying the readout scheme to reduce the load on the arbiter. For example, Son et al. [35] access eight pixels simultaneously to reduce the timing constraints on the arbiter. Li et al. [32] filter out faulty pixels and noise by counting and thresholding spikes emitted by groups of 2x2 pixels. It is not the same as having four times less pixels, because it permits to deliver a pre-processed stream of spikes with some defects, noises, and situational limitations (like flicker) removed. Both works propose to apply a synchronous readout strategy to manage timings and limit the bandwidth at the source more easily; more details about this point in section II.C.6.

We do not aim at implementing a synchronous readout of the grid by row or columns. On the contrary, we aim at keeping accurate readout timestamps in the order of the microsecond to avoid damaging the precision of an algorithm behind. With that goal in mind, we follow the lead of Finateu et al. [18] by deploying our filter node directly behind the pixel grid with the use of 3D IC technologies.

#### IV.B.1.b 3D Integrated Image Sensors

With this technology, Prophesee were able to demonstrate the complete realization of a functional 720p (0.92Mpx) event-based sensor at the ISSCC conference in 2020 [18]. The

readout module is instantiated on the bottom tier of the 3D stacked two-layer circuit and can access 1280 pixels - a full row - at once. Without 3D IC technologies this would require many supplementary wires that would add to all other metal lines required for the functional operation of the pixel. Indeed, in usual readout circuits, lines for accessing the pixels are shared by row and column, as illustrated Figure II.4 in chapter II. However, with 3D IC technologies, directly accessing each pixel individually or by group is easier as connection pins can be distributed throughout the top metal layer of the bottom tier. On top of that, EB pixels that directly integrate an event generation module only needs to output a single bit, which indicates that this pixel "*wants*" to spike - the request signal. And a second line can then be used to respond to the demand - the acknowledge signal. So, ideally, an event-based pixel grid can be fully readout by deploying two lines per pixel.

With 3D IC technologies interconnection pitches now reaching the micrometer scale, and pixels being approximately squares of  $5 \times 5 \mu\text{m}^2$ , Prophesee were thus able to design a 3D stacked imager with dense interconnection between the top and bottom tier [18]. Furthermore, the bottom tier is used as a filtering module based on programmed ROIs in addition to formatting the output data in a format more compact than the classical AER protocol. The number of events and the bandwidth is thus reduced by a factor 2.35x, from 2.5Gev/s in internal activity to 1.066Gev/s output at maximum capacity.

Non EB imagers have also been using 3D IC technologies for enabling advanced vision constructs. At first it was mostly used for maximizing pixel fill factors and reducing their size. Now, it is also used to add "intelligence" inside, i.e., the ability to realize advanced algorithmic operation on data acquired by the pixel grid directly behind it. A major realization in this perspective is the RETINE sensor [50]. It consists in a highly configurable image sensor, in terms of frequency of acquisition, with a shared ADC by group of  $2 \times 2$  pixels; under which comes a single instruction multiple data processor. The major realization relies on the fact that the processing operation, realized by several stack processors distributed throughout the pixel grid, is arranged by block of pixel and processor. These blocks - called macropixel (MPX) - are fully functional and work individually from their neighbors. They can interact with each other, e.g., when data should be transmitted in between them, but most of their operations *flow* from the pixel grid (where the data is acquired) to their own processor. Splitting up the operation of the sensor (pixels and processing) in blocks has the major advantage of enabling to scale such imagers to high resolutions, as each block integrates all the elements it requires to work on its own.

## IV.B.2 From Neuromorphic Circuits

### IV.B.2.a Bio-Inspired Near-Sensor Filtering

Hubel and Wiesel have shown that one of the first steps related to visual perception in mammals' brain is edge orientation detection [86]. Their work suggests that the biological vision pipeline relies on successive processing steps, where visual primitives (edges, color,

texture, etc.) are extracted first, and in parallel, by distinct neural feature extractors ("kernels"). All these extractors transmit the result of their operation to further downstream neural constructs of the visual cortex where other, more advanced, or abstracted, information is extracted - up to object recognition and spatial localization [14].

Similarly, in artificial neural systems, edge orientation discrimination is usually realized by the first layers of convolutional spiking neural networks when trained with (spike timing dependent plasticity) STDP methods on EB data, as has been shown by several research groups [143, 144]. As detailed in Appendix E, even if the STDP learning rule is not efficient for obtaining an accurate NN on complete tasks related to object recognition, it still permits to obtain filters - weight patterns - that corresponds to statistically salient features, especially when deploying a convolutional configuration [145]. So, deploying a spiking neural network as a filtering node behind an EB sensor would present several advantages. First, SNNs are naturally suited to operate on event-based data [141]. Secondly, leaky neuron models continuously update their states through time and naturally eliminate any residual - noisy - information after a brief period, at the scale of the tens of milliseconds. Hence, leaky SNNs deployed behind an EB may enable efficient filtering of EB data on a continuous and long-term basis. Finally, a convolutional network is easily distributable on large scale images because convolutional neurons are local filters, observing only a small portion of the input frame - their receptive field (RF). And hardware accelerators dedicated to the evaluation of SNNs are usually designed as distributed processors for a matter of efficiency, as will be discussed below section IV.B.2.b. So, we decide to implement a single layer convolutional spiking neural network that looks for oriented edges in the data directly behind a grid of event-based pixels. More details about the specificities of the algorithm implemented is discussed section IV.C.

#### **IV.B.2.b General Hardware Implementation Strategies**

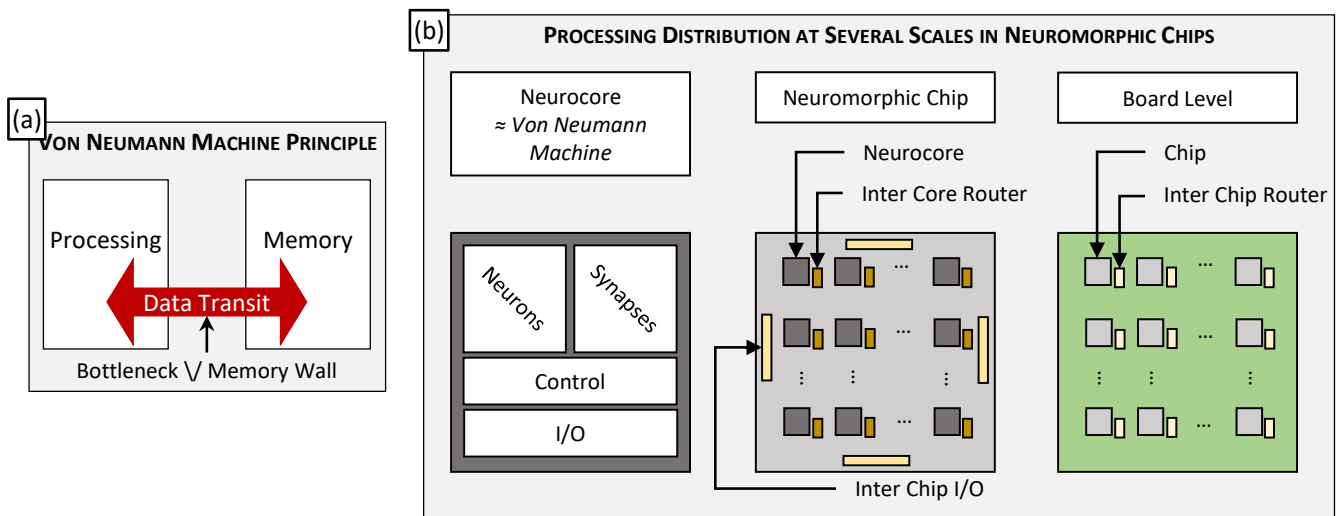
As already touched upon, SNNs are computationally heavy algorithms. Their dataflow naturally exploits the sparsity of spike-based communication between the neurons, so an inference task should ideally require less computations than an ANN architecture. However, as a single spike is not sufficient to entirely represent a value, membrane values undergo many additions before reaching their final state at every algorithmic time step. And as discussed section II.E.2.c.i, the membrane values and associated spike timestamps must be memorized in between each algorithmic tick. The huge consequence is only a dedicated hardware can take full advantage of the event-driven sparsity of spiking NNs. However, actual efficiency gain requires fine design of the accelerator.

##### *IV.B.2.b.i Hardware Platform*

**THE NEUROMORPHIC COMPUTING PARADIGM** Such accelerators are called "*neuromorphic*". They are organized as a fully distributed and parallel neuron state evaluators - similarly to an SIMD processor. The term "*neuromorphic*" is usually opposed to Von Neumann machines,

which denotes traditional computer organization, where the compute occurs apart from the main memory, as depicted Figure IV.3 (a). It is largely accepted that the main limitations, in terms of throughput and power consumption, when evaluating any type of NNs, comes from the memory bottleneck [146–148]. With inspiration taken from the brain, where every synapse and processing neuron are dispatched in 3D and operate simultaneously and in parallel, the ambition of the neuromorphic computing paradigm is to bring closer in space the memory and the processing elements (PE)s. In addition, it also relies on heavily parallelizing the evaluation of neural networks - which are usually event-driven in the neuromorphic paradigm - among many distributed neurocores.

**HIGH LEVEL NEUROMORPHIC CIRCUIT DESIGN CONSIDERATIONS** A neuromorphic circuit (or chip) is usually separated into many neuromorphic cores - *neurocore* for short -, where the memory and several PEs are specifically arranged to minimize the memory wall limitation discussed above. The layers of the neural network evaluated are distributed/mapped among all the available neurocores. In effect each neurocore is assigned a certain number of neurons to evaluate. They include memory modules (usually SRAMs) to stores the synaptic weights and the neuron states of the evaluated neurons. They continuously perform the



**Figure IV.3:** (a) Illustration of a Von Neumann machine and highlight of the data transit bottleneck in terms of computation speed and energy efficiency. (b) Illustration of the distributed architecture a neuromorphic accelerator usually relies on.

neuron updates for each input spike they receive. A neurocore exploits the address event representation protocol to communicate with other cores, usually by sending and receiving spikes on a communication network deployed as a network-on-chip (NOC) [15, 149, 150].

This type of architecture is scalable as long as the routers and the control circuitry can manage the AER requests. Still, the definition of the number of neurons to evaluate and synapses to be memorized per core, as well as the number of cores per chip limit the implementable topology. To overcome this limitation, some implementations go to multi-chip architecture [76, 151], up to multi-board architectures [152] and even multi-wafer [153]. Major



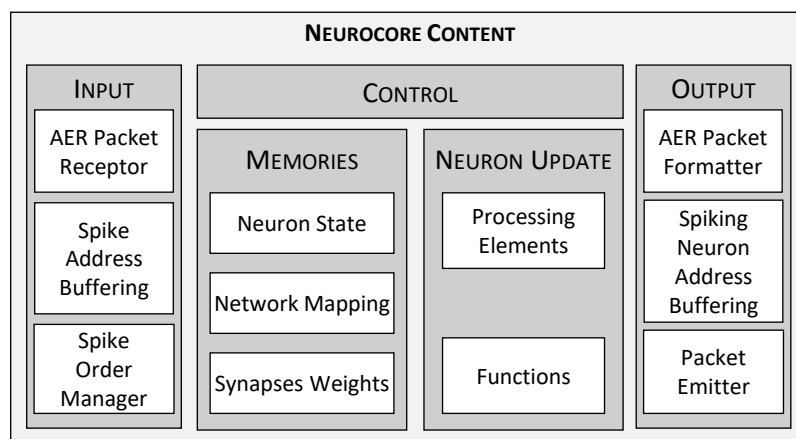
large scale neuromorphic accelerators demonstrated are BrainScaleS from the Human Brain Project [153], the recent Loihi from Intel [151], Neurogrid from Stanford University [74], MorphIC from the UCLouvain [150], SpiNNaker from the University of Manchester [154], and TrueNorth from IBM [76]. After an in-depth study of the literature in 2018, we wrote a survey about these large-scale processors, and many other smaller scale ones [25].

**NEUROCORE ORGANIZATION** As said, a neurocore is the hardware module that realizes the neuron updates. As any advanced hardware processor, a neurocore contains some control circuitry, input, and output modules. More specifically, it includes:

- **The memory**, for storing both the network map and associated synaptic weights, and neuron states. The neuron state contains both the neuron membrane potentials and at least the timestamps of the last input spikes and last output spike received by this neuron. These timestamps are used for evaluating time-based mechanisms, e.g., the leakage and the refractory period for each neuron.
- The hardware **module for neuron update**, which consists in the processing element(s) and several functionalities for realizing the various specific neuron functions and orchestrating data flow management in between the PEs.

A schematic illustration of the different components of a neurocore is represented Figure IV.4. From there the organization of data and instructions communication between the blocks, the precision of the blocks, the level of functionalities, etc., depends on the designers' scope and purposes.

**ORIENTATION OF OUR PLATFORM** For filtering data near an EB sensor, we distribute the computation of a full CSNN layer onto several neurocore, with one neurocore per macropixel. The cores do not communicate together via a NOC, they simply interact from one to the next with direct neighbor macropixels.



**Figure IV.4:** General organization of a neurocore, depicting the different modules usually contained in a core.

#### IV.B.2.b.ii Mapping the Algorithm to the Hardware

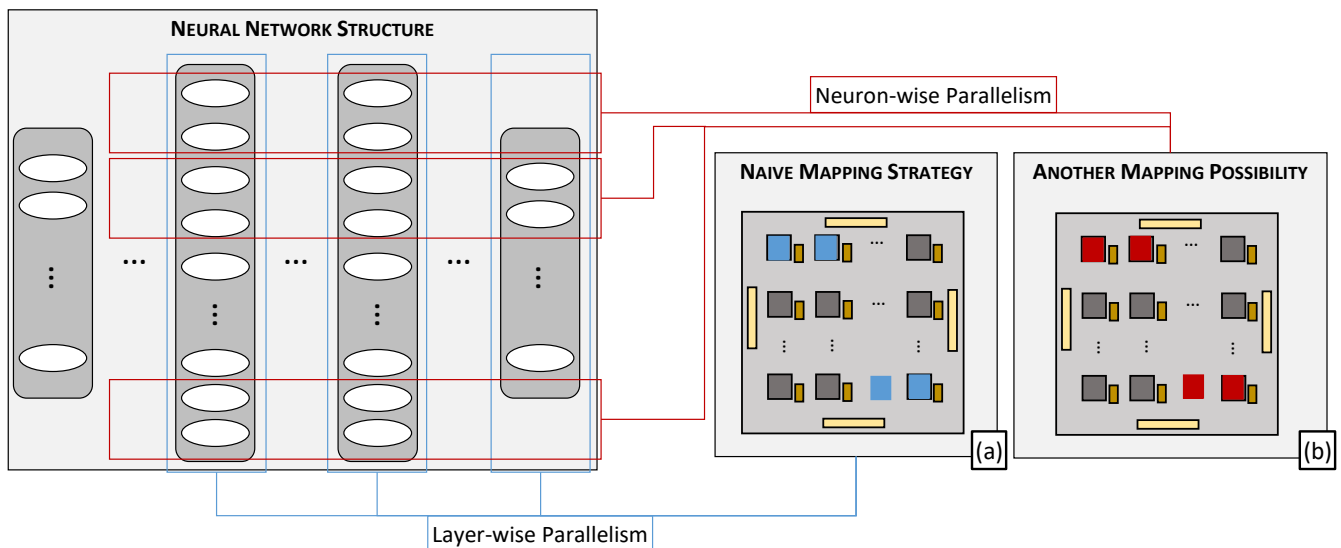
Once the architecture of the accelerator is defined, it is necessary to map the network onto it, i.e., to associate each neuron to evaluate with each neurocore of the chip.

**ASSOCIATING NEURAL LAYERS WITH NEUROCORES** The mapping strategy employed to evaluate an SNN onto a neuromorphic hardware largely affects its performances. A naive mapping strategy consists in having one layer evaluated per core, as illustrated by the blue rectangles on Figure IV.5 (a). With this solution the computational efficiency of the neuromorphic chip is highly dependent upon the network topology [96, 155].

Let us consider the following case study. One maps the network so that three identical neurocores evaluate a fully connected (FC) network with a relatively small topology of 100-1000-10 units. Each single core fully evaluates a single layer. The neurocore evaluating 1000 neurons will require 100 times more updates than the one evaluating only 10 neurons for each input spikes. On top of that, the number of spikes output per neuron usually decreases along the depth of the network [156]. So, with this naive mapping, the computational load of each neurocore, in events per second, will be drastically different, resulting in a non-optimal use of the neuromorphic cores. On top of that, implementing this layer-wise parallelism results in having each neurocore waiting the output of the previous one before starting its operation. It induces a latency between the input presentation and the output obtainment, latency that scales linearly with the network depth. A second mapping strategy would be distributing the neurons along the layer dimension (as opposed to the depth) of the network, as illustrated Figure IV.5 (b). The latter provides several benefits for fully connected networks, but less for convolutional neural networks as every neuron of every layers shares the same weights and thus implies an important memory redundancy between the neurocores. This short case study reveals that correctly mapping the algorithm to the hardware is essential to fully exploit neuromorphic hardware capabilities. This point is further discussed in section 2.3.2 of our survey [25].

**OUR MAPPING STRATEGY** We implement neuron-wise parallelism (blue case) onto our chip as it implements only a single CSNN layer. Each core thus evaluates the same kernels, but on a different portion of the input. On top of that, we introduce a specific mapping strategy that enables us reduce weight and mapping memory to a minimal 0.3kb. This point will be detailed section IV.C.2.e.

**NOTE ON PROGRAMMING AN APPLICATION-SPECIFIC CIRCUIT** Once the algorithm is set, and its mapping onto a specific hardware is decided, the chip has to be programmed so that it realizes the function. The amount of programmability of an application-specific integrated circuit (ASIC) - neuromorphic circuit here - is called *flexibility*. A general observation about programmability is the more the flexibility, the less the energy efficiency and/or the more



**Figure IV.5:** Illustration of different parallelisms possible when mapping a network topology to a dedicated large scale neuromorphic accelerator. **(a)** Parallel processing of each layer. **(b)** Parallelism along the depth of the network.

the circuit footprint. Indeed, more functionalities imply more control circuitry, resulting in increased overhead as well as the need for tools to facilitate the configuration of the chip.

In addition, unlike CUDA [157] or TPUs [11] cores, most of the neurocores are not general-purpose evaluators. So, the specifications of the SNN parameters - e.g., the bitlengths of the membrane and weight values - usually have to be revised, as energy efficient circuit design relies on techniques such as approximate computing, quantization, and others, and thus embeds reduced precision operators. Finally, being distributed architectures, neuromorphic accelerators present issues shared by any distributed architectures [157], with notably an overhead induced by the need of synchronizing and dispatching the data between the different cores. So, algorithms are often constrained to meet the specifications of the target neuromorphic platform.

To motivate the usage of such circuits, many groups thus propose a software - or at least an application programming interface (API) - for interfacing with their hardware. For examples, the SpiNNaker and the BrainScaleS realizations are now regrouped under the Human Brain Project [158] and propose a script interface for interacting with the material. Another example, the Loihi chip [151], is accessible through project proposals to the Intel Neuromorphic Research Community and can even be accessed remotely [159]. In this way, algorithms are designed specifically for the executing platform.

#### IV.B.2.b.iii Main Differences with the Brain

As the reader may have understood by now, traditional computers and human brains are highly different. In short, it is highly parallel with interwoven computation and memory elements, deployed in volume, and operates asynchronously.

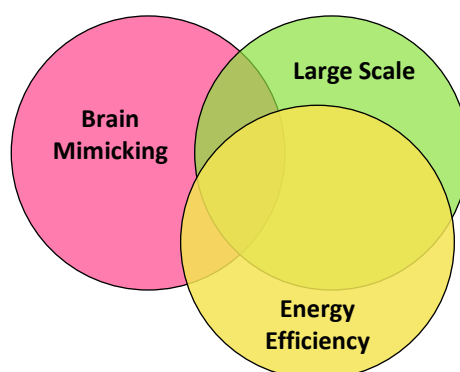
**SPACE ORGANIZATION** The neuron circuits are organized throughout the three-dimensional space the brain offers. Each neuron can communicate with many other neurons, reserving that it is connected to it. Hence, the distribution of neurons is relatively free. On top of that, neural circuits are organized by functions: neurons performing the same task are spatially close and distributed in groups sometimes isolated from each other [14].

**BRAIN PERFORMANCES** The brain counts around 86 billion neurons for a power consumption of around 25W [160]. Which is a relatively small power consumption for  $10^{15}$  operations per second, resulting in around 100TOPS/W (Tera operations per second per Watt). In comparison, Google's tensor processing unit (TPU) - a datacenter processor dedicated to the evaluation of large-scale ANN architectures - reaches 2.3TOPS/W [11]. The TPU is thus 50x times less efficient than the brain although it is designed specifically for its task.

**OTHER MECHANISMS** On top of that, many different mechanisms are occurring inside the brain, and are hardly reproducible on Silicon-based processing elements. For example, neuron firing synchrony, more universally known as neural waves, have been observed in the visual cortex [14, 161]. These researches suggest that not only information is processed and transmitted by individual neurons, but it is also communicated on a population level by the simultaneous activity of different units. Other examples can be given such as: communication between neurons is also realized through chemical interactions through the extracellular medium [162], or neuronal connection speed depends on the myelination levels of dendrites [163], etc. Most of these mechanisms are not considered for designing SNNs nor neuromorphic hardware.

### IV.B.2.c Three Scopes of Neuromorphic Circuit Design

We distinguish three main objectives for designing neuromorphic chips: brain-mimicking, large-scale EB machine learning acceleration, and power-efficient machine learning, illustrated Figure IV.6.



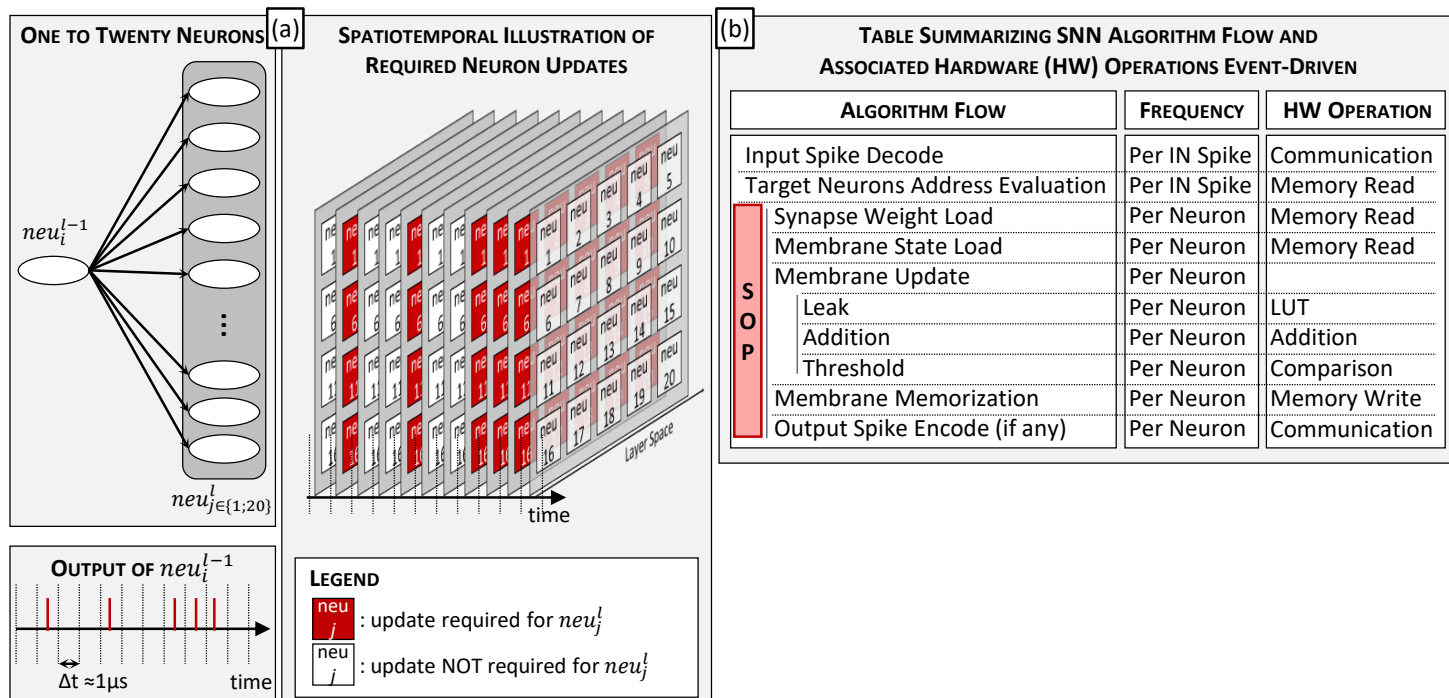
**Figure IV.6:** The three scopes of neuromorphic circuit design and their intertwining.

#### IV.B.2.c.i Bio-Mimicking for Cognitive Modeling

Neuromorphic design initially aimed at delivering new or advanced insights in neuroscience [15], following the philosophy C. Mead [26]. Mimicking complex brain behaviors sometimes go

deep into details with implementation of ionic channels and other bio-realistic components [74, 75, 164]. The precursors of this field first argued that analog asynchronous electronics was best suited to reproduce complex cognitive behaviors because neurons can be represented as analog devices, as illustrated Figure II.20 on page 36. However, latest chips designed for cognitive modeling are mostly realized in fully digital globally asynchronous locally synchronous (GALS) electronics. Analog circuits are hard to program, take more space than their digital counterparts, and are quickly limited by fanout, especially when the purpose is to connect thousands of neurons together. The only major processor realized in analog electronics is the ROLLS [75]. Other notable cognitive modeling oriented neuromorphic chips are SpiNNaker [154], BrainScaleS [153], and - to some extent - Loihi [151]. Also, an interesting demonstration of fully asynchronous digital design has been realized by Moradi et al. [165]. These processors, along with many others, are discussed in more details in our survey [25].

#### IV.B.2.c.ii Neuromorphic Circuits for Machine Learning



**Figure IV.7:** (a) Spatiotemporal representation of the computational requirements of spiking neural network in a 1x20 fully connected configuration. (b) Description of the algorithm flow of an SNN from an input neuron point-of-view, and associated hardware operation. Note that to keep the flow of time, an input spike is a spike output by an upstream neuron at the *previous time step*.

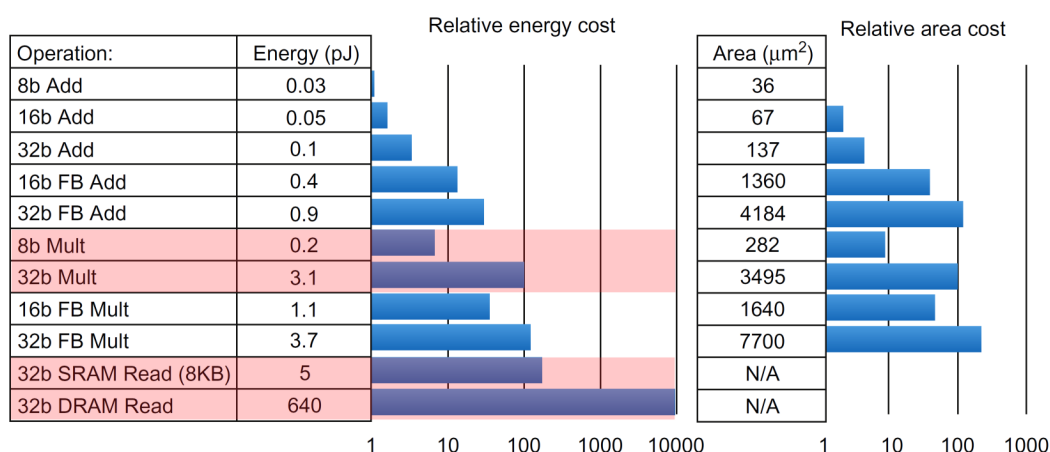
Simulating SNNs is computationally heavy, in terms of time and computational resources. They require to realize the inference of full networks for several time steps, as illustrated Figure IV.7 (a). For example, a sample of the N-MNIST dataset lasts around 300ms [166]. Data can be time-discretized with an exceptionally fine grain time step; as is the case of EB vision sensors that provide spikes sampled at the microsecond resolution [18, 35]. And because of the large amount of interconnection between neurons, a single input spike induces numerous neuron updates. All this combined lead to an extremely large complexity,

which depends on the average input data rate, and the various amount of NN parameters and functionalities. But the major difficulty with event-based data is their un-predictability. Because of their sparsity, knowing in advance which neuron will need to be updated is not possible. Effectively scheduling the evaluation of an SNN on classical computers is thus extremely costly with many cache misses. That is why intrinsic EB hardware for evaluation of deep spiking NNs is a necessity for any efficient large-scale deployment of these type of algorithms.

Several such chips have been designed, with notably Loihi [151] and TrueNorth [76]. The former is more recent, and interestingly clearly illustrates the concept of flexibility/energy efficiency tradeoff. Indeed, based on the minimum energy figures given in reference [151], Loihi consumes at least 4x times more energy per synaptic operation (SOP) than TrueNorth, while they are designed at the 14nm and 28nm CMOS node, respectively. This because Loihi implements many different neurons and synaptic models while TrueNorth only has binary weights.

IV.B.2.c.iii Event-Based Neural Networks for Energy Efficient Machine Learning

**MEMORY MOVES, A SMALL FEE?** Research around neuromorphic chips have been motivated by the idea that thanks to their important sparsity and low-cost information communication protocol, SNNs should be more energy efficient than an equivalent classical artificial neural network. But, as discussed in the previous section (IV.B.2.c.ii), their sparsity and many-step evaluation requirements make them hard to benefit from. Also, because a spiking neuron update requires only additions, and no multiplications, many advocated that its energy cost is significantly reduced with respect to the multiply-accumulate (MAC) operators of their classical counterparts.

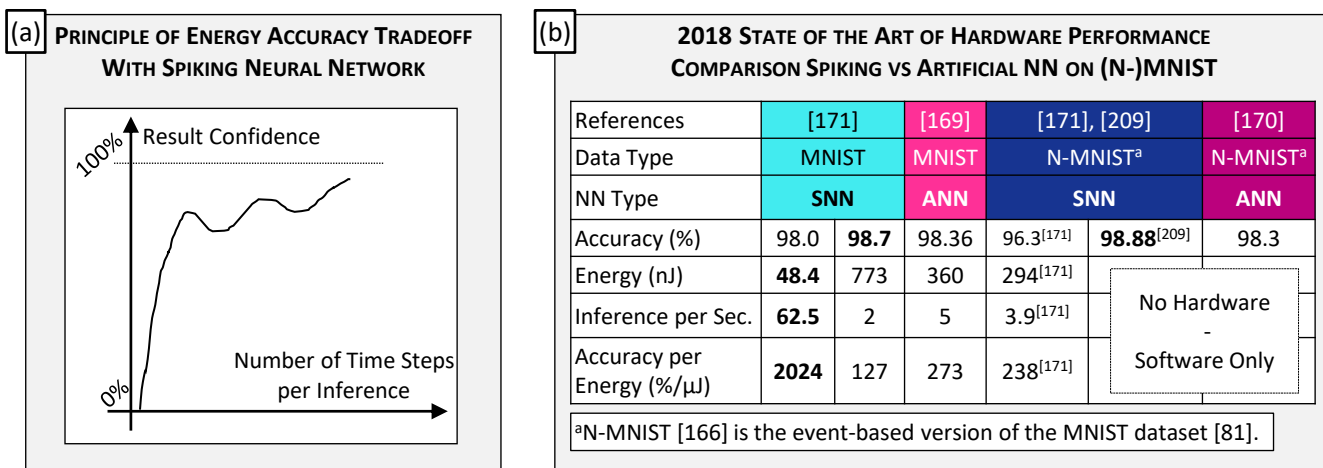


Energy numbers are from Mark Horowitz \*Computing's Energy problem (and what we can do about it)\*. ISSCC 2014  
 Area numbers are from synthesized result using Design compiler under TSMC 45nm tech node. FP units used DesignWare Library.

**Figure IV.8:** Energy and area costs of arithmetic operations and accesses to SRAM and DRAM in the TSMC 45nm technology. Extracted from [20].

However, what is often overlooked, is that spiking neurons require to load and store their states from and to memory. In contrast, ANNs conceptually do not require to store activation





**Figure IV.9:** (a) Illustration of the energy accuracy tradeoff available with SNNs when deciding the number of time steps evaluated per inference. (b) Table summarizing the software and hardware performances of state-of-the-art realization. Adapted from our survey [25] ©2019 ACM. The column MNIST by a standard NN is based on an efficient ANN accelerator realized by Whatmough et al. [168]. The column N-MNIST by a standard NN is based on software simulations by Neil et al. [169]. Note that the last accuracy percentages are harder to obtain than the first ones, so consider the Accuracy per Energy carefully as there is no actual linear relationship.

values. A full synaptic operation thus demands memory movements (see Figure IV.7 (b)). And as clearly stated by M. Horowitz [167], reading a 32b word from an 8kB SRAM consumes 1.66x and 25x more energy than a 32b-precision and 8b-precision Multiply operation; as illustrated Figure IV.8. It is even worst (more than a hundred times) for equivalent DRAM operations, because of the memory bottleneck.

**ENERGY EFFICIENT DESIGN STRATEGIES** So, as for their artificial counterparts, techniques for designing energy efficient event-driven hardware platforms have been conceived. An overview of these methods can be found in Appendix F. There, we discuss standard hardware optimization methods such as dynamic voltage and frequency scaling (DVFS), and quantized and phenomenological computing - which we mistakenly regrouped under the appellation "approximate computing". We also present optimization methods specific to spiking NNs such as data skip, data stream processing, energy accuracy tradeoff, and the AER protocol [25, 85]. A key point to highlight is the fact that usually, the operators' precision of dedicated accelerator is reduced (to at least 8b [85]). Hence, the energy gap between an SRAM Read and a Multiply operation is closer to the  $25/4=6.25x$  range than the 1.66x one.

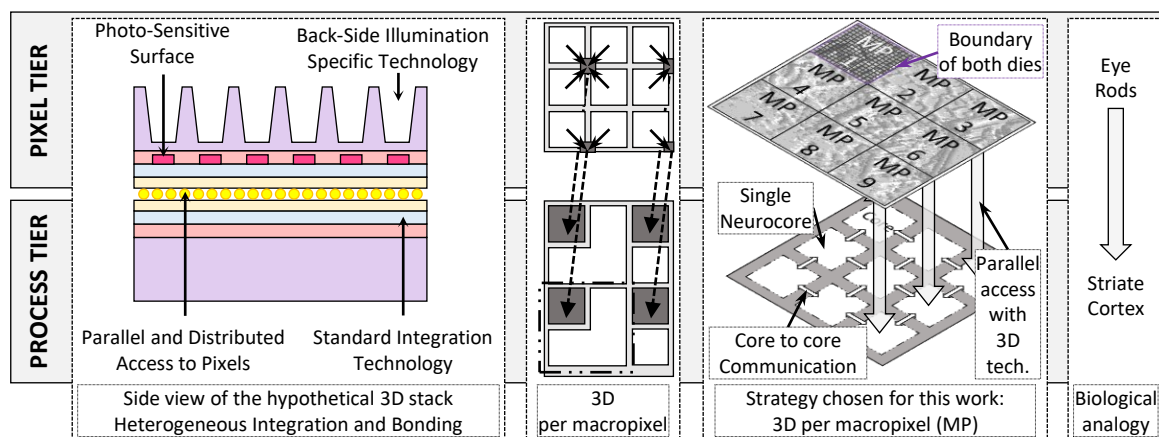
**TRADING ACCURACY FOR ENERGY** Nevertheless, a notable advantage of SNNs is the ease of trading off accuracy for energy. Indeed, spiking neural networks require several time steps to realize a single inference operation. And the confidence in the result tends to increase with time and number of time steps, as illustrated Figure IV.9 (a). However, most of the accuracy is gained in the first few time steps of evaluation, as discussed by Park et al. [170]. This means that the number of time steps evaluated per stimulus can be reduced while keeping the accuracy relatively high. A small loss is usually observed, but the network is still



fully functional. For example, Yin et al. [171] have shown that by decreasing the number of inference time steps, they could pass from 773nJ per classification on the MNIST dataset at an accuracy of 98.7%, to only 48.4nJ (around 16x times less) for an accuracy loss of only 0.7%. Another similar example has been demonstrated by Park et al. [172]. They reduce the number of time step to a single one, and still conserve an equivalent accuracy.

### IV.B.3 Efficient Near-Sensor Filtering

The circuit chip we designed thus mixes many design techniques and technologies to realize a particularly energy efficient near-sensor filtering. It consists in a convolutional spiking neural network accelerator distributed directly behind an event-based sensor, which realizes edge-orientation detection with leaky models of neuron. We apply the macropixel readout strategy to an event-based grid of pixels for enabling direct parallel communication between pixels and neurocores. The envisioned construct is illustrated Figure IV.10.



**Figure IV.10:** Left: side view of the envisaged 3D stacked imager. Middle: Top view of the interconnection strategy chosen for this work. Right: three-dimensional representation of the 3D stack. Right plus: analogous biological system.

Distributing the processing permits to reduce the arbiter size of each MPX which drastically reduces its sampling frequency requirements and could permit to design megapixel resolution sensors.

On top of that, we minimize the flexibility and size of the CSNN implemented to maximize the energy efficiency of our accelerator. We choose to implement the neural core in a data-stream processing perspective to take advantage of the EB nature of data: no computation or data movement is uselessly realized when no input data is available. We thus massively exploit clock gating to reduce the power consumption of the chip no activity is emitted by the pixel grid.

For clarification, we designed only the 3D processing tier of a hypothetical macropixel, by considering that it could receive spike requests from each individual pixel of a hypothetical pixel grid that would be on top of it; in a 3D stacked architecture implementation. We consider that the input pixels have pins distributed on the last metal layer - with pitches of

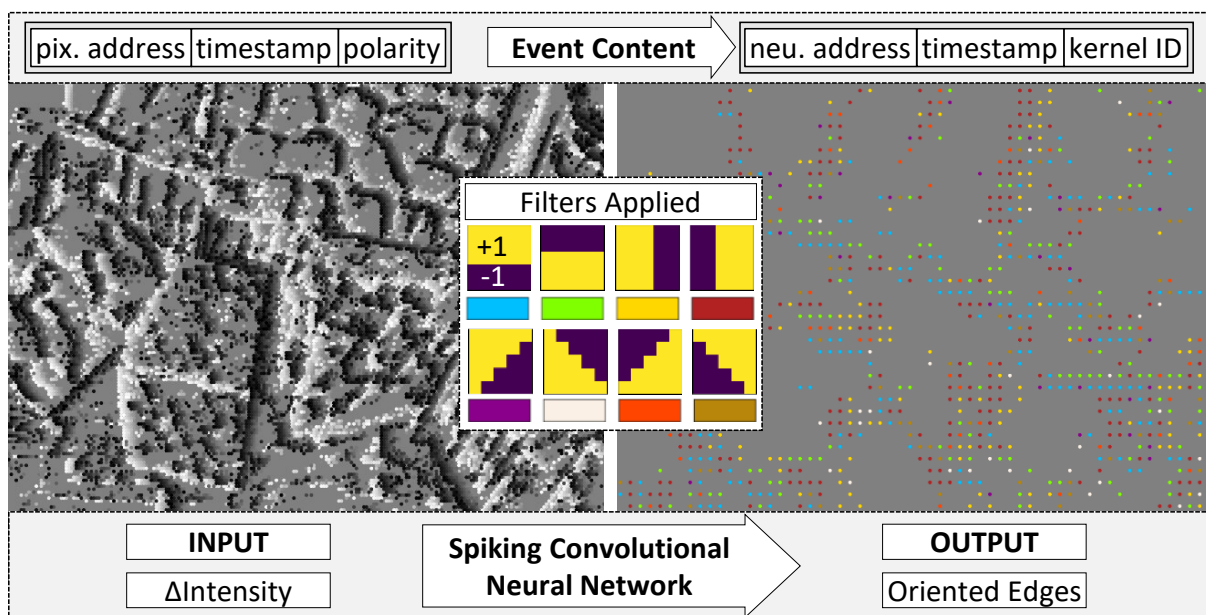
5 $\mu$ m - throughout the floorplan, which is restricted to only 32x32 pixels. To obtain data from neighboring pixels, the designed neurocore can communicate with cores of neighboring macropixels. It does not access these pixels directly.

Our main goal was to demonstrate that near sensor SNN acceleration - for reducing noise and bandwidth of EB sensors - could be realized efficiently and at a small energy cost. This, at the relatively accessible technological node of 28nm, with a restrained floorplan surface limited only by the dimensions of the pixels above the neurocore. The point of restricting the floorplan is essential as the designed macropixel can thus be tiled indefinitely for enabling high resolution EB sensors.

## IV.C Mapping a CSNN onto a 32x32 Macropixel

### IV.C.1 Definition of the Convolutional Spiking Neural Network

The neural network implemented is relatively simple. It is composed of a single spiking convolutional neural layer. Each neuron evaluates eight kernels, focusing on finding oriented edges.



**Figure IV.11:** CSNN results. Left: input data, B&W dots correspond to -1 & +1 event polarity, respectively. Right: corresponding output data, round dots correspond to vertical and horizontal filters, triangles dots correspond to diagonal ones. Input data from [21]. Simulations developed and realized in Python language (ver. 3.7.)

#### IV.C.1.a Neuron Connections and Mechanisms Implemented

Convolutional spiking neural networks conserve and exploit the spatiotemporal information contained in the input data. As neurons are regularly distributed throughout the input space, spatial information is conserved by labelling output events with the address of the emitting neuron. Temporal information is conserved and filtered thanks the two time-related mechanisms, namely the leakage and refractory period. As a reminder, the leakage consists

in continuously decreasing the membrane potential of every neuron, and the refractory period mechanism forbids a neuron to spike when the time between two consecutive spikes is smaller than the refractory period (duration), as illustrated Figure II.26.

The leakage filters out input spikes that are emitted too far apart in time by eliminating the contribution of old spikes from the membrane potential of the neuron. It thus limits the maximum observable time of a neuron, as would a sliding window. If two spikes impinge on the same neuron in quick succession, the contribution of the first spike will not be erased when the second spike is taken into account. The refractory period limits the maximum firing frequency of emission, limiting the bandwidth at the output of the layer and filtering out potentially redundant information. It has the advantage of naturally limiting contributions from faulty always-on pixels.

Hence, applying a monolayer CSNN with Leaky Integrate and Fire (LIF) neurons on a raw stream of events acquired by an EB camera enables to conserve spatial information while reducing noise and event-rates. The output is a new stream of events, each corresponding to a spatiotemporal pattern - called feature or kernel - that fired, as illustrated Figure IV.11. Note that a single neuron evaluates several kernels. Each kernel has its own associated membrane potential, and from a hardware point of view, a neuron actually simply corresponds to the address of a receptive field position.

**Table IV.1:** (a) CSNN algorithmic parameters and values. Possible weight values: -1 or +1. (b) Associated CSNN accelerator hardware parameters and values. Weight resolution: 8b.

(a) ALGORITHMIC PARAMETER SET			(b) HARDWARE PARAMETER SET		
PARAMETER NAME	SYMBOL	VALUE	PARAMETER NAME	SYMBOL	VALUE
Number of Kernels	$N_k$	8	Timestamps Bitlength ( <i>mem</i> )	$L_{TS}$	10b
Receptive Field Width	$W_{RF}$	5pix	Membrane Potential Bitlength ( <i>mem</i> )	$L_k$	8b
Threshold Voltage	$V_{th}$	8	Timestamp's LSB Duration	$\delta_{TS}$	25 $\mu$ s
Stride	$d_{pix}$	2pix	Nominal Root Clock Frequency	$f_{root}$	12.5MHz
Refractory Period	$T_{refrac}$	5ms	Maximal Root Clock Frequency	$f_{root}$	400MHz
Leakage Type	$f_{leak}$	exponential	Add Operator Precision	-	9b
Leakage Time Constant	$\tau$	6.67ms	Multiply Operator Precision	-	8b

## IV.C.2 From Algorithm to Hardware

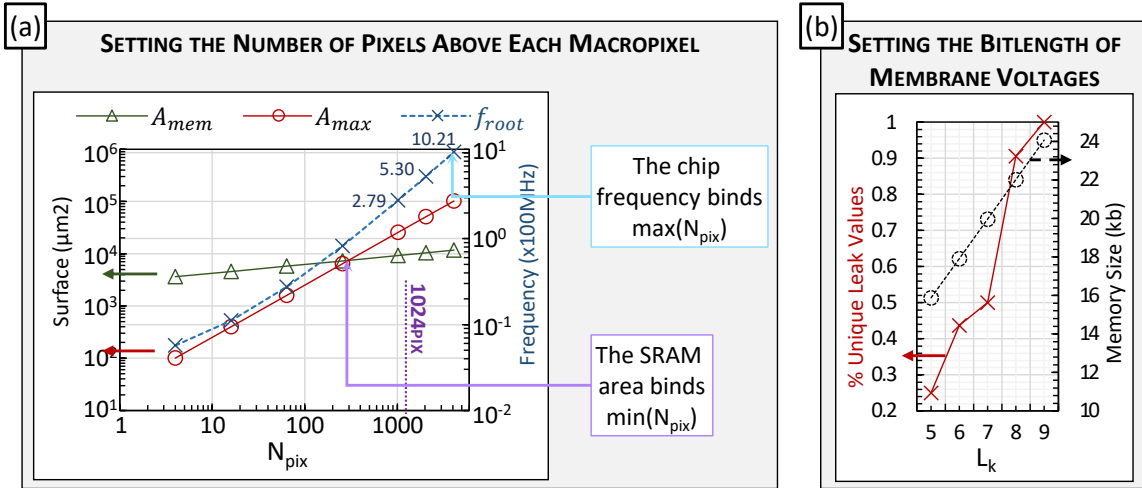
We aimed at implementing the smallest and most energy efficient SNN accelerator. As already discussed, the main problem with hardware implementation of SNNs is the memory [172–174]. A classical NN demands to memorize the weights and connection scheme of the network. This is also the case for event-driven networks. In addition, spiking networks also require storing their neuron states. So, we tried to find the set of algorithmic and hardware parameters that would permit to meet our objective, namely having our circuit routing below the pixel grid while consuming as little power as possible.

### IV.C.2.a Neuron States

A neuron state is composed of any information that is necessary to operate in between the algorithmic time steps. In our case it is fully represented with the membrane potential of each kernel, the timestamp of the last input spike and the timestamp of last output spike. The timestamps are shared between every kernels of a neuron. Because we deploy a sort of winner-take-all strategy, when a kernel spikes, all kernels of the neuron enter the refractory period. A full membrane potential is stored for each kernel, which is why we limited the number of kernels to eight.

### IV.C.2.b Memory Area and Number of Pixels

As the neuron states may *potentially* - due to the unpredictable nature of event-based data - be updated at each time step, they should be stored onto a memory energy efficient at writing. Hence, SRAM is preferred over any other non-volatile memory such as RRAMs. Standard SRAMs require space, and if the number of pixels  $N_{pix}$  considered per macropixel is not enough, all the required space may be taken by the SRAM, as depicted Figure IV.12 (b). The red line  $A_{max}$  shows the maximum allowed surface, which depends on the size of the pixels,  $25\mu\text{m}^2$  - and is directly proportional to the square of the number of pixels  $N_{pix}^2$ . An exploration of available standard cells has shown that below 256 pixels approximately, the amount of memory needed for storing all the neuron states would require an SRAM taking more space ( $A_{mem} = 7304\mu\text{m}^2$ ) than allowed by the 1024 pixels above ( $A_{max} = 6400\mu\text{m}^2$ ).



**Figure IV.12:** Design space exploration. **(a)** Illustration of the tradeoff between  $f_{root}$  and  $A_{mem}$  requirements for setting the number of pixels above the neural core in the macropixel construct  $N_{pix}$ . The memory areas  $A_{mem}$  values are obtained with SRAM cut generation exploration. **(b)** Impact of the neuron membrane bitlength  $L_k$  on the LUT precision and the required amount of memory  $M$ .

### IV.C.2.c Algorithmic Parameter Definition

#### IV.C.2.c.i Compression Ratio

Apart from the kernel patterns, the neuron threshold value  $V_{th}$ , and the refractory period duration  $T_{refrac}$ , every algorithmic parameter is fixed and hardwired in the design. We arbitrarily set the stride  $d_{pix}$  - the distance in pixel between two receptive field centers -, the

number of kernels  $N_k$  per neuron, the receptive field width  $W_{RF}$ , and the weight precision. The rest has been set after simulating the network on actual data [21], and by focusing on obtaining a compression ratio  $CR$  of approximately 10.  $CR$  is simply the ratio between the number of input and output spikes, expressed as  $CR = n_{ev_{in}} \div n_{ev_{out}}$ . It directly depends on the duration of the refractory period, the "strength" of the leakage, and the threshold voltage of the neurons. The parameter selected are listed Table IV.1 (a).

#### IV.C.2.c.ii Receptive Field Size

The size of the receptive fields  $W_{RF}$  was also chosen arbitrarily. To minimize the loss of information through the filtering operation, the receptive fields of neighboring neurons should overlap, hence the stride  $d_{pix}$  set at two with a receptive field width  $W_{RF}$  of 5. This results in a number of neuron four times smaller (2x2) than the number of pixels. Note that for keeping more precision at the output of the sensor, implementing an algorithm that could simultaneously manages receptive fields of varied sizes - multi-scale convolutions - may be necessary [44]. But, as already explained, this work aimed at exploring the possibility to implement such an algorithm directly behind an event-based sensor with pitch-constrained implementation. Hence, the algorithmic requirements were minimized.

#### IV.C.2.c.iii Kernel Patterns

The kernel patterns implemented are visible in the inlet of Figure IV.11. They represent oriented edges - which are usually obtained with STDP training [143] - and correspond to the first processing of visual information in mammals' brain, as demonstrated by Hubel and Wiesel [14, 86]. Each kernel corresponds to a single orientation, so each neuron applies winner-take-all in-between its kernel, to avoid generating false or redundant information. The first kernel to spike resets the membrane potential of all others and makes the neuron enter its refractory period. Finally, because near-binary weight distribution is sometimes spontaneously obtained by training [144], we reduced the possible weight values to -1 or +1 only, thus minimizing the required memory for storing the network information.

#### IV.C.2.d Standard Hardware Optimizations

We deploy a few optimization methods for improving the energy and area efficiency of our chip, namely clock-gating, static frequency scaling based on the required frequency of each module, and approximate computing. On top of that our neurocore behaves as a data-stream accelerator where each block works sequentially on a data, without storing intermediate results (apart from the module managing neuron states; the *computer*). Data transit one-way, from input to output, and no time nor energy is lost in a ping-pong between a central processing unit and an external memory, as would be in a Von Neumann machine. The frequency of each module is adapted to its local data rate; and if a module has no valid data in input, most of its components are clock gated.



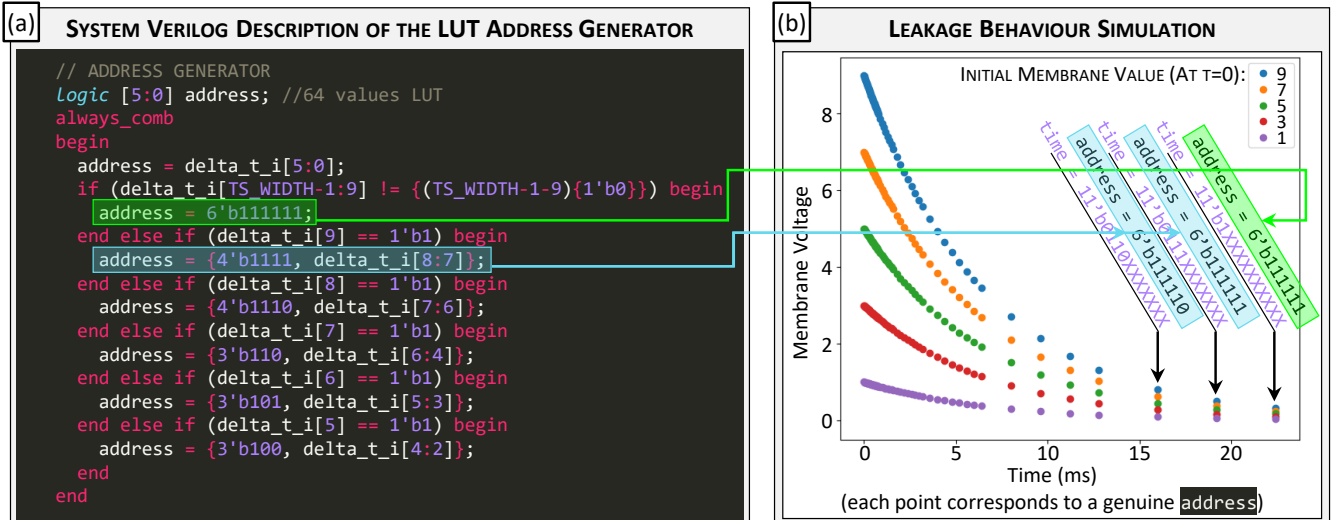
## IV.C.2.d.i Quantized Computing

The computations are realized with fixed-point representation of the timestamps and kernel potential values. The bitlengths of their stored values, respectively  $L_{TS}$  and  $L_k$ , are thus reduced so that the neuron state memory and combinatorial operators take less area and consume less energy per operation. The precision of these value is set following two points.

The leak is exponential, expressed as  $leak_{value} = \exp(-(t_{curr.} - t_{in})/\tau)$ , and if the difference between the current timestamp  $t_{curr.}$  and the timestamp at which the last input spike  $t_{in}$  was received is greater than 20ms,  $leak_{value}$  should be zero.  $L_{TS}$  is thus set to represent this full 20ms of leak range, and with a least significant bit (LSB) corresponding to 25 $\mu$ s, 10 bits are sufficient. An additional bit is used as a flag indicating overflow, resulting in  $L_{TS} = 11$ .

## IV.C.2.d.ii Exponential Leakage Design

The bitlength of the kernel potentials  $L_k$  is set to guarantee high precision exponential leakage. Each time a neuron state is loaded, leak is applied by multiplying every kernel



**Figure IV.13:** (a) System Verilog description of the address generator compressing the 11b difference between timestamps into a 6b input for the exponential leakage look up table. (b) Illustration of the Associated CSNN accelerator hardware parameters and values.

potential with the decrement factor  $leak_{value}$ , expressed as  $\exp(-(t_{curr.} - t_{in})/\tau)$ . Leak values are stored in a 64-input look up table (LUT). Membrane voltages being represented with fixed-point precision, the operation realized in hardware is approximated to  $V_{leaked}^k = V_{loaded}^k \times leak_{value} \div 2^{L_k}$ . The dividing factor  $2^{L_k}$  results from the fact that the decrement factors  $leak_{value}$  stored in the LUT should represent real numbers in the interval  $[0; 1]$  but are actually stored in the range scaled by  $2^{L_k}$ . Dividing a number by a power of two simply consists in taking the most significant bits (MSBs) of this number - a shift operation. Thus, the overhead induced is minimum.

The values  $leak_{value}$  stored in the LUT verify two constraints. They are designed so that the difference between two consecutive values in the LUT are approximately constant ( $\Delta leak_{value} \approx cst$ ), and so that the address converter operation does not rely on mathemati-

cal operators. We made this decision because the function  $f(x) = 2^x$  evolves exponentially - it is equivalent to  $g(x) = \exp(x * \ln(2))$ . The conversion between the timestamp and the addresses of the LUT are thus easier to realize.

The address converter is the module that converts the difference  $t_{curr.} - t_{in}$  to an address that is used for accessing one of the 64  $leak_{value}$  of the LUT. The interested reader can find its precise operation described in the System Verilog language Figure IV.13 (a). As it reduces the precision from 11b (timestamps) to 6b (64 values stored), satisfying the condition  $\Delta leak_{value} \approx cst$  becomes difficult. So, we chose the kernel voltage bitlength so that the least number of  $leak_{value}$  values inside the LUT are "naturally" identical. Figure IV.12 (b) depicts the variation of the "natural" precision of the LUT as a function of the bitlength of the kernel potentials. It shows that the number of "naturally" unique values inside the LUT drops by more than 50% when  $L_k$  decreases from 8b to 7b. So, we set  $L_k$  at 8b.

#### IV.C.2.e 3D-Enabled Optimization: The Smallest Repeatable Pattern

To take advantage of the distributed parallel interconnect of the macropixel stack, each core embeds its own readout module to directly access the pixels above it. The arbiter implemented thus has a fixed complexity of 1024 inputs, corresponding to the grid of 32x32 cells of the pixel tier. We conceived a fully customized method for addressing every neuron from the address of a pixel. It enabled us to design an efficient way of mapping to the network adapted for the macropixel meshing of the full network, we named it mapping by smallest repeatable pattern (SRP).

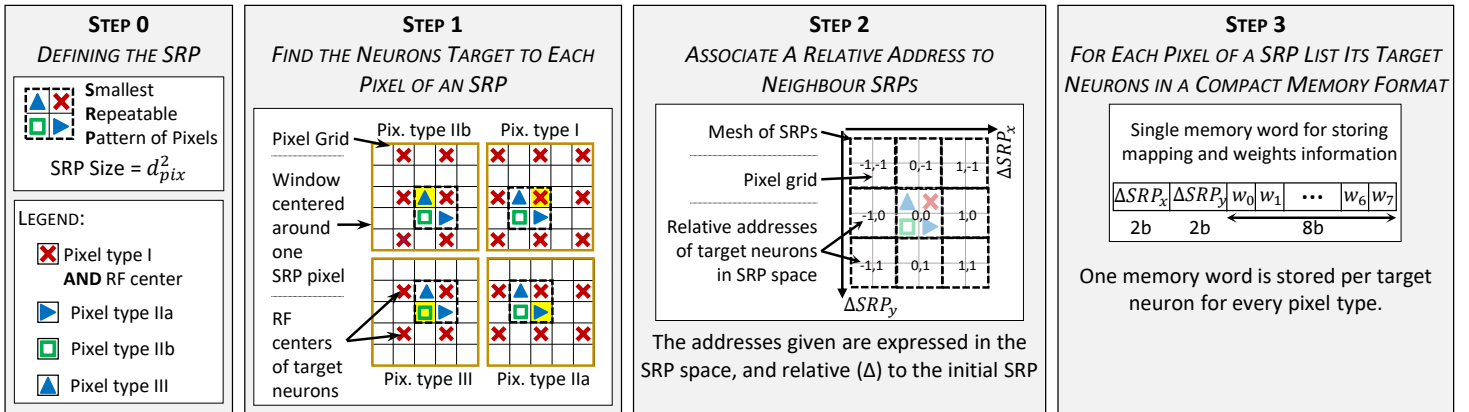


Figure IV.14: Smallest repeatable pattern definition and associated network mapping memory storage strategy.

An SRP is the smallest group of pixels and receptive field centers that permits to fully represent the full CSNN when meshed uniformly. We remind the reader that the goal is to be able to send any spike emitted by any pixel to the appropriate target neurons. With  $d_{pix} = 2$ , the SRP is a group of 2x2 pixels, as depicted Figure IV.14. Because of that, addresses in the SRP space contain 2b less than addresses in the pixel space. Thus, we store every addresses in the SRP space.



Obtaining all the information regarding an SRP is done with the following procedure. In step 1, the connections between every pixel of the SRP and the neighboring target neurons (red crosses) are found by taking a window of width  $W_{RF}$  around each pixel and looking at the receptive field centers inside this window. Then in step 2, for each pixel of an SRP, the associated target neurons are mapped by their relative addresses in  $\Delta SRP$ . Finally, in step 3, for each target neuron of each pixel composing an SRP a 12b word is stored in the mapping memory, composed of the  $\Delta SRP$ s coordinates (both stored on 2 bits) and the eight 1b weights  $w_i$  associated with each kernel potential. Pixels of type I, II (a & b), and III have 9, 6, and 4 target neurons, respectively. The full mapping memory thus requires only  $(9 + 6 + 6 + 4) \times 12 = 300b$ . Note that the SRP space coordinate system is actually identical to the neuron centers space, as there is a unique receptive field center in each smallest repeatable pattern.

Storing the mapping information using SRPs, all the connection information is only dictated by the stride  $d_{pix}$  of the convolutional layer. It is thus independent of the total number of neurons evaluated or of the core position with respect to the matrix of pixels. Hence, no overhead would be induced by tiling macropixel on a large-scale platform, as illustrated Figure IV.10. That is why convolutional networks are especially suited with the meshing by macropixel strategy.

### IV.C.3 Number of Pixels and Target Performances

Once the bitlength variables were set, we estimate the maximum number of pixels to evaluate per core. As we were not sure that the full system would fit into the space allowed, we decided to implement a single processing element per neurocore. Then, for setting the number of neurons  $N_{neuron}$  evaluated by a neurocore, we estimated the resulting required chip frequency  $f_{root}$  and core area  $A_{MP}$ .

#### IV.C.3.a Minimal Area Limit

As already discussed section IV.C.2, the maximal core area available  $A_{max}$  is directly dependent on the number of pixels  $N_{pix}$  of the macropixel stack and their pitch  $p_{pix}$ . And the core area is bounded from below by the memory cut area  $A_{mem}$ . Under 256 pixels the neuron state SRAM cannot fit, as visible Figure IV.12 (a).

#### IV.C.3.b Maximum Frequency Allowed

$f_{root}$  is directly proportional to the average input event rate per pixel  $f_{pix}$ , the number of target neurons per input spike  $N_{RF_{max}} - 9$  for pixel type I in our case -, the number of kernels evaluated per neuron  $N_k$ , and  $N_{neuron}$ . We defined  $f_{pix}$  (the average input event rate per pixel) as the maximum internal event-rate managed (2.5Gev/s) by Finatou et al. [18] divided by the total number of pixels  $N_{pix}$ . Now, the root frequency of our chip must permit to manage input rate. For any input spike, the maximum number of target neuron is  $N_{RF_{max}} = 9$  - corresponding to a pixel type I. For all these target neurons,  $N_k = 8$  kernels will be

updated sequentially by the same unique PE. The resulting required chip frequency can thus be expressed as the product of all these:

$$f_{root} = N_{pix} \times f_{pix} \times N_{RF_{max}} \times Nk \quad (IV.1)$$

With that, Figure IV.12 (a) depicts the evolution of  $A_{max}$  (red) and  $f_{root}$  (blue) as a function of  $N_{pix}$ . When  $N_{pix} \geq 2048$ ,  $f_{root}$  is at least 530MHz, which could lead to an important power consumption. And, as said, with  $N_{pix} \leq 256$  the SRAM takes more space than available. Hence, we set  $N_{pix}$  at 1024 which results in a macropixel composed of 32x32 pixels on top of a core evaluating 265 spiking convolutional neurons.

## IV.D Proposed Architecture

The overall architecture of our data-stream oriented neurocore is illustrated Figure IV.15. The event-based nature of the computation is kept thanks to a valid signal that propagates all the way from the pixels up to the computer module. The behavior is kept non-synchronous with combinatorial circuitry up to the input control module where the signal is sampled by a metastable tolerant synchronizer. After that it is fully synchronous with several clock domains and clock gating levels to reduce power consumption of each module individually.

### IV.D.1 Pixel Behavior Emulation

The pixels are not actually implemented, they are simply simulated at verification by emulating the behavior of a spiking pixel. That is, each pixel switches a valid signal to high at the  $f_{pix}$  frequency, which corresponds to a spike. It is then set to low when it receives the reset input signal.

### IV.D.2 Arbiter

The spike first passes through the arbiter, where its address is encoded. The arbiter implemented is adapted from the readout module of the Alice experiment of the CERN [142], with digital electronics and modified address encoding protocol to avoid using tristate buffers, which are replaced by multiplexers.

To emit an event, a pixel sets at high its valid signal which then propagates through the arbiter up to the input control module. The input control synchronously samples the valid signal and sends a reset pulse which passes inside the arbiter unit (AU) of layer N and generates a 2b address based on the low priority encoding of its input vector *state*, as illustrated Figure

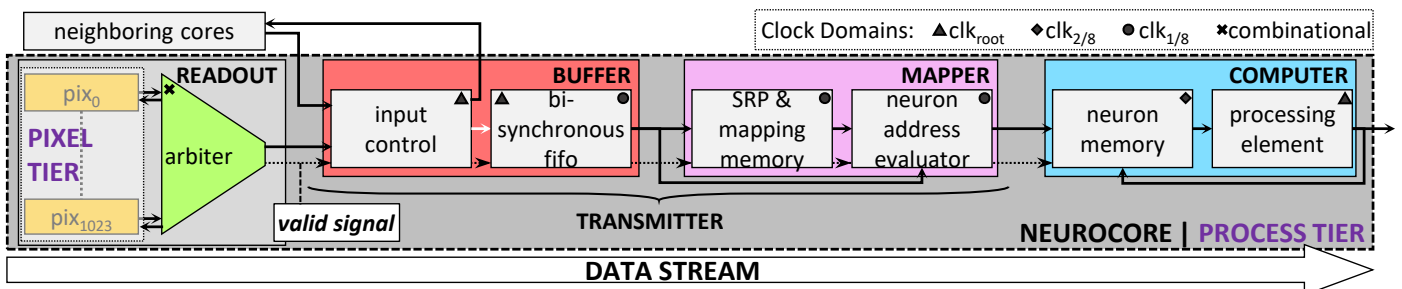
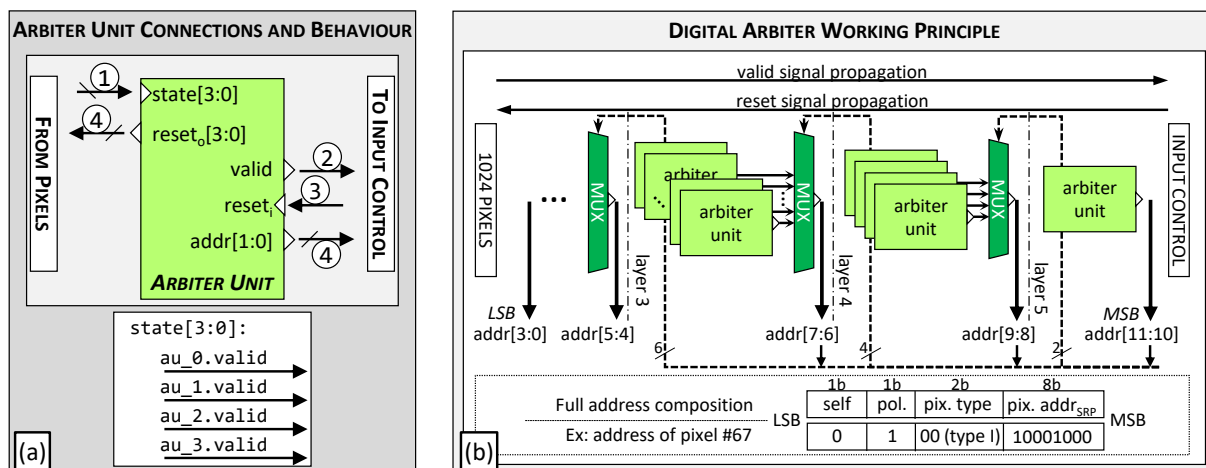


Figure IV.15: Overview of the data-stream architecture of the neural processing unit.



**Figure IV.16:** Schematic representation of the arbiter working principle. **(a)** Detailed illustration of the input and output vectors and associated dataflow of a single arbiter unit (AU). **(b)** Dataflow of the full arbiter, with multiplexing operations and associated resulting address vector.

IV.16 (a). Each AU reduces by four the number of signals; which results in a tree composed of  $\log_4(N_{pix})$  layer of AUs. The multiplexer of layer N - see Figure IV.16 (b) - uses this 2b address for selecting the next correct AU to which propagate the reset signal and from whom to read the next encoded 2b address. This new 2b address is concatenated with the previous one to generate a 4b-address that will be used by the multiplexer of layer N-1. This goes on sequentially. The full SRP address  $addr_{SRP}$  of the pixel is encoded by concatenation of all intermediate 2b addresses. Also, the AU closest to pixels directly encodes the pixel type inside its SRP, it takes in input the 4 pixels of the same SRP. It also conserves the bit of polarity  $pol.$  encoded by the pixel and set at 0 a bit called  $self$  to indicate that the event does not come from a neighbor macropixel. These four elements compose the full event address, as illustrated in the inlets of Figure IV.16 (b). The System Verilog code depicted Figure IV.17 illustrate the management of the address with multiplexing and valid AU selection.

**SYSTEM VERILOG DESCRIPTION OF THE ARBITER ADDRESS ENCODER**

```

genvar layer_i, unit_i;
generate
for (layer_i=0; layer_i<DEPTH; layer_i++) begin : layer
[...]
```

```

case (layer_i)
0 : always_comb begin [...] end
DEPTH - 1 : always_comb begin [...] end
default :
always_comb begin
//Get Valid Arbiter Unit ID
range = unsigned'(addr_o[2*DEPTH-1:2*(layer_i+1)]);
//Multiplexing Operation
addr_o[((layer_i+1)*2)-1:(layer_i*2)] = {addr_o_vec[((range+1)*2)-1], addr_o_vec[(range*2)]};
[...]
```

```

end
endcase
[...]
```

Semi-recursive definition that works only because of gates' delay.

```

endgenerate
endgenerate

```

**Figure IV.17:** Shorted System Verilog description of the arbiter. It focuses on the SRP address generator connection and multiplexing, and shows that the arbiter is made to be working only thanks to delay at the reset signal propagation time. To operate, the multiplexer of layer  $i$  needs the address of layer  $i - 1$  to be ready.

### IV.D.3 Transmitter

The transmitter is composed of a buffer and the pixel to neuron mapper module. The buffer manages the inputs from both the pixels on top of the neurocore and the ones sent by neighboring neurocores, thanks to the input control module. Then, the mapper loads the lists of all target neurons associated with the spike - stored under the SRP strategy as explained section IV.C.2.e. For every target neuron, it computes its absolute SRAM address vector  $addr_{RF}$  in the neuron state memory.

#### IV.D.3.a Buffer

The buffer includes the input control which manages spikes from the arbiter and neighboring MPXs (distinguished with the *self* bit) and stores them in a bisynchronous first in first out (FIFO) memory. The FIFO used is the one of Miro et al. [175]. The arbiter is sampled at the highest chip frequency  $f_{root}$  to avoid data congestion. To avoid continuously updating the FIFO registers at  $f_{root}$  when no data is available - thus keeping an event-based behavior -, the bisynchronous FIFO is clock gated on its input side. When a spike is available, a valid signal is transmitted from the arbiter to the FIFO which awakens it and makes it store the  $addr_{SRP}$  of the spike encoded by the arbiter. The valid signal controls the clock gating of the FIFO.

#### IV.D.3.b Mapper

Then, when the FIFO contains an event, the mapper fetches it and directly passes the valid signal and the full  $addr_{SRP}$  to downstream modules. The mapper is a finite state machine (FSM) hardwired for the 9 - 6 - 6 - 4 target neuron per pixel type configuration. The 2b pixel type contained in the  $addr_{SRP}$  vector dictates which target neurons mapping information (memory words containing  $\Delta SRP_s$  and associated weights, as depicted Figure IV.14) to load by simply setting the FSM in the right state. The weight values stored in the mapping memory are programmable.

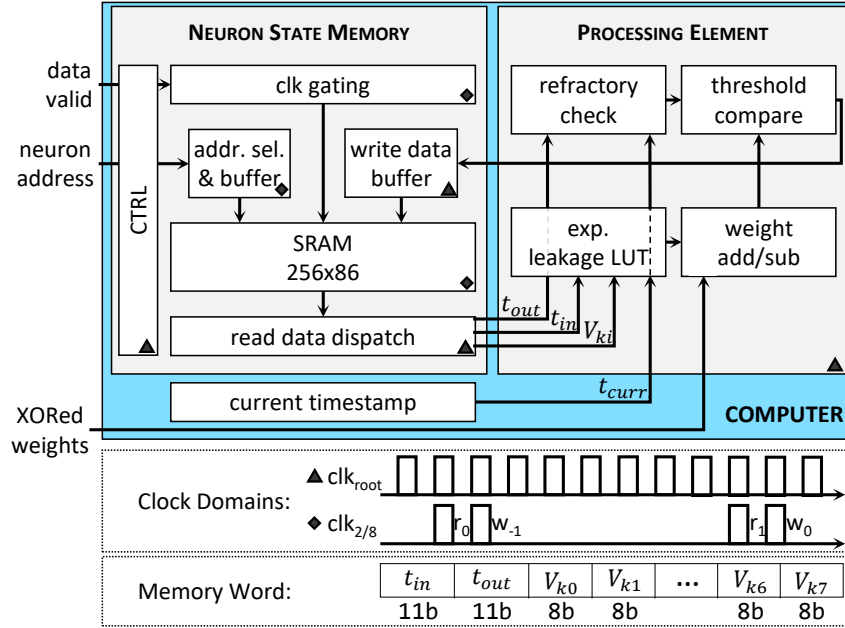
After that, the neuron address evaluator first decomposes the  $addr_{SRP}$  into SRP coordinates  $SRP_x$  and  $SRP_y$ . It then recomposes the target neuron address with:

$$addr_{RF} = [SRP_x + \Delta SRP_x; SRP_y + \Delta SRP_y] \quad (IV.2)$$

It also XORs the 8 weights with the event polarity and passes the resulting 8b vector to the computer module along with  $addr_{RF}$ . This operation is repeated at an eighth of the chip frequency ( $f_{1/8} = 1/8 \times f_{root}$ ) for every target neuron associated with the pixel type of the event. This because each neuron evaluates eight distinct kernels with a single PE,  $f_{1/8}$  is thus sufficient.

### IV.D.4 Computer

The computer is composed of the neuron state memory that stores the 256 neuron states and a single PE that updates the eight kernel potentials of each neuron sequentially. Figure IV.18 depicts the dataflow inside the computing unit of the core.



**Figure IV.18:** Detailed schematic representation of the computer behavior.

#### IV.D.4.a Neuron Memory

The SRAM address  $addr_{RF}$  received from the mapper is used to load the neuron's previous state at the read cycle  $r_0$  (see the digital timing diagrams of Figure IV.18). A full memory word contains the eight kernel potentials  $V_{k_i}$  and the timestamps  $t_{in}$  and  $t_{out}$ , corresponding to the times of the last input and output spikes, respectively. The  $V_{k_i}$  are sequentially sent to the PE for update, one  $V_{k_i}$  by clock cycle at  $clk_{root}$ . The SRAM is the smallest available for the 256 words of 86b configuration in the 28nm FDSOI technology, it is thus single port. When it does not write nor read, it is set in idle mode and clock gated. It thus works under the  $clk_{2/8}$  clock domain. To guarantee functional read/write synchronization with a single port SRAM, a write data buffer is placed at the input of the memory data port. It consists in seven registers in parallel, each sequentially storing an updated  $V_{k_i}$ . The last updated  $V_{k_7}$  is not stored in a register but directly written, at write cycle  $w_0$ , along with the seven others  $V_{k_i}$  and the timestamps  $t_{in}$  and  $t_{out}$ , at the same SRAM address  $addr_{RF}$  which has been memorized by the address selector and buffer. Thus, only 8 clock cycles are required for the full read, updates and write operations. On top of that, the bits corresponding to  $t_{out}$  are masked, excepted when the neuron fires, in which case all  $V_{k_i}$  are set to zero at write time - corresponding to the reset operation of the neuron potentials. The control module manages clock gating and read and write addresses selection throughout each read/write cycle. It is a hardwired FSM rolling between 8 main states that orchestrates every element of the computer module.

#### IV.D.4.b Processing Element

Finally, the processing element is a fully combinational module. It first applies leakage on  $V_{k_i}$  following the operation described section IV.C.2.d. Then, the XORed weight - received from the mapper module - associated with the  $i^{\text{th}}$  kernel is used as a selection bit that

define if the algorithmic logic unit of the PE should add or subtract 1 to or from the leaked  $V_{k_i}$ . Even if the weight values are stored as single bits, the weight added or subtracted from  $V_{k_i}$  is the real number 1 one encoded onto 8b. The kernel membrane  $V_{k_i}$  is now fully updated and compared with  $V_{th}$ . In parallel, the refractory checker evaluates the logic test  $t_{curr.} - t_{out} < T_{refrac.}$ . If the condition true the neuron is in refractory period, in which case the spiking operation is not allowed, even if  $V_{k_i} > V_{th}$  is true. If  $V_{k_i} > V_{th}$ , the neuron spikes, all kernel voltages at the address  $addr_{RF}$  are reset to zero. Otherwise, the memory simply stores the eight updated  $V_{k_i}$ , and the new time of input spike  $t_{in}$  at  $addr_{RF}$ . If an output spike is produced, the PE sends an event word composed of  $[addr_{SRP}, t_{curr.}, i]$  to a virtual output port.

## IV.E Results

### IV.E.1 Methodology

#### IV.E.1.a Design and Tools

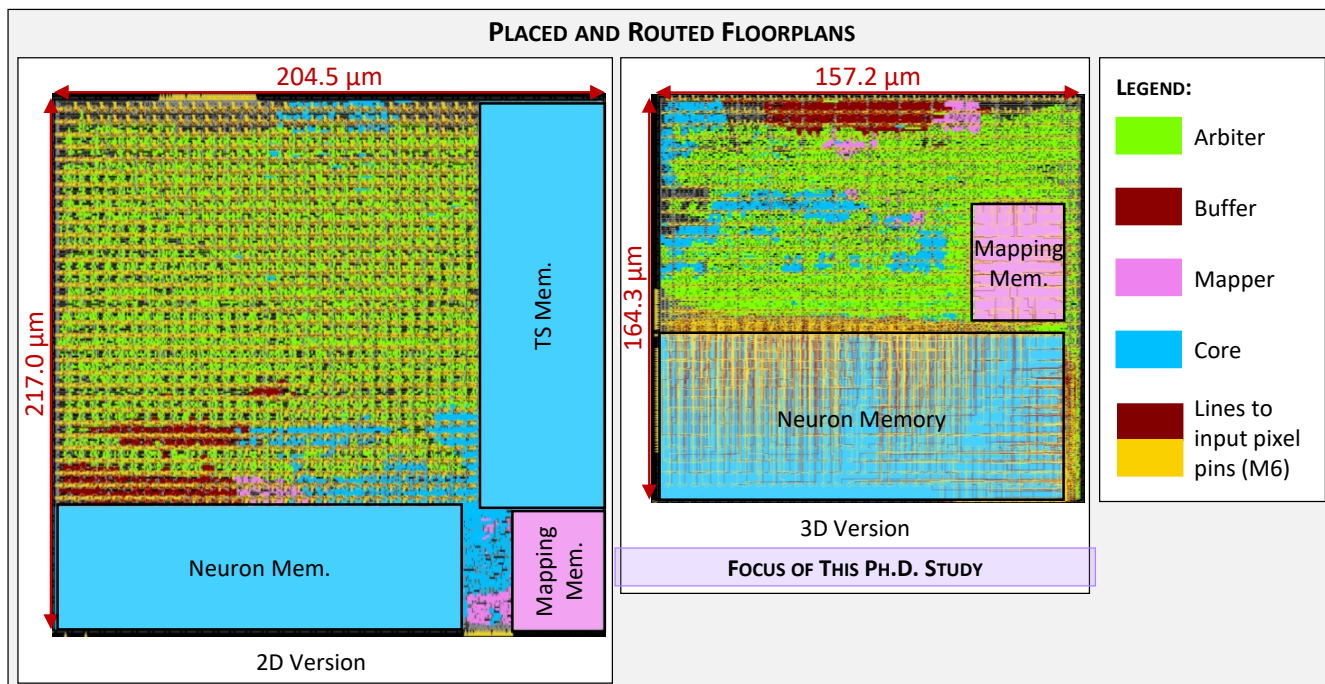
We evaluated the area and power consumption of the circuit based on post-layout simulations using uniform random spiking patterns as input to the neural core. All numbers presented below are associated to a single neural core. The power contributions of the pixels and 3D interconnects are not studied in this work. Only the neural processing block is designed, and pixel-related input pins are distributed on top the core at metal layer 6, with pitches of  $5\mu\text{m}$ . Nevertheless, the intellectual property (IP) presented here could be straightforwardly tiled and integrated within a full 3D stacked EB imager conception flow using available 3D process design kits.

The design was synthesized with Synopsys<sup>®</sup> Design Compiler using ST 28nm FDSOI technologies standard libraries. The place and route process was realized with Cadence<sup>®</sup> Innovus<sup>™</sup>. Finally, after signoff, timing, and power analysis with the Synopsys PrimeTime<sup>®</sup> suite. The core power evaluation is done through post-layout simulations under Mentor<sup>®</sup> Questa<sup>®</sup> Simulator. Snapshots of the layouts are shown Figure IV.19.

#### IV.E.1.b Validation

Once placed and routed, the behavior of our circuit was first validated by simulating sequential pixel spikes in fixed and known order. Each pixel would spike several times consecutively at a fraction of  $f_{pix}$ , and we verified that the corresponding neurons would fire after several spikes were received. This also permitted to confirm that everything up to the neuron update and the refractory period are functional. Once the fixed test pattern was checked; random pixel pulses were sent at the right target frequency  $f_{pix}$ . We then verified that random spikes would lead the corresponding neurons to spike after some time. We also verified that the neurons that were not stimulated often enough would not spike as their membrane potentials should be decreased because of leakage.





**Figure IV.19:** Layout of a single neural core for 32x32 pixels. **(a)** In a 2D configuration with SRAMs on the sides. **(b)** In a 3D configuration constrained by the pitches of the pixels above.

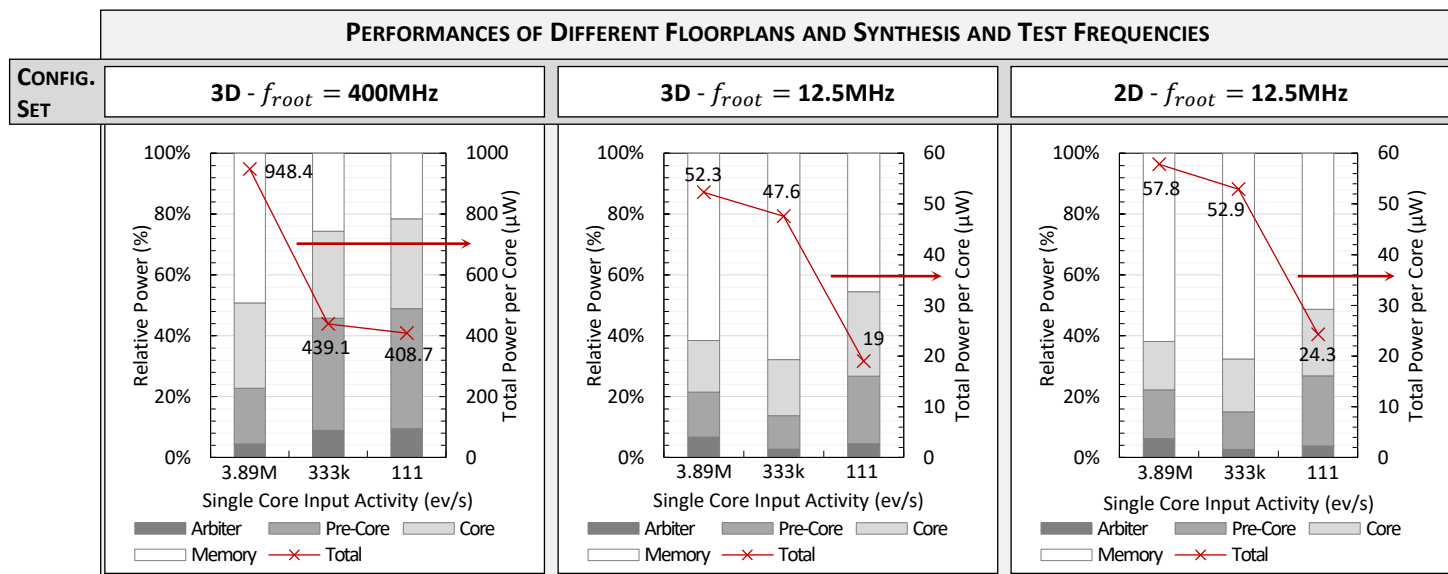
#### IV.E.1.c Equivalent 2D Model

In case the 3D constrained implementation would not route, we also evaluated the performance of a 2D model with SRAMs on the side. The performance of this version was slightly reduced as the neuron state SRAM was divided in two cuts of 256x64b and 256x22b. The former for storing membrane voltages, and the latter for storing the timestamps. Note that such a construct could still be distributed as tiles but would induce a grid effect visible in the acquired data as the pixels would be partially hidden by a sort of window contour. We thus focused on the 3D version - which did route successfully - for evaluations and power measurements.

#### IV.E.2 Input Rate, Chip Frequency, and Power Consumption

Two different targets  $f_{root}$  for synthesis and simulations were tested: 400MHz and 12.5MHz. These frequencies permit to easily represent timestamps with LSB corresponding to 25 $\mu$ s. They are adapted to manage the 720p equivalent input event rates of 3.5Gev/s and 300Mev/s, respectively. The former is close to the maximal internal event rate of 2.5Gev/s measured by Finateu et al. [18] but has been scaled to fit with the 25 $\mu$ s timestamp LSB, and the latter is the nominal event rate for comparing EB sensors [18, 32, 35]. As our core manages 900 times less pixels than a full 720p sensor, the input event rates of 3.5Gev/s, 300Mev/s, and 100kev/s are scaled down for simulation to 3.89Mev/s, 333kev/s, and 111ev/s, respectively. Figure IV.20 summarizes the power consumptions of a single neurocore obtained with post layout simulations at these different frequencies and event rates and floorplan configurations.





**Figure IV.20:** Post-layout power distribution for several input event rates, root frequencies  $f_{root}$ , and floorplans. The bar charts display the relative power consumptions (in percent %) at the configuration set and input event rate considered, normalized by the total power (red line) at this point.

#### IV.E.2.a Target 400MHz

The 400MHz clock frequency permits to handle the exceptionally large event rate of 3.5Gev/s. However, it leads to a power consumption of 948.4 $\mu\text{W}$ . Tiling this chip so that it reaches a 720p sensor would thus result in a total power consumption of at least 853.6mW, not adapted for an embedded device. Moreover, with a compression ratio of 10, the actual output event rate would be 350Mev/s, easily corresponding to a few Gbit/s when encoding spikes individually with a neuron address, a timestamp, and a kernel number. Thus, we demonstrate that managing a state-of-the-art event rate is achievable with the configuration of our chip, but it results in high ranges of power consumption and bandwidth, which is in opposition to the initial purpose of our filtering module.

#### IV.E.2.b Target 12.5MHz

On another hand, a clock frequency of 12.5MHz is more suited for embedding into an actual device. At the nominal input event rate of 300Mev/s for an equivalent 720p sensor (900x times more pixels), our core consumes only 47.6 $\mu\text{W}$ . The SRAMs and most of the registers being clock gated when no data is available, the power consumption drops by 2.5x to 19 $\mu\text{W}$  at the minimal input activity of 1kev/s - 720p equivalent. This power is mostly due to leakage of the neuron state SRAM and residual activity, especially at the level of the mapper, which was not clock gated with as much care as the computer module.

### IV.E.3 Discussion

#### IV.E.3.a Comparison with Other Works

A common metric for evaluating the efficiency of SNN accelerators is the energy per synaptic operation (SOP). An SOP is defined as a complete update operation onto one kernel potential of a neuron, as illustrated Figure IV.7 (b). For instance, a spike impinging from a pixel type I

**Table IV.2:** Comparison with state-of-the-art spiking neural network accelerators.

Reference	This Work		[176]	[177]	[150]	[172]	[151]	[174]		
Name	-		ODIN	SPOON	MorphIC	Park et al.	Loihi	Chen et al.		
Implementation	Digital		Digital	Digital	Digital	Digital	Digital	Digital		
IC Technology	28nm FDSOI		28nm FDSOI	28nm FDSOI	65nm LP	65nm	14nm FinFET	10nm FinFET		
Data Obtained From	Post-Layout		Chip	Post-Layout	Chip	Chip	Post-Layout	Chip		
NN Type	C-SNN		FC-SNN	EBCNN + FCNN	C-SNN	FC-BaNN <sup>c</sup>	Various	Various		
Core Area (mm <sup>2</sup> )	0.026		0.086	0.26 (excl. rails)	0.715 (excl. pads)	10.08	0.4	1.72		
Scaled Core Area (@28nm) <sup>f</sup>	<b>0.026</b>		0.086	0.260	0.257	3.63	-	-		
Neurone Behaviors (nb)	1 (exp. LIF)		<b>20 (LIF + Izhik.)</b>	1 (IF)	1 (LIF)	1	6	1		
Neurons per Core (nb)	256 <sup>l</sup>		256	922 <sup>j</sup>	512 <sup>j</sup>	<b>1194</b>	max. 1024	64		
Syn. Weights Storage	1bit SRAM		3+1bit SRAM	8bit SRAM	1bit SRAM	SRAM	1 to 9bit SRAM	7bit SRAM		
On-Chip Training	<b>No</b>		Yes	Yes	Yes	Yes	Yes	Yes		
Synapses per Core <sup>k</sup> (nb)	51.2k <sup>m</sup>		64k	2.2M	660k	238k	1M to 114k	16k		
Neu. Density (nb/mm <sup>2</sup> )	<b>9.8k</b>		3.0k	3.6k	716	0.1k	max. 2.6k	2.4k		
Syn. Density (nb/mm <sup>2</sup> )	1.97M <sup>m</sup>		741k	<b>7.8M</b>	738k	23.7k	2.5M to 285k	595k		
Supply Voltage (V)	0.8 <sup>m</sup>		0.55-1.0	0.6 (to 1.0)	0.8-1.2	0.8-1.2	0.5-1.25	0.45-0.9		
Chip Frequency (MHz)	400	12.5	75	150	210	55	20	-	506	105
SOP/s	194.4M <sup>d</sup>	<b>16.7M<sup>e</sup></b>	37.5M	-	105M	27.5M	-	min. 285.7M	393.8M	81.3M
Energy per SOP (pJ)	4.8	<b>2.86</b>	12.7 (0.55V)	6.8(CONV) <sup>g</sup> + 0.86(FC) (0.6V)	65	30	-	>23.6 (0.75V)	8.3 (0.9V)	3.8 (0.525V)
Single Core Power (μW)	948.4	<b>47.6</b>	476.3 <sup>a</sup>	2.7k <sup>h</sup>	6.8k <sup>i</sup>	825 <sup>i</sup>	23.6k (0.8V) <sup>b</sup>	6.7k	3.3k	308.75 <sup>a</sup>

<sup>a</sup>Includes IO and input generator.

<sup>b</sup>Obtained from SOP/s times energy per SOP values divided by the number of cores.

<sup>c</sup>BaNN stands for Binarized activations Neural Network.

<sup>d</sup>With 3.89Mev/s as input event rate to the core.

<sup>e</sup>With 333kev/S as input event rate to the core.

<sup>f</sup>Scaled by x0.6 from 60nm to 40nm and from 40nm to 28nm.

<sup>g</sup>Obtained from 15nj per event and 5x5x10 SOPs per event.

<sup>h</sup>Calculated as 313nJ/infer. divided by 117us/infer.

<sup>i</sup>Calculated as SOPmax/s times energy per SOP.

<sup>j</sup>One convolutional neuron includes several filter/kernel patterns.

<sup>k</sup>Each kernel *connection* is considered as a synapse (e.g., 5x5x8 Conv. => 200syn./neu.).

<sup>m</sup>Corrected with respect to DAC manuscript.[176, 177]

updates 9 neurons, each characterized by 8 kernels, thus resulting in  $9 \times 8 = 72$  SOPs. The four different pixel types characterizing an SRP lead to an average number of target neurons per input spike of  $25/4 = 6.25$ . The number of SOP per second at an input event rate of 333kev/s (equivalent to 300Mev/s for a 720p sensor) is thus  $6.25 \times 8 \times 333k = 16.65M$  SOP/s. With that in mind, the reader can find a comparison with other SNN accelerators Table IV.2.

On top of that, to characterize the power consumption of an eventual 3D stacked EB imager, we extract the *dynamic energy per event*, as proposed in [18, 32]. Aiming for a scalable metric independent of the matrix pixel size, we divide it by the number of pixels, resulting in an energy per event per pixel expressed in J/ev/pix. At 333kev/s and 12.5MHz, our core consumes  $93.0e^{-15}$  J/ev/pix. The reader can find an exhaustive comparison with EB imagers in Table IV.3.

### IV.E.3.b What is Good

The strongest point of our design resides in having the arbiter local to our core. Arbitrating 1024 pixels with 4-input to 1-output arbiter units requires only five arbitration layers. As illustrated by the floorplan snapshots, it still takes much room, but the resulting power consumption is minimal at 5.2% of the total power consumption in nominal conditions. Moreover, with  $f_{pix} = 3.16kHz$  the average inter-spike delay for 1024 pixels is 309ns, which

**Table IV.3:** Comparison with state-of-the-art event-based imagers.

Reference	This Work		[18]	[32]	[35]	
Name	-		Prophesee	Inivation	Samsung	
IC Technology	3D			2D		
Filter Type	<b>Convolutional Spiking Neurons</b>		Regions of Interest	Event Counting	None	
Technological Node	None <sup>a</sup> + 28nm FDSOI		90nm BI CIS + 40nm CMOS	65nm CMOS	90nm CIS BSI	
Implementation Stage	Post-Layout		Circuit Chip			
Resolution	N <sup>b</sup> x (32 x 32)		1280 x 720	132 x 104	640 x 480	
Pixel Size (μm <sup>2</sup> )	5,03 x 4,80		4.86 x 4.86	10 x 10	9 x 9	
Clk Frequency (MHz)	400	12.5	100	50	50	
Input event rate full res.	Low (kev/s)	100	100	100	100	
	High (Mev/s)	3500	300	300	300	
Power full res. (mW)	Low IN rate	367.83	17.1 <sup>c</sup>	32	0.25	27
	High IN rate	854.01	42.8 <sup>c</sup>	84	4.9	50
Power 1024 pix eq. <sup>d</sup> (μW)	Low IN rate	408.7	19	35.6	18.6	90.0
	High IN rate	948.9	<b>47.6</b>	93.3	365.5	166.7
Energy/event/pix <sup>e</sup> (aj/pix)		150.7	<b>93.0</b>	188.1	1882.8	249.6
Static Power <sup>f</sup> (nW/pix)		-	-	34.7	18.0	87.9
Power Without Pixels (mW)		854.0	42.8	52.0 <sup>h</sup>	3.9 <sup>h</sup>	23.0 <sup>h</sup>
P. Without Pixels (nW/pix)		926.7	<b>46.4</b>	56.4	71.0	74.9
Max. IN Event Rate (Mev/s)		<b>3500<sup>f</sup></b>	300	2920 <sup>f</sup>	180	300

<sup>a</sup>Pixel tier not implemented.

<sup>b</sup>Results shown for a block of 32x32 pixels, designed to be scalable.

<sup>c</sup>Values for an equivalent resolution of 1280x720 pixels; i.e., N = 900.

<sup>d</sup>Values for an equivalent resolution of 32x32 pixels.

<sup>e</sup>Dynamic Energy, defined and computed as in [32], divided by the total number of pixels.

<sup>f</sup>Peak internal activity.

<sup>g</sup>Static Power, defined and computed as in [32], divided by total number of pixels.

<sup>h</sup>Computed as in [32] : Total Power at High Frequency – (Static Power x Number of Pixels)

corresponds to a minimum sampling frequency of 324kHz. In comparison, a full 720p sensor would require 10 arbitration layers and a minimum sampling frequency of 2.5GHz [18]. This is a compelling argument in favor of 3D readout per macropixel. On top of that, mapping the network by smallest repeatable pattern requires only 300b of storage for mapping the full network. Note that we hardwired most of the algorithm parameters, drastically reducing the flexibility of the design. However, it enables us to make it fit in the 5μm pixel pitch constrained area representing only 0.026mm<sup>2</sup>.

### IV.E.3.c What Could be Improved

Because the floorplan is not 100% dense, which is visible Figure IV.19, several tracks can be followed to improve our design. We could add more programmability to our core, from example by enabling for convolutional filtering at multi-scale - i.e., with receptive field of both 3x3 and 5x5 pixel widths. This would be the hardest point to implement as modifying the number of neurons evaluated would directly impact the neuron state memory size, which is already taking more than 30% of the area available. The stride should be kept fixed at 2, as it reduces the number of operations per four with respect to the initial number of pixels.

An easy to realize upgrade would be to implement 4 processing elements instead of a single one. Doing this would permit to reduce the root frequency by 4, resulting in even more energy

efficiency passing from 12.5MHz to only 3.125MHz. However, the power consumption would not directly scale with the frequency because a residual power consumption still subsists, as the power measurements at 1kev/s shows.

Another point of improvement could be to avoid using an SRAM for the mapping memory, but instead deploy a non-volatile memory such as RRAMs. In contrast to the neuron state memory, the values stored in the mapping memory do not change during operation. Still, replacing the SRAM with a RRAM requires much care in the design, as the mapping memory is still read at each input spike, maximizing the number of bits accessed at each read could be required, notable by loading all the target neuron mapping information per pixel type at once - instead of sequentially as is currently the case.

Finally, a critical point would be to reduce even further the kernel potentials, passing from 8b to 6b for example. But modifying this sort of parameters must be done in parallel with an exploration of the impact on the accuracy of the algorithm behind.

## **IV.F Conclusion**

To solve limitations of EB imagers, we deploy a CSNN to filter noise and reduce the output event rate by 10x while conserving spatial and temporal information. Considering embedded evaluation of the algorithm, we implemented a data stream EB neural core mostly hardwired, to minimize the power consumption of the designed chip. On top that, by envisaging 3D stacking integration for near-sensor computing directly behind an EB imager, the core area is constrained to 5 $\mu$ m-pitch pixels and fits in only 0.026mm<sup>2</sup>. Our SNN accelerator consumes only 2.86pJ/SOP at a 720p equivalent nominal input event rate of 300Mev/s, corresponding to 93.0aJ/ev/pix.

We demonstrated that an SNN accelerator fits directly behind an EB pixel grid. This performance can be achieved thanks to direct access to pixels with 3D IC technologies. The core instantiates a direct distributed 1024-input pixel encoder, which enables to store the full monolayer CSNN parameters (weights and map) in only 300 bits and to drastically reduce the chip frequency. Moreover, it can be tiled without overhead to manage high resolution EB image sensors.

This work could be improved by adding more functionality to the design, namely with multi-scale convolutional filters with receptive fields dimensions different than the current 5x5 only. In addition, implementing several processing elements in parallel to reduce the required chip frequency by a factor, leading to an even better energy efficiency without reducing the throughput. By focusing on reducing everything at maximum, from both the hardware and algorithmic points of view, we demonstrated a never achieved 2.86pJ/SOP in digital electronics designed at the 28nm FDSOI technological node. Note that the bulk biasing available with FDSOI technology - enabling to reduce the power consumption of the chip - has not been used.

# V

## Conclusion

---

This Ph.D. study has been realized by targeting applications for cyber physical systems and embedded devices that analyze their surrounding environment. More specifically, these systems could be robots, mixed reality helmets or mobile phones, autonomous vehicles, and so on. This thesis aimed at designing a more robust and energy efficient visual perception system, combining an event-based (EB) image sensor with three-dimensional integration technologies. Hence, we explored a large diversity of perception algorithms, with for example depth extraction and visual inertial odometry (VIO), before designing a spiking neural network accelerator for near-sensor filtering.

We focused on EB sensors because they come with remarkably interesting properties for robust perception and should enable relatively low power budget operations. They intrinsically provide simultaneous high acquisition speed (HAS), high dynamic range (HDR), and event-based (EB) acquisition. They can thus work both in broad daylight, as well as in the dark, and measure movements in the milli- to micrometer scale. However, they come with major drawbacks that render their use costly in terms of power and computational requirements. They are noisy, have poor resolutions, and generate a large amount of data containing relatively few information, called events or spikes.

For long, they relied on asynchronous circuit output which made solving these limitations an impossibility. However, recent sensors are designed with synchronous readout techniques that alleviate this issue. But at the same time, the temporal precision of the acquired data is reduced, and the HAS property is lost. Moreover, with the rise of 3D integration technologies, novel sensors are being developed and take advantage of the 3D interconnection to access pixels individually in parallel. On top of that, these sensors integrate advanced processing capabilities inside.

The idea leading to the common thread followed in chapter III originates from the observation that visual processing inside our brain occurs on two dissociated neural circuits, one specialized in object categorization and the other in object and movement detection. Moreover, as our bibliographic research led us to observe that object classification with event-based data and spiking neural networks (SNN)s was poorly effective, we concluded that such data should preferably be used for detection tasks. From there, we developed object detection by clustering, depth extraction, and VIO algorithmic pipelines. We have shown that the two former tasks are hardly efficient, and that their implementation is slowed down by the issues inherited from the event-based imager properties. We thus realized a virtually generated

event-based dataset from scenes under Blender™ to dissociate the flaws of the algorithms from the quality of the data. We have then shown that even if depth extraction is functional, it does not permit to achieve results as accurate as standard methods, e. g., Block Matching. Nevertheless, we demonstrated that the VIO can be accurately done, even with non-ideal data, by generating frames based on the accumulation of events.

In addition, to avoid using classic frames into the loop, we proposed a solution for initializing the VIO module on IMU and EB data only. The EFR module is first given poses obtained by IMU integration and can thus provide motion compensated frames to the VIO backend module. The latter can initialize itself, and then provide poses to the EFR module in turn. This idea has been patented.

This algorithmic exploration also led to the conceptualization of a system, from sensor to application, which takes advantages of a hypothetical filter module operating directly at the sensor level, which would be integrated with 3D IC technologies. More precisely, we designed a convolutional spiking neural network (CSNN) as filtering module able to be deployed directly behind an EB sensor for reducing the amount of data emitted and modify the type of information output by the imager. This system, along with another one that relies on event-based data non-“naturally” acquired (emulated from high frequency frame-based data instead), have been deposited with two distinct patents. In the latter, we proposed a solution to detect objects based on frames generated by compensating the ego-motion of the sensor. It is based on the fact that objects moving with respect to the environment appear blurry on ego-motion compensated event frames. It should thus be possible to detect them based on simple contrast measurements.

To demonstrate the feasibility of the proposed systems, we designed a convolutional spiking neural network hardware accelerator. This circuit has been designed at the 28nm FDSOI technological node and simulated up to post-layout power evaluations. It is designed as a macropixel-oriented processor that can directly be integrated behind an EB pixel grid with the use of 3D IC technologies. It evaluates a single layer of a CSNN with filters based on spike timing dependent plasticity-learned features to extract simple visual primitives such as oriented edges. The resulting circuit measures  $0.026\text{mm}^2$ , consumes only 2.86pJ per synaptic operation, and could manage 300Mev/s in as input event-rate once scaled to a full 720p EB imager. In addition, our macropixel reduces by a factor 10x the number of events output, while conserving spatial and temporal information contained in the data. We have thus shown that evaluating an SNN in the space constrained by the pixel tile above the would-be processing die is possible and efficient.

Yet, there remains open questions that we did not answer. Would evaluating an agent localization with the type of data generated by our smart event-based sensor still be accurate? Also, it would be interesting compare the precision of such a system with a non-naturally event-based solution that realizes the same operation. Finally, the system chip designed is

not a full circuit, it is a block designed for being distributed onto a two-dimensional meshing. Thus, going to the industrialization with this block would require adding a readout module that would extract the data from the resulting multi-macropixel circuit.



# List of Contributions

---

## Scientific Communications

- M. Bouvier et al., “**Spiking Neural Networks Hardware Implementations and Challenges: A Survey**,” *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, p. 22:1-22:35, Apr. 2019, doi: 10.1145/3304103.
- P. Vivet et al., “**Advanced 3D Technologies and Architectures for 3D Smart Image Sensors**,” in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2019, pp. 674–679, doi: 10.23919/DATE.2019.8714886.
- M. Bouvier, A. Valentian, and G. Sicard, “**Scalable Pitch-Constrained Neural Processing Unit for 3D Integration with Event-Based Imagers**,” in *Proceedings of the 58th Annual Design Automation Conference*, San Francisco, CA, USA, 2021, p. X:1-X:6. (Conference to be held in December 2021)

## Patents Deposited

- M. Bouvier, A. Valentian, “**Dispositif de Compensation du Mouvement d’un Capteur Événementiel et Système d’Observation et Procédé Associés**”. - *patent 1*
- A. Bigé, M. Bouvier, “**Dispositif de Compensation du Mouvement d’un Capteur Événementiel, Systèmes et Procédés Associés**”. - *patent 2*
- M. Bouvier, A. Valentian, “**A Smart Imager Solution for Asynchronous Reinforced Computer Vision**”. - *patent 3*

## Notable Facts

- M. Bouvier was awarded the International Solid State Circuit Conference Student Travel Grant Awards (**ISSCC STGA**) in 2019.

# Bibliography

---

- [1] A. E. Gamal and H. Eltoukhy. "CMOS Image Sensors". *IEEE Circuits and Devices Magazine*. 2005, ISSN: 1558-1888. DOI: 10.1109/MCD.2005.1438751.
- [2] "Bayer Filter". *Wikipedia*. URL: [https://en.wikipedia.org/w/index.php?title=Bayer\\_filter&oldid=1002684290](https://en.wikipedia.org/w/index.php?title=Bayer_filter&oldid=1002684290) (visited on 02/25/2021).
- [3] A. E. Gamal and B. Wandell. "Stanford University Class Programmable Digital Camera Project - Lecture Notes 8 - SNR and Dynamic Range". Lecture Notes. Online, 2005.
- [4] H. Takahashi, M. Kinoshita, K. Morita, T. Shirai, T. Sato, T. Kimura, et al. "A 3.9  $\mu\text{m}$  Pixel Pitch VGA Format 10 b Digital Image Sensor with 1.5-Transistor/Pixel". *2004 IEEE International Solid-State Circuits Conference (IEEE Cat. No.04CH37519)*. 2004, DOI: 10.1109/ISSCC.2004.1332617.
- [5] R. M. Guidash. "Active Pixel Image Sensor with Shared Amplifier Read-Out". Pat. 2000.
- [6] C. Dupoirion. "Nouveaux paradigmes de capture d'images et traitements associés pour futurs SoC en nœuds CMOS nanométriques". PhD thesis. Université Grenoble Alpes, 2017.
- [7] F. Saffih. "Foveated Sampling Architectures for CMOS Image Sensors". 2005. URL: <https://uwspace.uwaterloo.ca/handle/10012/820> (visited on 03/01/2021).
- [8] J. Aoki, Y. Takemoto, K. Kobayashi, N. Sakaguchi, M. Tsukimura, N. Takazawa, et al. "A Rolling-Shutter Distortion-Free 3D Stacked Image Sensor with -160dB Parasitic Light Sensitivity in-Pixel Storage Node". *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. 2013, DOI: 10.1109/ISSCC.2013.6487824.
- [9] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. "Real-Time Camera Tracking: When Is High Frame-Rate Best?" *Computer Vision – ECCV 2012*. Ed. by A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, ISBN: 978-3-642-33786-4. DOI: 10.1007/978-3-642-33786-4\_17.
- [10] T. Schops, J. L. Schonberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, et al. "A Multi-View Stereo Benchmark With High-Resolution Images and Multi-Camera Videos".
- [11] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, ISBN: 978-1-4503-4892-8. DOI: 10.1145/3079856.3080246.
- [12] Q. Sun, E. Tseng, Q. Fu, W. Heidrich, and F. Heide. "Learning Rank-1 Diffractive Optics for Single-Shot High Dynamic Range Imaging". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020,
- [13] M. Sakakibara, K. Ogawa, S. Sakai, Y. Tochigi, K. Honda, H. Kikuchi, et al. "A Back-Illuminated Global-Shutter CMOS Image Sensor with Pixel-Parallel 14b Subthreshold ADC". *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, DOI: 10.1109/ISSCC.2018.8310193.
- [14] D. Purves, G. J. Augustine, D. Fitzpatrick, W. C. Hall, A.-S. LaMantia, R. D. Mooney, et al. "Neuroscience". Oxford University Press, 2018. ISBN: 978-1-60535-841-3.
- [15] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas. "Event-Based Neuromorphic Systems". Chichester, UK, John Wiley & Sons, Ltd, 2014. ISBN: 978-1-118-92760-1. DOI: 10.1002/9781118927601.
- [16] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128x128 120 dB 15  $\mu\text{s}$  Latency Asynchronous Temporal Contrast Vision Sensor". *IEEE Journal of Solid-State Circuits*. 2008, ISSN: 1558-173X. DOI: 10.1109/JSSC.2007.914337.
- [17] C. Posch, D. Matolin, and R. Wohlgenannt. "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS". *IEEE Journal of Solid-State Circuits*. 2011, ISSN: 1558-173X. DOI: 10.1109/JSSC.2010.2085952.

- [18] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, et al. "5.10 A 1280×720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86μm Pixels, 1.066GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline". *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2020, DOI: 10.1109/ISSCC19947.2020.9063149.
- [19] A. J. Martin and M. Nystrom. "Asynchronous Techniques for System-on-Chip Design". *Proceedings of the IEEE*. 2006, ISSN: 1558-2256. DOI: 10.1109/JPROC.2006.875789.
- [20] J. L. Hennessy, D. A. Patterson, and Gale (Firm). "Computer Architecture: A Quantitative Approach". 2019. ISBN: 978-0-12-811906-8.
- [21] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. "The Event-Camera Dataset and Simulator: Event-Based Data for Pose Estimation, Visual Odometry, and SLAM". 2016. DOI: 10.1177/0278364917691115.
- [22] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. "The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception". *IEEE Robotics and Automation Letters*. 2018, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2800793.
- [23] S. Chen. "Introduction of Celex Family Sensor and Event/Frame/Optical-Flow Hybrid Processing". Los Angeles, USA, 2019.
- [24] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck. "A 240×180 130 dB 3μs Latency Global Shutter Spatiotemporal Vision Sensor". *IEEE Journal of Solid-State Circuits*. 2014, ISSN: 1558-173X. DOI: 10.1109/JSSC.2014.2342715.
- [25] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, et al. "Spiking Neural Networks Hardware Implementations and Challenges: A Survey". *J. Emerg. Technol. Comput. Syst.*. 2019, ISSN: 1550-4832. DOI: 10.1145/3304103.
- [26] C. Mead. "Neuromorphic Electronic Systems". *Proceedings of the IEEE*. 1990, ISSN: 1558-2256. DOI: 10.1109/5.58356.
- [27] "iniVation | Neuromorphic Vision Systems". <https://inivation.com/>.
- [28] P. Furgale, J. Rehder, and R. Siegwart. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems". *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, DOI: 10.1109/IRoS.2013.6696514.
- [29] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart. "Extending Kalibr: Calibrating the Extrinsic of Multiple IMUs and of Individual Axes". *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA). 2016, DOI: 10.1109/ICRA.2016.7487628.
- [30] G. Taverni, D. P. Moeys, C. Li, C. Cavaco, V. Motsnyi, D. S. S. Bello, et al. "Front and Back Illuminated Dynamic and Active Pixel Vision Sensors Comparison". *IEEE Transactions on Circuits and Systems II: Express Briefs*. 2018, ISSN: 1558-3791. DOI: 10.1109/TCSII.2018.2824899.
- [31] C. Scheerlinck, H. Rebecq, T. Stoffregen, N. Barnes, R. Mahony, and D. Scaramuzza. "CED: Color Event Camera Dataset". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019,
- [32] C. Li, L. Longinotti, F. Corradi, and T. Delbruck. "A 132 by 104 10μm-Pixel 250μW 1kefps Dynamic Vision Sensor with Pixel-Parallel Noise and Spatial Redundancy Suppression". *2019 Symposium on VLSI Circuits*. 2019, DOI: 10.23919/VLSIC.2019.8778050.
- [33] J. Manderscheid, A. Sironi, N. Bourdis, D. Migliore, and V. Lepetit. "Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras". *arXiv:1903.11332 [cs]*. 2019. arXiv: 1903.11332 [cs].
- [34] "PROPHESSEE | Metavision for Machines". <http://www.prophesee.ai/>.
- [35] B. Son, Y. Suh, S. Kim, H. Jung, J. Kim, C. Shin, et al. "4.1 A 640×480 Dynamic Vision Sensor with a 9μm Pixel and 300Meps Address-Event Representation". *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, DOI: 10.1109/ISSCC.2017.7870263.
- [36] H. E. Ryu. "Industrial DVS Design; Key Features and Applications".
- [37] "CelePixel - Bio-Inspired Computer Vision". <https://www.celepixel.com/#/Home>.

- [38] iniVation. "Understanding the Performance of Neuromorphic Event-Based Vision Sensors". Tech. rep. iniVation, 2020.
- [39] C. Aerne. "iniVation List of Products".
- [40] L. Shi, Z. Yang, T. Wang, R. Zhao, W. He, and J. Pei. "Bimodal Signal Fusion System and Method". Pat. (CN). Beijing Lingxi Technology Co., Ltd. 2021. URL: <https://patents.google.com/patent/CN112188093A/en?q=CN112188093A> (visited on 03/05/2021).
- [41] R. Berner and C. Brändli. "Dynamic Vision Sensor with In-Pixel Digital Change Detection". Pat. Sony Advanced Visual Sensing Ag. 2020.
- [42] E. P. Gousev, A. Govil, and S. Y. Kim. "Event Based Computer Vision Computation". U.S. pat. Qualcomm Inc. 2020.
- [43] A. Verdant, G. Sicard, and C. Dupoiron. "An Event-Based, Frame-Based Image Acquisition Mechanism for CMOS Image Sensors". *2017 New Generation of CAS (NGCAS)*. 2017, DOI: 10.1109/NGCAS.2017.32.
- [44] M. Lefebvre, L. Moreau, R. Dekimpe, and D. Bol. "7.7 A 0.2-to-3.6TOPS/W Programmable Convolutional Imager SoC with In-Sensor Current-Domain Ternary-Weighted MAC Operations for Feature Extraction and Region-of-Interest Detection". *2021 IEEE International Solid-State Circuits Conference (ISSCC)*. 2021, DOI: 10.1109/ISSCC42613.2021.9365839.
- [45] J. J.-Q. Lu, D. Zhang, and P. Ramm. "Overview of Bonding and Assembly for 3D Integration". *Handbook of 3D Integration*. John Wiley & Sons, Ltd, 2014. Chap. 19, ISBN: 978-3-527-67010-9. DOI: 10.1002/9783527670109.ch19.
- [46] Z. Or-Bach. "Monolithic 3D Integration—An Update". *NANO-CHIPS 2030: On-Chip AI for an Efficient Data-Driven World*. Ed. by B. Murmann and B. Hoefflinger. The Frontiers Collection. Cham: Springer International Publishing, 2020, ISBN: 978-3-030-18338-7. DOI: 10.1007/978-3-030-18338-7\_8.
- [47] E. J. Marinissen and Y. Zorian. "Testing 3D Chips Containing Through-Silicon Vias". *2009 International Test Conference*. 2009, DOI: 10.1109/TEST.2009.5355573.
- [48] X. Zhang, J. K. Lin, S. Wickramanayaka, S. Zhang, R. Weerasekera, R. Dutta, et al. "Heterogeneous 2.5D Integration on through Silicon Interposer". *Applied Physics Reviews*. 2015, DOI: 10.1063/1.4921463. URL: <https://aip.scitation.org/doi/abs/10.1063/1.4921463> (visited on 03/03/2021).
- [49] "Apple M1 Chip". Apple. URL: <https://www.apple.com/mac/m1/> (visited on 03/03/2021).
- [50] L. Millet, S. Chevobbe, C. Andriamisaina, L. Benaissa, E. Deschaseaux, E. Beigne, et al. "A 5500-Frames/s 85-GOPS/W 3-D Stacked BSI Vision Chip Based on Parallel In-Focal-Plane Acquisition and Processing". *IEEE Journal of Solid-State Circuits*. 2019, ISSN: 1558-173X. DOI: 10.1109/JSSC.2018.2886325.
- [51] T. Haruta, T. Nakajima, J. Hashizume, T. Umebayashi, H. Takahashi, K. Taniguchi, et al. "4.6 A 1/2.3inch 20Mpixel 3-Layer Stacked CMOS Image Sensor with DRAM". *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, DOI: 10.1109/ISSCC.2017.7870268.
- [52] P. Chou, C. Chang, M. M. Mhala, C. C. Liu, C. Y. Chao, C. Huang, et al. "A 1.1 $\mu$ m-Pitch 13.5Mpixel 3D-Stacked CMOS Image Sensor Featuring 230fps Full-High-Definition and 514fps High-Definition Videos by Reading 2 or 3 Rows Simultaneously Using a Column-Switching Matrix". *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, DOI: 10.1109/ISSCC.2018.8310197.
- [53] M. Suzuki, Y. Sugama, R. Kuroda, and S. Sugawa. "Over 100 Million Frames per Second 368 Frames Global Shutter Burst CMOS Image Sensor with Pixel-Wise Trench Capacitor Memory Array". *Sensors*. 2020, DOI: 10.3390/s20041086.
- [54] O. Kumagai, A. Niwa, K. Hanzawa, H. Kato, S. Futami, T. Ohyama, et al. "A 1/4-Inch 3.9Mpixel Low-Power Event-Driven Back-Illuminated Stacked CMOS Image Sensor". *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018, DOI: 10.1109/ISSCC.2018.8310196.
- [55] A. Nose, T. Yamazaki, H. Katayama, S. Uehara, M. Kobayashi, S. Shida, et al. "Design and Performance of a 1 Ms High-Speed Vision Chip with 3D-Stacked 140 GOPS Column-Parallel PEs †". *Sensors*. 2018, DOI: 10.3390/s18051313.

- [56] Y.-H. Kim, J.-C. Kim, S.-y. Cha, and J.-c. Kim. "Image Sensor Package Having Multi-Level Stack Structure". Pat. 2020.
- [57] T. Takahashi, Y. Kaji, Y. Tsukuda, S. Futami, K. Hanzawa, T. Yamauchi, et al. "A Stacked CMOS Image Sensor With Array-Parallel ADC Architecture". *IEEE Journal of Solid-State Circuits*. 2018, ISSN: 1558-173X. DOI: 10.1109/JSSC.2017.2784759.
- [58] W. Maass. "Networks of Spiking Neurons: The Third Generation of Neural Network Models". *Neural Networks*. 1997,
- [59] A. S. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. V. Arthur, P. A. Merolla, et al. "Real-Time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with ~100×Speedup in Time-to-Solution and ~100,000×Reduction in Energy-to-Solution". *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '14. New Orleans, Louisiana: IEEE Press, 2014, ISBN: 978-1-4799-5500-8. DOI: 10.1109/SC.2014.8.
- [60] A. Binet and T. Simon. "Méthodes nouvelles pour le diagnostic du niveau intellectuel des anormaux". *L'Année psychologique*. 1904, DOI: 10.3406/psy.1904.3675.
- [61] D. Wechsler. "The Psychometric Tradition: Developing the Wechsler Adult Intelligence Scale". *Contemporary Educational Psychology*. 1981, ISSN: 0361-476X. DOI: 10.1016/0361-476X(81)90035-7.
- [62] U. Neisser, G. Boodoo, T. Bouchard, A. Boykin, N. Brody, S. Ceci, et al. "Intelligence: Knowns and Unknowns". *American Psychologist*. 1996, ISSN: 0003-066X.
- [63] B. J. MacLennan. "Connectionist Approaches". *International Encyclopedia of the Social & Behavioral Sciences*. Ed. by N. J. Smelser and P. B. Baltes. Oxford: Pergamon, 2001, ISBN: 978-0-08-043076-8. DOI: 10.1016/B0-08-043076-7/00537-4.
- [64] J. Garson. "Connectionism". 1997.
- [65] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, et al. "Building Watson: An Overview of the DeepQA Project". *AI Magazine*. 2010, ISSN: 2371-9621. DOI: 10.1609/aimag.v31i3.2303.
- [66] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". *Communications of the ACM*. 2017, ISSN: 0001-0782. DOI: 10.1145/3065386.
- [67] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, et al. "Mastering the Game of Go without Human Knowledge". *Nature*. 2017, ISSN: 1476-4687. DOI: 10.1038/nature24270.
- [68] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, K. Tunyasuvunakool, et al. "High accuracy protein structure prediction using deep learning". *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*. 2020,
- [69] "AlphaFold: a solution to a 50-year-old grand challenge in biology". <https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology/>.
- [70] A. L. Hodgkin and A. F. Huxley. "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve". *The Journal of Physiology*. 1952, ISSN: 0022-3751.
- [71] E. M. Izhikevich. "Which Model to Use for Cortical Spiking Neurons?" *IEEE Transactions on Neural Networks*. 2004, ISSN: 1941-0093. DOI: 10.1109/TNN.2004.832719.
- [72] D. E. Feldman. "Synaptic Mechanisms for Plasticity in Neocortex". *Annual Review of Neuroscience*. 2009,
- [73] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, et al. "Memristors with Diffusive Dynamics as Synaptic Emulators for Neuromorphic Computing". *Nature Materials*. 2016,
- [74] B. V. Benjamin, P. Gao, E. McQuinn, Swadesh Choudhary, A. R. Chandrasekaran, J.-M. Bussat, et al. "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations". *Proceedings of the IEEE*. 2014,
- [75] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, et al. "A Reconfigurable On-Line Learning Spiking Neuromorphic Processor Comprising 256 Neurons and 128K Synapses". *Frontiers in Neuroscience*. 2015,

- [76] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, et al. "A Million Spiking-Neuron Integrated Circuit with a Scalable Communication Network and Interface". *Science*. 2014,
- [77] M. A. Nielsen. "Neural Networks and Deep Learning". Determination Press, 2015. URL: <http://neuralnetworksanddeeplearning.com> (visited on 03/08/2021).
- [78] I. Goodfellow, B. Yoshua, and C. Aaron. "Deep Learning". MIT Press, 2016.
- [79] K. Hornik, M. Stinchcombe, and H. White. "Multilayer Feedforward Networks Are Universal Approximators". *Neural Networks*. 1989,
- [80] A. Karpathy. "CS231n Convolutional Neural Networks for Visual Recognition". <https://cs231n.github.io/>. 2021.
- [81] D. J. Finney. "Probit Analysis; a Statistical Treatment of the Sigmoid Response Curve". Probit Analysis; a Statistical Treatment of the Sigmoid Response Curve. Oxford, England: Macmillan, 1947,
- [82] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". *Proceedings of the IEEE*. 1998, ISSN: 1558-2256. DOI: 10.1109/5.726791.
- [83] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al. "Going Deeper With Convolutions". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015,
- [84] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016,
- [85] B Moons. "Embedded Deep Learning - Algorithms, Architectures and Circuits for Always-on Neural Network Processing". PhD thesis. KU Leuven, 2018.
- [86] D. H. Hubel and T. N. Wiesel. "Receptive Fields of Single Neurones in the Cat's Striate Cortex". *The Journal of Physiology*. 1959, ISSN: 0022-3751. pmid: 14403679. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130/> (visited on 11/19/2020).
- [87] A. Krizhevsky et al. "Learning multiple layers of features from tiny images". 2009.
- [88] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al. "ImageNet Large Scale Visual Recognition Challenge". *International Journal of Computer Vision*. 2015,
- [89] P. J. Werbos. "Backpropagation through Time: What It Does and How to Do It". *Proceedings of the IEEE*. 1990,
- [90] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. "Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image". *arXiv:1703.07570 [cs]*. 2017. arXiv: 1703.07570 [cs].
- [91] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. "Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion". 2018.
- [92] D. Scaramuzza and F. Fraundorfer. "Visual Odometry [Tutorial]". *IEEE Robotics Automation Magazine*. 2011, ISSN: 1558-223X. DOI: 10.1109/MRA.2011.943233.
- [93] M. Solinas, C. Galiez, R. Cohendet, S. Rousset, M. Reyboz, and M. Mermillod. "Generalization of Iterative Sampling in Autoencoders". 2020.
- [94] X. Zhou, D. Wang, and P. Krähenbühl. "Objects as Points". *arXiv:1904.07850 [cs]*. 2019. arXiv: 1904.07850 [cs].
- [95] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu. "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification". *Frontiers in Neuroscience*. 2017. ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00682.
- [96] A. Roy, S. Venkataramani, N. Gala, S. Sen, K. Veezhinathan, and A. Raghunathan. "A Programmable Event-Driven Architecture for Evaluating Spiking Neural Networks". *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 2017, DOI: 10.1109/ISLPED.2017.8009176.
- [97] W. Maass. "On the Computational Power of Winner-Take-All". *Neural Computation*. 2000, ISSN: 0899-7667. DOI: 10.1162/089976600300014827. URL: <https://doi.org/10.1162/089976600300014827> (visited on 01/29/2021).
- [98] S.-I. Amari and M. A. Arbib. "Competition and Cooperation in Neural Nets". *Systems Neuroscience*. Elsevier, 1977,

- [99] C. D. Salzman and W. T. Newsome. "Neural Mechanisms for Forming a Perceptual Decision." *Science (New York, N.Y.)* 5156 1994,
- [100] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, et al. "Event-Based Vision: A Survey". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020, ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2020.3008413. arXiv: 1904.08405.
- [101] M. A. Goodale and A. D. Milner. "Separate Visual Pathways for Perception and Action". *Trends in Neurosciences*. 1992, ISSN: 0166-2236. DOI: 10.1016/0166-2236(92)90344-8. URL: <https://www.sciencedirect.com/science/article/pii/0166223692903448> (visited on 02/11/2021).
- [102] M. Mishkin, L. G. Ungerleider, and K. A. Macko. "Object Vision and Spatial Vision: Two Cortical Pathways". *Trends in Neurosciences*. 1983, ISSN: 0166-2236. DOI: 10.1016/0166-2236(83)90190-X. URL: <https://www.sciencedirect.com/science/article/pii/016622368390190X> (visited on 02/11/2021).
- [103] J. Norman. "Two Visual Systems and Two Theories of Perception: An Attempt to Reconcile the Constructivist and Ecological Approaches". *Behavioral and Brain Sciences*. 2002, ISSN: 1469-1825, 0140-525X. DOI: 10.1017/S0140525X0200002X. URL: <https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/abs/two-visual-systems-and-two-theories-of-perception-an-attempt-to-reconcile-the-constructivist-and-ecological-approaches/9370D22D21BC4CB05FE387AFA52D3053> (visited on 02/11/2021).
- [104] I. Bramão, A. Reis, K. M. Petersson, and L. Faísca. "The Role of Color Information on Object Recognition: A Review and Meta-Analysis". *Acta Psychologica*. 2011, ISSN: 0001-6918. DOI: 10.1016/j.actpsy.2011.06.010. URL: <https://www.sciencedirect.com/science/article/pii/S0001691811001338> (visited on 02/11/2021).
- [105] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schon, B. Kohn, et al. "Embedded Vision System for Real-Time Object Tracking Using an Asynchronous Transient Vision Sensor". *2006 IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop*. Teton National Park, WY, USA: IEEE, 2006, ISBN: 978-1-4244-0535-0. DOI: 10.1109/DSPWS.2006.265448.
- [106] M. Litzenberger, D. Bauer, N. Donath, H. Garn, B. Kohn, C. Posch, et al. "Embedded Vehicle Counting System With 'Silicon Retina' Optical Sensor". *AIP Conference Proceedings*. 2006, ISSN: 0094-243X. DOI: 10.1063/1.2361239.
- [107] K. Hata and S. Savarese. "CS231A Course Notes 1: Camera Models".
- [108] J. Binns, D. Neil, S.-C. Liu, and T. Delbruck. "DDD17: End-To-End DAVIS Driving Dataset". 2017.
- [109] A. M. Andrew. "Multiple View Geometry in Computer Vision". *Kybernetes*. 2001, ISSN: 0368-492X. DOI: 10.1108/k.2001.30.9\_10.1333.2.
- [110] "OpenCV: Camera Calibration and 3D Reconstruction". [https://docs.opencv.org/4.5.1/d9/d0c/group\\_\\_calib3d.html/](https://docs.opencv.org/4.5.1/d9/d0c/group__calib3d.html/).
- [111] C. G. Harris, M. Stephens, et al. "A combined corner and edge detector." *Alvey vision conference*. Citeseer. 1988,
- [112] H. Bay, T. Tuytelaars, and L. Van Gool. "SURF: Speeded Up Robust Features". *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof, and A. Pinz. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, ISBN: 978-3-540-33833-8. DOI: 10.1007/11744023\_32.
- [113] 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*. 2017, ISSN: 2045-2322. DOI: 10.1038/srep40703.
- [114] S.-H. Ieng, J. Carneiro, M. Osswald, and R. Benosman. "Neuromorphic Event-Based Generalized Time-Based Stereovision". *Frontiers in Neuroscience*. 2018. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00442.
- [115] T. P. De, D. Perrone, and A. Sironi. "Method of Tracking Objects in a Scene". European pat. Prophesee. 2020.



- [116] B. B. Alagöz. “A Note on Depth Estimation from Stereo Imaging Systems”. *Computer Science* 1 2016, ISSN: 2548-1304. URL: <https://dergipark.org.tr/en/pub/bbd/306999> (visited on 02/24/2021).
- [117] “Anti-Aliasing — Blender Manual”. [https://docs.blender.org/manual/fr/2.79/render/blender\\_render/settings/antialiasing.html/](https://docs.blender.org/manual/fr/2.79/render/blender_render/settings/antialiasing.html/).
- [118] K. Konolige. “Projected Texture Stereo”. *2010 IEEE International Conference on Robotics and Automation*. 2010, DOI: 10.1109/ROBOT.2010.5509796.
- [119] “OpenCV: Cv::Stereo::StereoBinaryBM Class Reference”. [https://docs.opencv.org/3.4.13/d7/d8e/classcv\\_1\\_1stereo\\_1\\_1StereoBinaryBM.html#details/](https://docs.opencv.org/3.4.13/d7/d8e/classcv_1_1stereo_1_1StereoBinaryBM.html#details/).
- [120] H. Rebecq, D. Scaramuzza, and T. M. Horstschäfer. “Visual-Inertial Odometry With An Event Camera”. European pat. Universität Zürich. 2019.
- [121] “Kinetic - ROS Wiki”. <http://wiki.ros.org/kinetic/>.
- [122] T. Qin, P. Li, and S. Shen. “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator”. *IEEE Transactions on Robotics*. 2018, ISSN: 1941-0468. DOI: 10.1109/TR0.2018.2853729.
- [123] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad. “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics”. *Intelligent Industrial Systems* 4 2015, ISSN: 2199-854X. DOI: 10.1007/s40903-015-0032-7. URL: <https://doi.org/10.1007/s40903-015-0032-7> (visited on 09/25/2020).
- [124] E. M. Foxlin, M. Harrington, and Y. Altshuler. “Miniature Six-DOF Inertial System for Tracking HMDs”. *Helmet- and Head-Mounted Displays III*. Helmet- and Head-Mounted Displays III. International Society for Optics and Photonics, 1998, DOI: 10.1117/12.317434. URL: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3362/0000/Miniature-six-DOF-inertial-system-for-tracking-HMDs/10.1117/12.317434.short> (visited on 02/23/2021).
- [125] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier. “Asynchronous Event-Based Visual Shape Tracking for Stable Haptic Feedback in Microrobotics”. *IEEE Transactions on Robotics*. 2012, ISSN: 1941-0468. DOI: 10.1109/TR0.2012.2198930.
- [126] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza. “Continuous-Time Visual-Inertial Odometry for Event Cameras”. *IEEE Transactions on Robotics*. 2018, ISSN: 1941-0468. DOI: 10.1109/TR0.2018.2858287.
- [127] A. Zihao Zhu, N. Atanasov, and K. Daniilidis. “Event-Based Visual Inertial Odometry”. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2017/html/Zhu\\_Event-Based\\_Visual\\_Inertial\\_CVPR\\_2017\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2017/html/Zhu_Event-Based_Visual_Inertial_CVPR_2017_paper.html) (visited on 02/22/2021).
- [128] J. Bertrand, A. Yiğit, and S. Durand. “Embedded Event-Based Visual Odometry”. *2020 6th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*. 2020 6th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP). 2020, DOI: 10.1109/EBCCSP51266.2020.9291346.
- [129] Y. Zhou, G. Gallego, and S. Shen. “Event-Based Stereo Visual Odometry”. 2020. arXiv: 2007.15548 [cs]. URL: <http://arxiv.org/abs/2007.15548> (visited on 02/22/2021).
- [130] C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. Mahony, and D. Scaramuzza. “Fast Image Reconstruction with an Event Camera”. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2020, URL: [https://openaccess.thecvf.com/content\\_WACV\\_2020/html/Scheerlinck\\_Fast\\_Image\\_Reconstruction\\_with\\_an\\_Event\\_Camera\\_WACV\\_2020\\_paper.html](https://openaccess.thecvf.com/content_WACV_2020/html/Scheerlinck_Fast_Image_Reconstruction_with_an_Event_Camera_WACV_2020_paper.html) (visited on 02/23/2021).
- [131] S. Pini, G. Borghi, and R. Vezzani. “Learn to See by Events: Color Frame Synthesis from Event and RGB Cameras”. 2019. arXiv: 1812.02041 [cs]. URL: <http://arxiv.org/abs/1812.02041> (visited on 02/23/2021).
- [132] P. Shedligeri and K. Mitra. “Photorealistic Image Reconstruction from Hybrid Intensity and Event-Based Sensor”. *Journal of Electronic Imaging*. 2019, ISSN: 1017-9909, 1560-229X. DOI: 10.1117/1.JEI.28.6.063012. URL: <https://www.spiedigitallibrary.org/journals/journal-of-electronic-imaging/volume-28/issue-6/063012/Photorealistic->

- image-reconstruction-from-hybrid-intensity-and-event-based-sensor/10.1117/1.JEI.28.6.063012.short (visited on 02/23/2021).
- [133] Z. Zhang and D. Scaramuzza. "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry". *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, DOI: 10.1109/IROS.2018.8593941.
- [134] G. Gallego and D. Scaramuzza. "Accurate Angular Velocity Estimation With an Event Camera". *IEEE Robotics and Automation Letters*. 2017, ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2647639.
- [135] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". *2019 International Conference on Robotics and Automation (ICRA)*. 2019 International Conference on Robotics and Automation (ICRA). 2019, DOI: 10.1109/ICRA.2019.8793887.
- [136] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. "Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization". *The International Journal of Robotics Research* 3 2015, ISSN: 0278-3649. DOI: 10.1177/0278364914554813. URL: <https://doi.org/10.1177/0278364914554813> (visited on 10/01/2020).
- [137] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios". *IEEE Robotics and Automation Letters*. 2018, ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2793357.
- [138] M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana, and S. Apoorva. "Feature Extraction Using Convolution Neural Networks (CNN) and Deep Learning". *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*. 2018, DOI: 10.1109/RTEICT42901.2018.9012507.
- [139] K. Sakuma. "3D Integration in VLSI Circuits". 1st. Boca Raton: CRC Press., 2018. 233 pp.
- [140] P. Vivet, G. Sicard, L. Millet, S. Chevobbe, K. B. Chehida, L. A. Cubero, et al. "Advanced 3D Technologies and Architectures for 3D Smart Image Sensors". *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2019 Design, Automation Test in Europe Conference Exhibition (DATE). 2019, DOI: 10.23919/DATE.2019.8714886.
- [141] M. Bouvier, A. Valentian, and G. Sicard. "Scalable Pitch-Constrained Neural Processing Unit for 3D Integration with Event-Based Imagers". *Proceedings of the 58th Annual Design Automation Conference*. San Francisco, CA, USA, 2021,
- [142] P. Yang, G. Aglieri, C. Cavicchioli, P. Chalmet, N. Chanlek, A. Collu, et al. "Low-Power Priority Address-Encoder and Reset-Decoder Data-Driven Readout for Monolithic Active Pixel Sensors for Tracker System". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2015, ISSN: 01689002. DOI: 10.1016/j.nima.2015.02.063. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0168900215002818> (visited on 09/01/2020).
- [143] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier. "STDP-Based Spiking Deep Convolutional Neural Networks for Object Recognition". *Neural Networks*. 2018, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2017.12.005. URL: <http://www.sciencedirect.com/science/article/pii/S0893608017302903> (visited on 09/03/2020).
- [144] F. Paredes-Vallés, K. Y. W. Scheper, and G. C. H. E. de Croon. "Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 2020, ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2019.2903179. arXiv: 1807.10936. URL: <http://arxiv.org/abs/1807.10936> (visited on 09/03/2020).
- [145] J. Jo and Y. Bengio. "Measuring the Tendency of CNNs to Learn Surface Statistical Regularities". *arXiv:1711.11561 [cs, stat]*. 2017. arXiv: 1711.11561 [cs, stat].
- [146] J. Backus. "Can Programming Be Liberated from the von Neumann Style?: A Functional Style and Its Algebra of Programs". *ACM Turing Award Lectures*. New York, Association of Computing Machinery, 2007,
- [147] M. Horowitz. "Computing's Energy Problem (and What We Can Do about It)". IEEE, 2014,
- [148] N. R. Mahapatra and B. Venkatrao. "The Processor-Memory Bottleneck". *Crossroads* 3es 1999,

- [149] K. A. Boahen. "Point-to-Point Connectivity between Neuromorphic Chips Using Address Events". *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*. 2000,
- [150] C. Frenkel, J.-D. Legat, and D. Bol. "MorphIC: A 65-Nm 738k-Synapse/Mm<sup>2</sup> Quad-Core Binary-Weight Digital Neuromorphic Processor With Stochastic Spike-Driven Online Learning". *IEEE Transactions on Biomedical Circuits and Systems*. 2019, ISSN: 1940-9990. DOI: 10.1109/TBCAS.2019.2928793.
- [151] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, et al. "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning". *IEEE Micro*. 2018,
- [152] E. P. Frady, G. Orchard, D. Florey, N. Imam, R. Liu, J. Mishra, et al. "Neuromorphic Nearest-Neighbor Search Using Intel's Pohoiki Springs". *arXiv:2004.12691 [cs]*. 2020. arXiv: 2004.12691 [cs].
- [153] J. Schemmel, D. Briiderle, A. Griibl, Matthias Hock, K. Meier, and S. Millner. "A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neural Modeling". IEEE, 2010,
- [154] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. "The SpiNNaker Project". *Proceedings of the IEEE* 5 2014,
- [155] G. Urgese, F. Barchi, E. Macii, and Andrea Acquaviva. "Optimizing Network Traffic for Spiking Neural Network Simulations on Densely Interconnected Many-Core Neuromorphic Platforms". *IEEE Transactions on Emerging Topics in Computing*. 2016,
- [156] J. C. Thiele, O. Bichler, and A. Dupret. "Event-Based, Timescale Invariant Unsupervised Online Deep Learning with STDP". *Frontiers in Computational Neuroscience*. 2018,
- [157] D. Kirk. "NVIDIA Cuda Software and Gpu Parallel Computing Architecture". *Proceedings of the 6th International Symposium on Memory Management*. ISMM '07. New York, NY, USA: Association for Computing Machinery, 2007, ISBN: 978-1-59593-893-0. DOI: 10.1145/1296907.1296909.
- [158] "Silicon Brains". <https://www.humanbrainproject.eu/en/silicon-brains/>.
- [159] "Loihi Deep Dive Architecture, SDK, Examples". SUNY Polytechnic Institute, 2019.
- [160] S. Herculano-Houzel. "Scaling of Brain Metabolism with a Fixed Energy Budget per Neuron: Implications for Neuronal Activity, Plasticity and Evolution". *PLoS ONE*. 2011. Ed. by M. Perc,
- [161] W. Singer. "Neuronal Synchrony: A Versatile Code for the Definition of Relations?" *Neuron* 1 1999,
- [162] C. Bédard and A. Destexhe. "Local Field Potential Interaction with the Extracellular Medium". *Encyclopedia of Computational Neuroscience*. Ed. by D. Jaeger and R. Jung. New York, NY: Springer, 2013, ISBN: 978-1-4614-7320-6. DOI: 10.1007/978-1-4614-7320-6\_720-1. URL: [https://doi.org/10.1007/978-1-4614-7320-6\\_720-1](https://doi.org/10.1007/978-1-4614-7320-6_720-1) (visited on 03/09/2021).
- [163] "Myelination - an Overview | ScienceDirect Topics". URL: <https://www.sciencedirect.com/topics/medicine-and-dentistry/myelination> (visited on 03/09/2021).
- [164] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, et al. "Cognitive Computing Building Block: A Versatile and Efficient Digital Neuron Model for Neurosynaptic Cores". IEEE, 2013,
- [165] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri. "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)". *IEEE Transactions on Biomedical Circuits and Systems*. 2018, ISSN: 1940-9990. DOI: 10.1109/TBCAS.2017.2759700.
- [166] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades". *Frontiers in Neuroscience*. 2015. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00437.
- [167] M. Horowitz. "1.1 Computing's Energy Problem (and What We Can Do about It)". *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2014, DOI: 10.1109/ISSCC.2014.6757323.
- [168] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei. "14.3 A 28nm SoC with a 1.2GHz 568nJ/Prediction Sparse Deep-Neural-Network Engine with >0.1 Timing Error Rate Tolerance for IoT Applications". IEEE, 2017,

- [169] D. Neil and S.-C. Liu. "Effective Sensor Fusion with Event-Based Sensors and Deep Network Architectures". *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. Montréal, QC, Canada: IEEE, 2016, ISBN: 978-1-4799-5341-7. DOI: 10.1109/ISCAS.2016.7539039.
- [170] M. Pfeiffer and T. Pfeil. "Deep Learning With Spiking Neurons: Opportunities and Challenges". *Frontiers in Neuroscience*. 2018. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00774.
- [171] S. Yin, S. K. Venkataramanaiah, G. K. Chen, R. Krishnamurthy, Y. Cao, C. Chakrabarti, et al. "Algorithm and Hardware Design of Discrete-Time Spiking Neural Networks Based on Back Propagation with Binary Activations". 2017. arXiv: 1709.06206 [cs]. URL: <http://arxiv.org/abs/1709.06206>.
- [172] J. Park, J. Lee, and D. Jeon. "A 65-Nm Neuromorphic Image Classification Processor With Energy-Efficient Training Through Direct Spike-Only Feedback". *IEEE Journal of Solid-State Circuits*. 2020, ISSN: 1558-173X. DOI: 10.1109/JSSC.2019.2942367.
- [173] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol. "A 0.086-Mm<sup>2</sup> 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-Nm CMOS". *IEEE Transactions on Biomedical Circuits and Systems*. 2019, ISSN: 1940-9990. DOI: 10.1109/TBCAS.2018.2880425.
- [174] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy. "A 4096-Neuron 1M-Synapse 3.8-pJ/SOP Spiking Neural Network With On-Chip STDP Learning and Sparse Weights in 10-Nm FinFET CMOS". *IEEE Journal of Solid-State Circuits*. 2019, ISSN: 1558-173X. DOI: 10.1109/JSSC.2018.2884901.
- [175] I. Miro Panades and A. Greiner. "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures". *First International Symposium on Networks-on-Chip (NOCS'07)*. First International Symposium on Networks-on-Chip (NOCS'07). 2007, DOI: 10.1109/NOCS.2007.14.
- [176] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol. "A 0.086-Mm<sup>2</sup> 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28nm CMOS". *IEEE Transactions on Biomedical Circuits and Systems*. 2018, ISSN: 1932-4545, 1940-9990. DOI: 10.1109/TBCAS.2018.2880425.
- [177] C. Frenkel, J.-D. Legat, and D. Bol. "A 28-Nm Convolutional Neuromorphic Processor Enabling Online Learning with Spike-Based Retinas". *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, DOI: 10.1109/ISCAS45731.2020.9180440.
- [178] "Apple Unleashes M1". <https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/>.
- [179] "TSMC Developing 2nm Tech at New R&D Center - Taipei Times". <https://taipeitimes.com/News/front/archives/2020/08/26/200374229/5>. 2020.
- [180] K. J. Kuhn. "Considerations for Ultimate CMOS Scaling". *IEEE Transactions on Electron Devices*. 2012, ISSN: 1557-9646. DOI: 10.1109/TED.2012.2193129.
- [181] E. J. Nowak, I. Aller, T. Ludwig, K. Kim, R. V. Joshi, Ching-Te Chuang, et al. "Turning Silicon on Its Edge [Double Gate CMOS/FinFET Technology]". *IEEE Circuits and Devices Magazine*. 2004, ISSN: 1558-1888. DOI: 10.1109/MCD.2004.1263404.
- [182] N. Singh, F. Y. Lim, W. W. Fang, S. C. Rustagi, L. K. Bera, A. Agarwal, et al. "Ultra-Narrow Silicon Nanowire Gate-All-Around CMOS Devices: Impact of Diameter, Channel-Orientation and Low Temperature on Device Performance". *2006 International Electron Devices Meeting*. 2006, DOI: 10.1109/IEDM.2006.346840.
- [183] G. Hills, C. Lau, A. Wright, S. Fuller, M. D. Bishop, T. Srimani, et al. "Modern Microprocessor Built from Complementary Carbon Nanotube Transistors". *Nature*. 2019, ISSN: 1476-4687. DOI: 10.1038/s41586-019-1493-8.
- [184] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, et al. "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS". *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. 2007, DOI: 10.1109/ISSCC.2007.373606.
- [185] I. Jani. "Test and Characterization of 3D High-Density Interconnects". PhD thesis. Université Grenoble Alpes, 2019.

- [186] P. Garrou, M. Koyanagi, and P. Ramm. "Handbook of 3D Integration, Volume 3: 3D Process Technology". John Wiley & Sons, 2014. ISBN: 978-3-527-67012-3.
- [187] M. Bruel. "Process for the Production of Thin Semiconductor Material Films". Pat. 1994.
- [188] L. Brunet, C. Fenouillet-Beranger, P. Batude, S. Beaupaire, F. Ponthenier, N. Rambal, et al. "Breakthroughs in 3D Sequential Technology". *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018, DOI: 10.1109/IEDM.2018.8614653.
- [189] A. Vandooren, J. Franco, Z. Wu, B. Parvais, W. Li, L. Witters, et al. "First Demonstration of 3D Stacked Finfets at a 45nm Fin Pitch and 110nm Gate Pitch Technology on 300mm Wafers". *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018, DOI: 10.1109/IEDM.2018.8614654.
- [190] K. Gopalakrishnan, R. S. Shenoy, C. T. Rettner, K. Virwani, D. S. Bethune, R. M. Shelby, et al. "Highly-Scalable Novel Access Device Based on Mixed Ionic Electronic Conduction (MIEC) Materials for High Density Phase Change Memory (PCM) Arrays". *2010 Symposium on VLSI Technology*. 2010, DOI: 10.1109/VLSIT.2010.5556229.
- [191] F. Disegni, R. Annunziata, A. Molgora, G. Campardo, P. Cappelletti, P. Zuliani, et al. "Embedded PCM Macro for Automotive-Grade Microcontroller in 28nm FD-SOI". *2019 Symposium on VLSI Circuits*. 2019, DOI: 10.23919/VLSIC.2019.8778129.
- [192] Y. Cao, Y. Chen, and D. Khosla. "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition". *International Journal of Computer Vision* 1 2015,
- [193] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer. "Fast-Classifying, High-Accuracy Spiking Deep Networks through Weight and Threshold Balancing". IEEE, 2015,
- [194] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci. "Conversion of Artificial Recurrent Neural Networks to Spiking Neural Networks for Low-Power Neuromorphic Hardware". IEEE, 2016,
- [195] J. A. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, et al. "Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing—Application to Feedforward ConvNets". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 2013,
- [196] B. Rueckauer and S. Liu. "Conversion of Analog to Spiking Neural Networks Using Sparse Temporal Coding". *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018, DOI: 10.1109/ISCAS.2018.8351295.
- [197] N. Caporale and Y. Dan. "Spike Timing—Dependent Plasticity: A Hebbian Learning Rule". *Annual Review of Neuroscience* 1 2008,
- [198] T. Masquelier and S. J. Thorpe. "Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity". *PLoS Computational Biology* 2 2007,
- [199] S. Song, K. D. Miller, and L. F. Abbott. "Competitive Hebbian Learning through Spike-Timing-Dependent Synapticplasticity". *Nature Neuroscience* 9 2000,
- [200] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier. "Combining STDP and Reward-Modulated STDP in Deep Convolutional Spiking Neural Networks for Digit Recognition". 2018.
- [201] Y. Jin and P. Li. "AP-STDP: A Novel Self-Organizing Mechanism for Efficient Reservoir Computing". *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016 International Joint Conference on Neural Networks (IJCNN), 2016,
- [202] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, et al. "NVM Neuromorphic Core with 64k-Cell (256-by-256) Phase Change Memory Synaptic Array with on-Chip Neuron Circuits for Continuous in-Situ Learning". IEEE, 2015,
- [203] J. C. Tapson, G. K. Cohen, S. Afshar, K. M. Stiefel, Y. Buskila, R. M. Wang, et al. "Synthesis of Neural Networks for Spatio-Temporal Spike Pattern Recognition and Processing". *Frontiers in Neuroscience*. 2013,
- [204] R. V. Florian. "Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity". *Neural Computation* 6 2007,
- [205] F. Ponulak. "ReSuMe -New Supervised Learning Method for Spiking Neural Networks". *Technical Report*. 2005.

- [206] A. MOHEMMED, S. SCHLIEBS, S. MATSUDA, and NIKOLA KASABOV. "SPAN: Spike Pattern Association Neuron For Learning Spatio-Temporal Spike Patterns". *International Journal of Neural Systems* 04 2012,
- [207] Q. Yu, H. Tang, K. C. Tan, and H. Li. "Precise-Spike-Driven Synaptic Plasticity: Learning Hetero-Association of Spatiotemporal Spike Patterns". *PLoS ONE* 11 2013. Ed. by W. W. Lytton,
- [208] S. M. Bohte, J. N. Kok, and H. L. Poutrá. "Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons". *Neurocomputing*. 2002,
- [209] D. Huh and T. J. Sejnowski. "Gradient Descent for Spiking Neural Networks". *ArXiv e-prints*. 2017.
- [210] Y. Jin, W. Zhang, and P. Li. "Hybrid Macro/Micro Level Backpropagation for Training Deep Spiking Neural Networks". *ArXiv e-prints*. 2018.
- [211] N. Rathi, G. Srinivasan, P. Panda, and K. Roy. "Enabling Deep Spiking Neural Networks with Hybrid Conversion and Spike Timing Dependent Backpropagation". 2020. arXiv: 2005.01807 [cs, stat]. URL: <http://arxiv.org/abs/2005.01807> (visited on 03/09/2021).
- [212] T. Liu, Z. Liu, F. Lin, Y. Jin, G. Quan, and W. Wen. "MT-Spike: A Multilayer Time-Based Spiking Neuromorphic Architecture with Temporal Error Backpropagation". *arXiv:1803.05117 [cs, q-bio]*. 2018. arXiv: 1803.05117 [cs, q-bio].
- [213] C. Lee, P. Panda, G. Srinivasan, and K. Roy. "Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-Training Followed by Supervised Fine-Tuning". *Frontiers in Neuroscience*. 2018. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00435.
- [214] J. Wu, Y. Chua, M. Zhang, Q. Yang, G. Li, and H. Li. "Deep Spiking Neural Network with Spike Count Based Learning Rule". *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, DOI: 10.1109/IJCNN.2019.8852380.
- [215] H. Mostafa. "Supervised Learning Based on Temporal Coding in Spiking Neural Networks". *IEEE Transactions on Neural Networks and Learning Systems*. 2018, ISSN: 2162-237X. DOI: 10.1109/TNNLS.2017.2726060.
- [216] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida. "Deep Learning in Spiking Neural Networks". *Neural Networks*. 2019, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2018.12.002.
- [217] E. O. Neftci, H. Mostafa, and F. Zenke. "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks". *IEEE Signal Processing Magazine*. 2019, ISSN: 1558-0792. DOI: 10.1109/MSP.2019.2931595.
- [218] F. Zenke, S. M. Bohté, C. Clopath, I. M. Comşa, J. Göltz, W. Maass, et al. "Visualizing a Joint Future of Neuroscience and Neuromorphic Engineering". *Neuron*. 2021, ISSN: 0896-6273. DOI: 10.1016/j.neuron.2021.01.009.
- [219] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. "Regularization of Neural Networks Using DropConnect". *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, 2013,
- [220] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. "Efficient Neural Architecture Search via Parameter Sharing". *arXiv:1802.03268 [cs, stat]*. 2018. arXiv: 1802.03268 [cs, stat].
- [221] H. Bagherinezhad, M. Horton, M. Rastegari, and A. Farhadi. "Label Refinery: Improving ImageNet Classification through Label Progression". *arXiv:1805.02641 [cs]*. 2018. arXiv: 1805.02641 [cs].
- [222] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst. "14.5 Envision: A 0.26-to-10TOPS/W Subword-Parallel Dynamic-Voltage-Accuracy-Frequency-Scalable Convolutional Neural Network Processor in 28nm FDSOI". *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, DOI: 10.1109/ISSCC.2017.7870353.
- [223] A. P. Arechiga and A. J. Michaels. "The Robustness of Modern Deep Learning Architectures against Single Event Upset Errors". *2018 IEEE High Performance Extreme Computing Conference (HPEC)*. 2018, DOI: 10.1109/HPEC.2018.8547532.

## *Bibliography*

---

- [224] J. Park, J. Lee, and D. Jeon. "7.6 A 65nm 236.5nJ/Classification Neuromorphic Processor with 7.5% Energy Overhead On-Chip Learning Using Direct Spike-Only Feedback". *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2019, DOI: 10.1109/ISSCC.2019.8662398.
- [225] A. Yousefzadeh, G. Orchard, E. Stomatias, T. Serrano-Gotarredona, and B. Linares-Barranco. "Hybrid Neural Network, An Efficient Low-Power Digital Hardware Implementation of Event-Based Artificial Neural Network". IEEE, 2018,



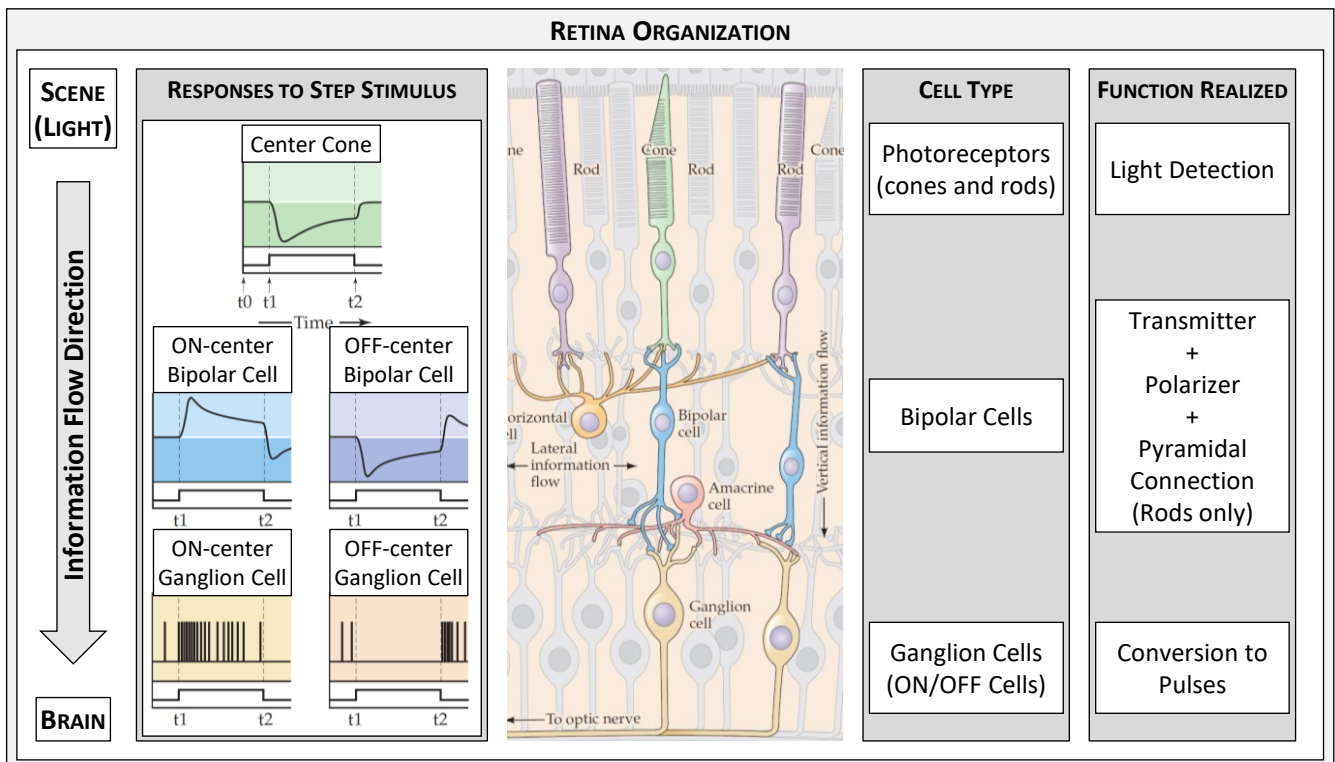
# A

## The Biological Vision Pipeline

All the content that follows is summarized from chapter 11 of [14]. It describes the first layer of information treatment processed inside the eye.

### A.1 The Eye's Retina Function

The retina converts spatial luminous information into an encoded information consisting in a two-dimensional train of spikes whose spike rate depends on light illumination condition and rate of change. The internal organization and operation of the retina is illustrated Figure 1.



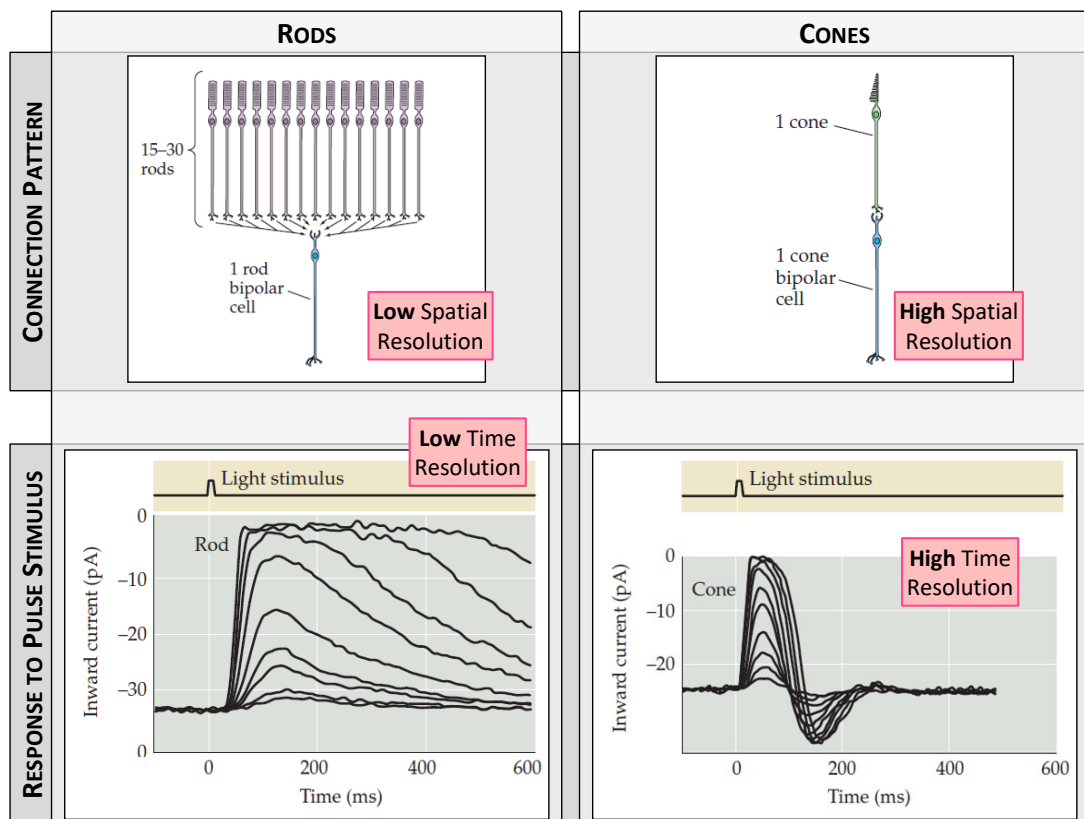
**Figure A.1:** Illustration of the internal organization of the retina, functional behavior from top to bottom. **(a)** Photoreceptor (cones and rods) realizing the phototransduction, i.e., converting luminous energy into electric membrane potential. **(b)** Bipolar cells, conducting the potential to the next stage; adding some diversity with inverted and non-inverted potentials distribution. **(c)** Ganglion cells, realizing the ON/OFF function and converting the membrane potential to spike trains that encode light information (intensity and rate of change) for downstream cell layers. Pictures taken and adapted from chapter 11 of [14] ©2018 Oxford University Press.

First light is collected by photoreceptor cells, the rods, and cones. They convert light into an electrical potential onto their membrane. This potential is then transmitted to ganglion

cells through the bipolar cells. Its behavior is realized thanks to 3 types of cells: the rods, the cones, and the ganglion cells.

### A.1.1 Rods and Cones

The internal operation of rods and cones are nearly identical: they convert luminous energy into electrical potential. However, they differ in their response time and spatial organization over the retina surface.



**Figure A.2:** Detailed behavior of cells composing the retina. Rods and cones playing the role of photodetectors. Pictures taken and adapted from chapter 11 of [14] ©2018 Oxford University Press.

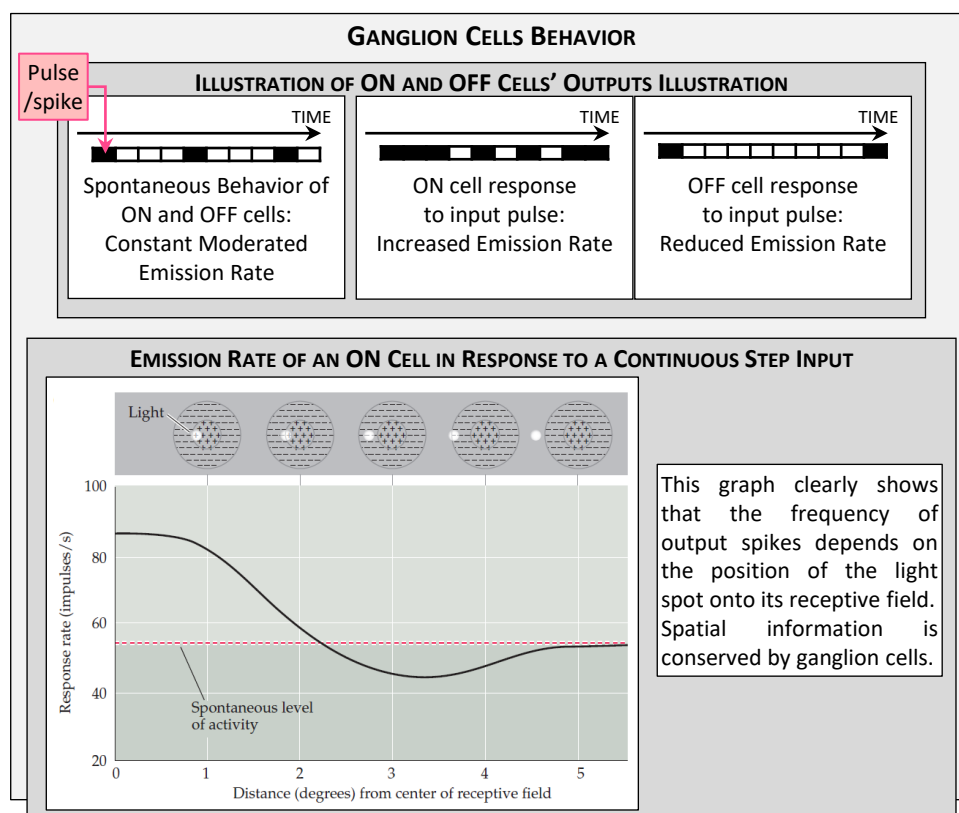
- The rods are more sensitive to low level illumination than cones. Also, their answer to bright environment saturates while the ones of the cones do not. The rods are thus favored for vision at night.
- The rods have larger receptive fields than cones. This comes from the fact that the bipolar cells conducting the output of the rods to the ganglion cells take several (15 to 30) rods as input while the ones conducting information from the cones take a single one in input. The cones thus enable a higher resolution than the rods.
- Cones are mostly present at the center of the retina - the *fovea* - whereas the fovea does not contain any rod. However, the peripheral surface of the retina is mostly populated by rods.
- Finally, both rods and cones have nearly instantaneous response to a sudden light pulse, but the cool down time of rods is about 800ms while the one of the cones is around 200ms. i.e., cones are four times faster than rods.

- Only the cones contribute to color vision. However, the diverse types of cones are not wavelength specific, they all respond to any wavelength of the visual spectrum, but the color sensation is reconstructed by a network of inhibition from downstream cells.

To sum up, the rods specialize in fast detection even under "bad" - in the sense of low brightness - illumination condition and approximate spatial analysis whereas the cones are favored for high resolution and precise "imaging". These points are illustrated Figure A.2.

### A.1.2 Ganglion Cells

The ganglion cells are electrical transducers and spatiotemporal filters that convert the electrical potential generated by the rods and cones into a pulse of spike train whose spike rate depends on the amplitude of their input potential and its rate of change. Under steady



**Figure A.3:** Detailed behavior of cells composing the retina. Ganglion cells play the role of electrical transducers, as the analog-to-digital converters of image sensors. However, they convert a temporally continuous physical value (the membrane potential) to a train of pulses discretized in time: event-based data. They do not directly code the illumination level information, they also analyze the spatial localization of the information. Pictures taken and adapted from chapter 11 of [14] ©2018 Oxford University Press.

light exposition condition, they continuously emit spikes at a certain pace. However, if they detect a spatial (not temporal) illumination contrast, positive or negative - in which case they are called ON or OFF cells respectively - the spike emission rate changes accordingly. E.g., as illustrated Figure A.3, ON cells emit spike more frequently when they detect bright spot in the center of the receptive fields, and inversely less frequently when the bright spot is in the periphery of their receptive field. OFF cells behave the same but with dark spots. On

top of that, ganglion cells are also temporal filters in the sense that when light exposition conditions change suddenly in time, for example with a flash of light, these cells will emit a burst of spike at high frequency before stabilizing at a steady pace.

To conclude, the human retina converts spatial and temporal luminous information into discrete two-dimensional trains of spikes that encode for spatial and temporal light distribution.

# B

## Arbitration Power Consumption

---

To evaluate the power consumption of the readout system independently of the pixel grid power consumption, the strategy we used was to first estimate the power consumption of the pixel grid, and then subtract it from the total chip power consumption.

The method for estimating the pixel grid power consumption is given in [32] and re-used in [18]. It consists in evaluating what they call the *dynamic energy*  $E_d$  and the *static power*  $P_s$ . The dynamic energy is evaluated as:

$$E_d = \frac{P_H - P_L}{F_{event_H} - F_{event_L}} \quad (\text{B.1})$$

Where  $P_H$  and  $P_L$  are the power at high and low activity respectively, and  $F_{event_H}$  and  $F_{event_L}$  are the event rate at high and low activity, respectively. The dynamic energy  $E_d$  is a measure of the energy consumed by a single spike. The static power per pixel can then be obtained by removing the dynamic power from the total power at low activity:

$$P_s = (P_L - F_{event_L} \times E_d) / N_p \quad (\text{B.2})$$

Where  $N_p$  is the total number of pixels. So, the static power is a measure of the power consumed by one pixel when it does not spike. Hence, one can argue that  $N_p \times P_s$  is the total power of the pixel grid only, the readout system consuming everything else. So, the power consumption of the readout system is:

$$P_{RO} = P_H - N_d \times P_s \quad (\text{B.3})$$

We evaluated these values for several published sensors and present them in Table B.1. The most recent sensors [18, 35] do not employ the classical readout system consisting in a 2D arbitration mechanism; they arbitrate the matrix of pixel in one dimension only, namely by row, and read the full row in parallel. Hence, temporal precision along a row is lost and traded-off against a faster readout method.

**Table B.1:** State-of-the-Art Event-Based Imagers: Details and Power Consumption

Appendix B. Arbitration Power Consumption

Reference		[18]	[33]	[35]	[24]
Resolution	Columns (X)	1280	640	640	240
	Rows (Y)	720	480	480	180
Number of pixels		921600	307200	307200	43200
Event Rate (ev/s)	High activity	300M	66M	300M	12M
	Low activity	100k	30k	100k	1k
Power (mW)	High activity	94	95	50	14
	Low activity	43	36	27	5
Dynamic Energy $E_d$ (pJ/event)		170	894	77	750
Static Power $P_s$ (nW/pixel)		47	117	88	114
Read Out Power $P_{RO}$ (mW)		51	59	23	9
<b>Relative RO Power (%)</b>		<b>54.2</b>	<b>62.2</b>	<b>46.0</b>	<b>64.8</b>
Readout Dimensionality		1D	-	1D	2D
IC Technology		3D: 90nm- 40nm	180nm	90nm	180nm

# C

## Details About Selected Recent Event-Based Sensors

---

### C.1 Samsung

The work of the EB team at Samsung is relatively close to the one of iniVation. They design EB sensors under the appellation dynamic vision sensors (DVS) and the characteristics of the Samsung Gen 3 [36] sensor are nearly exactly like the ones of the DVXplorer [39].

Throughout the years, they tested different strategies to enable high resolution sensors. For enabling VGA scale resolutions, they conceived an asynchronous arbitration scheme where a full column is asynchronously accessed in parallel. Spikes are then synchronously encoded by group of eight pixels, what they call the G-AER [35], which permits to rapidly dissociate spiking from non-spiking pixels. Passing from a classical 2D AER to G-AER permitted to increase the sensor throughput from 6.5Mev/s to 300Mev/s [36]. Then, they switched to a full synchronous readout mode, with a global shutter exposition of the grid of pixels at 2000fps, followed by sequential a column scan of the grid of pixels. It enables to reduce motion artifacts during exposition time - as for classical imagers - but reduce the fully EB aspect of the imager. Note that even if Samsung DVS sensors are not officially commercially available, the characteristics of this last sensor closely very closely to the ones of the DVXplorer from iniVation [39]. Finally, in 2019, they announced that they were deploying 3D IC technologies to shrink pixels from 9 $\mu$ m pitch to 4.95 $\mu$ m. The sensor is also able to filter data on-chip for eliminating flicker problems and de-noise the data.

### C.2 Prophesee

While the work of iniVation historically consisted in providing the world with prototype artificial retinas, Prophesee has come into being as a startup targeting industrial usage of EB sensors [34]. They claim to provide the industry novel vision solutions for problems that could not easily be solved with classical imagers, like particles counting or leak detection. For several years they developed sensors under both DVS (that they call contrast detection (CD)) and ATIS pixel modalities, i.e., respectively without and with luminance information outputted at each spike [33].

Their last realization - which is also their first detailed scientific communication about the internal operation of their sensor - is a 720p imager designed in three-dimensional integration technologies [18]. The readout module, working with an asynchronous arbiter at the scale of rows - takes advantage of the 3D interconnection to sample full rows of 1280 pixels at once.



On top of that, they added functionalities for reducing the output bandwidth of the imager. The main point is that they employ on-chip ROI-based filtering by groups of 40x23 pixels to reduce the activity of the imager where it is not useful application-wise. For example, continuously spiking because of clouds in the sky is not considered as a useful information for autonomous vehicles. In addition, they compress the outputted spike data - the AER information - by row, enabling them to reduce data size at up to 1bit per outputted spike when running at maximum event load.

### **C.3 CelePixel**

Finally, CelePixel technology, a startup that emerged from the Nanyang Technological University of Singapore, put on the market a 1.02Mpx sensor the CeleX V [23].

It realizes on-chip a variety of function non other megapixel resolution EB sensor can realize namely optical-flow or spikes with encoded luminance measurements. On top of that, each pixel can timestamp the moment of spike emission on a relatively reduced time scale. This in-pixel timestamp is combined with a to a larger scale timestamping at the grid level, which permits to conserve the temporal precision of the sensor independently of the readout method deployed. They thus designed a sort of 2D event-based scan strategy for outputting the data. The matrix is scanned by row and then arbitrated by column. If any pixel on a row is has sent a request command, the row is set active, and every spiking pixel is sequentially encoded with a low priority arbitration by column.

In usual situations this scheme is efficient, and again follows a less bio-inspired strategy to enable advanced functionalities. However, under maximum event load, the outputted data is closer to a 60fps rolling shutter acquisition scheme than an actual 2D asynchronous EB data flow. It is the drawback brought by a readout scheme where each pixel is encoded individually and sequentially - it easily brings a delay at the scale of the matrix of pixels.

# D

## 3D Integration Technologies

---

### D.1 The End of Moore's Law

Reducing the size of transistors has been done for decades with the aim of gaining in density, power consumption, and operation speed of designed circuit chips [20]. Nowadays, transistors are industrialized at the 5nm node [178], and the 2nm node should be reached in a couple of years [179].

However, transistor scaling arrives to an end. Indeed, the size of a Silicon atom being approximately 0.1nm, a 2nm gate represents only a few atoms. Reaching the nanometer scale, several device issues are becoming hardly manageable, for example quantum tunneling of electrons through the gate oxide of the transistors. At the circuit level, these small effects result in increased leakage - and thus power consumption-, and faulty behaviors with for example loss of memory on SRAM devices or transistor switch failures. With what many people called the end of Moore's law becoming a truth, solutions must be found to continue developing circuit chips that can do more calculations while consuming less energy and still having a smaller area footprint.

Thus, several research axes are being studied at all stages of integrated circuit development namely the device, system, and package levels.

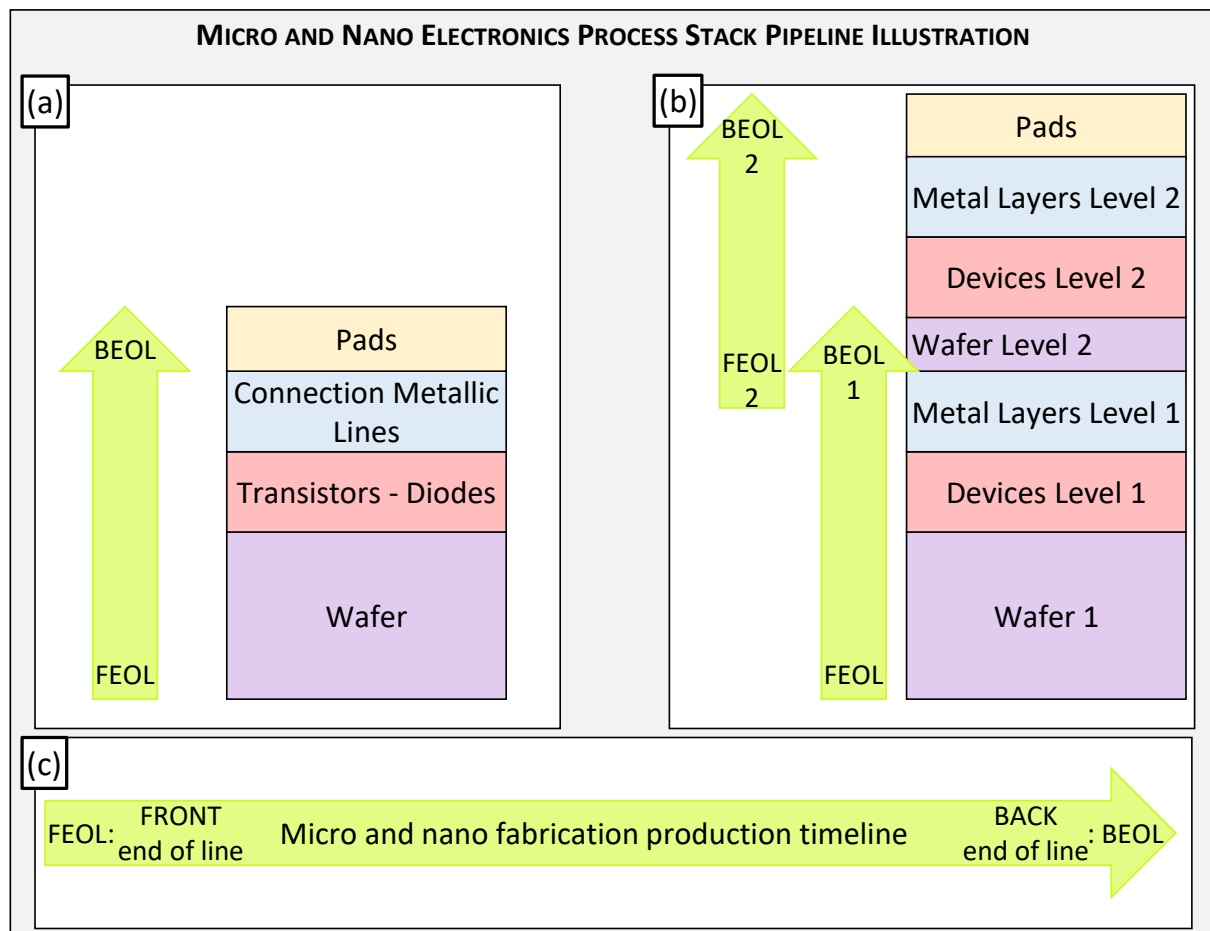
- **Device level.** Researchers are working on the implementation of complex structures for improving the performance of single transistors. Simply scaling down classical metal-oxide-semiconductor field-effect transistors (MOSFET)s does not provide the energy benefits anymore [180]. So, advanced transistor structures and materials are being studied. The most common are the multi-gate devices with fin field-effect transistors [181] (FinFETs) or nanowire gate all around transistor [182] (GAAT), and the carbon nanotube field-effect transistor [183] (CNTFET).
- **System level.** Circuit chips aims at realizing functions, and because of many factors, designing parts or full chips aiming at a single (or a relatively restrained scope) operation has become an important part of IC research and development efforts. One can think about dedicated chips like GPUs, but even the general-purpose CPUs actually rely on the combination of many dedicated small parts commonly called accelerators (like SIMD vector processors [184]). These chips are thus commonly called system-on-chip (SOC).
- **Package level.** Doing more on the same two-dimensional surface is possible when going three-dimensional [185]. In an analogous way to large cities growing vertically instead of

horizontally with multi-floor buildings, stacking circuit chips onto one another permits to gain in both efficiency and exploitable surface for functional operations.

In this section, we focus on 3D integration as it is a technology already widely used for image sensor design.

## D.2 Standard 2D Integration Manufacturing Process

Integrated circuit technologies are traditionally based on single wafer processing followed by dicing and packaging for final product obtainment. In a nutshell, the micro and nano fabrication pipeline follows this order: fabrication of the semi-conducting (usually in pure Silicon) substrate wafer, devices (transistors, diodes, etc.) realization on the surface of the wafer, isolation of the devices and inter-connections with metallic lines on several levels, and finally insulation of the full stack, pad realization and dicing of each circuit chip. The device fabrications step is commonly called the front end of line (FEOL) and the metal lines realization the back end of line (BEOL). Figure D.1 (a) schematizes the standard steps of integrated circuit manufacturing.



**Figure D.1:** Illustration of the manufacturing pipeline of micro and nano technologies in the cases of (a) standard 2D circuit chips, and (b) monolithic 3D integration. (c) Explicit illustration of the FEOL and BEOL concepts through the global processing pipeline.

## D.3 3D Processing Working Principle

Stacking a circuit on top of another one to gain in functionality requires to connect devices of both circuits together. Hence, solutions for interconnecting circuits through the stack - the 3D interconnections - are required. There are three main techniques for 3D integration [186].

- Bonding pieces together via pads with die-on-die, die-on-wafer and wafer-on-wafer bonding. This technic is commonly called microbumps or Cu-Cu bonding, making reference to the size or the material of the metallic depositions permitting to make the connection between the wafers.
- Digging vias into a wafer for integration on both sides, commonly called through silicon vias.
- Integrating several device levels in a single process on a single side of a wafer.

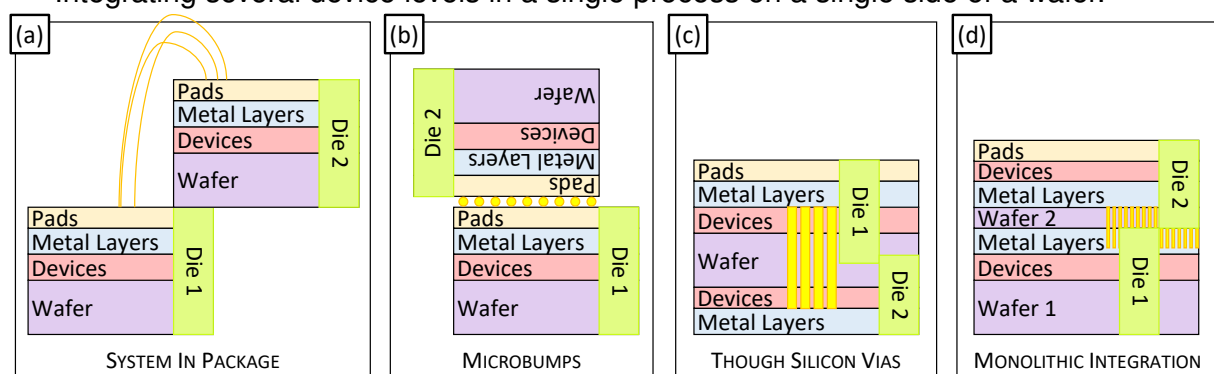


Figure D.2: The different techniques for 3D integration.

### D.3.1 Bumps and Through Silicon Vias

#### D.3.1.1 Stacking with Microbumps

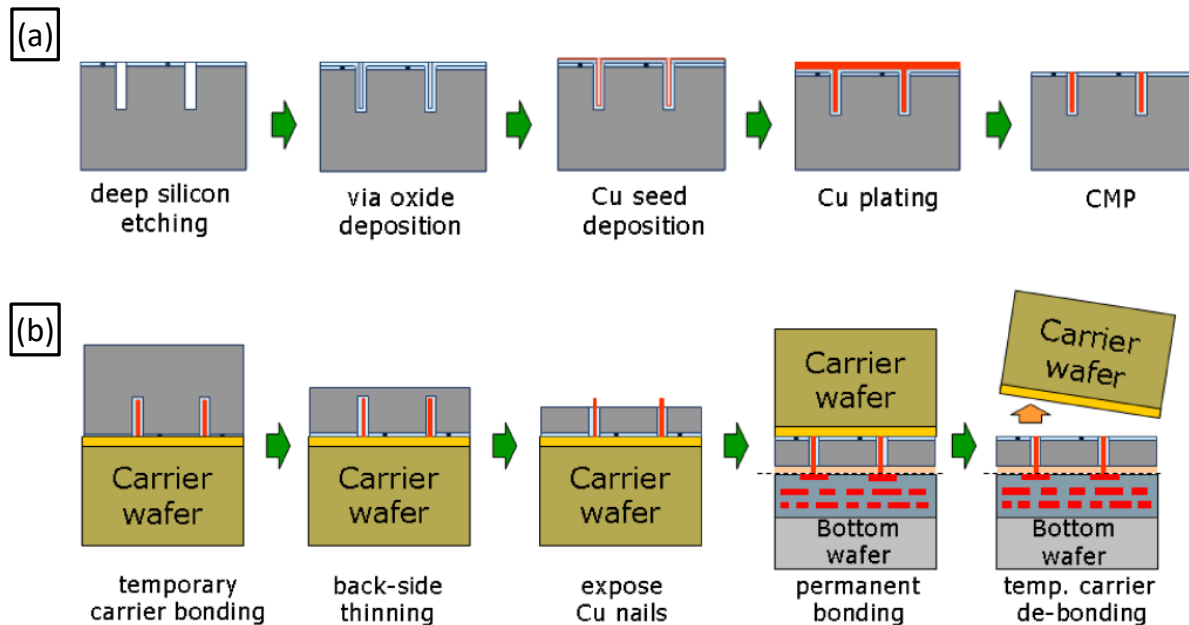
Microbump - a.k.a. solder ball - fabrication is conceptually simple. Instead of using the BEOL pads for making the connections to the external package with metallic wires, they are used for making the connection with another wafer or die [45]. Microbumps of metal, usually copper, are deposited on the pads, then one wafer is flipped, and the two wafers are put on top of each other, heated, and pressurized to weld the bumps together. Currently, distances between microbumps can reach micrometer scale interconnections [185].

#### D.3.1.2 Fabricating TSVs

Manufacturing through silicon vias consists in digging through the bulk wafer to connect pads on its the bottom side to the top side. The TSV fabrication can be summarized as a two-step process [47]:

1. Dig holes and fill them with conducting materials, as illustrated Figure D.3 (a).
2. Thin wafer on the opposite side, as illustrated Figure D.3 (b) .

On-the-shelf wafers are initially hundreds of micrometers thin. But, for filling a via homogeneously, its aspect ratio should not be smaller than one to ten. Therefore, wafers are thinned for making TSV. Nevertheless, below 50 $\mu\text{m}$ , they become unpractical for handling and may break. So currently, the minimum diameter of a via is 5 micrometers [47].



**Figure D.3:** Processing steps of TSVs. Taken from [47] ©2014 IEEE.

### D.3.1.3 Limitations of microbumps and TSVs

The TSVs takes an important active Silicon surface. Indeed, as holes are etched on the FEOL part of the wafer, transistors or other devices cannot be realized on the same surface than TSVs. On top of that, it still requires bonding to other wafers as supplementary devices cannot be realized on the bottom side of the substrate, due to BEOL thermal budget limitations.

Microbumps, on the other hand, do not consume much space because pads are sets on the top of the BEOL metal lines. However, they bring major technological challenges which limit their minimum pitch size to 1 micrometer.

## D.3.2 Sequential Monolithic 3D Integration

The last but not the least is Monolithic 3D Integration, which is not quite well defined yet [46]. The idea of monolithic 3D is mostly defined by the distance between the interconnections enabled by the technology, and by the idea that the integration of the two distinct tiers of the stack is done sequentially. Monolithic technology enables 3D interconnect with pitch of less than tens of nm, hence allowing even transistor-scale interconnection on several tiers - i.e., reaching the minimum functional granularity of any element of integrated circuits. In a sense, monolithic 3D is the possibility to have a single wafer with 2 levels of transistors, each layer integrated sequentially.

### D.3.2.1 Technological improvements enabling monolithic 3D

Several technological progresses have enabled reaching monolithic scale 3D interconnect pitches of tens of nanometers [46]:

1. Improvements in wafer aligning. Pitches of 50nm are now realizable thanks to advanced precision bonders.

2. Thinning *after* bonding [187]. This enables thin TSV, with diameters smaller than 5 $\mu\text{m}$ , and thus high-density interconnection and packaging possibilities.
3. Low temperature FEOL processes (below 500°C) [188, 189].

The latter was the main limiting challenge for sequential 3D integration. A standard processing pipeline sees its *thermal budget* decreasing along with the processing steps. This is because the physical topology of a transistor changes when undergoing treatments at high temperature.

### D.3.2.2 Examples of technological processes used for monolithic integration

There are many actors working for enabling monolithic 3D integration for building complex system-on-chip. We cite here a few notable researches, and a more exhaustive overviews of the SOA of monolithic 3D integration can be found in [46].

The CoolCube<sup>®</sup> project is the combination of the Smart Cut<sup>™</sup> process used to deposit a thin layer of Silicon on top of another wafer [187]. Then, a second FEOL integration procedure can be realized thanks to low temperature processes [188]. Reference [189] have shown that it is possible to sequentially stack FinFET and junctionless transistors during the second FEOL process.

One can also considered integrating specific devices during the BEOL steps as a sort of monolithic 3D integration. It is usually done for integrating novel non-volatile memory (NVM) devices - including PCM and OxRAMs [190, 191]. In these processing pipelines memristive elements are integrated at the metallic levels 3, 4 or 5, and the BEOL steps then go on. Several actors also use a sort of 3D where they implement arrays/matrices of memory elements, called RRAMs , in the BEOL process flow.

# E

## Training a Spiking Neural Network

---

Training an ANN is mostly done with backpropagation of the gradient, and relatively often with a combination of reinforcement learning and backpropagation [67]. For what concerns SNNs, three main techniques have been widely used. Some train classical NNs and convert them into spiking ones after training. Others deploy bio-inspired, usually unsupervised, learning rules. Finally, SNNs are also trained with supervised techniques, however, because of the dual dimensionality (network depth and time), such methods are harder and more computationally expensive to implement than for classical networks.

### E.1 Conversion

Converting an ANN into an SNN consists in training the artificial model and then transforming it into a spiking one by changing the neuron model and activation functions. The parameters obtained when training the artificial model are conserved, i.e., the resulting spiking network has the same weights and architecture - topology - as the artificial one. At first, ANN to SNN conversion was not trivial because spiking neurons behave quite differently than formal perceptrons, notably with nonlinear and not derivable activation functions - their membrane potential. Nevertheless, many conversion frameworks have been developed [95, 192–195]. Initially, these frameworks required to remove some internal ANN mechanisms to enable the conversion, such as batch normalization for example, thus resulting in a less performing model before conversion [192, 194]. Hence, state-of-the-art NN architectures could not be easily transferred from classical to event-driven version. However, it is now possible to design SNNs that benefit from techniques such as batch normalization, inception layers, and others thanks to the work of Rueckauer et al. [95]. In 2018, they achieved the best results among SNNs by deploying deep network topologies onto classical frame-based object recognition datasets that are the MNIST [82], CIFAR [87], and ImageNet [88].

However, converting ANNs to SNNs has the major drawback of requiring implementing rate coding for transmitting post-synaptic values - also called activation maps in classical NNs - in between neuron layers. Such coding dramatically increases the number of spikes required for the operation of the SNN as it requires to send several spikes to transmit a single activation value. See Figure E.1 (c) for an illustration of the diverse ways of representing a real number using binary spikes and time flow. And the problem lies in the fact that any supplementary spike represents an additional energy cost for the hardware evaluation of an SNN, see the discussion in section IV.B.2.c.iii. In addition, a classical network cannot naturally be trained

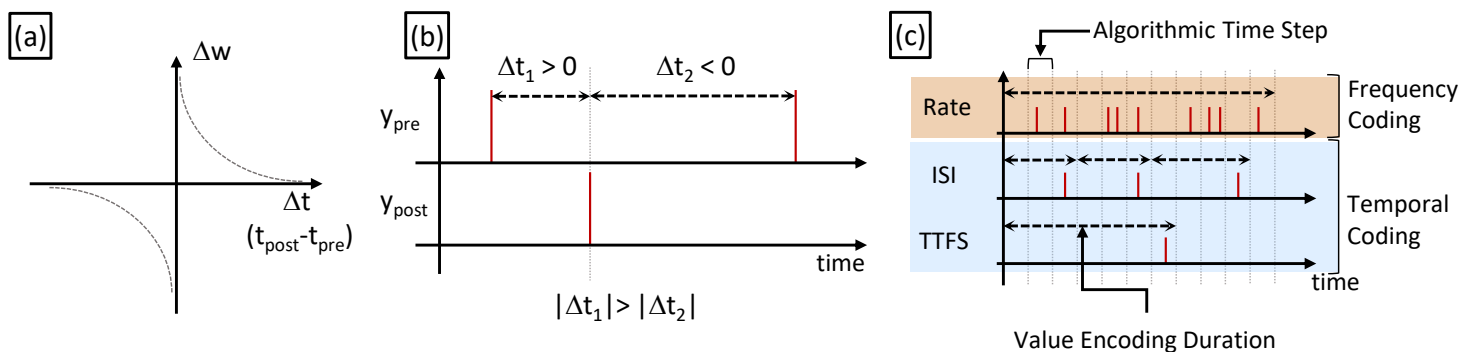


on event-driven datasets, such as [21, 22, 166]. So, deploying event-driven networks onto event-driven datasets requires to convert the dataset to frame-based data, then to train an ANN on it, and finally convert the trained network into an SNN; which does not really make sense.

Nevertheless, more recently, several works tried to exploit temporal coding for applying ANN to SNN conversion [196]. Temporal coding encodes an activation value with a single spike instead of several spike for rate coding. However, resulting networks are fixed and can hardly be fine-tuned, again all functions cannot be converted this way, and as for any conversion method it is not possible to exploit time-related benefits of SNNs when training a non-spiking network. Hence, a large amount of effort is being put into direct SNN training.

## E.2 Unsupervised Learning

The bio-inspiration and the expected consequent effectiveness of unsupervised learning led to consider the spike timing dependent plasticity (STDP) algorithm as a promising training method [197–199]. STDP states that the values of the weight updates of every synapse at each training step depend only on the relative timings of pre- and post-synaptic neuron spikes, as depicted in Figure E.1 (a-b). If the post-synaptic neuron fires shortly after the pre-synaptic one, a causality is expected between the two and their synaptic connection is strengthened, i.e., the weight value is increases. Inversely, if the first to fire is the post-synaptic one, then the relation is weakened.



**Figure E.1:** Spike timing dependent plasticity (STDP) rule illustration. **(a)** Diagram of the relation between weight update  $\Delta w$  and relative timings of post- and pre-synaptic spikes  $\Delta t$ . **(b)** Timing diagram illustrating the relative input-output spike timings and associated weight updates. **(c)** Illustration of the duration required for encoding values with binary spikes depending on the information representation strategy. Adapted from our survey [25] ©2019 ACM.

The main limitation of the STDP learning rule is its locality. Each layer adapts to the output of the preceding layer, without coordination between them or with other downstream layer. So, the full function of the network is the result of local modifications layer by layer, and the network cannot be "manually" oriented toward a specific function. It is thus difficult to train deep networks with STDP that have a high accuracy on the targeted task. Different methods have thus been combined with STDP to try and improve the resulting accuracy, such as adding supervision to the training [200], pre-processing the input data to bias the network

[143], or modifying the learning rule [156, 201]. These methods have shown interesting improvements with respect to STDP only methods, but results in accuracy that are behind the ones of SNN trained with supervised learning rule and backpropagated ANNs.

Nevertheless, the STDP learning rule is easily implementable within hardware [202] and is less computationally expensive than a global learning rule such as backpropagation of the gradient. Moreover, it has a proven biological basis [197], it is thus pertinent for cognitive modelling applications or for training shallow networks. We thus argue that STDP-trained layers could be used as pre-processing blocks that naturally extract statistically relevant information in full computer vision application. That is what we did for the work presented in chapter IV.

### **E.3 Supervised Learning**

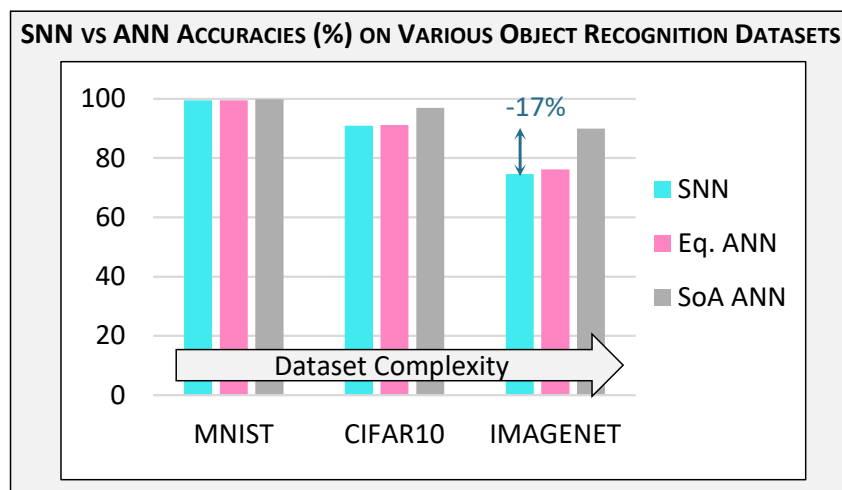
While unsupervised learning does not permit training SNN to obtain high accuracy networks on object classification tasks, supervised methods have emerged, and some of them bring back optimism in the field of spiking neural networks. Backpropagation cannot be realized in SNNs as they are because of the non-differentiability of neurons and synapses models.

Through years, many different strategies have emerged for supervising the training of these networks. Some analytically compute the values of the network parameters [203], which is of course hardly scalable. Older techniques exploit a modified version of STDP, with teachers that force the supervision [204, 205] But more recent ones realize backpropagation by exploiting various method, with for example approximated derivable neuron behaviors [206, 207], or exploitation of derivations enabled by the continuity of the time flow in temporally encoded networks [208–210]. Under certain condition error gradients can be evaluated and backpropagation can thus be exploited.

As training techniques start to be successful, some try to make them efficient and fast. In this purpose, several researchers have designed hardware-friendly methods for direct on-chip training backpropagation methods [171, 173]. A recent notable work of Rathi et al. [211] suggests combining ANN to SNN conversion with an event-driven backpropagation algorithm relying on a time-based consideration of derivability of neurons. They demonstrate an important gain in training complexity with respect to other works, and we believe that hybrid training may be an interesting way to train SNNs on tasks where classical networks perform well already.

Nevertheless, on frame-based datasets, event-driven networks are still behind the state of art deep learning equivalent networks, as illustrated Figure E.2. On top of that, the more the complexity of the dataset, the more the complexity of the network, and the more the difference in accuracy between classical networks and spiking ones; at least for object recognition tasks.

In addition, the two dimensions for differentiating the network - depth and time as illustrated Figure II.23 (c) - can be a major problem for dealing with time-related data. As it requires backpropagation to occur through two dimensions, it drastically increases the amount of computation required for training an SNN [89, 212]. Deploying single dimension backpropagation with SNNs composed of IF neurons is fine as long as the application does not require to operate through time [213–215] - e.g., static images against videos. In these works, neurons take as inputs rate or temporally coded values, which does not integrate actual time-related information apart from the coding itself. However, when dealing with temporal sequences that contain semantics through time - as most of the datasets acquired with event-based imagers -, such networks may not be able to extract the time-based semantic contained in the data. Several recent survey papers dealing with training an SNN have been proposed, and we re-direct the reader to these for more information [170, 216–218].



**Figure E.2:** Comparison of accuracies of SOA standard ANNs and Eq. ANNs with spiking NNs. SOA stands for state-of-the-art - to the best to our knowledge - and *Eq.* means that the classical network topology is identical to the spiking one. Results from [95, 219–221].

## E.4 Considerations About Training a SNN

### E.4.1 The MNIST Standard

Many SNN accelerators compared themselves on the famous frame-based dataset MNIST [82] for competing on figures of merit such as the energy per classification or the accuracy of the final spiking neural network [25]. However, we argue that this present two main drawbacks. First, the MNIST data must be converted to event-based data to be compatible with SNN computation flow. The method for conversion depends on the work considered - there is no standard for that - which may impact the results and limit the possibility to compare works together. It adds a bias in the data. Secondly, the MNIST dataset is actually a relatively simple one. ANNs are no more competing on it but try to improve their performances on more complex datasets [219–221]. However, SNNs' absolute performance is still based on it, especially for hardware design purposes. This is because MNIST is a simple dataset onto which even a one layer fully-connect network can perform well. It thus permits to design

small and energy efficient chips with relatively low efforts. Hence, we argue that training an SNN only to show that it works on MNIST is a useless effort, as it mostly depends on the training algorithm and not the network deployed.

#### **E.4.2 On-Chip or Off-Chip Training?**

On-chip learning should be considered only if the target application justifies it. If the final purpose of a designed neuromorphic chip is to detect objects, e.g., for surveillance purposes, there is no need for on-chip learning. Now, if the chip is designed for general purpose spiking neural networks acceleration, it could be great to enable on-chip learning. However, selecting an algorithm to be implemented efficiently is not an easy matter as it will limit the scope of algorithms the circuit can evaluate.

# F

## Methods for Energy Efficient Neuromorphic Hardware Design

---

### F.1 Techniques Applied for any NN Acceleration Platform

The hardware specific energy efficiency methods are relatively numerous, and most of them rely on the fact that the dynamic power consumption of a synchronous circuit can be expressed as:

$$P_{dyn} = \alpha \times f_{clk} \times V_{dd}^2 \quad (\text{F.1})$$

where  $f_{clk}$  is the clock frequency of the circuit and  $V_{dd}$  its supply voltage.  $\alpha$  is a scalar factor.

#### F.1.1 Dynamic Voltage and Frequency Scaling

Hence, a very efficient technic is to apply dynamic voltage and frequency scaling. It consists in adapting the frequency and/or the supply voltage of a circuit so that the computational bandwidth (operations per seconds) is adapted to the input data rate [85, 173, 174, 222]. For example, if a video stream is reduced from 60 frames per seconds to 30 frames per second, the frequency can be dynamically reduced by a factor 2x and thus the power consumption is halved. However, the total energy consumed is the sum of the contributions of the dynamic energy  $E_{dyn}$ , i.e., the energy consumed when the circuit realizes logical operations - in concrete terms when transistors are switched - and of the static energy  $E_{static}$ :

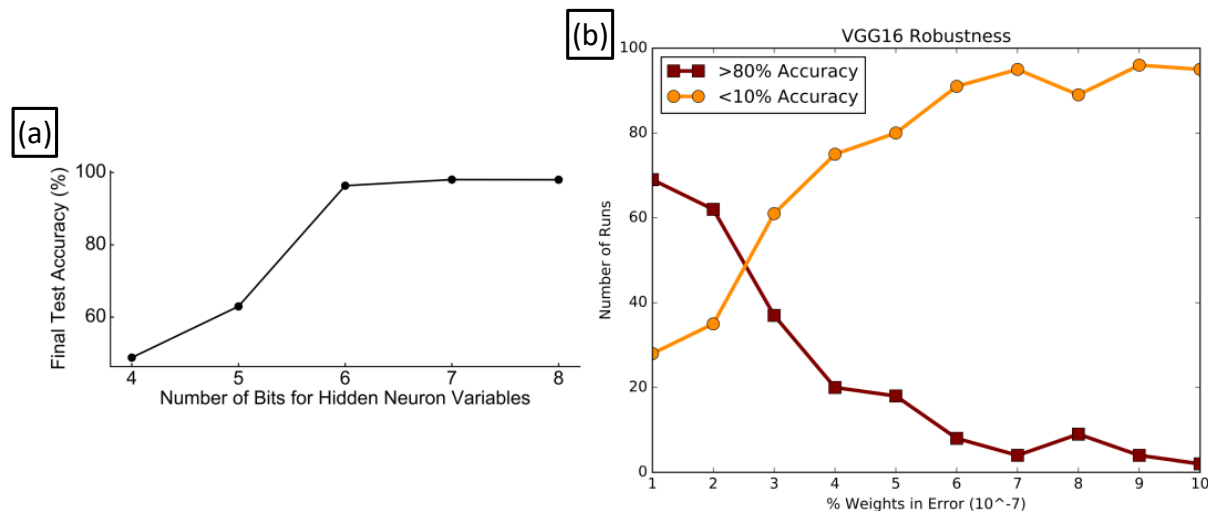
$$E_{tot} = E_{dyn} + E_{static} = \int P_{dyn}(t)dt + \int P_{static}(t)dt \quad (\text{F.2})$$

Static power consumption comes from transistors and other components leakage. So, a tradeoff is to be set between the computation time and the computation speed. When reducing dynamic power, at a certain point the static power becomes predominant in the total energy consumption of a circuit. It is thus also necessary to try and minimize the static power  $P_{static}$ . Also, note that usually a designer working toward an embedded VLSI circuit puts a significant amount of effort to minimize the required clock frequency as well as the nominal voltage.

#### F.1.2 Approximate Computing

Another knob onto which an algorithm hardware co-design can be optimized is the bitlength of the variables. Algorithms are conceived and tested at full precision, i.e., 32 to 64 bits in floating point representation for full dynamic range. Then, for embedding it, the precision of the variables - their bitlength - is reduced. This technic is called *approximate computing*. Reducing the bitlength usually impacts the precision of the application, as a small number, for example 0.001, could be approximated to zero when downsizing its binary representation.

This zero could then propagate throughout the computations resulting in a totally different solution.



**Figure F.1:** (a) Illustration of the impact of variables' bitlength (membranes and weights) on the final accuracy of a feed-forward spiking neural network on the MNIST classification task. Extracted from [172] ©2020 IEEE. (b) Illustration of the impact of single bit errors on weights values inside a VGG16 NN configuration. Extracted from [223] ©2018 IEEE.

But in the case of NNs, many studies have shown that they are resilient to *variable quantization*. A large amount of variation in the variable precision does not drastically impact the overall accuracy of the network [85, 172], as illustrated Figure F.1 (a). Weights and activations maps with reduced bitlength bring two advantages. It drastically reduces the energy consumption and area footprint of the circuit thanks to the implementation of shorter memories and numerical operators. For example, a 32b floating-point adder consumes 30x more energy and is 116x larger, as depicted Figure IV.8. Quantizing networks parameters and activation maps is usually called compression. Fault tolerance is the ability of an algorithm to be resilient to faulty hardware. And there is an erroneous belief that NNs are highly fault tolerant. But recent studies have shown that a few bitflips inside NN topologies can lead to dramatic failure of the solution evaluation, as depicted Figure F.1 (b).

## F.2 Techniques Exclusive to SNN Acceleration Platforms

For what concerns event-driven networks, other mechanisms can be exploited based on the properties of the data.

### F.2.1 Data Skip

Regarding fault tolerance, SNN enable dynamic accuracy-precision tradeoff by reducing the number of time steps to solution. It is indeed possible to stop the evaluation of a stimulus by skipping part of the spikes contained in the input sample. This permits to achieve near-equivalent accuracy while reducing the energy per classification by an order of magnitude, as illustrated in the table of Figure IV.9 (b) on page 92. Some even go to one-shot SNN by

reducing the number of time steps to a single one [224]. By doing so, they drastically reduce the number of operations required by at least 30%.

## **F.2.2 Data Stream Processing**

Spikes are a sparse sequential flow of binary events. It is thus possible to design accelerators that evaluate data on the flow, usually called *data stream processing* accelerators. A neurocore thus takes advantage of the fact that the data goes one way and update its state only when data is present. When no data is available, the core switch to idle mode, avoiding continuous computation. This requires storing the timestamps of each spike to keep track of the time that passes but permits to realize neuron updates upon load only, which avoids continuously updating neuron states when no data is present [96]. This allows, among other things, to easily manage clock gating between modules, as will be further demonstrated in chapter IV of this thesis.

## **F.3 Address Event Representation Protocol**

The AER communication format between the cores permits to reduce the bus size while keeping a large connection capability [15, 149]. Because the timestamps are shared by every core, and AER packet transmitted in-chip is reduced to a single address, thus reducing the switching activity for communication.

### **F.3.1 Drawbacks Exclusive to Spiking Neural Networks**


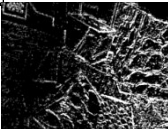



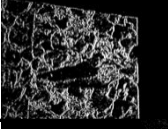



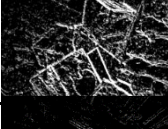


Nevertheless, with advantages come drawbacks. SNN are harder to train than classical ANNs. They thus do not permit to reach state-of-the-art art performances for at equivalent energy consumption, as illustrated Figure E.2 of Appendix E.

As already discussed, a memory movement is far more power consuming than a MAC operation. An SNN should thus present an exceptionally large amount of sparsity to minimize the number of spikes treated. Moreover, hybridization can be applied to ANNs to take advantage of the important sparsity of ANNs. After training, an ANN usually presents many null weights, resulting, because of the MAC operation, in an important data sparsity. Hence, adapting the AER protocol to communicate only non-nullified activations has been shown to be a good strategy for maximizing the chip efficiency [225].



# G

## EFR-VIO Results Comparison with State-of-the-Art

Dataset Sample [21, 135]			Method	VIO Only		SLAM	
Standard Frame	EB Frame (post EMC)	Name		RMSE Rotation (deg)	RMSE Translation (m)	RMSE Rotation (deg)	RMSE Translation (m)
		boxes_6DoF [21]	Ours EB	22.4	<b>0.348</b>	1.8	<b>0.062</b>
			RPG Zurich <sup>[126,137]</sup>	39.4	0.418	5.6	0.199
			Ours Std. <sup>[121]</sup>	111.6	0.529	7.0	0.263
		dynamic_6DoF [21]	Ours EB	9.0	<b>0.139</b>	2.8	<b>0.033</b>
			RPG Zurich <sup>[126,137]</sup>	21.2	0.220	5.0	0.076
			Ours Std. <sup>[121]</sup>	46.9	0.288	8.8	0.154
		poster_6DoF [21]	Ours EB	24.2	<b>0.257</b>	4.2	<b>0.063</b>
			RPG Zurich <sup>[126,137]</sup>	128.0	0.302	12.0	0.144
			Ours Std. <sup>[121]</sup>	88.4	0.325	83.7	0.327
		shapes_6DoF [21]	Ours EB	11.9	0.239	4.1	0.068
			RPG Zurich <sup>[126,137]</sup>	12.3	<b>0.225</b>	3.9	<b>0.065</b>
			Ours Std. <sup>[121]</sup>	171.8	0.295	167.7	0.293
		hdr_boxes [21]	Ours EB	9.7	<b>0.173</b>	5.5	0.081
			RPG Zurich <sup>[126,137]</sup>	11.0	0.286	6.0	0.161
			Ours Std. <sup>[121]</sup>	88.3	0.387	2.3	<b>0.052</b>
		drone indoor45_4 [135]	Ours EB	6.3	<b>1.540</b>	5.5	<b>1.178</b>
			RPG Zurich <sup>[126,137]</sup>	N.C.	N.C.	N.C.	N.C.
			Ours Std. <sup>[121]</sup>	13.8	3.645	10.9	2.551

**Figure G.1:** Samples of results obtained with our implementation of the EFG-VIO algorithm on various samples of the datasets of [21, 135]. RMSE computed as proposed by [133]. Different version of the algorithm are tested (with and without pose graph optimization). The results titled *RPG Zurich* are directly provided by the Robotic and Perception Group of the ETH Zurich [126, 137]. All other results are obtained with our ROS based implementation. N.C. Stands for *Not Communicated*.