KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Neuromorphic Sensor Fusion and Continual Learning for Drone Navigation and Radar Sensing

Ali Safa

Supervisors:
Prof. dr. ir. Georges Gielen
Prof. dr. ir. Francky Catthoor

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2024

Neuromorphic Sensor Fusion and Continual Learning for Drone Navigation and Radar Sensing

Ali SAFA

Examination committee:
Prof. dr. ir. Yves Willems, chair
Prof. dr. ir. Georges Gielen, supervisor
Prof. dr. ir. Francky Catthoor, supervisor
Prof. dr. ir. Marian Verhelst
Prof. dr. ir. Gert Cauwenberghs
(University of California San Diego, USA)
Dr. ir. Ilja Ocket
(imec, Belgium)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2024

© 2024 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Ali Safa, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgements

I would like to sincerely address my deepest gratitude to all the people that have helped and contributed to the work presented in this thesis.

First of all, I would like to express my heartfelt gratitude to my promoters Prof. Francky Catthoor and Prof. Georges Gielen for their exceptional guidance and enthusiasm. I felt highly encouraged during the research and I deeply appreciated the many meetings and brainstorms we had, as well as the highly detailed feedback that they provided along the way.

I would also like to thank my dear imec colleagues without whom this PhD would have not been possible. Special thanks to my daily advisor Dr. Lars Keuninckx for sharing his great wisdom and sharing highly memorable coffee breaks at the Java bar. Special thanks to Dr. Ilja Ocket who trusted me with this PhD research program and provided great support along the way. Many thanks as well to my manager Matthias Hartmann for his great kindness and general support during the completion of this PhD.

A great special thank to Dr. Tim Verbelen for his invaluable help during this research. His help was key during our drone navigation research and we enjoyed a great number of successful collaborations. I will never forget how much I enjoyed working with you Tim.

I would also like to express my special heartfelt gratitude to Prof. Gert Cauwenberghs for kindly welcoming me at UC San Diego during Spring 2023. I had such a great experience visiting your department and enjoyed many fruitful collaborations and brainstorming sessions with you and your team.

I am also grateful to the PhD jury and supervision committee. Thank you to Prof. Marian Verhelst for providing great feedback during the various supervision committee meetings, thank you to Prof. Bart Dhoedt and Prof. Yves Willems as well.

Many thanks to the scientific directors of our department at imec, André

Bourdoux and Prof. Hichem Sahli for providing great technical support in radar processing and AI.

I am also grateful to the whole ARF and neuromorphic departments at imec, many thanks to Eddy De Greef, Hans Capelle, Hamed Javadi, Achiel Colpaert, Wout Mommen, Mammoun Guenach, Thomas Gielen, Marc Bauduin, Ruoyu Feng, Adham Sakhnini, Bjorn Debaille, Adnan Al Baba, Meng Li, Yigit Ertugrul, Claude Desset, Nazar Idrees, Guanzhi Tang, Amirreza Yousefzadeh, Federico Corradi and many more for their precious help along the way.

A special thank to my KU Leuven colleagues: Mark Daniel Alea, Jonah Van Assche and Marco Carlino with whom I deeply enjoyed collaborating. Many thanks to my teaching assistant colleagues Sergio Massaioli and Francesco Lorenzelli with whom I enjoyed memorable teaching time during our lab sessions.

Thank you to the imec IDLab groups at U Antwerp and U Gent as well, it was great working with you Toon Van De Maele, Ing Jyh Tsang, Werner Van Leekwijck.

I also want to acknowledge the Flanders AI research program for partially funding the research presented in this thesis, as well as the Belgian Research Fund (FWO) for funding my research collaboration at UC San Diego.

Finally, a big thank you to my family for the great support along the way. You are such a indispensable part of the work achieved in this thesis.

Abstract

Today, there is an ever increasing demand for embedding advanced intelligence capabilities in energy- and area-constrained battery-powered devices such as advanced wearables, on-body health monitoring systems, AR / VR headsets and small robots such as drones. Therefore, there has been a surge in research effort focusing on the use of Deep Neural Networks (DNNs) and their acceleration in specialized hardware. However, these DNN solutions are still lagging far behind the remarkable efficacy of biological brains, such as, for example, in the honey bee, which is capable of a large range of complex tasks while consuming only 10 μW of power. Hence, the study of event-driven Spiking Neural Networks (SNNs) and local Hebbian plasticity rules (such as Spike-Timing-Dependent Plastity or STDP) has attracted great attention in recent years, as a more bio-plausible, neuromorphic model of computation, attempting to replicate in a more faithful manner the biological neural mechanisms found in the brain. This emerging AI field, termed Spiking Neuromorphic Computing, is seeking to take direct inspiration from biology in order to enable a highly accurate execution of complex AI tasks under the tightest of energy and chip area budgets, where conventional DNNs would significantly need more energy and area.

Currently, however, a landslide adoption of SNNs and local STDP for online learning has been jeopardized by a lack of high-accuracy, large-scale real-world demonstrations on tasks other than traditional pattern recognition using small-scale databases, and by the limited number of applications with sensory modalities other than image-type data. To help address these current issues, this thesis focuses on the algorithmic aspect of SNNs and their practical application for both radar processing and the navigation of small drones via radar-imaging sensor fusion, and online learning approaches using STDP. As such, significant contributions are made to both the theoretical understanding of SNNs and STDP, as well as to innovative neuromorphic application paradigms.

In this regard, this thesis provides the following contributions. Firstly, the novel use of SNNs with radar signals is studied with an in-depth comparison between SNN and DNN performance. It is shown that SNN implementations cannot simply be substituted for more heavy DNN implementations. In particular, the crucial importance of pre-processing for signal-to-spike encoding is demonstrated for the radar gesture application by experimenting with more than 10 different encoding approaches. It is shown that the impact of radar pre-processing on the task accuracy is far more important when using SNNs compared to using DNNs. By using an appropriate pre-processing, it is shown that a 4-bit-weight SNN can reach the same accuracy (93%) as a DNN using full-precision weights.

Secondly, the experience gathered in radar pre-processing is then used to set up a first-of-its-kind drone platform embarking a retina-inspired event-based camera and a 79 GHz radar sensor for studying drone navigation tasks such as Simultaneous Localization and Mapping (SLAM), and people detection in an indoor warehouse context, using a custom radar detector running in the on-board radar MCU.

Thirdly, this thesis provides a novel theoretical framework for the study of SNN-STDP learning systems. As opposed to prior works that explore SNN-STDP from a bottom-up perspective, by setting up randomly-connected networks following direct biological inspiration, this thesis explores SNN-STDP from a top-down perspective, by first mathematically formulating the optimization problem to be solved (based on joint Dictionary Learning and Basis Pursuit) and then deriving the underlying SNN-STDP substrate architecture that provably solves this optimization problem through its neural and weight dynamics. It is shown that the proposed top-down approach significantly outperforms the prior bottom-up works in terms of accuracy for common event camera benchmark datasets.

Fourthly, after the validation of the proposed SNN-STDP theory using benchmark tasks, this thesis proposes what is, to the best of our knowledge, the first SLAM system for drones, fusing event camera and radar data using an SNN-STDP architecture that continuously learns to adapt itself to its unknown environment in an unsupervised manner, and without the need for any offline pre-training phase. Instead, the drone starts flying in an unknown indoor warehouse that was not captured in the SNN during any prior offline training phase; during the flight, the SNN extracts features from its sensory input data by learning the structure of the environment on the fly using STDP. Remarkably, it is shown that the proposed system outperforms offline SNN training in terms of SLAM accuracy.

Fifthly, the applicability of the proposed SNN-STDP design framework to other large-scale real-world tasks is explored by studying the continual learning of people detection by drones, using the proposed top-down SNN-STDP theory under a convolutional network setting. Experiments are conducted for comparing the performance of the proposed continual learning system against conventional CNN models trained offline. Here, it is shown that the proposed SNN-STDP system outperforms the use of a same-size CNN, by gaining more than 19% on the peak F_1 score (used as a conventional measure of detection performance).

Finally, in order to pave the way towards the use of continual-learning SNN-STDP for the control of dynamical agents, this thesis explores how continual Hebbian learning networks can be used within an Active Inference (AIF) scheme in order to learn agent control from sensory observations. Experimental studies are conducted using the Mountain Car environment from the OpenAI gym suite, to study the effect of the various Hebbian network parameters on the task performance. It is shown that the proposed Hebbian AIF approach outperforms the use of Q-learning (as popularly used to solve the Mountain Car problem), while not requiring any replay buffer, in contrast to typical reinforcement learning systems.

In this way, this thesis significantly contributes to the algorithmic aspect of SNNs and STDP learning techniques, helping towards a wider adoption of neuromorphic computing for AI applications in the extreme edge.

Beknopte samenvatting

Vandaag de dag is er een steeds grotere vraag naar het inbouwen van geavanceerde intelligentiecapaciteiten in energie- en gebiedsarme apparaten op batterijen, zoals geavanceerde wearables, gezondheidscontrolesystemen op het lichaam, AR / VR-headsets en kleine robots zoals drones. Daarom is er een toename in onderzoek dat zich richt op het gebruik van diepe neurale netwerken (DNN's) en hun versnelling in gespecialiseerde hardware. Deze DNN-oplossingen lopen echter nog steeds ver achter op de opmerkelijke efficiëntie van biologische hersenen, zoals bijvoorbeeld in de honingbij, die in staat is een groot aantal complexe taken uit te voeren terwijl ze slechts 10 μW aan energie verbruiken. Daarom heeft de studie van event-driven Spiking Neural Networks (SNN's) en lokale Hebbiaanse plasticiteitsregels (zoals Spike-Timing-Dependent Plastity of STDP) de afgelopen jaren veel aandacht getrokken als een meer bio-plausibel, neuromorf rekenmodel, dat de biologische neurale mechanismen in de hersenen op een meer getrouwe manier probeert na te bootsen. Dit opkomende AI-gebied, dat Spiking Neuromorphic Computing wordt genoemd, probeert rechtstreeks inspiratie te halen uit de biologie om een zeer nauwkeurige uitvoering van complexe AI-taken mogelijk te maken met de krapste budgetten voor energie en chipruimte, waar conventionele DNN's aanzienlijk meer energie en ruimte nodig zouden hebben.

Op dit moment wordt een massale toepassing van SNN's en lokale STDP voor online leren echter in gevaar gebracht door een gebrek aan hoognauwkeurige, grootschalige demonstraties in de echte wereld op andere taken dan traditionele patroonherkenning met behulp van kleinschalige databases en door het beperkte aantal toepassingen met andere zintuiglijke modaliteiten dan beeldgegevens. Om deze huidige problemen aan te pakken, richt dit proefschrift zich op het algoritmische aspect van SNN's en hun praktische toepassing voor zowel radarverwerking als de navigatie van kleine drones via radar-beeldsensor fusie, en online leerbenaderingen met behulp van STDP. Als zodanig worden er significante bijdragen geleverd aan zowel het theoretische begrip van SNN's en STDP, als aan innovatieve neuromorfische toepassingsparadigma's.

In dit opzicht levert dit proefschrift de volgende bijdragen. Ten eerste wordt het nieuwe gebruik van SNN's met radarsignalen bestudeerd met een diepgaande vergelijking tussen SNN- en DNN-prestaties. Er wordt aangetoond dat SNN-implementaties niet simpelweg vervangen kunnen worden door zwaardere DNN-implementaties. In het bijzonder wordt het cruciale

belang van voorbewerking voor signaal-naar-spike codering aangetoond voor de radargebaartoepassing door te experimenteren met meer dan 10 verschillende coderingsbenaderingen. Aangetoond wordt dat de invloed van de radar voorbewerking op de taaknauwkeurigheid veel belangrijker is bij het gebruik van SNN's dan bij het gebruik van DNN's. Door een geschikte voorbewerking te gebruiken, wordt aangetoond dat een SNN met 4-bits gewichten dezelfde nauwkeurigheid kan bereiken (93%) in vergelijking met een DNN met volledige precisiegewichten.

Ten tweede wordt de ervaring die is opgedaan met radar pre-processing gebruikt om een uniek drone-platform op te zetten met een op het netvlies geïnspireerde, op gebeurtenissen gebaseerde camera en een 79 GHz radarsensor voor het bestuderen van navigatietaken voor drones, zoals SLAM (Simultaneous Localization and Mapping) en detectie van mensen in een magazijncontext, met behulp van een aangepaste radardetector in de MCU aan boord van de radar.

Ten derde biedt dit proefschrift een nieuw theoretisch kader voor de studie van SNN-STDP leersystemen. In tegenstelling tot eerdere werken die SNN-STDP onderzoeken vanuit een bottom-up perspectief, door willekeurig verbonden netwerken op te zetten volgens directe biologische inspiratie, onderzoekt deze dissertatie SNN-STDP vanuit een top-down perspectief, door eerst het op te lossen optimalisatieprobleem wiskundig te formuleren (gebaseerd op Dictionary Learning and Basis Pursuit) en vervolgens de onderliggende SNN-STDP substraatarchitectuur af te leiden die dit optimalisatieprobleem aantoonbaar oplost door middel van de neurale en gewichtsdynamica. Aangetoond wordt dat de voorgestelde top-down benadering significant beter presteert dan eerdere bottom-up werken in termen van nauwkeurigheid voor veelvoorkomende event camera benchmark datasets.

Ten vierde, na de validatie van de voorgestelde SNN-STDP-theorie met behulp van benchmark-taken, stelt dit proefschrift, voor zover wij weten, het eerste SLAM-systeem voor drones voor, waarbij gebeurteniscamera- en radargegevens worden samengevoegd met behulp van een SNN-STDP-architectuur die continu leert om zichzelf aan te passen aan de onbekende omgeving op een onbewaakte manier en zonder de noodzaak van een offline pre-trainingfase. In plaats daarvan begint de drone te vliegen in een onbekend binnenpakhuis dat niet is vastgelegd in het SNN tijdens een voorafgaande offline trainingsfase; tijdens de vlucht haalt het SNN kenmerken uit zijn sensorische gegevens door de structuur van de omgeving tijdens de vlucht te leren met behulp van STDP. Opmerkelijk is dat het voorgestelde systeem beter presteert dan offline SNN-training in termen van SLAM-nauwkeurigheid.

Ten vijfde wordt de toepasbaarheid van het voorgestelde SNN-STDP-ontwerpkader op andere grootschalige echte taken onderzocht door het continu leren van mensdetectie door drones te bestuderen met behulp van de voorgestelde top-down SNN-STDP-theorie in een convolutioneel netwerk. Er worden experimenten uitgevoerd om de prestaties van het voorgestelde systeem voor continu leren te vergelijken met conventionele CNN-modellen die offline zijn getraind. Hieruit blijkt dat het voorgestelde SNN-STDP-systeem beter presteert dan het gebruik van een CNN van dezelfde grootte door een winst van meer dan 19% te behalen op de piekscore van F_1 (gebruikt als een conventionele maat voor detectieprestaties).

Om de weg vrij te maken voor het gebruik van continu lerende SNN-STDP voor de besturing van dynamische agenten, onderzoekt deze dissertatie ten slotte hoe continu lerende Hebbiaanse netwerken kunnen worden gebruikt binnen een Active Inference (AIF) schema om de besturing van agenten te leren op basis van sensorische observaties. Experimentele studies worden uitgevoerd met behulp van de Mountain Car omgeving uit de OpenAI gym suite, om het effect van de verschillende Hebbiaanse netwerkparameters op de taakprestaties te bestuderen. Er wordt aangetoond dat de voorgestelde Hebbian AIF-benadering beter presteert dan het gebruik van Q-learning (zoals populair gebruikt om Mountain Car op te lossen), terwijl er geen replaybuffer nodig is, in tegenstelling tot typische versterkingsleersystemen.

Op deze manier levert deze dissertatie een belangrijke bijdrage aan de algoritmische aspecten van SNN's en STDP-leertechnieken, wat bijdraagt aan een bredere toepassing van neuromorfisch computergebruik voor geavanceerde AI-toepassingen.

List of Abbreviations

AWGN Additive White Gaussian Noise. 54

BPTT Back-Propagation Through Time. 49

CFAR Constant False Alarm Rate. 64

CL Continual Learning. 125, 126

CNN Convolutional Neural Network. 17

CUT Cell Under Test. 70

CW Continuous Wave. 37

DFT Discrete Fourier Transform. 40

DLBP Dictionary Learning and Basis Pursuit. 104

DNN Deep Neural Network. xii, xviii, xxv, 1–5, 8, 13, 14, 17–19, 22, 27, 33–38, 40, 45, 47, 48, 50–56, 62, 63, 126, 143, 147, 154

DVS Dynamic Vision Sensor. 14

FFT Fast Fourier Transform. 57

FMCW Frequency Modulated Continuous Wave. 17

FPS Frames per Second. 93

i.i.d Independent and Identically Distributed. 13

IF Intermediate Frequency. 34

LASSO Least Absolute Shrinkage and Selection Operator. 115

LIF Leaky Integrate and Fire. 4

LSM Liquid State Machine. 8

LSTM Long Short-Term Memory. 35

MAE Mean Absolute Error. 141

MCU Micro Controller Unit. 93

PSC Post-synaptic Current. 4

QBS Quadratic Bayesian Surprise. 131

RB Rate-Based. 48

RD Range-Doppler. 42

RDP Range Doppler Profile. 69

RP Range Profile. 68

SGD Stochastic Gradient Descent. 12

SIMO Single In Multiple Out. 67

SISO Single In Single Out. 36

SLAM Simultaneous Localization and Mapping. 17

SNN Spiking Neural Network. 3

STDP Spike-Timing-Dependent Plasticity. 2

STFT Short-Time Fourier Transform. 42

SVC Support Vector Classification. 130

SVM Support Vector Machine. 120

TS Teacher Signals. 159

TTFS Time-To-First-Spike. 48

UWB Ultra Wide Ban. 141

Contents

Abstract iii

Beknopte samenvatting v

List of Abbreviations x

List of Symbols xi

Contents xi

List of Figures xvii

List of Tables xxv

1 Introduction 1
- 1.1 Neuromorphic computing 1
- 1.2 SNN algorithmic fundamentals 4
  - 1.2.1 Leaky Integrate-and-Fire neuron 4
  - 1.2.2 Training SNNs via backprop using surrogate gradients . 5
  - 1.2.3 Liquid State Machines and STDP learning in SNNs . . 8
  - 1.2.4 Bottom-up vs. top-down neuromorphic approaches . . . 10
- 1.3 Emerging application and technological paradigms for the use of SNNs 11
  - 1.3.1 Continual learning for personalized edge-AI applications 11
  - 1.3.2 Trends in neuromorphic sensing and computation for small drones 14
- 1.4 Open research questions when scaling the applicability of SNNs and STDP learning to radar and drone navigation tasks 18
  - 1.4.1 Q1: What radar pre-processing and event encoding steps are best suited for SNN-based radar signal recognition? 18

1.4.2 Q2: How can we derive a top-down optimization-based theory of SNN-STDP learning? . . . 20
1.4.3 Q3: How can SNN-STDP be used for performing useful drone navigation tasks such as Simultaneous Localization and Mapping? . . . 22
1.4.4 Q4: How can SNN-STDP be further extended for performing other tasks in a continual-learning manner, such as for people detection on drones? . . . 25
1.5 Thesis contributions . . . 26
1.5.1 Investigating the impact of pre-processing for bridging the accuracy gap between SNNs and DNNs in radar signal classification . . . 27
1.5.2 A sensor-fusion drone platform for joint RGB-DVS-radar data acquisition . . . 28
1.5.3 A top-down optimization-based theory of SNN-STDP learning . . . 29
1.5.4 DVS-radar fusion SLAM for drones with continual STDP learning . . . 29
1.5.5 Using SNN-STDP for continually learning to detect people via DVS camera data on drones . . . 30
1.5.6 Active inference in Hebbian learning networks . . . 30
1.6 Conclusion . . . 31

2 Bridging the accuracy gap between SNNs and DNNs via the use of pre-processing for radar applications 33
2.1 Introduction . . . 33
2.2 Related works . . . 35
2.3 Radar data acquisition set-up . . . 38
2.3.1 FMCW radar theory . . . 38
2.3.2 Gesture data set . . . 40
2.4 Selected radar pre-processing approaches . . . 41
2.4.1 μDoppler signature . . . 41
2.4.2 Range-Doppler maps . . . 42
2.4.3 Proposed enhancements . . . 43
2.5 Neural network design and training approach . . . 45
2.5.1 Architectural choices . . . 47
2.5.2 Training approach . . . 48
2.6 Experimental results . . . 51
2.6.1 Discussion on the impact of pre-processing . . . 53
2.6.2 Varying event encoding to optimize the SNN accuracy . 56
2.6.3 Energy consumption and latency estimation . . . 57
2.6.4 Applicability of the proposed SNN on a different dataset 58
2.7 Towards radar gesture recognition with SNN-STDP . . . 59

2.8 Conclusion 62

3 Design of a drone platform for sensor-fusion data acquisition 63
3.1 Introduction to the first part 63
3.2 Related works 65
3.3 Background theory 67
3.3.1 FMCW radar in SIMO mode 67
3.3.2 Background on conventional CFAR detectors 70
3.3.3 State-of-the-art CFAR for multi-target scenarios 71
3.3.4 KRX detection principle 73
3.4 The proposed detector 74
3.5 Experimental performance assessment 78
3.5.1 Experimental setup 78
3.5.2 ROC-based results 80
3.5.3 Result confirmation with ground-truth data 81
3.5.4 Discussion 86
3.5.5 Effect of the D parameter 90
3.5.6 Ablation studies 91
3.5.7 System implementation and hardware metrics 93
3.6 Conclusion to the first part 95
3.7 Introduction to the second part 96
3.8 RGB-DVS-radar sensor suite integration on a drone 97
3.8.1 Background on Dynamic Vision Sensor (DVS) camera . 97
3.8.2 Sensor suite description 97
3.9 Dataset acquisition 98
3.9.1 People detection dataset 98
3.9.2 Dataset for Simultaneous Localization and Mapping (SLAM) 101
3.10 Conclusion to the second part 102

4 A top-down approach to SNN-STDP networks 103
4.1 Introduction 103
4.2 Background theory 105
4.3 SNN-STDP methods 106
4.3.1 High-level view of the proposed SNN-STDP topology . 106
4.3.2 Solving LASSO via the SNN dynamics 109
4.3.3 Learning the network weights via STDP 111
4.3.4 A rate-based STDP model 112
4.3.5 Using STDP for dictionary learning 114
4.4 Network hyperparameter study and methods 116
4.4.1 How to tune the STDP parameters? 116
4.4.2 How to terminate STDP training? 118
4.5 Using the SNN-STDP as feature descriptor for classification tasks 120

4.5.1 Global feature descriptor . . . 120
4.5.2 Action descriptor . . . 121
4.6 Experimental results . . . 121
4.6.1 N-MNIST classification . . . 122
4.6.2 IBM DVS128 gesture classification . . . 123
4.7 Conclusions . . . 124

5 Sensor-fusion SLAM with continual STDP learning 125
5.1 Introduction . . . 125
5.2 Continual learning to encode vision . . . 128
5.2.1 What are the challenges posed by continual learning? . 128
5.2.2 Proposed method: the Quadratic Bayesian Surprise . . 129
5.2.3 Effect of QBS on DLBP . . . 133
5.2.4 Applying QBS to the SNN-STDP network . . . 134
5.3 Forming the SLAM system . . . 135
5.3.1 RGB SLAM using DLBP . . . 135
5.3.2 DVS-radar fusion SLAM using SNN-STDP . . . 136
5.3.3 Radar-gyroscope odometry approach . . . 138
5.3.4 Radar-based obstacle aggregation . . . 139
5.4 Experiments . . . 140
5.4.1 Experimental setup . . . 140
5.4.2 Results with RGB and DLBP . . . 141
5.5 Results with DVS-radar fusion via SNN-STDP . . . 145
5.5.1 Normal lighting conditions . . . 146
5.5.2 Under strong lighting variations . . . 149
5.5.3 Comparing pre-trained SNN-STDP vs. continual-learning SNN-STDP . . . 150
5.6 Considerations regarding the safety of the SLAM system . . . 152
5.7 Conclusions . . . 154

6 Continual-learning people detection from DVS data 155
6.1 Introduction . . . 155
6.2 Network architecture . . . 157
6.2.1 SNN-STDP in a convolutional setting . . . 157
6.2.2 SNN-readout system overview . . . 158
6.2.3 Semi-supervised learning in SNN-STDP . . . 159
6.2.4 Anti-Hebbian STDP for reducing over-fitting . . . 162
6.3 Experimental setup and results . . . 163
6.3.1 Continual-learning strategy . . . 164
6.3.2 Post-processing for performance evaluation . . . 164
6.3.3 System assessment . . . 164
6.3.4 Impact of the focal loss parameter γ . . . 166
6.3.5 Impact of unsupervised and supervised strengths λ_u, λ_s 166

6.3.6 Continual SNN-STDP vs. CNN with the same memory size 168
6.4 Conclusions . . . 169

7 Active inference in Hebbian learning networks 171
7.1 Introduction . . . 171
7.2 Background theory on Hebbian learning networks . . . 173
7.3 Active inference in Hebbian learning networks . . . 174
7.3.1 Minimizing the Variational Free Energy . . . 175
7.3.2 Minimizing the Expected Free Energy . . . 178
7.4 Experimental results . . . 179
7.4.1 Mountain car environment . . . 179
7.4.2 Impact of the number of neurons in the posterior and state transition networks . . . 181
7.4.3 Impact of the sparsity of the output activity in the posterior and state transition networks . . . 182
7.4.4 Impact of the time-lag buffer length on task performance 184
7.4.5 Comparing Hebbian active inference against the use of a replay buffer and against Q-learning . . . 184
7.5 Conclusions . . . 185

8 Conclusions and future work 187
8.1 Main contributions . . . 187
8.2 Suggestions for future work . . . 190
8.3 Perspective on continual SNN-STDP learning . . . 193
8.4 Closing remarks . . . 195

A Using SNN-STDP for radar gesture recognition 197

B Baseline CNN design for people detection on drones 201
B.1 Input pre-processing . . . 201
B.2 CNN design . . . 205
B.3 Choice of the loss function . . . 205
B.3.1 Training approach: a curriculum learning strategy . . . 206
B.4 Experimental results . . . 208
B.4.1 Precision-recall analysis . . . 208
B.4.2 Ablation studies . . . 209
B.5 Conclusion . . . 211

C Enhancements added to the RatSLAM back-end 213
C.1 Graph-based map correction . . . 213
C.1.1 New node registration . . . 213
C.1.2 Loop closure detection . . . 214
C.1.3 Graph correction . . . 216

C.1.4 Data association and edge information 216
C.2 Conclusion 217

Bibliography **219**

Curriculum Vitae **247**

List of publications **249**

List of Figures

1.1 *Conceptual illustration of the operation of the LIF neuron. In this siplified example, the LIF neuron is connected to a single spiking source s_{in} via a unit weight.* 5
1.2 *Example of Gaussian (1.4) and Laplacian (1.5) surrogate derivatives, approximating the ill-defined spike derivative from (1.1).* 6
1.3 *Conceptual illustration of STDP learning. The weight W_{ij} is modified according to the double exponential STDP rule (1.6) with Long Term Depression (LTD) and Potentiation (LTP) regions.* 9
1.4 *Conceptual illustration of an Liquid State Machine (LSM). Two populations of neurons (excitatory and inhibitory) are randomly initialized with lateral competition between excitatory neurons. Input spikes are connected to the excitatory neurons via the input weights subject to STDP plasticity. An output readout classifier can be trained (e.g., linear regression) to classify the LSM activity.* 10
1.5 *Conventional offline training with a shuffled training set available a priori. Once the network is deployed for inference, no continual model adaptation is provided anymore, potentially leading to poor generalization if the test data are not well captured in the training set.* 12
1.6 *Continual learning scenario where a neural network (e.g., using an SNN-STDP architecture) continuously refine its weights using data points received in their natural, real-time order (without buffering and shuffling past observations in memory). At the same time, the neural network is performing inference at the edge.* 13

1.7 *A selection of neuromorphic drone platforms (among other works). a) the drone platform presented in this thesis equipped with a DAVIS 346 event camera and TI radar. b) The drone platform in [160] used for event camera object detection and tracking. c) A stereo event camera mounted on a drone for obstacle avoidance [57]. d) A drone equipped with a Loihi SNN chip for RGB optical flow-based landing control [50].* 15
1.8 *Conceptual comparison between the output of a standard RGB camera and a DVS event-based camera. As opposed to the standard camera which outputs frames in a periodic manner, the DVS camera outputs a stream of asynchronous spikes corresponding to the region in the image where change is occurring due to the black dot (figure adapted from [196]).* 16
1.9 *Outline of the contributions of the thesis.* 27

2.1 *Frequency-Modulated Continuous Wave (FMCW) radar sensing pipeline summarizing the steps from (2.1) to (2.4). For the sake of simplicity, a single reflector is assumed in this example.* . . . 39
2.2 *Gesture acquisition set-up.* 41
2.3 *Example patch for the vertical gesture, using $\Delta\mu Doppler$.* . . . 44
2.4 *Example patch for the vertical gesture, using ΔRD (Range-Doppler).* 45
2.5 *SNN architecture used. Each pixel of the input radar maps is first encoded into a spike train of length T_{inf}, either through RB or TTFS coding. Each spiking map slice corresponding to each time step is then fed one by one to the network, and the I&F neurons change state according to their self recurrence (as denoted by the black recurrence arrows). Output spikes emitted by the σ_3 layer are accumulated during the T_{inf} time steps and the final accumulation vector A is transformed through SoftMax to class probabilities.* 46
2.6 *Corresponding DNN architecture. The network retains a similar topology to Fig. 2.4 but with I&F neurons replaced by ReLU and the output of the σ_2 layer directly fed to the SoftMax non-linearity, as conversion from the event domain to the frame domain by accumulation is not needed.* 47
2.7 *Confusion matrix for the 60-GHz radar gesture recognition dataset of [261]. The recognition accuracy of the 4-bit-weight SNN is* $94.6\% \pm 2\%$. 60
2.8 *High-level description of a generic SNN architecture for unsupervised personalization via STDP learning. In the SNN ensemble, the blue and orange neural ensembles denote the excitatory and inhibitory neurons.* 61

3.1 *A typical range-angle profile (RAP) from radar data captured by a flying drone in an indoor warehouse environment (see Fig. 3.3). The peaks indicate reflecting obstacles.* 69
3.2 *A typical range profile being filtered by a CFAR detector. The CUT is the cell S in the middle, the training cells form a vector Z[x] and the remaining bins (in gray) are the guard cells.* . . . 70
3.3 *Our drone setup with the 79-GHz Texas Instruments radar.* . . 79
3.4 *An illustrative example RP obtained when the drone is moving towards the racks.* 79
3.5 *Effect of applying our detector to the RP of Fig. 3.4. The detection is performed as in (3.21) by thresholding* $||\Phi(\bar{x}) - \bar{N}_{\Phi}^{m}||_{\infty}$ *(purple plot) against* $T_2 = 0.95$*. The detections are shown in red. The D parameter was set to* 15 *in this example. Some peaks in Fig. 3.4 are not detected leading to false negatives, and some peaks in Fig. 3.4 are detected multiple times (due to the peak tail), leading to false positives.* 80
3.6 *ROC curves for our proposed detector, for different combinations of* N_{tc} *and* N_g*.* $D = 15$*.* 82
3.7 *ROC curves for our method compared to OS-CFAR with* $k = 0.5$*. The average gain is 40% on* P_D *compared to OS-CFAR for* $P_{FA} < 1\%$*.* 82
3.8 *ROC curves for our method compared to OS-CFAR with* $k = 0.7$*. The average gain is 19% on* P_D *compared to OS-CFAR for* $P_{FA} < 1\%$*.* 83
3.9 *ROC curves for our method compared to OS-CFAR with* $k = 0.9$*. The average gain is 21% on* P_D *compared to OS-CFAR for* $P_{FA} < 1\%$*.* 83
3.10 *ROC curves for our method compared to OR-CFAR with* $\gamma = 1.3$*. The average gain is 20% on* P_D *compared to CHA-CFAR for* $P_{FA} < 1\%$*.* 84
3.11 *ROC curves for our method compared to TS-LNCFAR with* $\gamma = 1.8$*. The average gain is 35% on* P_D *compared to CHA-CFAR for* $P_{FA} < 1\%$*.* 84
3.12 *ROC curves for our method compared to CHA-CFAR with* $m = 0.65 \times N_{tc}$*. The average gain is 16% on* P_D *compared to CHA-CFAR for* $P_{FA} < 1\%$*.* 85
3.13 *Ground-truth and radar acquisition. The two people walking in an empty room hold their indoor GPS beacon on their head during the radar acquisition.* 86
3.14 *ROC curves for our method compared to OS-CFAR with* $k = 0.7$*. The average gain is 19 % on* P_D *compared to OS-CFAR for* P_{FA} *values smaller than 1%.* 87

3.15 *ROC curves for our method compared to CHA-CFAR with* $m = 0.4 \times N_{tc}$. *CHA-CFAR achieves a better performance compared to OS-CFAR (Fig. 3.14) but is still outperformed by our proposed method by an average gain of* 14%. 88
3.16 *Log-normal fitting of the measured PDF. The KS distance is* 0.048. 88
3.17 *Exponential fitting of the measured PDF. The KS distance is* 0.056. 89
3.18 *ROC curves of the proposed detector for different D values. The diamond-shaped points denote the* (P_D, P_{FA}) *achieved in the limit of* $T_2 \to 1$ *(which are the limit values).* 90
3.19 *ROC curves for different methods (k = 0.7 for OS-CFAR). Our method uses (3.17). The higher ROC performance attained by the proposed detector confirms the use of the heuristic rule (3.17).* 92
3.20 *ROC curves for different methods (k = 0.7 for OS-CFAR). Our method uses the* L_∞ *norm. The higher ROC performance attained by the proposed detector confirms the use of the* L_∞ *norm.* . . . 92
3.21 *Radar-camera snapshot of our drone flying through indoor industrial aisles with racks on each side. The right and left shelves can easily be identified in the detection point clouds (in blue).* 94
3.22 *Conceptual illustration of a DVS event-based camera.* 98
3.23 *Our drone setup with the sensor suite mounted on it, as indicated with the arrows.* 99
3.24 *Snippet from the f-aisles-1 acquisition. The static background has a non-zero Doppler velocity with regard to the drone as the drone is flying forward. Detections from the right and left shelves are thus returned and can be used for obstacle avoidance.* 100
3.25 *An example drone flight path in the indoor warehouse environment where we collect data for the SLAM experiments.* 101

4.1 *High-level view of the SNN-STDP architecture.* 107
4.2 *A push-pull pair of LIF neurons approximates* $Prox_{\lambda_1||.||_1}$ *(4.2).* . 111
4.3 *Synaptic modification due to STDP vs. the rate-only weight change model (4.23).* 113
4.4 *Spectra of the DVS data and STDP kernels. The non-matching STDP kernel strongly attenuates the information-bearing bandwidth of the DVS spectra, leading to the learning of noisier weights (see Fig. 4.5), while this is not the case for the matching kernel.* 117
4.5 *SNN weight subset learnt on N-MNIST: a) matching STDP kernel leads to crisp atoms; b) non-matching STDP kernel leads to noisy atoms.* 118
4.6 *Evolution of the inner loss function during SNN-STDP training on the IBM DVS128 gesture dataset.* 120

4.7 *Subset of the learned SNN weights. a) N-MNIST, b) IBM gesture.* 122

5.1 *High-level continual-learning SLAM architecture.* 127
5.2 *Effect of the QBS on the DLBP performance and stability.* . . . 134
5.3 *Block diagram of the continual learning (CL) DLBP-based RGB SLAM system, with QBS adaptation ($\mathcal{S}_2$).* 136
5.4 *DVS-radar fusion SLAM using continual SNN-STDP networks.* 137
5.5 Trajectory result of raw radar-gyro odometry and obstacle aggregation. 140
5.6 *Sensor fusion drone setup.* 141
5.7 *Map of the warehouse and drone flight paths, with pictures taken at each marked locations.* 142
5.8 *SLAM result for flight 1, blue path in Fig. 5.7.* 144
5.9 *SLAM result for flight 2, red path in Fig. 5.7.* 145
5.10 *SLAM result for flight 3, both red and blue paths in Fig. 5.7.* . 146
5.11 *SLAM results for flight a (dimensions in meter); the obstacle modelling of section 5.3.4 is not plotted for clarity.* 148
5.12 *SLAM results for flight b.* 148
5.13 *SLAM result for flight c (strong lighting variations). As opposed to the previously-proposed SLAM setups, our proposed system is robust to the strong lighting variations as it can better recover the map of the room.* 149
5.14 *Pre-training vs. continual learning for SLAM on flight b) of section 5.5. The continual-learning setup clearly outperforms the pre-trained method in terms of mapping quality.* 150
5.15 *Pre-training vs. continual learning for SLAM on flight c) of section 5.5. The continual-learning setup clearly outperforms the pre-trained method in terms of mapping quality.* 151
5.16 *The drone can follow multiple different paths from its starting position to its end goal. Each path comes with a trade-off between hardware and navigational overheads versus safety costs. Each trade-off can be illustrated as a Pareto curve. The dynamic scenario method exploits the correlations inside the complex dynamic system that it controls by clustering the different paths as distinct scenarios.* 153
5.17 *During drone navigation, the lower and upper bounds on the drone safety are estimated by the dynamic scenario algorithm. As the drone gets closer to e.g., a human being, these bounds become tighter, which pushes the drone to have more emphasis on safety by trading related hardware and navigational overheads.* 154

6.1 *SNN-STDP architecture for continually learning to produce attention maps for people detection.* 156

6.2 *Sweeping the SNN-STDP of Chapter 4 across the image plane in order to produce a convolutional output map.* 158
6.3 *Convolutional SNN-STDP and readout architecture for studying continual SNN-STDP learning with the availability of teacher signals (TS).* 159
6.4 *The standard positive (1.6) vs. the negative (6.16) STDP rules as a function of the difference between the post- and the pre-synaptic spike times* $\tau_{ij} = t_{post,i} - t_{pre,j}$. 162
6.5 *Attention maps inferred from the DVS data by our SNN-STDP system. The label bounding boxes are also shown.* 165
6.6 *Impact of the focal loss parameter* γ *on the precision-recall curves. The peak* F_1 *score is maximized near* $\gamma = 0.5$. 167
6.7 *Impact of the semi-supervised hyper-parameters* $\lambda_{u,s}$ *on the precision-recall curves. The peak* F_1 *score is maximized near* $\lambda_u = 0.2$ *and* $\lambda_s = 0.8$. 167
6.8 *Cluttered scene with no humans. a)* $\lambda_u = 0$ *(negative STDP not used) and* $\lambda_s = 1$*, b)* $\lambda_u = 0.2$ *(negative STDP used) and* $\lambda_s = 0.8$. 168
6.9 *Detection precision vs. recall for the presented SNN-STDP and the conventional CNN with offline training. Our proposed method outperforms the CNN by more than* 19% *on the peak* F_1 *score.* . 169

7.1 *Baseline Hebbian network architecture used in this work. The dynamics of the network follow (7.4) and minimize (7.3), given subsequent input vectors* o*. Each layer possesses its own weight matrix* Φ, Ψ *which evolve through Hebbian plasticity (*$\Psi \sim \Phi^T$ *in (7.4), as an independent, local set of weights).* 174
7.2 *Hebbian Active Inference Architecture. a) The state-transition network takes as input the* L_{buf} *previous latent states produced by the posterior network, and projects them onto its internal representation* c_S *via the learned* Φ_S*. When producing roll-outs, the state-transition network estimates the next state* $\hat{s}_l$ *by re-projecting the output activity* c_S *due to* $[s_{l-L_{buf}}, \ldots, s_{l-1}]$ *back onto the input space via (7.10). b) The posterior network takes observation-action pairs as input and produces latent states* s *(corresponding to the output in Fig. 7.1). In addition to the Hebbian mechanisms depicted in Fig. 7.1, the weights* Φ_P *of the posterior network are also subject to a top-down Hebbian learning mechanism for minimizing the second term in (7.9 b).* 177
7.3 *Impact on the success rate when changing the number of neurons* M_Q *in the posterior network.* 181
7.4 *Impact on the success rate when changing the number of neurons* M_P *in the state transition network.* 182

7.5 *Impact on the success rate when changing the sparsity hyper-parameter λ_Q in the posterior network.* 183
7.6 *a) Impact on the success rate when changing the sparsity λ_P in the state transition network. b) Success rate when changing the time-lag buffer length L_{buf}.* 183
7.7 *Hebbian AIF versus the use of a replay buffer and Q-learning (a). Q-learning needs two orders of magnitude more episodes in order to converge (b).* 185

A.1 *Converting the μDoppler patches to a spike train as input to the SNN-STDP network.* 198
A.2 *Radar gesture recognition accuracy and standard deviation as a function of the number of coding neurons.* 199

B.1 *Our CNN architecture with cross-fusion highways. The RGB, DVS and radar inputs are concatenated and fed to the input of the SqueezeNet backbone. In addition, the DVS and radar inputs are concatenated and fed to the input of the cross-fusion highway. During training, the network learns at which scale fusion must happen the most along the backbone. Finally, two MLP heads are used for bounding box regression and for human presence detection.* 202
B.2 *Radar columns obtained by projecting the range-azimuth radar detections on the image plane by homography (star-shaped markers) and extending them in height to cover their spatial context.* 204
B.3 *Precision-recall curves of the CNN detector with an IoU threshold of* 0.5. 209
B.4 *Ablation studies showing precision-recall curves of a) cross-fusion vs. early-fusion of RGB, DVS and radar modalities, and b) fusion vs. single modality networks.* 210

List of Tables

2.1 *Dataset content used in the gesture recognition demonstration experiments.* 41
2.2 *Network input sizes.* 47
2.3 *Impact of the radar data pre-processing on the SNN classification accuracy compared to the DNN classification accuracy. For the SNN, the $N_q = 5$ last epochs are trained with quantized weights. RB coding is used throughout the table. The results clearly motivate the need for using a well-suited pre-processing method in the SNN case.* 52
2.4 *Impact of event-domain encoding on the SNN classification accuracy. Pre-processing is done with $\Delta\mu$Doppler with band-limiting, max_{48} coding (exception made for encoding 8), and normalization on the whole map.* 53

3.1 *Parameters of the used 79-GHz radar.* 78
3.2 *Measured average inter-frame processing time margin. The radar frame rate is set to 10 FPS.* 93
3.3 *Radar parameters: θ_{res} is the AoA resolution, d_{res} is the range resolution, d_{max} is the maximum range that can be sensed, v_{min} is the Doppler velocity threshold, v_{res} is the velocity resolution, and v_{max} is the maximum velocity that can be sensed.* 99
3.4 *Hover (h) and fly (f) sets with 6 different walking subjects (30 acquisitions in total).* 100
3.5 *SLAM flight acquisitions.* 102

4.1 *Learning and network parameters.* 123
4.2 *N-MNIST performance.* 123
4.3 *IBM DVS128 gesture performance.* 123

5.1 *Mean Absolute Errors of Localization and Mapping: MAE_L | MAE_M (the lower the better).* 143
5.2 *SNN-STDP network parameters.* 146
5.3 *MAE_L (Localization, L) and MAE_M (Mapping, M). The lower the value the better.* 147

6.1 *SNN-STDP learning parameters. The simulation time step is* 0.005 *s. The LIF decay τ_m is in seconds. λ_1 in (6.1) is implicitly set by the LIF threshold μ (see section 4.3.2).* 166
6.2 *Architectural parameters. $H \times W$: input size. $k_H \times k_W$: SNN convolutional kernel size, d: SNN stride, M: number of SNN kernels (neurons in the coding layer), $r_H \times r_W \times M$: kernel size of the readout. $D_H \times D_W$: shape of the output attention map. The readout has a stride of 1 and zero padding.* 166

Chapter 1

Introduction

1.1 Neuromorphic computing

In recent years, the world has witnessed the advent of a new technological revolution: the ubiquity of Artificial Intelligence (AI) in our everyday lives. In particular, the use of Deep Neural Networks (DNNs) has enabled applications that were thought to be out of reach a few decades ago. Many teams in both academia and industry have been exploring the use of DNNs in diverse domains, from computer vision to natural language processing. The success of DNNs has mainly been fueled by the remarkable performance of the back-propagation of error (backprop) algorithm and the massive deployment of GPU-enabled servers in the cloud, necessitating large facilities that burn huge amounts of electrical power [249].

This comes in striking contrast with the recent surge in the demand for energy- and area-efficient, low-latency AI computing operating close to or inside the sensor front-ends in a decentralized manner (i.e., not depending on cloud servers). This computing paradigm has been coined as *(extreme) edge AI* and constitutes a crucial ingredient in the design of the next-generation IoT devices, human-machine interfaces (e.g., speech recognition), wearables such as Augmented-Virtual Reality (AR-VR) headsets, personalized healthcare, and small robotics applications such as micro drones [10, 50, 106, 160, 257, 266]. Many of these devices are battery-powered and are cost sensitive. Hence the tight constraints on energy consumption and on the size or volume of the computing unit. The market forecast for edge-AI devices in general is expected to reach more than 70 billion U.S. dollars in 2032 [256]. This provides strong motivation

for the study and design of edge-AI devices.

Within this context, many teams in both academia and industry have been exploring the design of hardware accelerators for running DNNs in significantly more efficient manners compared to GPUs, for use in resource-constrained applications [18, 38, 106, 162–164, 257]. Using these DNN accelerators, a gain of $\sim$ 2 orders of magnitude more compute throughput per watt has been demonstrated compared to GPUs [249]. Still, it has been remarked that these DNN solutions come in huge contrast with the methods of cognition found in biology (i.e., the brain). For example, the honey bee is capable of remarkable perception abilities and extremely complex navigation capability while using only about one million neurons and consuming around 10 μW of power. Needless to say that the current DNNs running in the most advanced edge AI accelerators are still far away from the remarkable performance of the honey bee's brain from a hardware-algorithmic complexity perspective [65]. In contrast to DNNs, a promising approach towards reaching biological-level efficiency lies in the advent of neuromorphic (or bio-plausible) spiking neural networks (SNNs) using potentially different algorithmic paradigms to DNNs (instead of replicating DNN behavior with SNNs [54] which leads to limited energy gains), with recent hardware demonstrations showing a gain of more than one order of magnitude in power consumption compared to both binary neural networks (BNNs) and DNNs [67].

Indeed, to help uncover roads toward bio-level cognition, the field of **neuromorphic computing** is currently attracting huge attention as it explicitly proposes to study neural networks under a more faithful bio-plausible point of view. Unlike conventional DNNs, a biologically-plausible neural architecture complies with the following characteristics [223]:

- Learning must strongly rely on unsupervised and *continual* local learning rules (i.e., using the local information of each neuron to learn in real-time streams of sensory data), such as via the *Spike-Timing-Dependent Plasticity* (STDP) rule as empirically observed in the brain [252] (which modifies neural weights according to the time difference between input and output spikes), instead of conventional supervised offline (i.e., non-continual) training via backprop as used in DNNs.

- Neurons must communicate using binary *action potentials* (also called *spikes*) as opposed to continuous activation functions as used in DNNs, since spikes are actually observed in the brain as a more efficient event encoding of temporal data.

- The inter-neuron communication must be event-driven in the sense that neurons must only push data when significant information needs to be

propagated at that point in time (vs. frame-based in DNNs) [53].

- Although not the focus in this thesis, an additional meta-characteristic of biologically-plausible neural networks is their support for multiple time scales, from short- and mid-term to long-term memory (see the recent European MemScales project[1] for a comprehensive study [105]).

This has led to the study of Spiking Neural Networks (SNNs) as a new generation of bio-plausible, event-driven neural networks that can be implemented in asynchronous and massively parallel non-Von Neumann compute architectures, where computation is done close to the memory [165], solving also the memory bottleneck issues. Furthermore, SNNs have also been implemented and successfully demonstrated in highly innovative emerging technologies such as analog sub-threshold accelerator hardware and memristor computing devices that closely attempt to replicate plasticity learning rules found in biology [41, 86, 101, 147, 257].

Currently, however, a landslide adoption of SNNs for edge AI has been jeopardized by a lack of high-accuracy, large-scale real-world demonstrations on tasks other than traditional pattern recognition using simpler datasets such as MNIST [69, 113]. Indeed, thanks to availability of user-friendly frameworks such as PyTorch and TensorFlow [3, 190], DNNs have already been applied with great success in a plethora of challenging applications with many different types of sensory settings, achieving high algorithmic performance (such as e.g., perception accuracy). In contrast, SNNs and neuromorphic concepts have until now been demonstrated only on a smaller amount of diverse and challenging use cases, and have often been shown to have a much harder time to reach DNN-level accuracy on similar tasks.

In particular, key neuromorphic concepts, such as providing competition between excitatory and inhibitory neurons (as in Liquid State Machines [151]), *homeostasis* regularization [130] and STDP learning (which is believed to enable low-power learning at the edge) have rarely been applied in applications other than simple digit-like image recognition problems, often reaching a significantly lower accuracy compared to SNNs trained via backprop [113]. Hence, this lack of greater *algorithm-level* success has been a major thorn in the side of the neuromorphic computing paradigm.

To help alleviate these issues, this thesis focuses on the algorithm-level aspect of SNNs and their practical application in the *autonomous navigation* domain, by *i)* significantly contributing to the theoretical understanding of bio-plausible learning techniques such as STDP, and *ii)* providing many real-world

[1] https://memscales.eu/

demonstrations of neuromorphic approaches in novel sensing and application domains such as radar processing and drone navigation tasks.

Next, the fundamentals behind SNNs algorithmic concepts will be introduced. These are needed as basic building blocks for the description of our contributions in the next chapters of this thesis.

1.2 SNN algorithmic fundamentals

This section covers the background elements needed for the design of the various SNNs in this thesis (note that these background elements *per se* are not the main focus of the contributions presented in this thesis).

1.2.1 Leaky Integrate-and-Fire neuron

In contrast to traditional DNNs, SNNs make use of *spiking* neurons as non-linearity, often modelled by a Leaky Integrate-and-Fire (LIF) activation [54] (see Fig. 1.1):

$$\begin{cases} \frac{dV}{dt} = \frac{1}{\tau_m}(J_{in} - V) \\ \sigma = 1 \textbf{ if } V \geq \mu \textbf{ else } 0 \\ V(t+dt) = 0 \textbf{ if } V \geq \mu \end{cases} \tag{1.1}$$

with J_{in} the input current to the neuron (1.2), σ the spiking output, V the membrane potential, τ_m the time constant governing the membrane potential decay, dt the simulation time step, and μ the neuron threshold [53]. The scalar input current J_{in} is continuously integrated in V following (1.1). When V crosses the firing threshold μ, the membrane potential is reset back to zero and an output spike is emitted. The input current J_{in} is obtained by filtering the inner product of the neural weights and the spiking inputs through a *post-synaptic current* (PSC) kernel [53] (estimating the *instantaneous* spiking rate):

$$J_{in} = \mathcal{PSC}\{\bar{\phi}^T \bar{s}_{in}(t)\} \tag{1.2}$$

with $\bar{s}_{in}(t)$ the input spiking vector (originating from e.g. an event camera or other spiking neurons), $\bar{\phi}$ the weight vector and:

$$\mathcal{PSC}\{x(t)\} = x(t) * \frac{1}{\tau_s} e^{-t/\tau_s} \tag{1.3}$$

the effect of PSC filtering with time constant τ_s. Fig. 1.1 conceptually illustrates the LIF neuron behavior. The LIF neuron is connected to a pre-synaptic spiking input s_{in}.

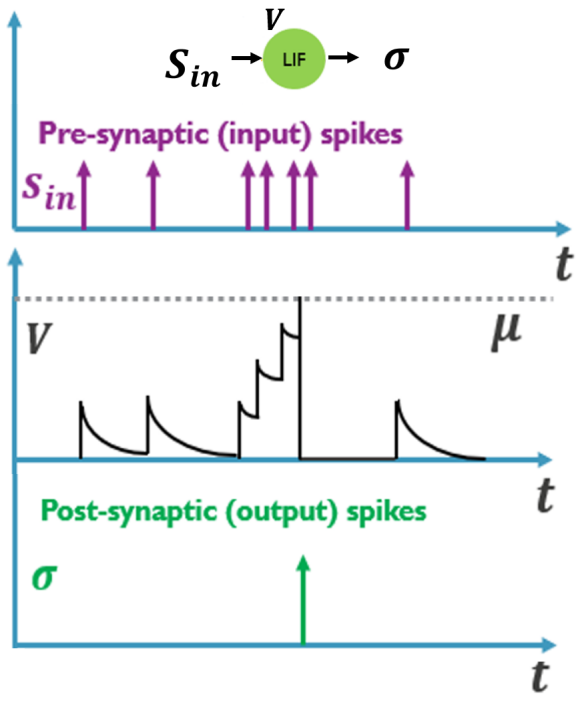

Figure 1.1: *Conceptual illustration of the operation of the LIF neuron. In this siplified example, the LIF neuron is connected to a single spiking source s_{in} via a unit weight.*

Using the LIF neuron model described above, dynamical neural networks working along the time domain can be designed using a similar connectionist approach as in conventional DNNs and their weights can be learned in order to perform useful tasks.

1.2.2 Training SNNs via backprop using surrogate gradients

For a long time, it was unclear how SNNs could be trained via the standard backprop algorithm, due to the non-differentiability of the output spikes with regard to the input of spiking neurons (such as in the LIF neuron model of (1.1)). Indeed, at first glance, the Dirac pulses generated in SNNs have an ill-defined (infinite) derivative *at the moment of the spike time* and a zero-valued derivative elsewhere in time, which jeopardizes a direct application of backprop training as in DNNs.

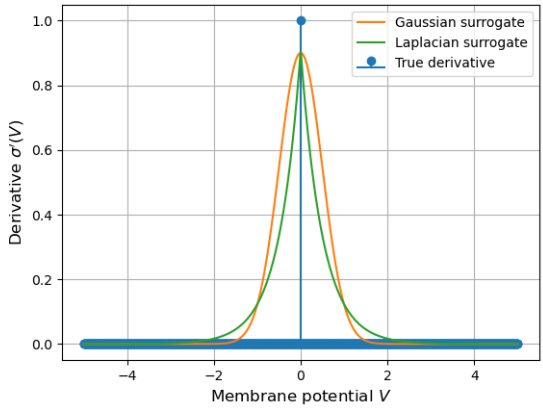

Figure 1.2: *Example of Gaussian (1.4) and Laplacian (1.5) surrogate derivatives, approximating the ill-defined spike derivative from (1.1).*

More recently, the use of *surrogate gradients* has constituted a major milestone for training SNNs and has enabled the use of backprop training even in the presence of the non-differentiable spiking non-linearity [54]. The surrogate gradient technique proposes to approximate the ill-defined derivative of the Dirac spike as a function of the neuron membrane potential (V in (1.1)) as a smooth, symmetric, decaying function such as a Gaussian kernel [54]:

$$\sigma'_g(V) \approx \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{V^2}{2\sigma^2}} \tag{1.4}$$

or other similarly behaving surrogates, such as a Laplacian kernel [224]:

$$\sigma'_l(V) \approx \frac{1}{\beta} e^{-\frac{|V|}{\beta}} \tag{1.5}$$

where the parameters σ and β set the precision of the surrogate functions approximating the Dirac spike derivative (see Fig. 1.2). The use of symmetric decaying surrogate functions, such as (1.4) and (1.5), is motivated by the fact that in the limit of $\sigma \to 0$ and $\beta \to 0$, the surrogate functions converge to a Dirac pulse of infinite value, effectively corresponding to the derivative of the spiking non-linearity.

With these surrogate approximations, automatic differentiation packages, such as the pyTorch library [189], can be used to train arbitrary networks of spiking neurons by defining a custom neuron model following (1.1) in forward

propagation, and using either (1.4) or (1.5) (or other similar variants) in the backward pass (see Source Code 1.1).

A number of popular python libraries such as *snnTorch* [54] have explicitly been proposed for democratizing the design of SNNs trained via surrogate gradients and backprop, and are currently enabling and facilitating a wider adoption of SNNs within the edge-AI research community.

But even though training through surrogate gradient backprop has been helpful for increasing the accuracy of SNNs, its use is still limited to *offline* training scenarios and is not well suited for continual on-chip learning in extreme edge devices, due to the significant complexity and memory access overheads of using backprop [68].

Therefore, to enable continual on-chip learning in resource-constrained devices, the use of local plasticity learning rules such as STDP has attracted much attention [42, 69, 281]. It will be reviewed in the next section.

Source Code 1.1: *LIF neuron with Gaussian surrogate gradient.*

```
sigma = 0.5 #Surrogate gradient precision parameter
decay_neu = 0.9 #Membrane potential decay constant
#Define the surrogate firing function
class Surrogate_ActFun(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input): #input=membrane-threshold
        ctx.save_for_backward(input)
        return input.gt(0).float() #if firing, returns 1

    @staticmethod
    def backward(ctx, grad_output): #Gaussian surrogate
        input, = ctx.saved_tensors
        grad_input = grad_output.clone()
        temp = np.exp((-(input) ** 2 / (2 * sigma ** 2)))
               / (np.sqrt(2 * np.pi) * sigma)
        return grad_input * temp.float()

#Define Leaky Integrate and Fire neuron
def Leaky_Integrate_and_Fire(inputs, mem, thr):
    mem = decay_neu*mem + inputs #Leaky membrane potential
    inputs_ = mem - thr #Centralize with regard to threshold
    spike = Surrogate_ActFun.apply(inputs_) #Pass through custom
```

```
                                    #activation with surrogate gradient
    mem = mem*(1-spike) #Set membrane to zero if spiked
    negative_ = (mem < -thr).float() #Clamp membrane to zero
                                                #if going negative
    mem = (mem*(1-negative_)) - (thr*negative_)
    return mem, spike
```

1.2.3 Liquid State Machines and STDP learning in SNNs

Even though back-propagation training using surrogate gradients has shown remarkable success in conventional GPU-based supervised training settings [207], its use is still ill-suited for ultra-low-power edge learning as it suffers from the same compute-expensive error assignments issues faced in standard DNN back-propagation [70]. In addition, the surrogate gradient method is non-local (i.e., error information must be propagated from the network head to each layer) and hence, biologically implausible, making it ill-suited for the study and modelling of cortical neural ensembles. In addition, the non-local aspect of backprop is more costly in terms of memory and compute complexity compared to local learning approaches [68]. Therefore, the study of Spike-Timing-Dependent Plasticity (STDP) is actively pursued by the SNN community as a biologically plausible and local *Hebbian* learning rule [94], avoiding the high memory-access-expensive problem of backprop [68] by *locally* modifying the weights of each neuron following the difference τ_{ij} between the post- and pre-synaptic spike times [24]:

$$W_{ij} \leftarrow \begin{cases} W_{ij} + A_+ e^{-\tau_{ij}/\tau_+}, & \text{if } \tau_{ij} \geq 0 \\ W_{ij} - A_- e^{\tau_{ij}/\tau_-}, & \text{if } \tau_{ij} < 0 \end{cases} \tag{1.6}$$

with A_+, A_- the potentiation and depression weights, τ_+, τ_- the potentiation and depression decay constants, W_{ij} the jth element of the ith neuron weight vector $\bar{W}$, and τ_{ij} the time difference between the post- and the pre-synaptic spike times across the jth synapse of neuron i (see Fig. 1.3) [24].

Using (1.6), the design of most SNN-STDP systems, such as typical *Liquid State Machines* (LSMs) equipped with STDP [151], is highly empirical and typically consists of randomly initializing a network topology (or *reservoir*) composed of excitatory and inhibitory neurons (i.e., *positive* and *negative* contribution to the post-synaptic input currents) [151]. The network topology is randomly initialised in a way such that competition is created between the excitatory neurons by preventing inhibitory neurons to input the same excitatory neurons that they receive as inputs. Then, STDP learning can be optionally added to the input weights connecting the input spike trains to the excitatory neurons [42]

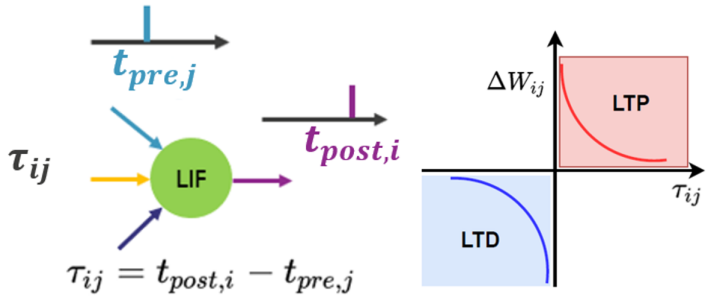

Figure 1.3: *Conceptual illustration of STDP learning. The weight W_{ij} is modified according to the double exponential STDP rule (1.6) with Long Term Depression (LTD) and Potentiation (LTP) regions.*

in order to learn a more efficient set of input weights compared to the use of purely random weights in the original LSM formulation [151]. For the sake of completeness, it is important to note that LSMs are an upgrade of the non-spiking *reservoir computing* concept [219] towards the spiking domain, with the original LSM formulation [150] not benefiting from STDP learning but solely relying on random weights. Fig. 1.4 conceptually illustrates the LSM-STDP approach.

LSMs equipped with STDP are therefore composed of a large number of parameters (neuron threshold, time constants, connection probabilities, STDP parameters and so on), which are typically hand-tuned through a lengthy and unsystematic process [42]. Since their topology is random, it is unclear what precise loss function these networks are in fact minimizing through their neural and synaptic dynamics, adding additional uncertainty to their behavior. Also, since the link between LSM-like networks and the specific loss function is unclear, usual cross-validation techniques cannot directly be used to detect when over-fitting starts to occur in order to stop training, potentially degrading the performance. Finally, it can be remarked when analyzing the STDP learning rule (1.6) that nothing prevents the weights W_{ij} to be updated continuously and to grow indefinitely. Indeed, as long as a neuron subject to STDP is spiking and receiving input spikes, the STDP rule (1.6) will keep on potentiating and depressing the synapses, leading to run-away behaviors, as commonly known in STDP research [36] (i.e., the norm of the weight vectors will grow indefinitely). But even though STDP has been observed physically in the brain, there has been no clear evidence of run-away behaviors, which adds even more uncertainties to our understanding of brain-like plasticity rules.

In order to alleviate this run-away phenomenon, weight normalization schemes

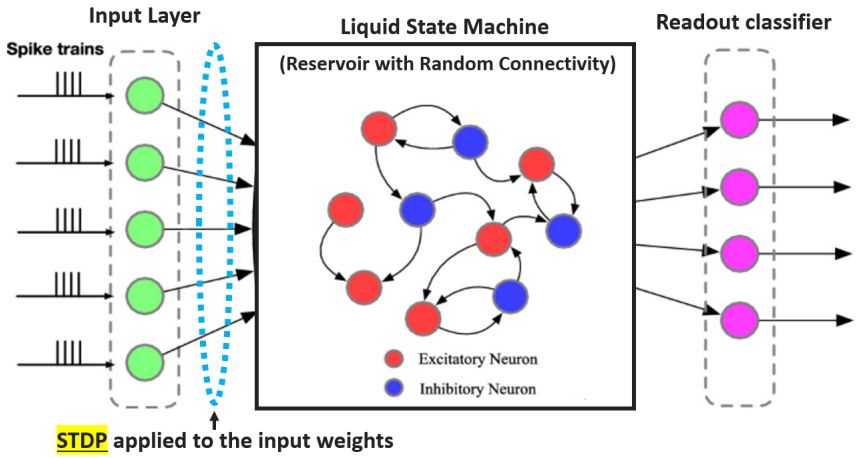

Figure 1.4: *Conceptual illustration of an Liquid State Machine (LSM). Two populations of neurons (excitatory and inhibitory) are randomly initialized with lateral competition between excitatory neurons. Input spikes are connected to the excitatory neurons via the input weights subject to STDP plasticity. An output readout classifier can be trained (e.g., linear regression) to classify the LSM activity.*

such as (1.7) have been proposed, where the weight vectors $\bar{W}$ are constantly normalized to a user-specified l_2 norm α [130]:

$$\bar{W} \leftarrow \alpha \frac{\bar{W}}{||\bar{W}||_2} \tag{1.7}$$

But from a functional point of view, it is unclear whether all weights must exhibit the same norm as in (1.7) after the learning process, since (1.7) could potentially be an additional constraint degrading accuracy is some applications. Even more, using the normalization rule (1.7) makes the process non-local and identifying the proper α value in (1.7) adds yet another hand-tuning parameter, adding additional difficulty to the neural network design in practice.

1.2.4 Bottom-up vs. top-down neuromorphic approaches

The difficulties currently faced when training STDP-based systems have jeopardized a wider adoption of the STDP learning rule in SNNs, lagging far behind the use of surrogate gradient back-propagation due to significant

gaps in reported performances and task complexity [239]. This issue is tightly linked to an important distinction that has been drawn by Frenkel et al. [66] in the design of neuromorphic techniques, between *bottom-up* and *top-down* approaches.

In general, a neuromorphic technique is said to be *bottom-up* when its design attempts at mimicking the underlying biological evidence *a priori* in order to perform some computational task *a posteriori*. Currently, most STDP-based approaches, such as LSMs, fall into this bottom-up category. Indeed, these networks are heuristically designed using direct inspiration from *what we believe is taking place in the biology*. Hence, this approach is rather *qualitative* than *quantitative*: the goal is to *qualitatively* replicate biological behavior, *hoping* that it will lead to a successful computational outcome [66].

In contrast, this thesis crucially argues that the shortcomings of current STDP systems, as discussed in section 1.2.3 are mostly due to the lack of a holistic *top-down*, optimisation-based theoretical framework, linking the neural and synaptic dynamics of SNN-STDP systems to the minimization of a certain unsupervised loss function, chosen wisely so as to model the aforementioned dynamics. This point will be developed further in section 1.4.2 and chapter 4, where SNN-STDP systems will be studied from a top-down point of view, as *substrate of computations* that solve clearly defined optimization problems through their neural and synaptic dynamics.

1.3 Emerging application and technological paradigms for the use of SNNs

This section provides an overview of two application domains for SNNs: *a)* continual learning at the edge, and *b)* neuromorphic computing for small drones. Both have been gaining significant interest in recent years. These two emerging research applications for neuromorphic computing will serve as the main drivers throughout this thesis, where novel continual learning SNN techniques will be developed and demonstrated on drone navigation tasks.

1.3.1 Continual learning for personalized edge-AI applications

A promising approach for the conception of online-learning edge-AI applications, such as personalized healthcare, AR-VR, wearables and so on, lies in the design of SNN-STDP models that can continuously learn and personalize themselves to the environment in which they are operating, as opposed to conventional

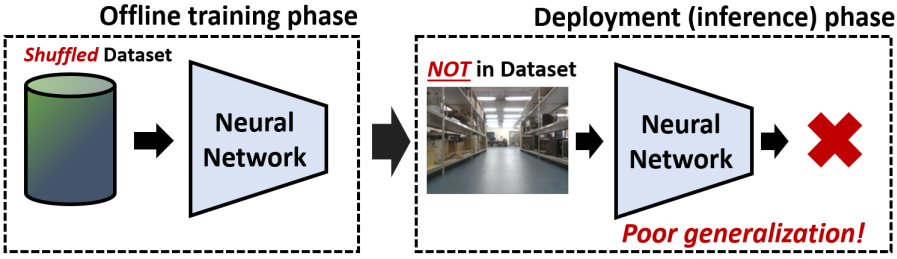

Figure 1.5: *Conventional offline training with a shuffled training set available a priori. Once the network is deployed for inference, no continual model adaptation is provided anymore, potentially leading to poor generalization if the test data are not well captured in the training set.*

offline training (see Fig. 1.5). Indeed, since local plasticity rules such as STDP are significantly less compute-expensive compared to backprop learning, the use of SNN-STDP networks has been identified as a sound technology for continual on-device learning [42, 179], with a growing number of hardware accelerators being proposed for on-chip learning at the extreme edge [35, 67, 69, 265].

Currently, however, most demonstrations of SNN-STDP limit themselves to simple use cases such as small pattern recognition [69] and MNIST-like tasks [142]. Therefore, studying how continual learning can be done *in more complex* edge applications via SNN-STDP networks is of paramount importance, as it will potentially enable a whole new paradigm of sensing and IoT applications, where the underlying AI models do not need to send any data to the cloud for offline pre-training phases (preserving privacy, security, reducing power consumption, etc.) and seamlessly personalize to their working environment, avoiding model generalization issues found in conventional offline training methods (see Fig. 1.5). This thesis will experimentally demonstrate this point in chapter 5.

In contrast to the conventional *offline* training of neural networks with training sets captured beforehand and *shuffled* before being fed to the networks, continual learning is concerned with the more challenging scenario whereby a neural network must learn while being fed with a continuous stream of data presented in its natural, real-time order (see Fig. 1.6).

To illustrate this, let us consider a toy example where a neural network must learn to distinguish between class A and class B. To achieve this, the typical method for training the network would consist of first *shuffling* the training set containing both classes and then using this *shuffled* dataset for training the network using a mini-batch stochastic gradient descent (SGD) approach. In contrast, let us now consider the continual learning case, where the network

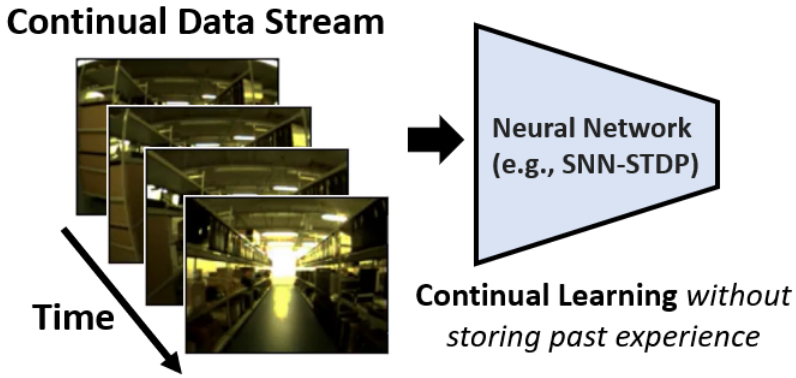

Figure 1.6: *Continual learning scenario where a neural network (e.g., using an SNN-STDP architecture) continuously refine its weights using data points received in their natural, real-time order (without buffering and shuffling past observations in memory). At the same time, the neural network is performing inference at the edge.*

must learn while receiving the class A and class B data in a *non-shuffled* order, for example, by feeding all the class A data first, followed by all the class B data during training. It is then clear that learning does not take place efficiently as the network will keep on over-fitting, alternatingly on e.g., class A when all the class A data are shown, followed by an over-fitting on class B when all the class B data are shown, effectively *forgetting* what it learned using the class A images. This phenomenon is famously referred to as the *catastrophic forgetting* problem and constitutes a major challenge for continual learning systems [45].

The non-shuffled data streams found in continual-learning applications are often referred to as *non-stationary* data streams [46] because it cannot be assumed that the prior distribution Γ from which data points are sampled during learning stays the same throughout the learning process. Indeed, when a dataset is shuffled, data points end up being sampled in an independent and identically distributed (i.i.d) manner, as opposed to continuous streams, where subsequent data points $\bar{o}_k$ and $\bar{o}_{k+1}$ are strongly correlated through time k. In the i.i.d case of a shuffled data source, the *expected value* $\mathbb{E}$ of the stochastic gradients used for example by SGD during DNN learning correctly converges to the gradient of the *full* loss function $\mathcal{L} = \frac{1}{N_d}\sum_{k=1}^{N_d} l(\bar{o}_k)$ computed on all the N_d data points:

$$\mathbb{E}[\frac{\partial l(\bar{o}_k)}{\partial W}] \rightarrow \frac{\partial \frac{1}{N_d}\sum_{k=1}^{N_d} l(\bar{o}_k)}{\partial W} \tag{1.8}$$

where W denotes the network weights and l denotes the energy or loss to be minimized (e.g., mean square error, cross-entropy,...).

On the other hand, (1.8) does not hold anymore in the continual-learning case, where the data points are sampled from a non-stationary source (i.e., a prior distribution Γ that changes during the learning process). Since (1.8) does not hold, learning is conducted *outside* the assumptions made when using SGD-style techniques and hence learning becomes jeopardized. This discussion does not only hold for SGD, but also for other *greedy descent* learning techniques like the standard STDP rule (1.6), which also assume data source stationarity (see [22] for a description of STDP under the light of SGD).

To help alleviate catastrophic forgetting using conventional backprop-based DNNs, weight regularization techniques such as Elastic Weight Consolidation (EWC) and Memory Aware Synapse (MAS) have been proposed [7, 121]. These attempt to identify the importance of each weight in the DNN in order to keep the important weights fixed and to only allow significant modification of the weights that are currently judged to be less important when the continual learning process takes place. More closely related to SNNs, Aljundi et al. [8] have shown that providing activation sparsity in DNNs using lateral inhibition between the neurons of each layer (as in an SNN-STDP system) has a beneficial effect on continual-learning performance, usually leading to a significant boost in network's accuracy after learning. Following this observation on the importance of sparsity for continual learning, this thesis will derive an SNN-STDP design framework based on sparse coding theory [185] in chapter 4. The proposed sparse-coding-based SNN-STDP framework will then be applied in continual-learning settings in chapters 5 and 6.

1.3.2 Trends in neuromorphic sensing and computation for small drones

Recently, there has been a considerable interest from many teams in applying neuromorphic sensing and computation techniques to small drone navigation tasks (see Fig. 1.7). This application choice is motivated by the fact that small drones are highly constrained in terms of power consumption. They cannot lift large payloads, prohibiting the use of large-capacity batteries, which highly limits the overall power budget. This makes the use of neuromorphic technologies attractive. Indeed, neuromorphic sensors such as Dynamic Vision Sensors (DVS) (also called event-based cameras), coupled with neuromorphic computing techniques such as SNNs, come with the promise of significantly reducing data bandwidth, computing energy, latency and area utilization.

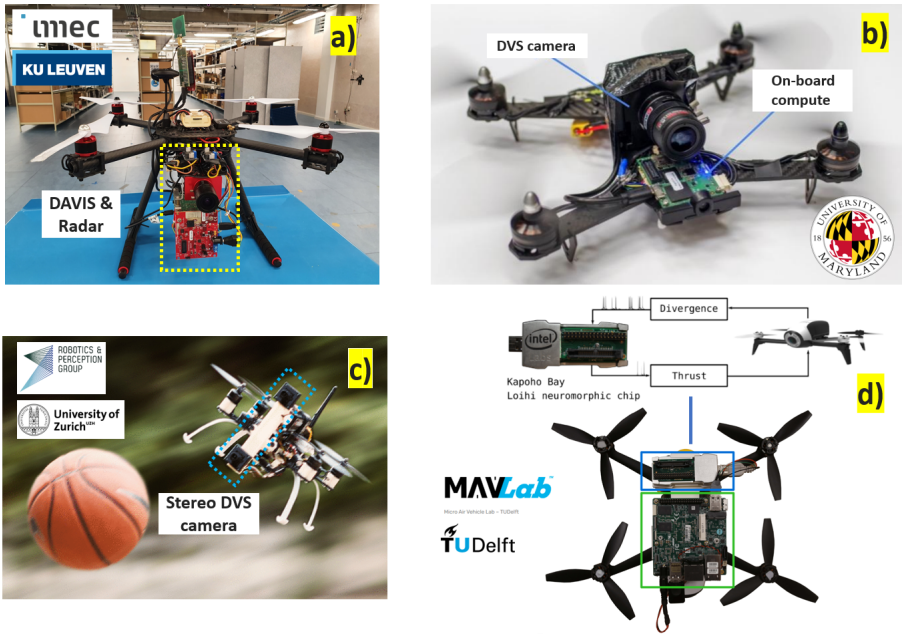

Figure 1.7: *A selection of neuromorphic drone platforms (among other works). a) the drone platform presented in this thesis equipped with a DAVIS 346 event camera and TI radar. b) The drone platform in [160] used for event camera object detection and tracking. c) A stereo event camera mounted on a drone for obstacle avoidance [57]. d) A drone equipped with a Loihi SNN chip for RGB optical flow-based landing control [50].*

Therefore, many teams are currently exploring the use of DVS cameras and SNNs for small drones [57, 160, 254]. DVS cameras are a novel type of imaging sensor specially well suited for low-power and low-latency applications such as in drone navigation contexts. Indeed, compared to standard RGB imaging shutters, a DVS camera outputs spikes for each independent pixel $\vec{x}_k$ in an asynchronous manner whenever the change in log-intensity at time t_k, $\Delta L(\vec{x}_k, t_k)$, crosses a certain threshold C [75].

It can be shown that events are generated by moving edges [75] (since edges have a high contrast $|\Delta L(\vec{x}_k, t_k)|$). Fig. 1.8 shows how DVS data convey spikes corresponding to areas on the image plane that are subject to movement only, significantly reducing the data bandwidth, which in turn, reduces the memory consumption and processing power overheads. In addition, DVS cameras are about two orders of magnitude faster than standard RGB cameras since they

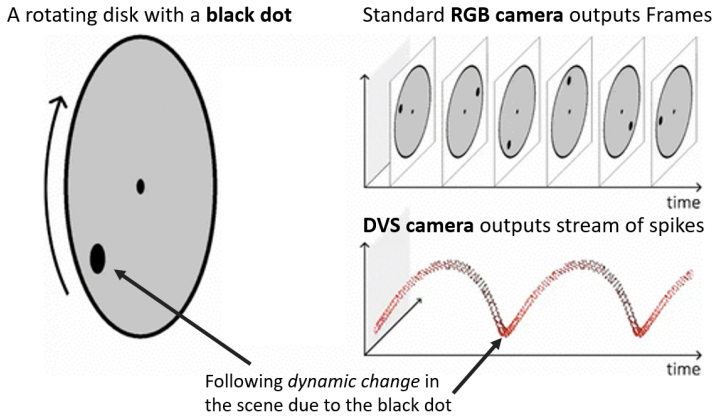

Figure 1.8: *Conceptual comparison between the output of a standard RGB camera and a DVS event-based camera. As opposed to the standard camera which outputs frames in a periodic manner, the DVS camera outputs a stream of asynchronous spikes corresponding to the region in the image where change is occurring due to the black dot (figure adapted from [196]).*

can generate asynchronous events with a time resolution as low as 1 μs, making them well suited for the navigation of highly dynamical quadcopters, which have smaller mechanical time constants compared to larger aerial vehicles with more inertia. Furthermore, DVS cameras provide a significantly higher dynamic range than standard cameras ($\sim$ 140dB vs. $\sim$ 60dB) [75], making them better suited for low-light navigation conditions.

Due to their complementary features to standard RGB shutters, DVS and RGB cameras have often been used in *sensor fusion* setting, in applications ranging from visual odometry in low-light conditions (thanks to the high dynamic range of DVS) to fast object detection for collision avoidance and video super-resolution exploiting the ultra-fast $\sim$ 1 μs time resolution of DVS cameras [57, 243, 251].

In addition to RGB-DVS, other promising sensor fusion settings such as *RGB-radar* are actively being researched for robot and automotive navigation, albeit not in a neuromorphic context [14, 174]. Indeed, using imaging sensors such as RGB and DVS alone, still exhibits a number of drawbacks. For example, sensing in darkness or *extremely* low light is still difficult; sensing in fog, rain and other harsh environment can still be challenging due to sensor occlusions; and so on. In addition, imaging cameras do not provide ranging information on their own and must be processed through rather compute-expensive depth estimation algorithms in order to provide ranging [90]. For these reasons, a

growing number of works is addressing the use of radar sensing in drone and automotive navigation contexts. For instance, Aziz et al. [14] proposed a road target classification and tracking system using a 79-GHz Frequency Modulated Continuous Wave (FMCW) radar and a standard imaging camera. Nobis et al. [174] proposed a cross-fusion network to classify objects in an automotive context with radar and camera data. Closer to the use case of small drone navigation, Wessendorp et al. [264] proposed an obstacle avoidance system using an FMCW radar mounted on a small drone and running multi-target tracking algorithms for obstacle detection and tracking when the drone navigates in an indoor scenario.

Motivated by the research trends discussed above, this thesis will explore the use of both a DVS camera and radar in drone navigation and sensing contexts. Compared to the prior works mentioned above that predominantly focus on RGB and DVS, this thesis innovates by being amongst the first works that address the novel case of using radar sensing with SNNs, both for performing signal recognition tasks (see chapter 2) and for drone navigation purposes using continual STDP learning (see chapter 5). In addition, this thesis further innovates by proposing first-of-its-kind DVS-Radar fusion algorithms using SNNs for applications such as Simultaneous Localization and Mapping (SLAM) for small drones. Furthermore, compared to most neuromorphic drone research, which mainly either makes use of neuromorphic DVS cameras processed by conventional DNNs, or processed through Spiking CNNs trained *offline* using backprop (like in DNNs) [40, 48], this thesis proposes what is, to the best of our knowledge, the first application of SNNs equipped with *continual* STDP learning for drone perception, where the SNN model is not trained offline and then deployed for inference (as in DNNs), but rather learns continuously its weights while performing inference at the same time (see section 1.3.1). This *continual learning* (or SNN personalization) enables the model to reach a better SLAM performance compared to the offline training case, while not requiring any pre-training phase and hence any training data acquisition step (as will be discussed in detail in chapter 5).

Following these observations, this thesis will explore the use of SNN-STDP in continual-learning scenarios, for building networks that do not require any offline training. Furthermore, since drone navigation has been identified as a well-suited use case for assessing SNNs in more difficult situations, this thesis validates the use of continual-learning SNN-STDP systems for performing challenging drone tasks such as SLAM, without any offline model pre-training.

1.4 Open research questions when scaling the applicability of SNNs and STDP learning to radar and drone navigation tasks

Although a growing number of works is focusing on extending the domain of application of SNNs to novel sensing use cases, and to extend the capability of SNN-STDP learning systems towards more complex real-world scenarios, these domains of research are still relatively understudied compared to the use of SNNs trained via backprop and surrogate gradient for DVS and RGB data processing. The following open research questions should be adressed to advance the state of the art.

1.4.1 Q1: What radar pre-processing and event encoding steps are best suited for SNN-based radar signal recognition?

SNNs have predominantly been used with imaging-type sensor data such as RGB images and event-based camera data. The extension of their use with other types of sensors has been lagging behind compared to DNNs. Indeed, a competitive SNN accuracy compared to the DNN counterpart has mainly been demonstrated on standard benchmarks such as MNIST [97], and it is unclear how SNNs compare to DNNs when used for other applications such as in radar signal recognition tasks.

It is therefore worthwhile to study the application of SNNs to modalities other than camera-type sensors in order to extend the domain of application of SNNs. In particular, studying the application of SNNs to radar signal recognition tasks is highly attractive since radars are often used as *complementary* sensors to imaging data. Indeed, unlike cameras, radars preserve privacy, are independent of lighting conditions and are able to sense even when occluded, e.g. in fog [148].

Existing research

Tsang et al. [242] have proposed an SNN architecture for radar gesture recognition. Their method is applied to two public datasets (Google Soli [261] and Dop-NET [200]) containing *already pre-processed* radar data (vs. raw radar ADC data). But since public datasets containing already pre-processed data were used, Tsang et al. did not provide an in-depth comparison between the use of different radar pre-processing schemes and their impact on the SNN accuracy. Arsalan et al. [12, 13] explored the use of SNNs trained through

surrogate-gradient back-propagation for 60-GHz radar gesture recognition, using range-Doppler pre-processing via both Fast Fourier Transforms (FFTs) [12] and special SNN layers mimicking the operations done by FFT processing [13]. But since both approaches essentially perform range-Doppler pre-processing, a comparison of their approach against other popular radar pre-processing techniques such as μDoppler signatures [250] remains missing. Using range-Doppler pre-processing is also carried out by Gerhards et al. [80], who perform radar-SNN gesture recognition experiments with a custom dataset composed of only three gesture classes. Although Gerhards et al. compare the performance of their SNN solution to a DNN model using Long Short-Term Memory blocks (LSTM), they do not provide comparative studies on the importance of the pre-processing between the SNN and DNN cases. Finally, Stuijt et al. [227] have demonstrated 4-class radar gesture recognition using μDoppler signature pre-processing (instead of range-Doppler), but they did not provide any comparative study of their radar to spike conversion strategy against other pre-processing approaches such as range-Doppler.

Many different strategies for radar pre-processing and data-to-spike conversion can be carried out prior to feeding the radar data into the SNN. However, a thorough comparison between the different radar pre-processing and spike encoding representations has not been carried out, and it is unclear what forms of *radar to spike* conversion are the most well-suited for achieving a high recognition accuracy. Crucially, a comparison on the impact of radar pre-processing schemes between DNNs and the important case of resource-constrained SNNs using few-bit quantized weights has not been investigated, even though sensor to spike encoding is usually a paramount step in any SNN processing application [63].

In contrast to the above previous works, chapter 2 of this thesis will provide an in-depth comparison between more than 10 different radar pre-processing and spike encoding strategies in order to thoroughly assess the impact of the radar data pre-processing on the downstream SNN accuracy. Chapter 2 will also compare the importance of the radar pre-processing on the recognition accuracy between the SNN and DNN cases, by setting up a DNN architecture that replicates as closely as possible the topology of the SNN, and by investigating in which cases pre-processing has the most impact. Furthermore, chapter 3 will further build on top of the experience gathered in chapter 2 in order to set up a drone platform embarking a radar, as well as a spiking event camera that will be used in the subsequent chapters as part of their experimental investigations.

1.4.2 Q2: How can we derive a top-down optimization-based theory of SNN-STDP learning?

As discussed in section 1.2.4 above, most SNN-STDP systems are designed from a *bottom-up* approach, by taking direct inspiration from the underlying biological mechanisms and trying to replicate these mechanisms as SNN models equipped with STDP plasticity. One of the most widely used approaches for building SNN-STDP systems is through the use of LSMs (see Section 1.2.3), where a recurrent reservoir of spiking neurons (both excitatory and inhibitory) is initialized following a random connectivity topology, with random weights. The excitatory neurons contribute positively to the input of downstream neurons and vice versa for the inhibitory neurons. The random connectivity in the network is initialized in a way such that lateral competition is initiated between the excitatory neural population, by avoiding the self-inhibition of the excitatory neurons during initialization (i.e., an inhibitory neuron I_j connected to an excitatory neuron E_i cannot feed back its output back into the excitatory neuron E_i).

Then, STDP plasticity is added on top to the input weights that usually connect the input spike trains to the excitatory neural population. After a highly empirical and non-systematic hyperparameter search, it is expected that the application of STDP to the input synapses will lead to the learning of weights that embed input signal features *complementary* to each other, thanks to the lateral competition between the excitatory neurons.

In recent years, the study of SNN-STDP systems has greatly been motivated by the fact that SNN-STDP may enable a generation of ultra-low-power and -area edge-AI systems [59] integrating unsupervised, on-line learning in the sensor chips themselves, thus reducing the latency and preserving the privacy of the user data. Due to its local nature, STDP learning has therefore been identified as a potential candidate for carrying neural network learning in extreme-edge devices [131] (where the use of backprop is too compute expensive), with many SNN-STDP accelerator chips proposed in the literature [69, 85, 116, 272, 281].

Next, the existing research will be reviewed, followed by a description of some research challenges that still need to be tackled.

Existing research

Liu et al. [142] have proposed the Multiscale Spatio-Temporal (MuST) framework as an SNN-STDP system that tries to capture multi-scale information in both time and space. Their architecture is built using the traditional LSM-

inspired approach involving excitatory and inhibitory connections. The latter are connected in a way such that lateral competition among the excitatory neurons is established. STDP is applied, together with a weight normalization scheme, to ensure that the learning does not diverge (since conventional STDP is notoriously subject to run-away behaviors if weight normalization is not provided [222]). The authors demonstrate a close to 90% recognition accuracy on the N-MNIST dataset, which is $\sim 9\%$ less accurate compared to conventional event-based classification methods and therefore, not competitive [225]. The architecture proposed by Liu et al. is empirically designed, using a *bottom-up* approach, and it is therefore unclear what unsupervised optimization problem their network is solving. Iyer et al. [104] and George et al. [78] have proposed SNN-STDP approaches based on the reservoir computing (LSM) approach for solving the 11-class IBM DVS128 gesture recognition dataset [9]. Here again, their network is not designed from a top-down, optimization theory approach but rather follows in the footsteps of empirically designed LSMs (see section 1.2.4). In addition, they both reach a very low recognition accuracy of 60.37% and 65%, respectively, which highlights the weakness of bottom-up SNN-STDP design approaches. Zhang et al. [279] have proposed a hybrid STDP-backprop system that achieves 84.76% of recognition accuracy on the IBM DVS128 gesture dataset. But their proposed approach uses a mixture of STDP learning and backprop. This use of backprop makes their system unsuited for SNN-STDP learning at the edge and an optimization-based understanding of the effect of the STDP is still unclear.

Khacef et al. [113] have provided what is, to the best of our knowledge, the most recent comprehensive survey on synaptic plasticity in SNNs, but little discussion is given on the underlying theoretical understanding of the STDP mechanism from an optimization-based point of view. This further evidences the fact that a top-down theoretical framework clearly linking the neural and synaptic dynamics of SNN-STDP systems with the minimization of a clear objective function is currently lacking.

Currently, SNN-STDP architectures are less competitive compared to conventional learning methods in terms of accuracy and task complexity:

1. They have been demonstrated on a significantly simpler and smaller number of applications, often reaching inferior inference performances (such as classification accuracy) compared to feature-based and backprop methods [116].

2. Designing SNN-STDP systems currently involves the careful hand-tuning of a large number of hyper-parameters (STDP parameters, neuron thresholds, time constants and so on) [42].

3. It is often unknown which exact *optimization objective* STDP learning is solving, adding uncertainty to the design and behavior [113].

Earlier work [2, 58] shed light on the existence of links between unsupervised SNN-STDP architectures and conventional machine learning methods such as Principal Component Analysis (PCA), dictionary learning and auto-encoders. Inspired by these findings, it is argued in this thesis that the main limiting factor for the design of high-performance SNN-STDP systems could be the lack of a top-down, *optimization-based* theoretical approach that bridges the gap between traditional feature learning and the biologically-inspired, bottom-up SNN-STDP setting. Currently, it is however unclear how to derive such theoretical framework.

Therefore, in contrast to the above previous works, chapter 4 of this thesis will first identify the joint dictionary learning and sparse coding (DLBP) problem [185] (conventionally used in unsupervised learning and LASSO-type sparse coding [43]) as a well-suited energy function for explaining the dynamics of SNN-STDP systems. Then, the derivation of a top-down SNN-STDP design framework will be provided that provably minimizes the unsupervised DLBP optimization problem. Thanks to this top-down point of view, additional rules will be derived to ease the tuning of the SNN-STDP network hyper-parameters. Therefore, the derivations proposed in chapter 4 will show how an SNN-STDP network can be built as a *substrate* that solves the unsupervised DLBP problem through its neural and synaptic dynamics, when input data are fed in the network and STDP is applied to the network weights.

1.4.3 Q3: How can SNN-STDP be used for performing useful drone navigation tasks such as Simultaneous Localization and Mapping?

Most SNN-STDP systems proposed in literature have been tested on classical benchmark-grade tasks such as MNIST-like datasets, but the amount of more complex, real-world applications demonstrated with SNN-STDP systems remains limited. This is currently jeopardizing the *confidence* of the broader AI community in continual SNN-STDP learning solutions, since SNN-STDP systems have enjoyed much less demonstrations in large-scale, real-world applications compared to conventional DNNs. Hence, it is important to expand the domain of application of SNN-STDP networks by applying them to more complex use cases not tackled so far by prior works.

Performing useful drone navigation tasks such as Simultaneous Localization and Mapping (SLAM) constitutes a highly challenging application domain for SNN-

STDP. Indeed, compared to indoor *ground-based* (wheeled) robot navigation, drone navigation adds additional practical sensing and perception challenges due to a *non-constant* sensing altitude, more fuzzy trajectory paths and increased platform vibrations.

Existing research

Tang et al. [235] have proposed an SNN-based mapping system for wheeled robots and have demonstrated mapping in a *simulation* environment consisting of a double T maze. Tang et al. [236] later proposed a real-world demonstration of a *unidimensional* SLAM system composed of a robot equipped with a depth camera. The robot is placed in the middle of a square room and can only rotate around its axis (only one degree of freedom) in order to reconstruct the square shape of the room. But it is currently unclear how the proposed system can be extended to at least two degrees of freedom in order to enable mapping while having a robot with full 2D navigational capabilities. Espino et al. [55] have proposed an SNN-based path planner that makes use of the supervised E-prop learning rule [20] for finding paths of the least cost during robot navigation from a starting position to a desired goal. During training, a grid-like SNN representing the environment and embedding the cost of moving from one place to another is learned. Real-world demonstrations of path planning are provided using a wheeled robot navigating in wasteland using satellite images as input sensory data. Even though useful, their work does not make use of STDP learning and continual personalization, but rather uses the eligibility propagation (E-prop) learning algorithm for backpropagating errors during training. Hussaini et al. [98] have proposed an unsupervised SNN-STDP system for place recognition from RGB images. Even though using an unsupervised STDP rule, their system is trained *offline* following a conventional train-test procedure (vs. continual SNN-STDP personalization on the fly). In addition, their system can currently only perform place recognition and it is therefore unclear how it will behave when integrated into a complete SLAM system. Hines et al. [91] have proposed a *supervised* SNN architecture for place recognition from RGB images. Their SNN is trained via both STDP and a supervised learning rule which enforces each output neuron to respond to only a single place. After the *offline* training phase of their SNN, they plan to use the obtained place recognition system as part of the loop closure mechanism in SLAM applications (not demonstrated yet). Even though they make use of STDP, the use of additional supervised learning mechanisms jeopardizes the deployment of their SNN as a fully continual-learning system that can learn new environments on the fly. This makes the motivation and actual use of STDP unclear in this supervised training scenario, since backprop-based systems have

been shown to outperform STDP in conventional offline supervised training settings [59, 116].

This growing number of works has investigated the use SNNs for robot navigation tasks such as localization, mapping and path planning [55, 98, 235, 236], but they most often use backprop-based learning rules and their variants instead of the more hardware-efficient STDP rule. In addition, these prior works mostly demonstrate their findings in virtual simulation environments which could be considered over-simplistic compared to the real-world case (with exceptions such as [55]). Crucially, none of these prior works addresses the important use of STDP for the continual *personalization* of the underlying SNN models to environments not captured by any prior training dataset. This is a crucial point since in practice, all environments that a robot will visit, are not necessarily known in advance, strongly motivating the capability of **continual SNN-STDP personalization** to new, unknown environments. Currently, it is still unclear how SNN-STDP systems can be leveraged for performing useful drone tasks such as SLAM in complex real-world contexts such as indoor drone navigation.

In contrast to the above previous works, chapter 5 of this thesis will provide what is, to the best of our knowledge, the first SLAM system fusing a DVS event camera and a radar sensor using SNNs that continuously learn to adapt themselves to their environment via STDP, in a fully unsupervised manner and without the need for any prior offline training phase. This use of SNN-STDP in a continual-learning setting strongly deviates from prior learning-based SLAM systems that typically require offline training on a dataset of the working environment captured beforehand. Following a number of neuroscience investigations that discuss the importance of *surprise* for continual learning [187], the continual SNN-STDP setup proposed in this thesis makes use of a newly-formulated surprise-modulation technique which enables the filtering of redundant data during learning, significantly reducing catastrophic forgetting. Indeed, learning on data points that do not add novelty does not help the learning system and could even drive the model into over-fitting. Hence, learning on data points which bring significant novelty to the learning system will help model generalization while reducing the chances of over-fitting on old data. As such, chapter 5 will contribute significantly to the extension of the application domain of SNN-STDP systems towards complex, real-world tasks.

1.4.4 Q4: How can SNN-STDP be further extended for performing other tasks in a continual-learning manner, such as for people detection on drones?

This research question (Q4) directly follows from Q2 and Q3 described in sections 1.4.2 and 1.4.3. Indeed, other than SLAM, people detection and avoidance constitute another useful real-world application for drone navigation [141]. Hence, it is important to study how the SNN-STDP theory adressed in (Q2) can be extended for performing people detection under a continual-learning setting, where the SNN-STDP is not pre-trained a priori but rather initialized randomly before learning to detect people on the fly from spiking DVS camera data, while the drone is exploring the environment. Most if not all research addressing real-world people and object detection cases (vs. MNIST-like problems) have been carried out in an *offline training* context using CNNs [25, 100, 283] or SNNs trained using surrogate-gradient backprop [117, 228]. Hence, it is currently unclear how bio-plausible SNN-STDP systems can continually learn to detect people in an online manner when operating in environments not captured by training data, without any offline pre-training phase.

Existing research

Kim et al. [117] have proposed a spiking version of the *You Only Look Once* (YOLO) CNN for object detection [197]. Their system, composed of a deep spiking CNN (S-CNN), is trained *offline* using surrogate-gradient backprop and it is shown that the proposed approach reaches an accuracy close to the use of a Tiny-YOLO CNN detector [61], while potentially consuming less power when implemented in neuromorphic hardware. Closer to our STDP learning context, Kheradpisheh et al. [114] have proposed a deep spiking CNN equipped with STDP learning for object and image recognition. Their system is trained *offline* by feeding input data from a training set in random order, while allowing STDP plasticity to modify the convolutional weights during the training phase. As such, their proposed system does not work in an online, continual-learning way, but is rather trained in a similar manner as backprop-based SNNs, jeopardizing online SNN personalization. Inspired by [114], Mozafari et al. [166] proposed another spiking CNN equipped with *reward-modulated* STDP, as a *supervised* learning version of the normally unsupervised STDP rule. The authors discuss the fact that reward modulation might help neurons to learn and forget features in a controlled manner, but continual-learning demonstrations of object detection are not provided as their system is learned using a conventional offline training approach. Therefore, it is currently unclear how SNN-STDP systems can be

leveraged for performing people detection on drones in a continual-learning manner.

In order to extend the application domain of continual SNN-STDP learning within the drone navigation context, chapter 6 of this thesis will investigate the applicability of continual-learning SNN-STDP networks to the task of human detection from DVS event camera data captured from a flying drone. In addition, in order to uncover ways towards the use of continual-learning SNN-STDP for the control of dynamic agents, chapter 7 will provide a first investigation on how the theory of joint Dictionary Learning and Basis Pursuit (DLBP), used as the basis of the SNN-STDP developments in chapter 4, can be unified with the theory of Active Inference [71, 158], which describes how bio-plausible agents can perform actions given world observations.

1.5 Thesis contributions

Addressing the open research questions listed in Section 1.4 above, this section will summarize the major contributions made in this thesis. In order to provide further insight, Fig. 1.9 shows the principal domains of contribution of each chapter, amongst the following domains:

- **Algorithm Theory:** Data to spike conversion, SNN-STDP theory, signal processing (radar and DVS pre-processing, detection and so on).
- **Applications and Demonstrations:** Experimental results (benchmarking SNN-based radar processing performance, SNN-STDP accuracy and so on), SNN demonstrations on drone data.
- **Hardware and Drone Platform:** Drone platform integration, data acquisition.

As shown in Fig. 1.9, this thesis mostly focuses on the algorithmic aspect of SNNs while providing demonstrations using real-world applications. This is *complementary* to the extensive body of work done by many SNN hardware design teams [41, 86, 101, 147, 257], which mostly covers the hardware cluster of the diagram in Fig. 1.9. We will now elaborate the contributions in more detail.

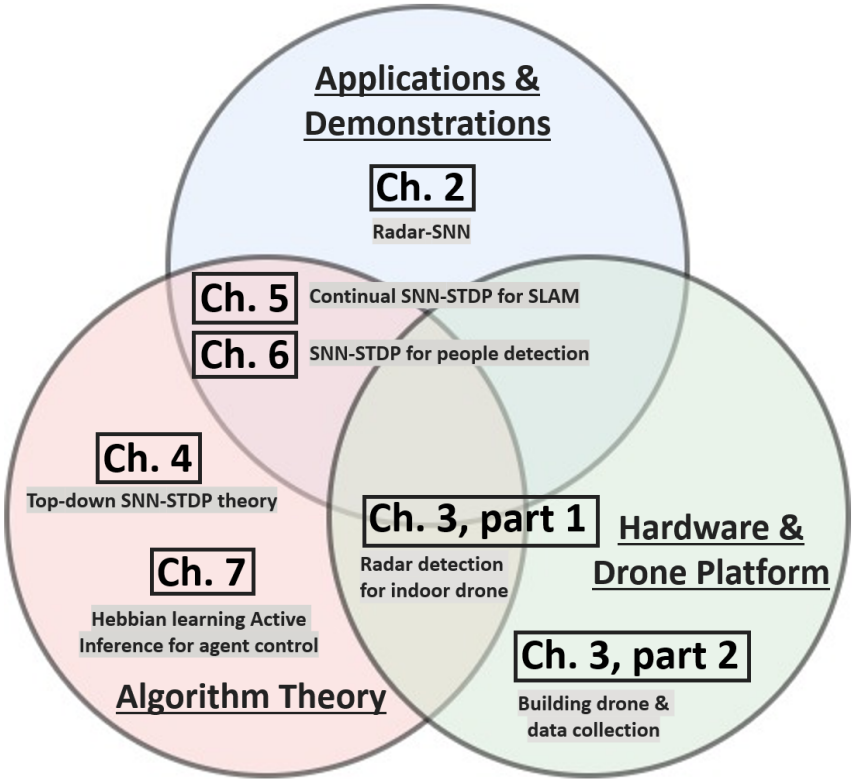

Figure 1.9: *Outline of the contributions of the thesis.*

1.5.1 Investigating the impact of pre-processing for bridging the accuracy gap between SNNs and DNNs in radar signal classification

To enable a wider adoption of SNNs for novel sensory applications of interest other than imaging, a highly detailed analysis on the use of SNNs for radar signal classification will be proposed in **chapter 2**. It will be shown that resource-constrained SNNs using only 4-bit precision weights can indeed reach the accuracy of conventional same-class CNNs that use full-bit weights and ReLU activations, *under the condition that a carefully-designed data pre-processing queue is used prior to feeding the sensory data into the SNN.* In addition, chapter 2 will extensively compare the effect of more than 10 different radar data pre-processing techniques (range-Doppler, μDoppler and their variants) on the SNN and CNN classification accuracy, and will draw important observations

regarding the SNN behavior compared to the CNN case. In addition to radar pre-processing, chapter 2 will also study the effect of signal to spike encoding on the radar gesture recognition accuracy of the proposed SNN architecture. Experiments conducted using imec's low-power 8-GHz FMCW radar chip [143] specially designed for edge applications will be reported. In contrast to the limited amount of pre-processing investigations in prior radar-SNN works, the demonstrations in chapter 2 will provide what is, to the best of our knowledge, the most complete and thorough study of radar signal conditioning and spike encoding for SNNs. Finally, the radar SNN processing experience gathered in chapter 2 will provide technical background throughout the remainder of this thesis for the design of a sensor fusion drone platform using RGB, DVS and radar sensing in chapter 3, as well as for the use of radar data in the design of a DVS-radar SNN-STDP SLAM system proposed in chapter 5.

1.5.2 A sensor-fusion drone platform for joint RGB-DVS-radar data acquisition

Chapters 3 of this thesis will present the various works done towards the integration of a sensor fusion drone platform combining RGB, DVS and radar sensing, in order to investigate the use of SNN-STDP networks for performing useful drone tasks.

In the first part of chapter 3, a novel radar detection algorithm will be presented that outperforms state-of-the-art Constant False Alarm Rate (CFAR) detectors by more than 10% in detection rate when used in indoor drone flight applications. This is achieved by proposing a novel approach for radar detection, which performs detection in a feature space instead of the radar space. The proposed compute-efficient feature mapping of the radar data into a higher-dimensional nonlinear feature space enables a better linear separability between the background clutter and the detection signals, leading to a higher detection rate. Since capturing raw radar ADC data typically requires a large bandwidth through the use of gigabit Ethernet, the novel radar detector presented in chapter 3 is directly implemented in the on-chip micro-controller unit (MCU) of the radar to perform detection at the sensor side and to avoid the transfer of high-bandwidth raw radar ADC data. Doing so, radar detections can be recorded at ~ 30 frames per second (FPS) without causing a bottleneck on the recording bandwidth allocated to the other sensors that are mounted on the drone.

The second part of chapter 3 will present our drone setup, which carries a DAVIS 346 camera providing joint RGB and DVS event camera data, as well as a 79-GHz Frequency Modulated Continuous Wave (FMCW) radar running

our custom detector, connected through USB-UART to a Raspberry Pi 4B. The Raspberry Pi is used to log the sensory data during flights and to collect datasets for both the SLAM and the people detection tasks. Hence, the data collected in the second part of chapter 3 will be used in chapters 5 and 6 to perform SLAM and people detection using SNN-STDP models.

1.5.3 A top-down optimization-based theory of SNN-STDP learning

In order to provide a sound mathematical underpining of SNN-STDP systems, a top-down optimization-based framework will be developed for the design of SNN-STDP networks (as opposed to the bottom-up approach previously used in literature). This top-down framework will shed new light at the SNN-STDP problem and will enable us to look at SNN-STDP networks as *a substrate of computation that provably solves the DLBP optimization problem through its neural and synaptic dynamics*. It will be shown that this top-down formulation presented in **chapter 4** enables a more systematic design and provides novel insights on hyper-parameter tuning, which greatly improves both the accuracy and the design time of SNN-STDP systems compared to prior art. In chapters 5 and 6, this SNN-STDP design framework developed in chapter 4 will be utilized in the subsequent chapters for SLAM and people detection applications on drone.

1.5.4 DVS-radar fusion SLAM for drones with continual STDP learning

To address the lack of more complex, real-world demonstrations of SNN-STDP networks with continual learning and personalization, a first-of-its-kind SLAM system will be presented in **chapter 5**, which makes use of multiple SNN-STDP ensembles to fuse DVS camera and radar data, as part of the loop closure mechanism of the SLAM system. Crucially, the SNN-STDP SLAM system that will be proposed in chapter 5 works in a continual-learning manner. The SNN-STDP networks are initialized randomly and learn to extract features on the fly as the drone is navigating and performing SLAM. As such, there is no need for any offline pre-training phase since the networks continually personalize themselves on the sensory data captured from the environment. Experiments conducted in a warehouse environment will show that the proposed system compares favorably to previously-proposed learning-based SLAM systems.

1.5.5 Using SNN-STDP for continually learning to detect people via DVS camera data on drones

To further extend the application domain of continual SNN-STDP learning in our drone navigation context, **chapter 6** proposes another complex, real-world application of continual SNN-STDP for the task of people detection from DVS camera data. The SNN-STDP continual learning process works as follows: a drone flies in an environment where human subjects are walking and the SNN-STDP system must learn to detect where the walking human subject is to be found on the DVS event camera data frame. Experiments are conducted for comparing the performance of the proposed SNN-STDP system to the performance of a conventional CNN model trained offline, with a similar memory footprint as the SNN-STDP network. Together with the SLAM application tackled in chapter 5, this demonstration of human detection on drones via SNN-STDP constitute what is, to the best of our knowledge, amongst the most challenging real-world applications of continual SNN-STDP learning reported so far.

1.5.6 Active inference in Hebbian learning networks

Finally, in order to pave the way towards the use of continual-learning SNN-STDP for the control of dynamical agents, **chapter 7** studies how Hebbian networks, designed following the joint Dictionary Learning and Basis Pursuit (DLBP) algorithm, can be unified with the theory of Active Inference (AIF). AIF is a bio-plausible theoretical framework describing how intelligent agents can learn to take action from their environmental observations. A generative model capturing the environment dynamics is learned by a network composed of two distinct Hebbian learning networks: a *posterior* network, which infers latent states given the observations, and a *state transition* network, which predicts the next expected latent state given current state-action pairs. Experimental studies are conducted using the Mountain Car environment from the OpenAI gym suite, to study the effect of the various Hebbian network parameters on the task performance. It is shown that the proposed Hebbian AIF approach outperforms the use of Q-learning (as popularly used to solve Mountain Car), while *not requiring* any replay buffer, in contrast to typical reinforcement learning systems. Since the Hebbian learning networks used here have been modelled using the same DLBP framework used in chapter 4 to form the SNN-STDP architectures, the investigations of chapter 7 open the door to the advent of continual-learning and unsupervised SNN-STDP for the control of dynamical agents.

1.6 Conclusion

This chapter has provided a general introduction to the research topics that will be addressed in this thesis, focusing on the algorithm-level aspect of SNNs and STDP learning with applications to radar sensing and drone navigation. This chapter has also provided the SNN algorithmic fundamentals and the description of the various research questions that will be studied in this thesis. The next chapter will address the research question (Q1) about what radar pre-processing and event encoding steps are best suited for SNN-based radar signal recognition.

Chapter 2

Bridging the accuracy gap between SNNs and DNNs via the use of pre-processing for radar applications

2.1 Introduction

In the past decade, radar sensing via neural networks has gained huge interest because of the advantages that radar sensors present over vision-based technologies. Unlike cameras, radars preserve privacy, are independent of lighting conditions and are able to sense even when occluded, e.g. in fog [148]. DNNs have successfully been used for radar processing in a wide range of applications, from indoor person identification [247] and heart-rate estimation [267] to target classification [118] and gesture recognition [261]. As described in chapter 1, those DNN-based solutions are however ill-suited for embedded AI applications at the extreme edge (such as in IoT devices), where the current trend is to embed neural network accelerators within the sensor chip

Adapted from: *A. Safa et al., "Improving the Accuracy of Spiking Neural Networks for Radar Gesture Recognition Through Preprocessing," in IEEE Transactions on Neural Networks and Learning Systems, June 2023.*

itself and where the very tight size and energy budgets generally cannot cope with the latency requirements, memory size and computing power needed to run such deep learning solutions, despite the remarkable recent efforts of many teams in DNN acceleration [37, 163]. In an effort to bring radar-based sensing to the ultra-low-power edge devices, we will investigate in this chapter the use of SNNs for radar sensing and gesture recognition, and explore the design challenges that must be faced to make them effective compared to conventional DNNs.

Indeed, a competitive SNN accuracy compared to the DNN counterpart has mainly been demonstrated on standard vision benchmarks [97], and it is unclear how SNNs compare to DNNs when used for other applications such as radar gesture recognition. More specifically, we will show in this chapter that SNN processing of the sensory data requires the use of carefully crafted pre-processing, but there has been little discussion so far about the impact of such data pre-processing (such as *denoising*, *sparse coding* or *dimensionality reduction*) on the SNN performance compared to any impact of pre-processing on the DNN performance. DNNs have shown remarkable success when applied to unprocessed data sets corrupted by non-stationary noise [26], with experimental evidence suggesting that learning with noisy data might even be beneficial [171]. It is unclear if and how those observations generalize to SNNs.

Within this context, we will demonstrate the crucial importance of adding pre-processing to a resource-constrained SNN for the task of gesture recognition in an extreme-edge device, using a custom *imec* 8-GHz ultra-low-power Frequency Modulated Continuous Wave (FMCW) SISO radar [143] especially designed for edge applications, and we will compare the importance of the pre-processing between the SNN and the DNN case. The resulting SNN architecture with pre-processing achieves DNN-level accuracy while using LIF neurons with 4-bit quantized weights.

Therefore, the major contributions of this chapter are the following:

1. **Algorithm:** We will demonstrate the importance of radar data pre-processing for SNNs compared to its importance in the DNN context. We will demonstrate that conclusions drawn from DNN design cannot be extended to the SNN counterpart, even if a similar training methodology is used (backpropagation of error in this case, to keep the comparison between the SNN and DNN as close as possible).

2. **Radar Signal Processing:** We will introduce multiple methods for pre-processing and encoding radar *Intermediate Frequency* (IF) signals into the event domain (spike trains) and will extensively compare the different pre-processing approaches.

3. **Experiments:** Our experimental study will demonstrate a resource-constrained SNN architecture with one convolutional layer and two fully-connected hidden layers, using 4-bit weights, that achieves DNN-level 93% of recognition accuracy within only 4 spike time steps.

This chapter is organized as follows. Related works will be discussed in section 2.2. The radar data acquisition setup will be introduced in section 2.3. The different radar pre-processing approaches studied will be detailed in section 2.4. The neural network architectures used throughout this chapter will be introduced in section 2.5. Results and their discussion will be presented in section 2.6. Finally, conclusions will be provided in section 2.8.

2.2 Related works

Radar gesture recognition is actively being explored using conventional DNN solutions [229, 231, 261, 280]. Using Google's Soli sensor [135], 11-class gesture recognition with 87% accuracy has been demonstrated in [261]. The authors use a network of *rectified linear units* (ReLU) neurons composed of multiple convolutional and max-pooling layers, followed by multiple fully-connected layers and ending with a recurrent layer using *Long Short-Term Memory* blocks. As input to the network, they perform range-Doppler processing on the radar signals and feed their magnitude as images. Compared to our work, they are able to recognize more gesture classes at the expense of a much higher network complexity (three 3×3 convolutional layers with 32, 64 and 128 filters, respectively; two fully-connected layers both with 512 neurons; an LSTM layer of size 512; and a fully-connected SoftMax layer of 11 outputs). In addition, the authors point out that their remarkable ability to recognize 11 classes is also due to their use of a 60-GHz radar [135, 261], providing a close to 8 times finer grain Doppler resolution compared to our 8-GHz radar chip (as Doppler resolution is inversely proportional to the radar frequency [250]). On the other hand, their 60-GHz radar consumes around 300 mW of power [135] compared to 680 μW for our 8-GHz radar [143], which makes our sensor much better suited for an ultra-low-power applications, such as in IoT edge devices. Also, they do not quantize the weights nor the input to make their network compatible with applications requiring an ultra-low power consumption. Therefore, even though better suited for solutions that can make use of GPU compute power, their solution is unsuited for ultra-low-power edge devices with limited energy availability.

As mentioned above, a key observation made by the authors in [135, 261] is the need for fine-grain Doppler resolution in order to achieve high-accuracy gesture

recognition with many classes. This is a key limitation of our 8-GHz sensor as we trade off frequency for ultra-low power consumption and die area utilization ($1.8mm^2$ [143] vs. $144mm^2$ for Google's Soli [135]). Integrating an SNN with such a small-area, low-power 8-GHz radar chip is therefore a well-suited choice, as SNN architectures can lead to substantially higher energy and area-efficient implementations compared to current DNN accelerators [44, 69, 165], even when the network architecture is shallow (see [35] for a discussion about the accuracy-area-energy tradeoff). But using SNNs comes at the cost of a potentially severe accuracy loss compared to using DNNs (see table 2.3 below). This motivates the exploration of pre-processing strategies for the SNNs, as provided further in this chapter. Indeed, it is thanks to a well-suited pre-processing strategy that we are able to achieve a high-accuracy, 5-class gesture recognition with our simple 8-GHz radar and a small 4-bit-weight SNN.

In order to bring radar gesture recognition closer to the edge, the authors in [231] proposed a 12-class gesture recognition system using a 60-GHz FMCW radar with an Nvidia Jetson Nano board running a network of ReLU neurons with four convolutional layers and three fully-connected layers achieving 94% accuracy. Compared to our work with SISO setting, their radar system [231] uses multiple receive antennas to provide azimuth and elevation information, at the expense of an increased power consumption in the radar front-end due to the use of multiple receive antennas. As input to their network, the authors first compute the magnitude of range-Doppler maps for each antenna and sum the resulting maps for each antenna. Then, they extract the 25 largest bins and compute the azimuth and elevation angles for those bins. Finally, they aggregate the range, Doppler and angular information as a feature cube that is fed to the neural network. Compared to [231], our pre-processing chain is much simpler as we have a SISO setting (angle processing is not possible since only one receive antenna is available). While the system of [231] is well suited for GPU-enabled edge computing platforms, it is much less suited for extreme-edge applications requiring an ultra-low power consumption. [231] also does not perform aggressive quantization on the inputs and network weights compared to our SNN queue.

Other DNN solutions have reported similar results by extracting micro-Doppler features of the gestures (joint time-frequency representations) instead of range-Doppler map processing [280]. In this work, both micro-Doppler and range-Doppler approaches will be compared in order to illustrate the importance of choosing the right pre-processing approach for resource-constrained SNNs.

Compared to our work, the DNN solutions described above are all frame-based and therefore cannot be deployed on the growing number of event-based SNN processors well suited for ultra-low-power AI applications. The authors in [165] proposed a mixed-signal SNN processor implementing an asynchronous and

event-based packet routing methodology in order to minimize latency and memory requirements. The authors in [44] proposed a fully digital asynchronous and event-based SNN accelerator, whereas [69] proposed a fully synchronous, yet event-based digital architecture.

Although not fully comparable to our work, most SNN-based gesture recognition systems use Dynamic Vision Sensor (DVS) cameras with pixels that directly output spikes when their change in luminosity crosses a certain threshold [157, 224]. Compared to standard cameras, that output image frames at a certain frame rate, DVS cameras output events asynchronously, which makes them a natural choice for SNN-based processing. Compared to our small 8-GHz radar chip, DVS cameras provide signals of significantly better fidelity (images vs. sparse number of radar returns) at the expense of being bulkier, more power hungry (23 mW for an *iniVation* DVS128 camera vs. 680 μW for our 8-GHz radar) and more expensive (in the $\sim$2000 \$ range compared to $\sim$10 \$ for ubiquitous radar chips).

Even then, as will be shown below, the performance of our work compares favorably with results in the DVS gesture recognition literature. Using the DvsGesture data set [11], the reported accuracies using various machine learning models and SNNs fall in the 91%$-$96% range for 10-class recognition [11,109,156, 201, 224]. Recently, the authors in [157] proposed a 11-class gesture recognition system achieving 89.64% of accuracy by first training a DNN composed of four convolutional layers and a fully-connected SoftMax layer, and then converting this trained DNN to an SNN through the conversion of ReLU activations into spike rates (as opposed to our work where we train the SNN directly in the spiking domain). More closely related to our work, the authors in [156] have reported an accuracy of 93% (similar to ours) using the NavGesture data set, containing 5 similar gestures compared to our work.

Also closely related to our work, 8-class human action recognition with CW radar and SNN processing has been proposed in [17], achieving an accuracy of 85%. Although their pre-processing shares commonalities with ours, they do not perform sparse coding nor dimensionality reduction after the radar processing, which helps to increase the SNN performance in terms of energy-area efficiency and accuracy, as reducing the dimensionality of the radar signals leads to overall smaller network architectures as less neurons are required. In addition, dimensionality reduction can be used to reject out-of-band noise, and sparse coding can increase the separability of the input signals, which can lead to a higher accuracy [83]. The authors in [17] use an SNN topology composed of multiple *Convolutional Spiking* layers and *Winner-Take-All* classification, coupled to the *Spike-Timing-Dependent Plasticity* (STDP) learning rule. Compared to the topology in [17], our proposed network architecture is simpler. The work in [17] makes use of *Leaky Integrate and Fire*

neurons, while we use *Integrate and Fire* (I&F) neurons, that are simpler to implement in hardware. Furthermore, [17] does not quantize the weights, while we apply 4-bit quantization to the weights in our network. As we are aiming for keeping the neural network constrained in size, the architecture of [17] is not appropriate for our goal.

In this chapter, our aim is to illustrate the importance and the *lessons learned* on the impact of using pre-processing for SNNs compared to its impact on DNNs. To the best of our knowledge, an extensive discussion about the impact of pre-processing on the accuracy of resource-constrained SNNs compared to non-constrained DNNs with the same topology, both trained via back-propagation, has not yet been reported. Such a discussion is important within the radar context of this chapter, since state-of-the-art radar gesture recognition DNNs targeting the edge computing domain [231] are principally being deployed in embedded platforms such as the Nvidia Jetson board, providing support for high bit-width integer and floating-point computation [231]. This will be discussed and illustrated for the example case of radar gesture recognition in section 2.6.

2.3 Radar data acquisition set-up

This section gives a brief overview of FMCW radar theory, as well as the radar data set we use for our SNN-based gesture recognition experiments.

2.3.1 FMCW radar theory

Using a SISO FMCW radar (see Fig. 2.1), a chirp signal $p_t(t)$ with slope α and carrier frequency f_c is emitted at the transmit antenna. This can be modelled as follows [153]:

$$p_t(t) = \sin\left(2\pi f_c t + \pi\alpha t^2\right) \tag{2.1}$$

Bouncing off an object located at a distance d from the radar, the signal $p_r(t)$ at the receive antenna can be modelled as a delayed and attenuated version of the transmitted signal (assuming a single point target):

$$p_r(t) = \xi p_t(t - T_d) \tag{2.2}$$

where ξ denotes the attenuation coefficient. The time-of-flight delay T_d depends on the round-trip distance $2d$ from the radar to the target:

$$T_d = \frac{2d}{c} \tag{2.3}$$

where c is the speed of light. By mixing the received signal $p_r(t)$ with a replica of $p_t(t)$ at the *In-phase* receiver of the radar chip [143], and by low-pass filtering the resulting signal such that the frequency component f_c is removed, the following Intermediate Frequency (IF) signal is obtained:

$$r(t) = -\frac{\xi}{2}\cos\left(2\pi\alpha T_d t + 2\pi f_c T_d - \pi\alpha T_d^2\right) \tag{2.4}$$

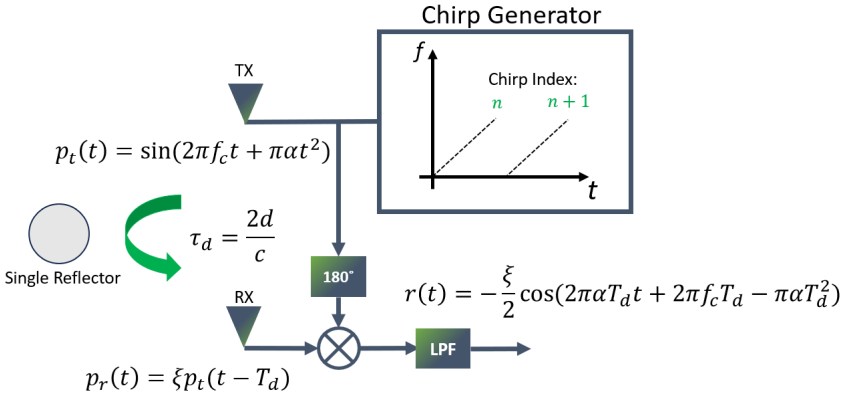

Figure 2.1: *Frequency-Modulated Continuous Wave (FMCW) radar sensing pipeline summarizing the steps from (2.1) to (2.4). For the sake of simplicity, a single reflector is assumed in this example.*

As the frequency of the baseband signal is proportional to T_d and therefore to d, its spectrum shows peaks at certain frequencies corresponding to the distance between the radar and the surrounding objects (the receive signal being in fact a sum of many signals of the form given by (2.4) when multiple targets are present). After the analog to digital conversion by the *In-phase* ADC, the discrete-time signal $r[i]$ is found:

$$r[i] = r(t = iT_f) \tag{2.5}$$

The choice of the ADC sampling period T_f impacts the maximum range coverage d_{max} of the radar [153]:

$$d_{max} = \frac{c}{2\alpha T_f} \tag{2.6}$$

The slope α is defined as follows:

$$\alpha = \frac{\Omega}{T_c} \tag{2.7}$$

where T_c is the chirp duration and Ω is the bandwidth of the chirp, which impacts the range resolution [153]:

$$d_{res} = \frac{c}{2\Omega} \tag{2.8}$$

For chirp n, the range profile $R_n[k]$ is defined as the DFT of $r[i]$. Peaks in the *magnitude* of $R_n[k]$ indicate the presence of a target, while the *phase evolution* of $R_n[k^*]$ for a target at range bin k^* between successive chirps represents the micro motions of the target [250] (a small radial displacement Δd induces a phase shift $\Delta\phi = \frac{4\pi}{\lambda}\Delta d$). The spectrum of the micro motions is called the Doppler profile and is defined as the DFT of $R_n[k]$ along n [250].

2.3.2 Gesture data set

We use a custom *imec* ultra-low-power 8-GHz SISO radar [143] to acquire a 5-class gesture data set. Table 2.1 shows the data set content used for the demonstration experiments reported in section 2.6 later on. The gestures were recorded by us at a distance of 2 meter from the antennas (RX and TX) and were obtained by swinging the right or left arm in the vertical direction (gesture 1-vertical), by swinging the right or left arm in the horizontal direction (gesture 2-horizontal), by waving with the right or left hand while keeping the palm facing out (gesture 3-hello), by moving the hand with the palm facing out towards and away from the radar (gesture 4-towards), and finally by recording background activity in which none of the above gestures appears in a static background (gesture 5-background). It should be noted that such data set is particularly well suited for comparing the importance of pre-processing in SNNs and DNNs as radar signals represent the environment with a lower fidelity compared to images [62], making them more sensitive to proper pre-processing and feature extraction.

The radar parameters were set as follows: the number of ADC samples per chirp is 512, the number of chirps per frame is 192, the time between chirps $T_s = 1.2$ ms, while the chirp duration T_c is 41 μs. Therefore, the total duration for a frame capture is 238 ms and $T_f = 80$ ns. Figure 2.2 shows the gesture acquisition set-up with the antennas and the radar read-out boards.

Class	Number of frames
Vertical	827
Horizontal	894
Towards	907
Hello	843
Background	704

Table 2.1: *Dataset content used in the gesture recognition demonstration experiments.*

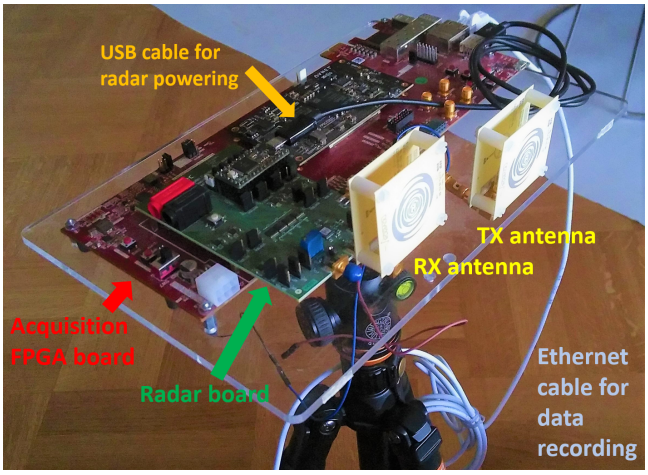

Figure 2.2: *Gesture acquisition set-up.*

2.4 Selected radar pre-processing approaches

This section will present the two widely used radar pre-processing approaches that will then be compared in section 2.6, followed by a description of our proposed *variants* and *add-ons* to these two techniques.

2.4.1 μDoppler signature

We acquire the μDoppler signature plot [250] of each gesture acquisition in the data set by computing the 1D vector $R_n[k^*]$ for $n = 1, ..., N_{tot}$ (N_{tot} is the total number of chirps), where k^* is the range bin in which the gestures are performed (which is known *a priori* since the distance between the human target and the

radar is fixed) and $R_n[k]$ is acquired by DFT as follows:

$$R_n[k] = \frac{1}{\sqrt{L}} \sum_{i=0}^{L} w[i] r_n[i] e^{-j2\pi \frac{ki}{L}} \tag{2.9}$$

where $L = 512$ is the number of ADC samples of r_n (the received IF signal for chirp n), and w denotes the *Blackman* window used [87].

Then, we apply the Short-Time Fourier Transform (STFT) on $R_n[k^*]$ as [17]:

$$\Theta[m, f] = \sum_{n=-\infty}^{\infty} R_n[k^*] g_s[n - mR] e^{-j2\pi fn} \tag{2.10}$$

where g_s is the *Hanning* window of length s and R is the hop size ($s = 192$ and $R = 8$ in this work). Finally, the μDoppler signature plot is found by taking the magnitude of $\Theta[m, f]$ [280]. The size of the $\Theta[m, f]$ matrix for each acquisition of Table 2.1 is $(N_T \times s)$ with $s = 192$ frequency bins, and the number of time bins N_T given by [17]:

$$N_T = \left\lfloor \frac{N_{frames} N_{chirps} - N_{overlap}}{s - N_{overlap}} \right\rfloor \tag{2.11}$$

where $N_{overlap}$ is the number of overlapping bins between subsequent analysis windows and is equal to $s - R$.

We also investigate a variation of this method, that we call $\Delta\mu$Doppler, by applying the STFT on the sequence $\tilde{R}_n[k^*] = R_n[k^*] - R_{n-1}[k^*]$ to remove the strong DC component during each analysis window [248].

In order to obtain example patches to feed to our neural network, we cut $|\Theta[m, f]|$ along dimension m into patches of 48 time samples, resulting in $\lfloor \frac{N_T}{48} \rfloor$ examples for each acquisition. Finally, we construct a balanced data set with a total of 1695 examples by randomly selecting 339 examples per acquisition class. The choice of 339 examples comes from the fact that the background acquisition (which is the one containing the smallest amount of frames) gives 351 example patches according to (2.11), but we discard the first and the last 6 example patches to remove any start-up and ending artefacts.

2.4.2 Range-Doppler maps

We acquire the range-Doppler (RD) maps [261] for each gesture acquisition in the data set by first acquiring (using (2.9)) $R_n[k], n = 1, ..., N_{win}$ as a matrix

of size $(L \times N_{win})$ where N_{win} is the number of chirps in each RD map (set to 192 in this work). Then, the RD map Γ is found as follows [153]:

$$\Gamma[m,k] = |\frac{1}{\sqrt{N_{win}}} \sum_{n=0}^{N_{win}} w[n] R_n[k] e^{-j2\pi \frac{mn}{N_{win}}}| \tag{2.12}$$

This operation is repeated for each successive N_{win} chirps, and $\left\lfloor \frac{N_{tot}}{N_{win}} \right\rfloor$ RD maps are acquired during gesture acquisition with N_{tot} the total number of chirps for a particular gesture recording. Only the first 50 range bins of each RD map are kept in order to reject far-away clutter.

In order to remove DC reflectors and to exploit the correlation between successive RD maps, we also investigate a custom variation of this method, that we call ΔRD, by aggregating the *difference* between successive RD maps as follows (where b denotes the RD map index):

$$\tilde{\Gamma}^b[m,k] = \max(\Gamma^b[m,k] - \Gamma^{b-1}[m,k], 0) \tag{2.13}$$

Similarly to the first pre-processing method from section 2.4.1, we construct a balanced data set with a total of 1695 examples, randomly selecting 339 examples per class to enable a fair comparison between the two pre-processing approaches.

2.4.3 Proposed enhancements

For both pre-processing methods described above, and for their respective Δ-variations, we explore the effect of a representative subset of possible enhancements [248]:

a) Dimensionality reduction through Doppler spectrum band-limiting.

b) Sparse coding through soft thresholding.

c) Image normalization.

a) In order to reject *out-of-band* noise, we explore the effect of band-limiting the Doppler spectra by keeping only a reduced portion of the Doppler axis between the normalized frequencies $[-0.26, 0.26]$ (frequency band found by visually identifying the maximal significant extent of the Doppler spectra in the data set).
b) In order to remove *in-band* noise, we use a fast approximation to *Lasso*

coding [269] by soft thresholding each Doppler spectrum, for each time bin in case of the pre-processing method *A)* μDoppler or for each range bin in case of the pre-processing method *B)* Range-Doppler. For soft thresholding, we use the max_k operator, which keeps the k largest values and replaces the others with 0 (we do not subtract the threshold from the values to be kept). To choose k, we consider that more than half of the Doppler samples within the normalized frequencies $[-0.26, 0.26]$ are significant, which leads to the choice of $k = \left\lfloor \frac{192 \times (0.26 - (-0.26))}{2} \right\rfloor - 1 = 48$ throughout this work.
c) Before encoding the pre-processed radar signals into event streams to be fed to the SNN, we normalize the pixel values between 0 and 1, either by normalizing each Doppler spectrum individually or by normalizing the complete maps (which gives a smaller amount of large pixel values compared to the normalization of each individual Doppler spectrum).

For the sake of illustration, Fig. 2.3 and 2.4 show example patches of the *vertical* gesture, for each pre-processing method, to be fed to the neural network. As seen in Fig. 2.3, $\Delta\mu$Doppler processing leads to a periodic-like pattern in time, which captures the Doppler information of the range bin in which the gestures are performed. On the other hand, ΔRD (Fig. 2.4) provides no time information, but rather captures the Doppler information carried by any wave that was reflected from the gestures in the environment (hence the range dimension).

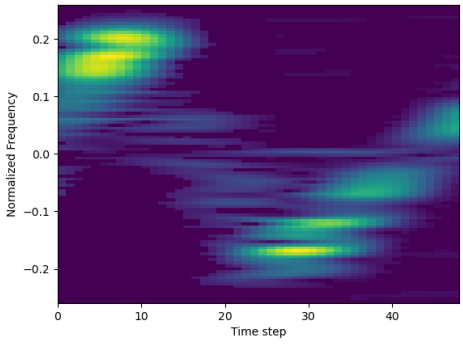

Figure 2.3: *Example patch for the vertical gesture, using $\Delta\mu$Doppler.*

Since back-end classifier algorithms are needed to evaluate the impact of the different pre-processing approaches, section 2.6 will explore the effect of the above pre-processing methods *A)* μDoppler and *B)* Range-Doppler, their respective Δ-variations and the enhancements **a)**, **b)** and **c)** on the classification accuracy

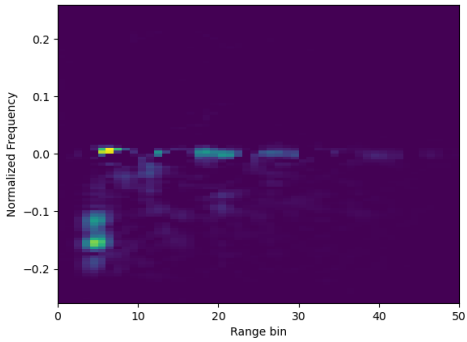

Figure 2.4: *Example patch for the vertical gesture, using ΔRD (Range-Doppler).*

of a low-complexity SNN compared to the classification accuracy of a DNN with the same topology.

2.5 Neural network design and training approach

We use two neural networks within this work: an SNN with *Integrate and Fire* (I&F) neurons (see (2.14), where V^k is the neural membrane potential at time step k, J_{in} is the neuron input and S is the spiking output), and the corresponding DNN architecture with Rectified Linear Unit (ReLU) neurons. Fig. 2.4 and 2.6 show the topologies used for both networks. The input size to the neural network varies depending on the pre-processing combination used, as shown in Table 2.2.

$$\begin{cases} V^{k+1} = V^k + J_{in} \text{ and } S = 0 & \text{if } V^k < 1 \\ V^{k+1} = 0 \text{ and } S = 1 & \text{if } V^k \geq 1 \end{cases} \tag{2.14}$$

In Table 2.2, the input size is different for each pre-processing variant as each pre-processing produces results in different domains: Doppler frequency against the number of sliding window hops through time for μDoppler, and Doppler frequency against range for RD. Band-limiting also affects the size as it crops the original dimension of the Doppler axis from 192 Doppler bins to 100 in our case (rejecting higher-frequency bins).

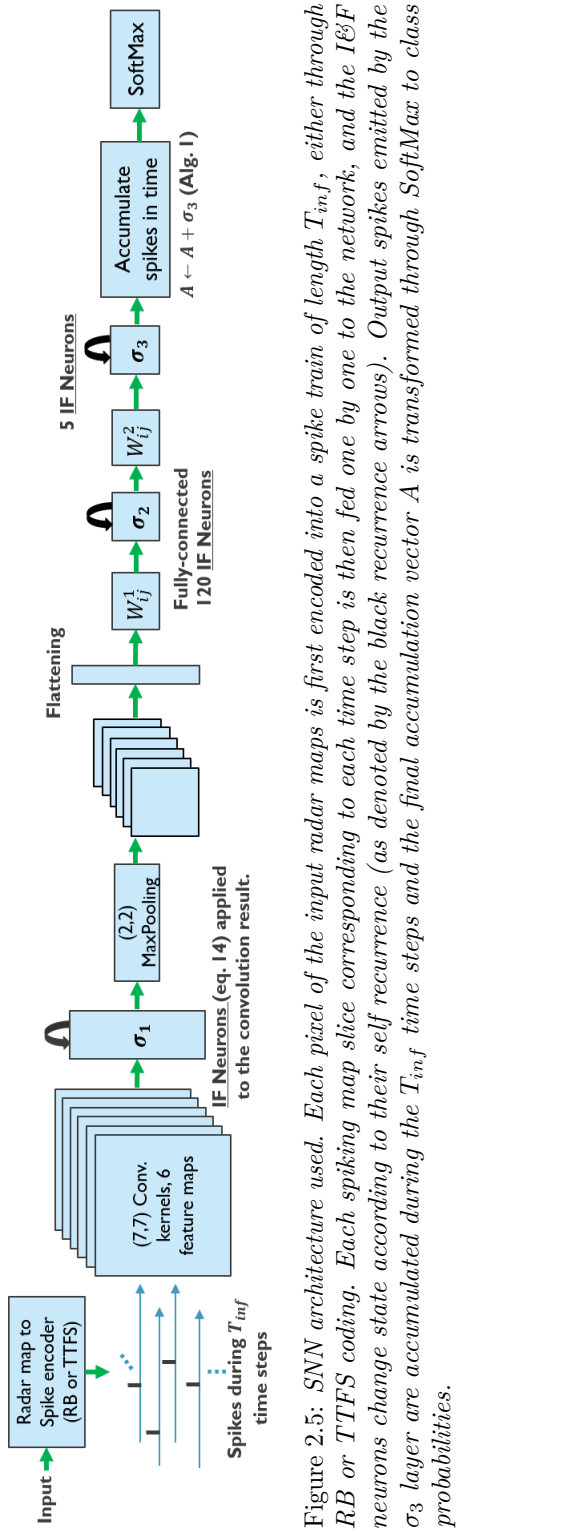

Figure 2.5: *SNN architecture used. Each pixel of the input radar maps is first encoded into a spike train of length T_{inf}, either through RB or TTFS coding. Each spiking map slice corrrespondig to each time step is then fed one by one to the network, and the I&F neurons change state according to their self recurrence (as denoted by the black recurrence arrows). Output spikes emitted by the σ_3 layer are accumulated during the T_{inf} time steps and the final accumulation vector A is transformed through SoftMax to class probabilities.*

Pre-processing	Band-limiting a)	Input size
μDoppler-based (2.4.1)	no	(192, 48)
μDoppler-based (2.4.1)	yes	(100, 48)
RD-based (2.4.2)	no	(192, 50)
RD-based (2.4.2)	yes	(100, 50)

Table 2.2: *Network input sizes.*

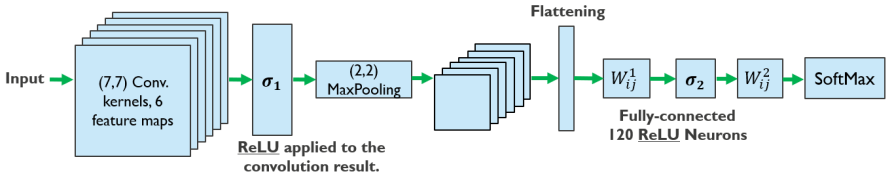

Figure 2.6: *Corresponding DNN architecture. The network retains a similar topology to Fig. 2.4 but with I&F neurons replaced by ReLU and the output of the σ_2 layer directly fed to the SoftMax non-linearity, as conversion from the event domain to the frame domain by accumulation is not needed.*

2.5.1 Architectural choices

The architectural choices made during the design of the spiking neural network (Fig. 2.4) are motivated as follows. First, we decided to use a convolutional layer at the network input to better capture spatial patterns in the input maps and to avoid over-fitting through weight sharing. Then, we opted for a *Max Pooling* layer instead of *Average Pooling*, as we wanted to keep the spiking nature of the data to be given as input to the pooling layer. Indeed, pooling is performed on the output of the spiking neurons in layer σ_1. With *Average Pooling*, the resulting map would not contain spikes anymore, as the output values are between 0 and 1. On the other hand, *Max Pooling* preserves the spiking nature of the data (output values are either equal to 0 or 1), which is beneficial for hardware implementation, as it is equivalent to a simple OR gate passing the spikes to the next layer. It is important to note that *Max Pooling* in SNNs has been shown to be rather complex to deploy for those SNNs that result from continuous-valued DNN to SNN conversion (see [205], where an existing DNN is converted into an SNN through rate-based coding). This is not a problem in our case as we train our SNNs (containing a standard *Max Pooling* layer) directly in the spiking domain. Finally, limiting the size of the network to only one convolutional layer and only two fully-connected layers is motivated by hardware considerations (a smaller network consumes less energy

and is easier to implement in hardware). We have decided to use I&F neurons over more complex models as the I&F neuron is the simplest to implement in hardware, again reducing the chip area and energy overhead.

For the SNN output, the conversion from the event to the frame domain is done by counting spikes during the inference time T_{inf} for each of the 5 output neurons in the σ_3 layer in Fig. 2.4. Then, the 5-element vector containing the spike counts is transformed via *SoftMax* to class probabilities. For the DNN in Fig. 2.6, the *SoftMax* layer is directly connected to weights W_{ij}^2.

In the SNN case, each pixel of the radar input maps must first be encoded into event trains of length T_{inf} (number of time steps for inference) to make the input compatible with the spiking, event-based nature of the SNN. In section 2.6, we will explore the effect of two different spike encoding approaches: *rate-based* (RB) and *time-to-first-spike* (TTFS) coding [28], as well as the effect of the inference time T_{inf}.

RB and TTFS encoding are done as follows. After normalizing the pixel values between 0 and 1, a value v is coded into a periodic event train with event rate $\frac{\lfloor vT_{inf} \rfloor}{T_{inf}}$ for RB, while v is coded into an event train containing one spike located at index $T_{inf} - \lfloor vT_{inf} \rfloor$ for TTFS. In both cases, if $\lfloor vT_{inf} \rfloor = 0$, no spike is generated.

2.5.2 Training approach

As a training approach in all our experiments, we use the *backpropagation of error* (backprop) algorithm in order to have a similar, consistent training method between both the SNN and the equivalent DNN. Indeed, since the aim of our experiments is to compare the impact of signal pre-processing on the SNN vs. the equivalent DNN performance, it is important to not only compare similar network architectures, but also to keep other parameters such as the training method as consistent as possible between the SNN and DNN setups.

To train the SNN, we use the *pyTorch* framework [189] by defining a custom I&F neuron model behaving as (2.14). As the derivative of a spike as a function of the neuron membrane potential $S'(V)$ is ill-defined, we approximate it using a Gaussian function (2.15) as surrogate derivative [173] to enable back-propagation (see section 1.2.2 in the Introduction chapter):

$$S'(V) \approx \frac{1}{\sqrt{2\pi}} e^{-2V^2} \tag{2.15}$$

Even if the network itself is not recurrent, back-propagation should be carried out through time (BPTT) as each spiking neuron can be seen as a recurrent unit by itself [173]. Indeed, an input event to a spiking neuron at time n still affects its membrane potential at future times, as evidenced by (2.14). In the general case of BPTT, the total loss function $\mathcal{L}_{tot}$ can be written as the sum of the losses for each time step as follows [263]:

$$\mathcal{L}_{tot} = \sum_{n=0}^{T_{inf}-1} \mathcal{L}[n] \tag{2.16}$$

Then, the derivative of $\mathcal{L}_{tot}$ as a function of weight W_{ij}^{l} in layer l is given as follows:

$$\frac{\partial \mathcal{L}_{tot}}{\partial W_{ij}^{l}} = \sum_{m=0}^{T_{inf}-1} \frac{\partial V_i^{l+1}[m]}{\partial W_{ij}^{l}} \sum_{n=m}^{T_{inf}-1} \frac{\partial \mathcal{L}}{\partial V_i^{l+1}[m]} \tag{2.17}$$

where $V_i^{l+1}[m]$ is the membrane potential of neuron i in layer $l+1$ at time step m. Expanding (2.17) shows the effect that a spike at time step n has on all future evaluations of the loss:

$$\begin{aligned}\frac{\partial \mathcal{L}_{tot}}{\partial W_{ij}^{l}} = \frac{\partial V_i^{l+1}[0]}{\partial W_{ij}^{l}} \sum_{n=0}^{T_{inf}-1} \frac{\partial \mathcal{L}[n]}{\partial V_i^{l+1}[0]} \\ + \frac{\partial V_i^{l+1}[1]}{\partial W_{ij}^{l}} \sum_{n=1}^{T_{inf}-1} \frac{\partial \mathcal{L}[n]}{\partial V_i^{l+1}[1]} + ... \\ + \frac{\partial V_i^{l+1}[T_{inf}-1]}{\partial W_{ij}^{l}} \sum_{n=T_{inf}-1}^{T_{inf}-1} \frac{\partial \mathcal{L}[n]}{\partial V_i^{l+1}[T_{inf}-1]}\end{aligned} \tag{2.18}$$

Equation (2.18) shows how BPTT applies to SNNs: the membrane potential at time 0 resulting from an input spike at time 0 affects the loss function evaluation from time step 0 onward; the membrane potential at time 1 resulting from an input spike at time 0 and 1 affects the loss function evaluation from time step 1 onward; and so on until the final inference time step $T_{inf}-1$. For our gesture classification task, the loss function only takes a non-zero value at the last time step as we accumulate all the output spikes after layer σ_3 before applying *SoftMax* (so-called *many-to-one* correspondence). In that case, (2.18) simplifies to the following:

$$\frac{\partial \mathcal{L}_{tot}}{\partial W_{ij}^{l}} = \sum_{m=0}^{T_{inf}-1} \frac{\partial V_i^{l+1}[m]}{\partial W_{ij}^{l}} \frac{\partial \mathcal{L}[T_{inf}-1]}{\partial V_i^{l+1}[m]} \tag{2.19}$$

Our *feedforward* SNN can thus be seen as a *recurrent* network where the recurrence is due to the state of the membrane potentials being passed to the next time step. This enables the use of BPTT just like in regular Recurrent Neural Networks (RNNs). Algorithms 1 and 2 below summarize the forward and backward pass during SNN training and inference. The 3D inputs $I[n]$ (resulting from the spike train encoding of each pixel in the 2D radar maps) are sliced out as 2D binary maps at each spike time step and fed one by one to the input of the network.

Algorithm 1 SNN Forward Pass

Input: $I[n], n = 0, ..., T_{inf}$: radar map, y : one-hot label
Output: $\hat{y}$, $\mathcal{L}$
V_i^l : all membrane potentials, A : (5×1) spike accumulator
Initialization: $V_i^l \leftarrow 0 \; \forall i, l$, $A \leftarrow 0$
for $n = 0$ to $T_{inf} - 1$ **do**
  $\sigma_3 \leftarrow \text{Network}(I[n], V_i^l)$
  $A \leftarrow A + \sigma_3$
end for
$\hat{y} \leftarrow \mathit{SoftMax}(A)$
$\mathcal{L} \leftarrow \mathit{CrossEntropy}(\hat{y}, y)$
return $\hat{y}$, $\mathcal{L}$

Algorithm 2 SNN Backward Pass (BPTT)

Input: $\hat{y}$: predicted class, y : one-hot label
$\mathcal{L} \leftarrow \mathit{CrossEntropy}(\hat{y}, y)$
for $n = T_{inf} - 1$ to 0 **do**
  $\Delta W_{ij}^l \leftarrow \frac{\partial V_i^{l+1}[n]}{\partial W_{ij}^l} \frac{\partial \mathcal{L}[T_{inf}-1]}{\partial V_i^{l+1}[n]}$
end for

For training both the SNN and the DNN, we initialize the weights of our network using the *Uniform Glorot* method (2.20) proposed in [82], where U denotes the uniform distribution and n_j is the number of neurons in layer j) and the biases set to zero, helping the variance at the input of the layers to be equal to the variance at the output in the forward pass and vice versa for the backward pass. Unlike the usual random initialization, where statistics are not layer-dependent, this method helps avoid the vanishing and exploding gradient problems during training [82].

$$W_j \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}] \tag{2.20}$$

We use the *Adam* optimizer [119] with the learning rate $\eta = 10^{-3}$, the gradient moving average coefficient $\beta_1 = 0.9$ and the gradient-square moving average coefficient $\beta_2 = 0.999$. Training is done on an Nvidia Geforce RTX 2080 and the batch size is set to 128. In the SNN case only, the training is done in two phases. First, the network is trained with full-bit weights for N_{full} epochs. Then, training continues with 4-bit quantized weights in the forward pass and full-bit weights in the backward pass for N_q epochs (the total number of epochs is therefore $N_{full} + N_q$). The choice of the number of epochs depends on the radar pre-processing configuration, as will be detailed in section 2.6.

In this next section, we will use the SNN and DNN networks presented above, alongside the radar pre-processing variations introduced in section 2.4, to extensively compare the impact of the pre-processing and event encoding on the classification accuracy of the 4-bit-weight SNN and its full-bit-weight DNN counterpart.

2.6 Experimental results

First, we will explore the effect of the different pre-processing variations described in section 2.4 on the classification accuracy of the SNN compared to the DNN. The results are reported in Table 2.3, where pre-processing techniques **1)** to **6)** use the μDoppler approach, while pre-processing techniques **7)** to **13)** use the RD (Range-Doppler) approach. In Table 2.3, all SNN results have been generated using Rate-based (RB) coding. The number of epochs needed to have effective training (without falling into over-fitting) varies depending on the pre-processing method used. For the SNN, the last $N_q = 5$ epochs of each training are done with 4-bit quantized weights in the forward pass [241]. We use 6-fold cross-validation to assess the classification accuracy performance of each set-up (282 examples in the test set and 1413 examples in the training set for each validation pass).

Then, we will take the best pre-processing set-up (in this case, pre-processing technique **6)** from Table 2.3), and vary the event-domain encoding (as described in section 2.5). Here, the goal is to optimize the classification accuracy of the network without making the inference time T_{inf} significantly longer. Table 2.4 shows the accuracy results for each event-encoding variant.

Pre-processing	Δ-variant	Band-limiting $[-0.26, 0.26]$	max_{48}	Normalization	SNN/DNN epochs (total)	SNN Acc.	DNN Acc.
1) (μDoppler)	No	No	No	Doppler	30/20	$36\% \pm 11\%$	$89\% \pm 3\%$
2)	Yes	No	No	Doppler	30/10	$34\% \pm 7\%$	$91\% \pm 1\%$
3)	No	Yes	No	Doppler	30/20	$36\% \pm 8\%$	$89\% \pm 1\%$
4)	Yes	Yes	No	Doppler	10/10	$87\% \pm 2\%$	$91\% \pm 2\%$
5)	Yes	Yes	Yes	Doppler	10/10	$88\% \pm 2\%$	$93\% \pm 1\%$
6)	**Yes**	**Yes**	**Yes**	**Complete map**	**10/10**	**89% ± 1%**	**93% ± 1%**
7) (Range-Doppler)	No	No	No	Doppler	20/20	$63\% \pm 8\%$	$90\% \pm 2\%$
8)	Yes	No	No	Doppler	20/20	$72\% \pm 3\%$	$85\% \pm 2\%$
9)	No	Yes	No	Doppler	20/20	$62\% \pm 5\%$	$92\% \pm 2\%$
10)	Yes	Yes	No	Doppler	20/20	$72\% \pm 3\%$	$89\% \pm 1\%$
11)	No	Yes	Yes	Doppler	20/20	$58\% \pm 4\%$	$91\% \pm 2\%$
12)	Yes	Yes	Yes	Doppler	20/20	$72\% \pm 3\%$	$91\% \pm 2\%$
13)	Yes	Yes	Yes	Complete map	30/30	$76\% \pm 3\%$	$90\% \pm 2\%$

Table 2.3: *Impact of the radar data pre-processing on the SNN classification accuracy compared to the DNN classification accuracy. For the SNN, the $N_q = 5$ last epochs are trained with quantized weights. RB coding is used throughout the table. The results clearly motivate the need for using a well-suited pre-processing method in the SNN case.*

Encoding	T_{inf}	Epochs (total)	Accuracy
1) RB	16	20	$90\% \pm 2\%$
2) TTFS	16	15	$92\% \pm 2\%$
3) RB	8	15	$92\% \pm 2\%$
4) TTFS	8	15	$92\% \pm 2\%$
5) RB	4	10	$91\% \pm 1\%$
6) TTFS	**4**	**10**	**93% ± 2%**
7) TTFS	2	10	$91\% \pm 1\%$
8) TTFS (no max_{48})	4	10	$87\% \pm 2\%$

Table 2.4: *Impact of event-domain encoding on the SNN classification accuracy. Pre-processing is done with $\Delta\mu$Doppler with band-limiting, max_{48} coding (exception made for encoding 8), and normalization on the whole map.*

2.6.1 Discussion on the impact of pre-processing

Our observations on the impact of pre-processing are given below. Observations **1)** and **2)** highlight differences between the SNN and DNN behavior, while observations **3)** and **4)** compare our different radar pre-processing approaches.

1) Table 2.3 highlights one of the major observations of this work: *The radar pre-processing parameters **drastically** affect the SNN classification accuracy, while their effect is **significantly more limited** for the DNN accuracy, even though the same network topology and the same learning approach are used.* A possible explanation is the following: in our setting, the differences between the SNN and the DNN are the neuron model used and the nature of the input (spiking or not). The output of I&F neurons is of binary nature (a spike) whereas the output of a ReLU neuron is continuous. The ReLU neuron has more *expressiveness* compared to the I&F neuron [184]. In addition, the ReLU neuron does not suffer from the vanishing gradient problem [83], while the I&F neuron does (see (2.15)), which may lead to a more effective training for the DNN. Furthermore, encoding input maps into spike trains can be seen as a *quantization in time* [49] which may potentially discard useful information, requiring better pre-processing to compensate for this information loss. During our experiments, we have noticed that those observations hold even when no weight quantization is performed.

The effect of each pre-processing parameter can be explained as follows. The Δ-variant acts as a first-order high-pass filter for both the μ-Doppler and the RD pre-processing, only keeping the more useful AC information, but it also amplifies the noise in higher-frequency bands. This filtering provides a

classification accuracy gain when RD pre-processing is used (entry 7 in Table 2.3), but it does not significantly affect the accuracy for the case of μ-Doppler pre-processing (entry 2 in Table 2.3) because of the larger noise amplification in the μ-Doppler case compared to the RD case. This larger amplification can be understood as follows. The RD maps are obtained using (2.12), as the absolute value of the range-Doppler processed radar ADC data. Noise in radar data is considered to be *Additive White Gaussian Noise* (AWGN) and remains AWGN after range-Doppler processing, since Fourier transforms are linear operations with zero mean and variance σ_w. Consequently, this noise distribution becomes Rayleigh-distributed with variance $\sqrt{\frac{4-\pi}{2}}\sigma_w$ after taking the absolute value [153]. Then, the resulting variance of the difference between two Rayleigh-distributed noise samples is given by $\sigma_{RD} = \sigma_w\sqrt{4-\pi}$ [153].

For $\Delta\mu$Doppler, on the other hand, differentiation is done directly on the range-processed data (2.9), while the noise is assumed AWGN. In that case, the resulting variance after differentiation is given by $\sigma_{\mu D} = \sigma_w\sqrt{2}$, which is clearly larger than σ_{RD}. Then, band-limiting discards the higher-frequency bins predominantly containing noise. This provides a large enhancement of the classification accuracy for the case of μDoppler pre-processing (entry 4 in Table 2.3). On the other hand, band-limiting does not seem to affect the accuracy for the RD pre-processing case (entry 10 in Table 2.3).

Finally, the application of max_{48} provides a smaller yet significant accuracy boost for both pre-processing methods, as it only keeps the 48 largest elements of each Doppler spectrum untouched and pads the rest to zero. Thus, this sparse coding step retains the *support* of each Doppler spectrum while discarding potential noise bins. Interestingly, using μDoppler pre-processing results in a wider accuracy range compared to RD pre-processing. Indeed, the discussion above motivates the fact that μDoppler and its Δ-variant introduce a larger noise contribution, which explains its lower classification accuracy compared to RD when band-limiting and max_{48} are not applied. But compared to RD, applying band-limiting and max_{48} seems to capture better features in the case of $\Delta\mu$Doppler, which explains its larger final accuracy (entry 6 in Table 2.3).

2) A second major observation found in Table 2.3 is the following: *Changing the pre-processing parameters can have* ***antagonistic effects*** *on the SNN and DNN classification accuracy, even though the same network topology and the same learning approach are used.* This can be seen by comparing the pre-processing techniques **7)** and **8)** in Table 2.3, where using the Δ variant enhances the SNN accuracy, while degrading the DNN accuracy significantly. This effect can be explained as follows. One of the advantages for using DNNs over SNNs is the limited need for pre-processing due to the *analog* nature of DNN neurons over the *spiking* nature of SNN neurons. The various pre-processing strategies of

Table 2.3 sparsify the original radar maps, which helps the SNN classification accuracy by increasing the separability between classes, but inevitably results in a loss of information, which can degrade the DNN accuracy. This demonstrates that conclusions drawn from DNN design cannot generally be applied to SNN design, even though the same network topology and the same learning approach are used.

3) Comparing pre-processing techniques **1)** to **4)** in Table 2.3, we see the following. *For μDoppler, it is not the sole effect of the Δ-variant or the band-limiting that affects the SNN accuracy,* ***rather a combination of both***. This can be explained as follows: using the Δ-variant alone helps removing the strong DC and near-DC components in each STFT window such that mostly useful AC information is processed. But the Δ-variant heavily amplifies the higher-frequency noise components, which degrades the learning. Band-limiting helps by rejecting out-of-band noise and keeping only the signal components, leading to an efficient feature extraction, which helps the SNN learning.

4) Globally, we see in Table 2.3 that the μDoppler approach clearly achieves a much higher accuracy than the RD approach with our SNN settings, with the best accuracy achieved for the $\Delta\mu$Doppler pre-processing with band-limiting, max_{48} coding (keeping the 48 largest entries in each Doppler spectrum) and normalization on the complete map (pre-processing technique **6**). This is specific to our data set and can be explained as follows. As the gestures are executed at a nearly constant distance from the radar, the range information in RD maps does not significantly relate to the nature of gestures. Training the SNN to find the relevant features then becomes more challenging as less useful information is available compared to the μDoppler pre-processing.

Yet, the SNN accuracy is still 89% compared to 93% for the DNN. It should be noted that during our experiments, we have also investigated the use of a one-hidden-layer fully-connected SNN with 50 I&F neurons in the hidden layer and the same output layer than the SNN shown in Fig. 2.4. We generally have observed the same recognition accuracy trends as presented above compared to an equivalent one-hidden-layer fully-connected ReLU DNN, although achieving a drastically lower accuracy because of the over-simplistic network architecture.

The major conclusions for this use case of radar-based gesture recognition are therefore:

*The pre-processing parameters **drastically** affects the SNN accuracy, while their effect is **significantly more limited** for the DNN accuracy.*

*Changing the pre-processing parameters can have **antagonistic effects** on the SNN and DNN accuracy.*

*For μDoppler, it is not the sole effect of the Δ-variant or the band-limiting that affects the SNN accuracy, **rather a combination of both**.*

*Globally, the μDoppler approach clearly achieves a **higher accuracy** than the RD approach with our SNN settings.*

Next, we will take the best pre-processing solution (entry **6** in Table 2.3) and vary the event encoding parameters to optimize the SNN classification accuracy.

2.6.2 Varying event encoding to optimize the SNN accuracy

By varying the event-encoder type (RB or TTFS) and T_{inf}, the best result is obtained with pre-processing technique **6)** of Table 2.3 and encoding method **6)** of Table 2.4, achieving 93% of recognition accuracy. Table 2.4 shows that TTFS gives on-par or better results compared to RB for the same T_{inf} (even though the advantage is limited). This limited boost in classification accuracy can be explained as follows. TTFS-encoding the input generates less spikes than RB-encoding, which makes the network activity sparser. Network sparsity may lead to better information disentangling and linear separability of the input representations in the neural network, as discussed in [83]. For completeness, we come back to the effect of the pre-processing by removing the max_{48} coding out of our best pre-processing and encoding queue (entry **8** of Table 2.4). Removing max_{48} has a significantly larger negative effect compared to the event-encoding parameters (it degrades the accuracy by 6% compared to entry **6**) which again emphasises the importance of using pre-processing on the SNN classification performance.

Using the pre-processing technique **6)** of Table 2.3 and the encoding **6)** of Table 2.4, our SNN achieves an accuracy on-par with the DNN, while being significantly cheaper to implement in hardware. Firstly, the SNN requires only *add* operations as the output of the I&F neurons are binary, whereas the DNN requires more expensive *multiply* and *add* operations, asking for more die area [206]. Secondly, the *Max Pooling* layer in the SNN can be implemented as a pool of simple OR gates, whereas the *Max Pooling* layer in the DNN must be implemented as a *search* operation, as the values at the input of the pooling layer are not binary in this latter case. Finally, an important limitation of current computing architectures is the energy cost of memory transfer due to the

memory bottleneck problem [92]. As our SNN architecture is event-based, it can be deployed on the growing number of massively-parallel and energy-efficient SNN processor architectures that aim to solve the memory bottleneck problem and are designed specifically for extreme-edge applications (see e.g., [44,69,165]).

2.6.3 Energy consumption and latency estimation

As discussed in section 2.2, most SNN-based gesture recognition systems use DVS cameras with a typical power consumption of around 100 mW (like for instance the one used in [157]). In our system, we use an imec 8-GHz SISO radar consuming only 680 μW at the expense of the number of gesture classes that can be distinguished because of the lower-fidelity nature of our radar signals compared to DVS cameras and higher-frequency radars such as Google's Soli (consuming $300m$ W) [135].

The different pre-processing approaches presented above introduce different latency and energy consumption overheads. In both the μDoppler and RD cases, two Fourier-based transforms are executed, which can efficiently be implemented with ubiquitous FFT accelerators (integrated in most commercial radar chips). Using a 256-point FFT only requires 2×1668 instruction cycles (56 μs at 60 MHz) and 2×407.2 nJ of energy in typical micro-controllers as in [1]. Similarly, the other pre-processing steps are cheap to implement as they do not iterate over the input data. Rather, the bottleneck to the pre-processing latency depends on the amount of radar chirps that must be collected during each acquisition. For μDoppler-based pre-processing, a radar acquisition time of 476ms can be derived using (2.11) and the radar parameters specified in section 2.3.2. For RD pre-processing, an acquisition time of 238 ms is needed, while for ΔRD an acquisition time of 476 ms, similar to the μDoppler case, is needed as two successive range-Doppler maps are used.

The SNN energy, latency and area overheads depend on the accelerator chip onto which the architecture is mapped, but also on the total number of neurons, the number of time steps needed for inference and the number of events per time step. With the input size of our best pre-processing technique (100×48), the convolutional layer contains $94 \times 42 \times 6 = 23688$ neurons and the fully-connected layers contain $120 + 5 = 125$ neurons for a total of 23813 neurons. The *Max Pooling* layer must also be mapped and, as discussed in section 2.5.1, our *Max Pooling* layer behaves as a pool of OR gates with 4 inputs, providing a downsampling of 2 on the convolution result. Such layer can be mapped as a layer of I&F neurons with synaptic weights and threshold tuned such that if any of the four input synapses of the neuron receives a spike, then the neuron will spike (e.g., by setting the four synaptic weights and the threshold to unity).

This leads to $47 \times 21 \times 6 = 5922$ neurons. Therefore, a total number of 29735 neurons is used.

Even though emerging mixed-signal SNN chips promise to be even more energy-efficient than digital ones [165], using the popular Intel Loihi chip [44] as our reference design for energy consumption and inference latency (as done by most SNN works in the literature), and using the *Nengo-Loihi* tool in *python*, we have estimated an energy cost per inference of 7.85 μJ (ignoring the cost of transferring data on and off the device) and a maximum inference time of around 4 ms (which could be reduced when deploying the model into Loihi through further optimisation). Therefore, the total latency is around 480 ms which is similar to most radar-based gesture recognition systems proposed in literature [135, 229, 231, 261, 280]. This value is feasible for a real-time system given the typical time scale of a hand gesture (some state-of-the-art systems such as [231] exhibit a total latency of more than 1 s from radar data acquisition to inference while still being useful in practice). This leads to an SNN power consumption of 7.85 μJ/480 ms= 16.35 μW. Compared to the DVS-based gesture recognition systems proposed in [11] (which reports power estimates on the IBM True North neuromorphic chip), our system consumes around three orders of magnitude less power (16.35 μW for our system vs. 88.5 mW for theirs), due to the significantly larger network used in their work with 16 convolutional layers vs. only one convolutional layer in our case. As explained earlier, this is at the expense of the number of classes that can be recognized (5 for our system vs. 11 for theirs), with a comparable accuracy (93% for our system vs. 91.77% for theirs). Compared to state-of-the-art radar-based gesture recognition systems [231, 261], our solution trades off the number of gesture classes that can be recognized for a better energy and die area efficiency at the sensor and neural network level (e.g., 11 classes for [261] and 12 classes for [231] at the expense of 60-GHz, multi-antenna radar sensors consuming more than 300 mW vs. our 8-GHz, SISO radar consuming only 680 μW), while still achieving a high recognition accuracy and fast inference time without making use of a desktop GPU as in [261] or of an Nvidia Jetson platform as in [231] (with an inference time of 25.84 ms against 4 ms in our case), both being too bulky and power-hungry for small-sized, ultra-low-power IoT applications.

2.6.4 Applicability of the proposed SNN on a different dataset

Finally, it is important to verify the applicability of our proposed 4-bit-weight SNN model to a different dataset than the 8-GHz one used in section 2.6.1. To the best of our knowledge, most of the state-of-the-art radar gesture recognition datasets such as [261] do not provide raw ADC data, but rather radar data that have already been pre-processed. Therefore, we train our SNN model

(Fig. 2.4) on the dataset proposed in [261], featuring 11 gesture classes. The dataset in [261] contains a total of 5500 sequences of RD maps (see section 2.4.2), acquired using a 60-GHz FMCW radar.

We note $\mathrm{RD}_i[t,k,m]$ for the ith sequence of RD maps, where t is the frame index, l is the range bin and m is the Doppler bin. The number of frames $M_i \geq 28$ in each sequence i can vary, but is always larger than $M_{\min} = 28$ throughout the dataset. Before feeding the data to our SNN, accumulation and sub-sampling is performed along t on each $\mathrm{RD}_i[t,k,m]$ as follows:

$$\mathrm{RD}_i[l,k,m] = \sum_{t=l}^{l+\frac{M_i}{M_{\min}}} \mathrm{RD}_i[l,k,m] \tag{2.21}$$

where l is the frame index after sub-sampling. Finally, spiking input sequences are obtained by thresholding $\mathrm{RD}_i[l,k,m]$ with regard to $\epsilon = 10^{-2}$ (found empirically). Values larger than ϵ are assigned to 1, while smaller values are assigned to 0. Training is conducted using the same learning parameters as in section 2.5.2, by iterating through 16 cycles of 20 epochs, with the first $N_{full} = 19$ epochs of each cycle trained with full-bit weights, and the last epoch of each cycle trained with 4-bit weights in the forward pass ($N_q = 1$). Similarly as in section 2.6.1, the model performance is assessed using 6-fold cross-validation. Fig. 2.7 shows the obtained confusion matrix, with a gesture recognition accuracy of $94.6\% \pm 2\%$. This demonstrates the applicability of our proposed 4-bit-weight SNN model to a second radar gesture dataset with more classes.

2.7 Towards radar gesture recognition with SNN-STDP

This chapter has shown that DNN-level accuracy could be reached using the proposed resource-constrained SNNs thanks to the use of a carefully-designed data pre-processing queue. In this Section, we provide additional motivation and explanations as to why these pre-processing (and also possible post-processing steps) do not make the SNN system more compute-intensive compared to the DNN case.

First, the pre-processing steps that are performed typically (see Section 2.4) rely on Fourier transforms that can efficiently be executed by ubiquitous FFT accelerators found in Digital Signal Processors (DSPs). In addition, also the other pre-processing steps are of low complexity in nature, relying on operations such as vector summing, linear data scaling (i.e., normalizing the data to

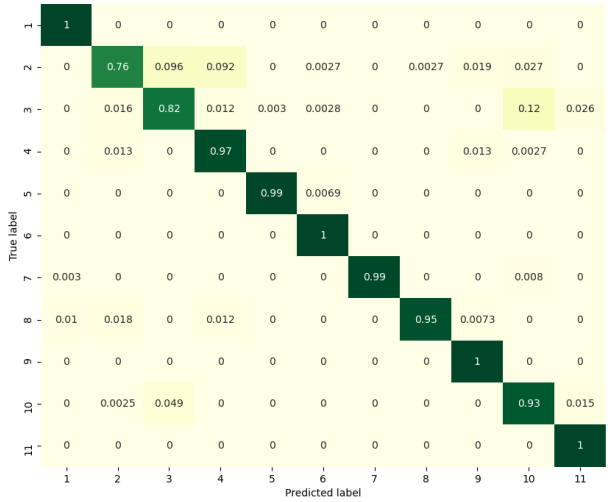

Figure 2.7: *Confusion matrix for the 60-GHz radar gesture recognition dataset of [261]. The recognition accuracy of the 4-bit-weight SNN is* $94.6\% \pm 2\%$.

have values between 0 and 1) and data cropping. These compute-efficient pre-processing steps are also often encountered when using DNNs [215] and therefore, do not constitute an additional compute overhead for the SNN case vs. the DNN setting.

The same can be said about possible post-processing steps. For example, in the case of the SNN-STDP network that will be presented in chapter 4, the post-processing consists of using a linear Support Vector Machine (SVM) as readout classifier. Such readout can be implemented with the usual multiply-accumulate (MAC) operations and can be accelerated in hardware as well by resorting to the ubiquitous parallel instruction set architectures found in common DSPs.

For the sake of completeness, Fig. 2.8 provides a high-level view of how we envision generic SNN systems for enabling online on-chip learning via STDP. The sensory data are first pre-processed by a feature extraction and data cleaning pipeline designed for the targeted application by an expert. The relevant signals are encoded as spike trains and fed to the SNN ensemble equipped with STDP plasticity for continuously personalizing its internal model to the specific environment in which it is operating. Finally, the output neural activity of the SNN can be fed to a supervised readout classifier trained only once by an expert. Doing so, the SNN ensemble personalizes the system to changes in the environment and constitutes the part of the network that can be re-trained and trained continually. This is also specially well-suited for extreme edge

applications since weight storage and activation data buffering is typically much lower in the SNN-STDP case compared to the DNN case, reducing the overall energy costs of accessing a large system memory.

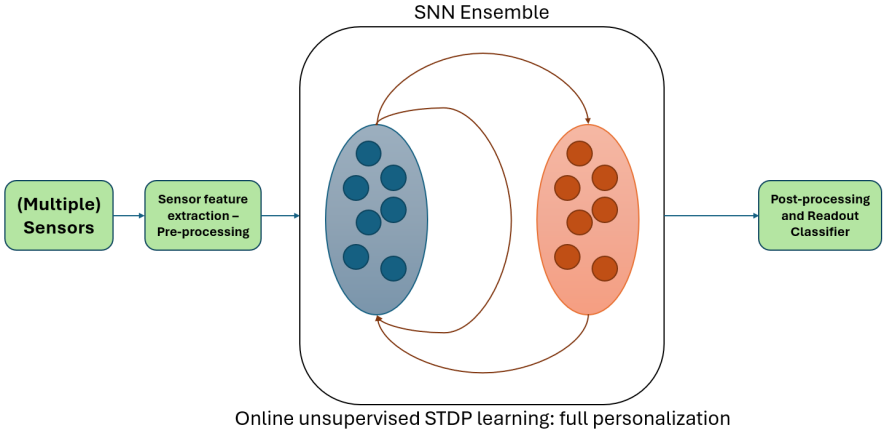

Figure 2.8: *High-level description of a generic SNN architecture for unsupervised personalization via STDP learning. In the SNN ensemble, the blue and orange neural ensembles denote the excitatory and inhibitory neurons.*

In order to provide energy consumption estimations for the radar gesture use case discussed in this chapter, the FFT-based radar processing steps can be performed using the DSP accelerator found in ubiquitous radar chips (e.g., Texas Instruments's AWR1443 chip), consuming very little energy per FFT operation (407.2 nJ with a latency of 56 μJ) [1]. Hence, it can be clearly seen that the energy and latency overheads of the FFT-based operations executed in the pre-processing phase of Fig. 2.8 can be safely ignored in comparison to the power consumption overheads of typical SNN-STDP accelerator hardware (e.g., consuming 4 mW in the mixed-signal SNN-STDP online-learning chip proposed in [192]). The same can be said about the post-processing since it essentially consists of a multiply-accumulate operation for the linear readout, which is of lower complexity than the FFT circuit in the DSP of the radar chip [1]. Overall, this SNN-based approach enables a reduction of more than two orders of magnitude in energy consumption compared to state-of-the-art radar gesture recognition systems using DNNs [231], while enabling online learning through STDP.

As further discussion, Appendix A provides supplementary material on the applicability of SNN-STDP for the use case of radar-based gesture recognition.

2.8 Conclusion

This chapter has shown the impact of various pre-processing and event-encoding techniques on the accuracy of a 4-bit-weight SNN within the context of radar gesture recognition for extreme-edge applications with a limited power and die area budget. In particular, it has been shown that, while using the same network topology and learning approach, pre-processing drastically affects the SNN recognition accuracy while its effect is significantly more limited on the DNN recognition accuracy. In addition, it has been demonstrated that conclusions drawn from the DNN design cannot directly be generalized to the SNN design as the pre-processing parameters can have antagonistic effects on the SNN and DNN recognition accuracy, even if the same topology and learning approach are used. Also the impact of event encoding to optimize the accuracy of the network has been explored, be it that it has less impact than the pre-processing.

Then, after extensive comparison between different approaches, a radar pre-processing and event-encoding queue has been identified to be used with a small, 4-bit-weight SNN to achieve 93% of recognition accuracy within only 4 time steps. The applicability of the proposed 4-bit-weight SNN approach has also been demonstrated on a second radar dataset, reaching 94.6% of recognition accuracy.

In addition, a supplementary discussion on the applicability of using an SNN with *Spike-Timing-Dependent Plasticity* (STDP) learning (instead of backprop) for this use case of radar gesture recognition is proposed in Appendix A.

Finally, we expect that many of the discussions and findings provided in this chapter generalize to the case of audio signals, where raw ADC data are often pre-processed with Fourier-type transformations. We also expect our discussions to generalize to other cases where signal propagation and demodulation prior to ADC sampling is done in a similar way to radar signals (e.g., SONAR, FMCW LIDAR, etc.). We leave the exploration of these other sensing modalities as future work (see Section 8.2 of the conclusion chapter).

Since this chapter has demonstrated the importance of radar pre-processing when using SNNs with radar data, the first part of the next chapter (chapter 3) will introduce a novel radar detection algorithm specially tailored for indoor drone navigation. This detector will then be integrated on a drone platform in the second part of chapter 3, and used to pre-process the raw radar data into a set of detections (indicating the presence of obstacles in the environment), before being fed to SNNs in chapter 5 for performing Simultaneous Localization and Mapping (SLAM).

Chapter 3

Design of a drone platform for sensor-fusion data acquisition

Part 1: A low-complexity high-performance radar detector for indoor drones

3.1 Introduction to the first part

The previous chapter has provided a first study of radar signal processing using conventional DNNs and SNNs, since incorporating a spiking radar sensor readout and processing SNN was identified in chapter 1 as a crucial aspect of the sensor suite that will be used to build our *neuromorphic sensor fusion* drone platform in the next chapters. To this end, the first part of this chapter will investigate the design of a low-complexity, yet high-performance radar obstacle detection algorithm that will be used in both the second part of this chapter as part of the neuromorphic drone platform design, and in chapter 5 to perform Simultaneous Localization and Mapping (SLAM) by fusing the radar detections with an event camera via SNNs.

Adapted from: *A. Safa et al., "A Low-Complexity Radar Detector Outperforming OS-CFAR for Indoor Drone Obstacle Avoidance," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2021.*

Recently, the use of radar sensors for indoor applications has been enabled thanks to the enormous progress in radar miniaturization [186,253] and energy efficiency improvement [144], making use of the *Frequency Modulated Continuous Wave* (FMCW) design principle [259]. Among those indoor sensing applications, the navigation of small *Unmanned Aerial Vehicles* (UAVs) or drones is extensively being investigated for applications ranging from automated logistics [240] and indoor maintenance inspection [276] to people search and rescue in partially damaged infrastructures [169], all requiring robust obstacle avoidance. Currently, most autonomous drone systems solely use camera-based sensing to perform obstacle avoidance, path planning [230] and SLAM [30]. As standard cameras have sensory limitations such as sensitivity to lighting conditions, occlusion by dirt or fog, and no intrinsic ranging [175], many teams are investigating the use of complementary sensors such as *Light Detection and Ranging* (LIDARs) [107] as well as radars [250]. The latter is the focus of this chapter.

Radars are well suited as a complementary sensing modality because, unlike cameras, they intrinsically provide range information, are very robust towards environmental effects and are independent of lighting conditions [148]. Usually, radar sensors are tightly coupled to pre-processing units that provide range, velocity and angle of arrival spectra, as partially covered in Section 2.4 of chapter 2 (so-called range-Doppler and range-angle profiles). These must be filtered afterwards by detection algorithms in order to separate target peaks from the background noise and clutter [154].

Traditional detectors used within the radar community are the so-called *Constant False Alarm Rate* (CFAR) class of algorithms, from low-complexity *Cell Averaging* (CA) CFAR to more computationally complex *Ordered Statistics* (OS) CFAR [154]. As CA-based methods require a minimal spatial separation distance between targets, OS-CFAR has been proposed as a better approach to the CA-CFAR-based methods for *multi-target* scenarios, where reflecting objects are closer to each other [99]. OS-CFAR has thus widely been adopted in critical areas such as *outdoor* automotive applications as its probability of detection P_D for a given probability of false alarm P_{FA} is larger compared to CA-based methods [202]. On the other hand, *indoor* scenes are usually composed of a much larger number of reflecting objects placed closer to each other compared to the outdoor case.

But OS-CFAR can fail to detect all potential targets systematically when processing radar signals during an indoor drone flight (a discussion on this point will be provided in section 3.5.4), while achieving a P_D close to 1 is essential for a safe drone flight. Thus, reliable radar target detection remains a challenge as missed detections can jeopardize the obstacle avoidance mechanisms and may crash the drone.

Therefore, there is a need for a better, yet *low-complexity* detection algorithm featuring a higher P_D for a certain P_{FA} compared to the conventional CFARs, not only including OS-CFAR but also more recent, state-of-the-art CFAR variants such as OR-CFAR [5], TS-LNCFAR [6] and CHA-CFAR [278]. Achieving a low-complexity solution is indeed crucial for applications such as small drones since the compute power on these resource-constrained platforms is usually limited due to the restricted available power consumption, area and payload. Hence, the major contributions of the first part of this chapter are the following:

1. **Algorithm:** We propose a high-performance, yet low-complexity radar target detector derived from the *Kernel Reed-Xiaoli* (KRX) non-linear detection principle [126] that outperforms a number of state-of-the-art CFAR detectors for our specific task of indoor drone navigation.
2. **Experiments:** We conduct many experiments for comparing our proposed detector against OS-CFAR (standard in automotive), OR-CFAR, TS-LNCFAR and CHA-CFAR, and we report a boost of $> +10\%$ in detection rate compared to the previously proposed techniques. In addition, we conduct many ablation studies to show the effect of each parameter and design choice in the proposed detector.
3. **Hardware implementation:** We implement the proposed detector in the on-chip micro-controller integrated in the 79-GHz Texas Instruments (TI) AWR1443 radar used during the data acquisitions, and we compare the processing speed of the proposed method against TI's standard pipeline.

The first part of this chapter is organized as follows. Section 3.2 will briefly review the related state-of-the-art works. Section 3.3 will provide background on CFAR detection. Section 3.4 will present our proposed methods. Section 3.5 will describe our experimental validation. Finally, section 3.6 will conclude the first part of this chapter.

3.2 Related works

To the best of our knowledge, radar sensing for drones has mainly been investigated for *outdoor* navigation above wasteland containing a sparse number of ground objects (while our work addresses the indoor setting). Most recent among those works, the authors in [250] demonstrated outdoor drone navigation using an *Inertial Measurement Unit* (IMU, magnetometer, gyroscope and accelerometer), a camera and a 24-GHz FMCW radar, and reported a lower

RMS navigation error against the ground-truth GPS compared to older radar-based drone solutions [216, 229]. In addition, the authors in [250] clearly demonstrated that conventional CFAR filters can suffer from a large number of missed detections and therefore proposed an alternative detection mechanism by first filtering the radar maps with a Gaussian kernel and then keeping the *five largest peaks* as detection. While such detection algorithm is well suited for navigation above wasteland containing an *a priori* known number of targets, it is much less the case for the dense indoor setting where the (potentially large) number of scatterers is unknown a priori.

Regarding the use of radar in indoor drone navigation, the authors in [271] proposed an *indoor* FMCW SAR imaging system mounted on a drone and tested in an indoor setting with four corner reflectors and one large obstacle placed in the middle of the scene. Although promising, a challenge faced with such system is the lack of sufficient space in constrained indoor environments in order to form a large aperture. In addition, as only a few strong reflectors were present in the scene, it remains unclear how the system would behave in realistic indoor scenarios in terms of effective detection and false alarms as no explicit detection is performed (only imaging is provided). Another work aiming at indoor SAR imaging using FMCW radars was proposed in [282], still without a discussion on radar target detection.

As conventional CFAR filters can severely suffer from missed detection, a large number of alternative methods have been proposed. The authors in [258] proposed a detector which combines multiple conventional CFAR variants and chooses the best one depending on the clutter homogeneity. The authors in [51] proposed to estimate the detection threshold of conventional CFARs using polynomial regression on the maps to be filtered, which, compared to the conventional CA-CFAR, provided a better threshold estimate as higher statistical moments are used during the estimation.

Recently, a number of CFAR detectors based on the *truncated statistics* principle have been proposed for multi-target detection in outdoor environments (e.g., ship detection from SAR images), using a careful modelling of the clutter statistics [5, 6, 278]. In addition to OS-CFAR, the applicability of those aforementioned methods to indoor scenarios is also investigated in this work. Using modern machine learning techniques, a CFAR detector using Support Vector Machines (SVMs) was proposed in [260] and a neural-network-based CFAR was proposed in [203]. Although highly efficient in scenarios well captured by the training data, those methods based on supervised training are not guaranteed to perform well across radically different scenes.

Closely related to our work, the authors in [126] proposed the *kernel RX* (KRX) detector as an enhancement of the classical *Reed-Xiaoli* (RX) detector

used in hyperspectral imagery [198]. They showed that KRX significantly outperforms RX (in terms of the reachable P_D for a certain P_{FA}) without being significantly more computationally complex. Still, the KRX detector can be too computationally expensive, as it requires a large number of matrix inversions [126]. Therefore, we have derived our detector based on the same principles used within KRX while proposing several modifications to simplify the computational cost of the algorithm, such that implementation is possible in extreme edge devices, in contrast to the KRX detector.

3.3 Background theory

In section 2.4 of chapter 2, background theory about FMCW radar processing has been provided under the *Single Input Single Output* (SISO) case (where the radar setup cannot provide angle of arrival estimation). Here, we will briefly introduce FMCW radar theory and processing in the Single Input *Multiple Outputs* (SIMO) case (where angle estimation can be done). Then, a number of state-of-the-art conventional CFAR detectors will be discussed. Finally, the principles behind the KRX detector will be presented. These are needed to explain our methods in the next section.

3.3.1 FMCW radar in SIMO mode

In Single Input Multiple Outputs (SIMO) mode, a radar uses one TX antenna and $m = 1, ..., N_{RX}$ RX antennas, usually spaced by an inter-antenna distance $d_a = \frac{\lambda}{2}$, where λ is the radar carrier wavelength [81]. In each sensing frame, the radar sends a sequence of $q = 1, ..., N_c$ chirps defined by:

$$p_q(t) = \exp j(2\pi f_c t + \pi\alpha t^2) \tag{3.1}$$

where f_c is the radar carrier frequency (77 GHz in this work) and α is the chirp slope [136]. By reflection on surrounding objects, a mix of delayed and attenuated versions of $p_q(t)$ is received at each RX antennas. Those received signals are demodulated by the IQ receiver such that for each chirp q and each antenna m, an *Intermediate Frequency* (IF) signal is obtained as follows:

$$r_{qm}(t) = \sum_{i=1}^{N_t} \xi_i e^{j(-2\pi\alpha T_{d_i} t - 2\pi f_c T_{d_i} + \pi\alpha T_{d_i}^2 + m\phi_i)} \tag{3.2}$$

where N_t is the number of target reflectors in the scene, ξ_i is the attenuation of the target reflector i, ϕ_i is the phase shift of target i induced by its angle of

arrival, and T_{d_i} is the round-trip time from the radar antenna to the target i, which is proportional to the distance d_i between the radar and the target as $T_{d_i} = \frac{2d_i}{c}$ (where c is the speed of light) [154]. The phase shift ϕ_i for target i is given by $\phi_i = \pi \sin\theta_i$ [81], where θ_i is the angle of arrival of the signal reflected by target i.

The IF signals $r_{qm}(t)$ (3.2) are then sampled by an ADC with a sampling period T_f to obtain the discrete-time signals $r_{qm}[n]$ for each chirp q and each antenna m with N_s samples. It is then possible to retrieve the range, the Doppler radial velocity and the angle of arrival of each target i by computing the Discrete Fourier Transform (DFT) of $r_{qm}[n]$ along n, q and m, respectively (a 3D DFT for each radar frame) [232]:

$$\Omega_{\mathrm{RDA}}[K,L,P] = \left|\frac{1}{\sqrt{N_{RX}N_cN_s}}\sum_{m=0}^{N_{RX}}\sum_{q=0}^{N_c}\sum_{n=0}^{N_s} r_{qm}[n]e^{-j2\pi\left(\frac{m}{N_{RX}}K+\frac{q}{N_c}L+\frac{n}{N_s}P\right)}\right| \tag{3.3}$$

The magnitude of the resulting data cube $\Omega_{\mathrm{RDA}}[K,L,P]$ exhibits N_t peaks at the bin locations corresponding to the range, velocity and angle of arrival of each target i in the scene. For the sake of illustration, Fig. 3.1 shows a typical range-angle profile (RAP) obtained by averaging the range-Doppler-angle cube $\Omega_{\mathrm{RDA}}[K,L,P]$ along the Doppler dimension L. Peaks in the profile indicate the presence of a reflecting obstacle.

Noise, clutter (ground echo) and multipath also result in additional undesired peaks that must be filtered using a detection algorithm, such as conventional Constant False Alarm Rate (CFAR) algorithms [154]. To reduce the number of DFT and detection operations (and thus, to increase the maximum frame rate achievable by the sensing system), the 3D DFT is not done at once; rather, a 1D DFT is performed first along n to obtain the so called *range profile* (RP) with its magnitude containing peaks at target locations. The magnitude of the RP is then filtered by a detector, such as a CFAR, obtaining a subset of *populated* range bins. It is then possible to perform the DFTs along q and m for each populated range bin only, to reduce the computational costs. It must be noted that this solution is well suited when the relative velocity between the radar and the obstacles is relatively low, so that the target remains within the same range bin at different time epochs. This is the case in indoor navigation, where the drone speed is relatively low due to the size of the drone and the weight of the payloads (for a drone flying at 10 kph with a typical coherent processing interval of 12 ms as used in our radar setup, the maximum range migration of obstacles is about 2.8 m/s $\times$ 0.012 = 3.36 cm, which falls below the range resolution of 7.1 cm used in our radar setup).

Another, more expensive approach is to first perform the 2D DFT along n and

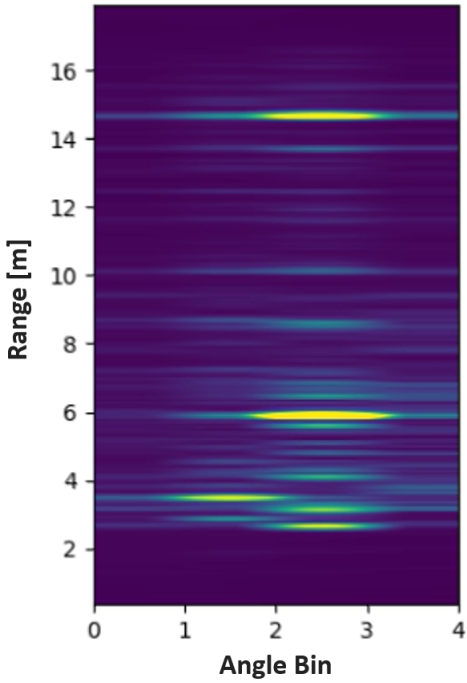

Figure 3.1: *A typical range-angle profile (RAP) from radar data captured by a flying drone in an indoor warehouse environment (see Fig. 3.3). The peaks indicate reflecting obstacles.*

q to obtain the so-called *range-Doppler profile* (RDP). The RDP is then filtered along each range and Doppler bin to find the subset of populated bins. As this second method introduces an additional decorrelation, conventional CFAR detectors will perform better at the expense of a severe increase in computation (i.e., the detection in this second approach must be performed along each range and Doppler bin with typical dimensions of e.g. 256×128 bins versus only one detection pass for the first approach) [274].

As high frame rates are desired for our indoor drone application to increase the dynamic responsiveness of the drone, we adopt the first approach, even though this method trades off lower signal processing gain for higher speed. In addition, this allows us to assess the performance of our algorithms on more challenging data in section 3.5 as a second decorrelation introducing the additional Doppler processing gain is not performed.

3.3.2 Background on conventional CFAR detectors

CFAR filters are used to detect targets in unknown background noise [154]. The most traditional CFAR is the CA-CFAR, which computes a local detection threshold for the *cell under test* (CUT) proportional to the average of the local *training* cells in a sliding analysis window (see Fig. 3.2 where S is the CUT, and the $Z_l[x]$ and $Z_r[x]$ are the left and right training cells, respectively). For locally

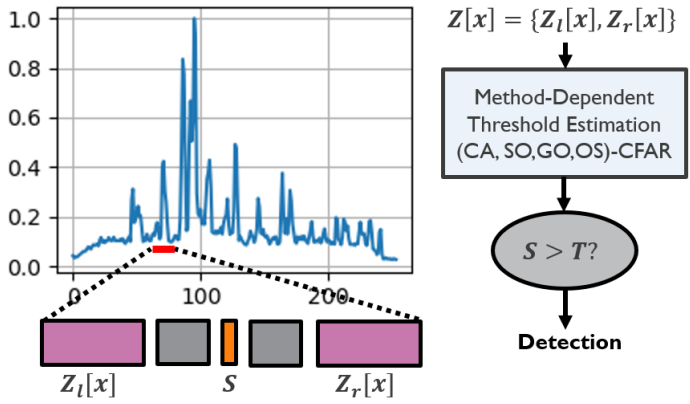

Figure 3.2: *A typical range profile being filtered by a CFAR detector. The CUT is the cell S in the middle, the training cells form a vector Z[x] and the remaining bins (in gray) are the guard cells.*

open environments, where scatterers are well isolated, CA-CFAR performs well, but its performance can strongly degrade in non-homogeneous environments where multiple targets and clutter edges are present [260]. The *greatest of* (GO) CA-CFAR (which chooses the greatest average between the left training cells $Z_l[x]$ and the right training cells $Z_r[x]$) performs better compared to CA-CFAR in environments with several clutter edges, but suffers from a loss in P_D when mulitple targets are present locally. On the other hand, the *smallest of* (SO) CA-CFAR (which chooses the smallest average between the left training cells $Z_l[x]$ and the right training cells $Z_r[x]$) performs better in the multi-target scenario, but is very sensitive to clutter edges, resulting in a significant increase in P_{FA} [154].

It has been reported that the OS-CFAR performs significantly better than CA-based CFARs for the multi-target case (its P_D is larger than CA-CFAR for a certain P_{FA}) and therefore has widely been adopted in (outdoor) automotive applications [99,133,202]. OS-CFAR first sorts all the training cells in increasing order and selects the kth element as the noise estimate $\hat{\beta}^2$. Then, the local

detection threshold is derived as $T = \alpha\hat{\beta}^2$, where α sets the P_{FA} and k is a parameter to be chosen.

Clearly, as this method requires a sorting operation per CUT, it is significantly more computationally expensive than CA-based methods [133]. OS-CFAR has a typical complexity of $O(LN_{tc}^2)$ when using *Bubble sorting* and $O(LN_{tc}\ln(N_{tc}))$ when using *Quick sorting*, where N_{tc} is the number of training cells (noted as $Z[x]$ in Fig. 3.2) and L is the total number of range bins [99]. On the other hand, CA-based methods have a complexity of $O(L)$ as *multiply-accumulate* operations can be done in one clock cycle on CPUs with vectored instructions, unlike sorting. The goal of our proposed detector is to improve the performance beyond OS-CFAR, while reducing the computational complexity. In section 3.5, we will benchmark our proposed detector against OS-CFAR as it is the standard detector adopted in most multi-target applications [154].

3.3.3 State-of-the-art CFAR for multi-target scenarios

Even though still widely used in practice, the conventional CFAR detectors discussed in section 3.3.2 have shortcomings in a number of situations where their clutter statistics estimation performance becomes poor (multi-target environments, non-homogeneous clutter and so on) [262]. Thanks to the progress made by the remote sensing community, a number of novel radar detectors specially tailored for the multi-target scenario have been proposed, achieving state-of-the-art result in anomaly detection from satellite SAR images [5, 6, 237, 278]. Among those, we consider three candidate detectors in this work, in addition to OS-CFAR.

Outlier-robust CFAR (OR-CFAR)

OR-CFAR [5] has been proposed as a well-suited detector for *Gaussian clutter* scenarios subject to an important number of outliers that can jeopardize the noise estimation performance of conventional CFAR methods. OR-CFAR achieves a robust noise estimation performance by using a truncated statistics (TS) mechanism where outliers in the training cells $Z[x]$ are removed if:

$$Z[x] - \mu > \gamma\sigma \tag{3.4}$$

where γ is the truncation depth, μ is the mean of the training cells and σ is their standard deviation. Then, the *maximum-likelihood* (ML) estimate of the clutter mean $\hat{\mu}$ and the standard deviation $\hat{\sigma}$ from the remaining training cells $\tilde{Z}[x]$ after truncation by (3.4) are given by [5]:

$$\hat{\sigma} = \sqrt{\frac{1}{\chi}\{\frac{1}{n}\sum_{i=1}^{n}\tilde{Z}[i]^2 - (\frac{1}{n}\sum_{i=1}^{n}\tilde{Z}[i])^2\}} \tag{3.5}$$

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}\tilde{Z}[i] + \alpha\hat{\sigma} \tag{3.6}$$

$$\alpha = \frac{e^{-\frac{\gamma^2}{2}}}{\sqrt{2\pi}\phi(\gamma)}, \quad \beta = 1 - \gamma\alpha, \quad \chi = \frac{1}{\beta - \alpha^2} \tag{3.7}$$

where $\phi(.)$ is the CDF of the standard normal distribution and n is the number of training cells after truncation. A CUT of amplitude S is reported as a detection if $S > \hat{\mu} + t\hat{\sigma}$, where t is a parameter setting the desired P_{FA}.

Truncated statistics log-normal CFAR (TS-LNCFAR)

TS-LNCFAR [6] has been proposed as a well-suited detector for *log-normal clutter* scenarios subject to an important number of outliers. Similarly to OR-CFAR, TS-LNCFAR uses a truncated statistics (TS) approach for outlier rejection, followed by a maximum likelihood (ML) estimation of the clutter mean and standard deviation. Outliers in the training cells are removed if:

$$\ln Z[x] - \mu > \gamma\sigma \tag{3.8}$$

with γ the truncation depth and μ, σ the mean and the standard deviation of the log training cells $\ln Z[x]$. The ML estimate of the clutter mean $\hat{\mu}_{ln}$ and standard deviation $\hat{\sigma}_{ln}$ are then given by a set of equation similar to those in OR-CFAR (3.5)-(3.7) [6]. A CUT of amplitude S is reported as a detection if $\ln S > \hat{\mu}_{ln} + t\hat{\sigma}_{ln}$, where t is a parameter setting the desired P_{FA}.

Censored harmonic averaging CFAR (CHA-CFAR)

CHA-CFAR [278] has been proposed as a well suited detector for *exponential clutter* scenarios affected by a large number of outliers. In contrast to methods that seek to *hardly* remove outliers by truncation (such as in OR-CFAR and TS-LNCFAR), CHA-CFAR *softly* removes the effect of outliers using the harmonic mean and the OS principle to estimate the noise level $\hat{\omega}$:

$$\hat{\omega} = (z_{m+1}^{-1} + z_{m+2}^{-1} + ... + z_N^{-1})^{-1} \tag{3.9}$$

where $\{z_1, ...z_N\}$ is a set obtained by sorting the elements of $Z[x]$ in increasing order, and m is the number of discarded samples (with the smallest amplitude) [278]. A CUT of amplitude S is reported as a detection if $S > t\hat{\omega}$, where t is a parameter setting the desired P_{FA}.

As those state-of-the-art, multi-target techniques have mostly been demonstrated for earth and ocean observation, it is interesting to study their applicability in indoor environments. Therefore, in addition to OS-CFAR (the standard detector for automotive applications), we will also benchmark our proposed detector against OR-CFAR, TS-LNCFAR and CHA-CFAR in section 3.5.

3.3.4 KRX detection principle

The novel detector that we will propose in this work (in section 3.4) is based on the principles behind *kernel* RX (KRX) detection [126]. The KRX detector has initially been proposed for hyperspectral images, where each pixel $\bar{x}$ is a vector with each vector entry corresponding to a certain spectral band (as opposed to the RP that we aim to filter in this work, obtained after applying range processing on (3.2), where each pixel $\bar{x}$ of the RP reduces to a scalar). In order to introduce KRX, the original RX detector, on which KRX is built, must be introduced first. The RX detector can be seen as a generalization of the CA-CFAR method to the multivariate Gaussian case (where pixels are not scalars but rather vectors) [198]:

$$RX(\bar{x}) = (\bar{x} - \hat{\bar{\mu}})^T \hat{C}^{-1} (\bar{x} - \hat{\bar{\mu}}) \tag{3.10}$$

where a pixel $\bar{x}$ is detected when $RX(\bar{x}) > T$, where T is a threshold setting the P_{FA}. $\hat{\bar{\mu}}$ and $\hat{C}$ respectively denote the mean vector and the covariance matrix of the multivariate Gaussian noise estimated from the training cells in the sliding window (similar to Fig. 3.2). When the background statistics cannot be modelled by a single Gaussian distribution (as in our multi-target scenario), the RX detector performance degrades significantly [170].

KRX was proposed in [126] as a nonlinear version of RX, which expresses the original RX model of (3.10) in a high-dimensional feature space $\mathcal{F}$ of dimension D through a nonlinear transformation $\Phi(\bar{x})$ as:

$$KRX(\bar{x}) = (\Phi(\bar{x}) - \hat{\bar{\mu}}_\Phi)^T \hat{C}_\Phi^{-1} (\Phi(\bar{x}) - \hat{\bar{\mu}}_\Phi) \tag{3.11}$$

where $\hat{\bar{\mu}}_\Phi$ and $\hat{C}_\Phi$ denote the estimated mean and covariance in $\mathcal{F}$. The covariance matrix in the feature space $\mathcal{F}$ is defined as $\hat{C}_\Phi = (\Phi(X_{tc}) -$

$\hat{\bar{\mu}}_\Phi)^T(\Phi(X_{tc}) - \hat{\bar{\mu}}_\Phi)$, where X_{tc} defines a matrix containing the training pixel vectors along each column and $\Phi(X_{tc})$ denotes the application of Φ to each of the individual vectors in X_{tc}. Eq. (3.11) is derived by considering that, once projected to $\mathcal{F}$, the model is well approximated by a Gaussian distribution, even though the process is non-Gaussian in the original space [170]. As $\mathcal{F}$ can possibly be of infinite dimensions, (3.11) is simplified using the *kernel trick* [108], which replaces the inner products found by expanding the terms in (3.11) by a *kernel* function $k(\bar{x}_i, \bar{x}_j) = \langle \Phi(\bar{x}_i), \Phi(\bar{x}_j) \rangle$.

A common choice for $k(\bar{x}_i, \bar{x}_j)$ is the *Radial Basis Function* (RBF) $k(\bar{x}_i, \bar{x}_j) = \exp(-\gamma ||\bar{x}_i - \bar{x}_j||_2^2)$, for which the Gaussian assumption in $\mathcal{F}$ holds [126]. Thus, KRX nonlinearly projects the input data in a space $\mathcal{F}$, which is such that a simple threshold decision region in $\mathcal{F}$ corresponds to a much more complex decision region in the original space. This enables the algorithm to perform significantly better in environments with complex background statistics such as highly cluttered and multi-target indoor scenes [126]. Also, other hyperspectral detectors have been extended using the nonlinear projection principle described above and it was demonstrated that the nonlinear versions always outperform the original ones in terms of reaching a high P_D for a given P_{FA} [127].

Although promising for our near-sensor detection application, the approach given above still presents three challenges for implementation on resource-constrained computing platforms:

1. Even when the kernel trick is used, (3.11) requires a large matrix inversion per CUT ($N_{tc} \times N_{tc}$) [126], making it unsuited for fast near-sensor detection.
2. The RBF kernel is an exponential model requiring a high bit precision as its scale changes rapidly [188]. Can we find a nonlinear transformation Φ that is well suited for heavy quantization, even down to 1-bit precision?
3. If such Φ exists, that is well-suited for 1-bit quantization, then the Gaussian assumption in the feature space will not hold anymore. How can we then correct for this initial assumption?

These challenges will be addressed by our proposed detector (described in the next section).

3.4 The proposed detector

In this section, we derive our novel detector starting from (3.11). We first note that (3.11) is the square of the Mahalanobis distance between $\Phi(\bar{x})$ and $\hat{\bar{\mu}}_\Phi$,

which is a measure of *distance* between $\Phi(\bar{x})$ and the background statistics in $\mathcal{F}$ [170]. When this distance is *larger* than a threshold T_0, $\bar{x}$ is reported as a detection. We consider the dual problem [123]: when the *correlation* between $\Phi(\bar{x})$ and the background statistics in $\mathcal{F}$ is *smaller* than a threshold T_1, $\bar{x}$ is reported as a detection. The original detection problem:

$$(\Phi(\bar{x}) - \hat{\bar{\mu}}_\Phi)^T \hat{C}_\Phi^{-1} (\Phi(\bar{x}) - \hat{\bar{\mu}}_\Phi) > T_0 \tag{3.12}$$

then becomes:

$$(\Phi(\bar{x}) - \hat{\bar{\mu}}_\Phi)^T \hat{C}_\Phi (\Phi(\bar{x}) - \hat{\bar{\mu}}_\Phi) < T_1 \tag{3.13}$$

where the computationally expensive matrix inversion does not appear anymore. Equations (3.12) and (3.13) are dual in the sense that searching for a $\Phi(\bar{x})$ with a *distance larger* than a certain threshold from a distribution parametrized through $\hat{\bar{\mu}}$ and $\hat{C}_\Phi$ (3.12) is conceptually the same as searching for a $\Phi(\bar{x})$ with a *correlation smaller* than a certain threshold from the same distribution (3.13). It is important to note that, although solving the same problem, (3.12) and (3.13) are not identical *per se*, as they are solutions to different optimization problems (see [123] for a review). In order to simplify (3.13) even more, we choose a transformation Φ such that $\hat{\bar{\mu}}_\Phi \to \bar{0}$ with $\hat{\bar{\mu}}_\Phi$ defined as:

$$\hat{\bar{\mu}}_\Phi = \frac{1}{N_{tc}} \sum_{i=1}^{N_{tc}} \Phi(\bar{x}_i) \tag{3.14}$$

where N_{tc} is the number of training cells in the window from which the noise estimation is performed. We therefore choose a Φ which induces *sparse* projections $\Phi(\bar{x})$ in $\mathcal{F}$, with only a few (possibly one) non-zero entries, such that $\hat{\bar{\mu}}_\Phi$ can be dropped in (3.13). Under this *sparse* projection, (3.13) becomes:

$$\Phi(\bar{x})^T \Phi(X_{tc}) \Phi(X_{tc})^T \Phi(\bar{x}) < T_1 \tag{3.15}$$

where X_{tc} is a $D \times N_{tc}$ matrix with D the dimension of $\mathcal{F}$, containing the projection of the training cells in $\mathcal{F}$. We can further rewrite (3.15) as:

$$\Phi(\bar{x})^T \bar{N}_\Phi < T_1 \tag{3.16}$$

where $\bar{N}_\Phi$ can be seen as a summary of the background noise statistics integrated through $\Phi(\bar{x})$ against which the correlation is measured. As remarked in section 3.3.4, (3.16) holds under Gaussian assumption in the feature space. As we are aiming for a sparse Φ, well-suited for 1-bit quantization, the Gaussian assumption does not hold and we need to modify how the background noise statistics are summarized in $\mathcal{F}$ through a modification of $\bar{N}_\Phi$.

To this end, we use the heuristic used by SOCA-CFAR, which states that the larger the norm of a pixel in the initial space, the less likely it is noise, such

that its effect should be attenuated when summarizing the background noise statistics [154]. Using this heuristic, we replace $\bar{N}_\Phi$ by a *centroid* across the training cells where each weight is inversely proportional to the L_2-norm of its corresponding training pixel:

$$\bar{N}_\Phi^m = \frac{1}{\Gamma}\sum_{j=1}^{N_{tc}} \frac{1}{||X_{tc}[j]||_2}\Phi(X_{tc}[j]) \tag{3.17}$$

where X_{tc} is assumed to be normalized to a maximum pixel L_2-norm of 1, with

$$\Gamma = \sum_{j=1}^{N_{tc}} \frac{1}{||X_{tc}[j]||_2} \tag{3.18}$$

being the amplitude normalization factor to effectively compute a *centroid*. The test (3.16) then becomes:

$$\Phi(\bar{x})^T \bar{N}_\Phi^m < T_1 \tag{3.19}$$

Further simplifications can be made to reduce the amount of *multiply* and *divide* operations in (3.17), (3.18) and (3.19). First, we can truncate the Taylor expansion of x^{-1} around 1 to the first-order term as $\frac{1}{x} \approx 2 - x$ where we can substitute $x = 2||X_{tc}[j]||_2$ (with $x \in [0, 2]$ as we considered $||X_{tc}[j]||_2 \in [0, 1]$) to obtain:

$$\frac{1}{||X_{tc}[j]||_2} \approx 2(1 - ||X_{tc}[j]||_2) \tag{3.20}$$

Therefore, instead of computing $\frac{1}{||X_{tc}[j]||_2}$, we compute $(1 - ||X_{tc}[j]||_2)$, where the constant 2 is dropped after normalization by Γ. Secondly, we can remove the expensive *multiply-accumulate* operations in (3.19) by solving the dual problem: a *small* ($< T_1$) correlation between $\Phi(\bar{x})$ and $\bar{N}_\Phi^m$ corresponds to a large distance between $\Phi(\bar{x})$ and $\bar{N}_\Phi^m$:

$$||\Phi(\bar{x}) - \bar{N}_\Phi^m||_p > T_2 \tag{3.21}$$

We choose $p = \infty$ instead of $p = 2$ for two reasons. Firstly, in order to replace the explicit norm computation involving *multiply-accumulate* operations by a sequence of simple *comparison* operations in parallel. Secondly, to attenuate the data-dependent bias introduced in the relation between the correlation in (3.19) and the L_2-norm $||\vec{a} - \vec{b}||_2^2 = \sum_i a_i^2 + \sum_i b_i^2 - 2\sum_i \vec{a}_i\vec{b}_i$, where the term $\sum_i x_i y_i$ is the correlation and the term $\sum_i x_i^2 + \sum_i y_i^2$ is a data-dependent bias. The L_∞-norm attenuates this bias, as $||\vec{a} - \vec{b}||_\infty \leq ||\vec{a} - \vec{b}||_2$ always holds [226]. At this point we have our final detector (3.21). In the sliding window, the CUT $\bar{x}$ is reported as a detection if (3.21) holds with $p = \infty$.

Finally, we need to find a nonlinear and sparse transformation Φ. As our goal is to filter a radar RP in this work, the pixels $\bar{x}$ that were generically considered as

vectors, become scalars that we note as $\xi = \bar{x}$ in the remainder of the derivation. We found that a well suited Φ is the *one-hot encoding* of ξ as follows:

$$\Phi(\xi)_i = \begin{cases} 1, & \text{if } i = D - \left\lfloor \frac{D}{\xi} \right\rfloor \\ 0, & \text{else} \end{cases} \tag{3.22}$$

where D is the dimension of the destination space $\mathcal{F}$. Such transformation is well suited as it is binary (1-bit precision), sparse (as at most one element is non-zero, $\hat{\tilde{\mu}}_\Phi$ can be ignored in (3.13)) and promotes linear independence between the pixel projections in $\mathcal{F}$ (which better complies with the constraint of non-singular covariance matrix $\hat{C}_\Phi$ in the original problem of KRX (3.12)).

In summary, our proposed detection algorithm performs the following steps:

1. Φ is applied to the CUT and the training cells in the sliding window using (3.22).
2. $\bar{N}_\Phi^m$ is computed following (3.17), (3.18) with the simplification of (3.20).
3. Each entry i of the CUT projection $\Phi(\xi)[i]$ is compared to $T_2 + \bar{N}_\Phi^m[i]$, and the CUT is reported as a detection as soon as an entry i gives $\Phi(\xi)[i] > T_2 + \bar{N}_\Phi^m[i]$ (by virtue of the infinite-norm in (3.21)).

Similar to the original KRX detector, finding a closed-form relation between the threshold T_2 and the P_{FA} is intractable. The threshold T_2 must rather be selected empirically by 1) acquiring a labelled dataset, 2) measuring the experimental P_{FA} and P_D values by sweeping the threshold, and 3) selecting a threshold returning the desired P_{FA} and P_D [170].

The complexity of our proposed algorithm is $O(LD)$ compared to $O(LN_{tc}^2)$ and $O(LN_{tc} \ln(N_{tc}))$ for OS-based methods (such as OS-CFAR and CHA-CFAR) and $O(L)$ for CA-based methods (such as OR-CFAR and TS-LNCFAR) [133]. D is typically rather small, as equation (3.22) projects scalars to vectors (we found $D \leq 15$ to be well suited during our experiments, see section 3.5).

Because of the simple heuristic rule used in (3.17), we expect the performance of the algorithm to degrade for degenerated situations where an overwhelmingly large number of neighbouring targets are present, since most bins in the sliding window will then miss-represent the noise statistics. Those situations are, however, unlikely to happen as realistic indoor scenes always have some empty space between obstacles. Nevertheless, we found this rule to be both necessary and efficient during our experiments in a regular indoor scenario: removing it caused a severe performance degradation while incorporating it provided excellent detection results (see the *ablation studies* in section 3.5.6).

3.5 Experimental performance assessment

In this section, we will experimentally assess the performance of our proposed novel detector against OS-CFAR by generating the *Receiver Operating Curves* (ROC: P_D versus of P_{FA}) of both solutions for labelled radar data acquired using a drone flying in a dense indoor scene.

3.5.1 Experimental setup

Fig. 3.3 shows the NXP *HoverGames* drone that we used to acquire the radar data. This drone setup uses an off-the-shelve 79-GHz Texas Instruments radar sensor, which is well suited for precise obstacle ranging due to its centimeter-scale range resolution. The radar parameters are given in Table 3.1, where L is the number of ADC samples per chirp, T_f is the ADC sampling period, N_c is the number of chirps in one frame, T_c is the chirp period, α is the chirp slope and d_{max} is the maximum distance that can be sensed. It is important to note that the choice of parameters in Table 3.1 impacts both the radar sensing accuracy and the energy consumption. For example, by duty cycling down the number of chirps N_c, the energy consumption is reduced while also reducing the sensing accuracy. We leave the further detailed study of these energy-accuracy trade-offs to future work.

L	T_f	N_c	T_c	α	d_{max}
256	160 ns	128	80 μs	50 MHz/μs	18.3 m

Table 3.1: *Parameters of the used 79-GHz radar.*

We have navigated the drone in an indoor environment composed of benches, chairs, shelves and so on (see Fig. 3.3), and we have obtained a dataset of nearly 400 radar frames. Out of that dataset, we selected a subset of 100 frames corresponding to the cases where the drone was flying towards a dense part of the environment, either towards the three racks visible in Fig. 3.3 or towards benches and chairs. Those cases represent the challenging, dense scenarios where missed detection can be detrimental to the drone and the object hit. We finally obtained a dataset of 100 RPs and we visually labelled as a detection each significant peak in each of the 100 RPs. Fig. 3.4 shows an illustrative example of an RP acquired when the drone was flying towards the racks. The red stars denote the labelled indices against which the detection algorithms are compared. Fig. 3.5 shows the effect of applying our detector to the RP



Figure 3.3: *Our drone setup with the 79-GHz Texas Instruments radar.*

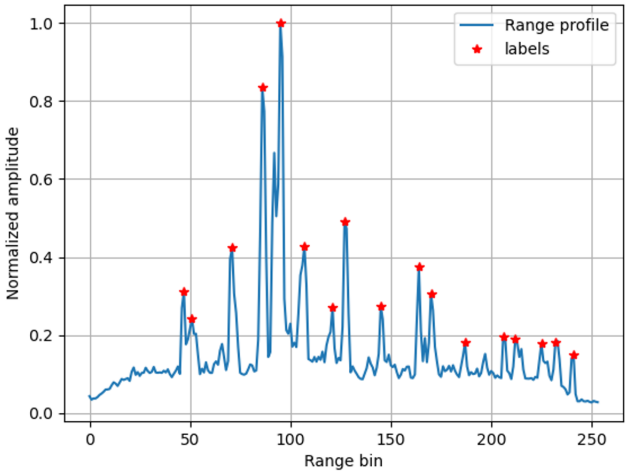

Figure 3.4: *An illustrative example RP obtained when the drone is moving towards the racks.*

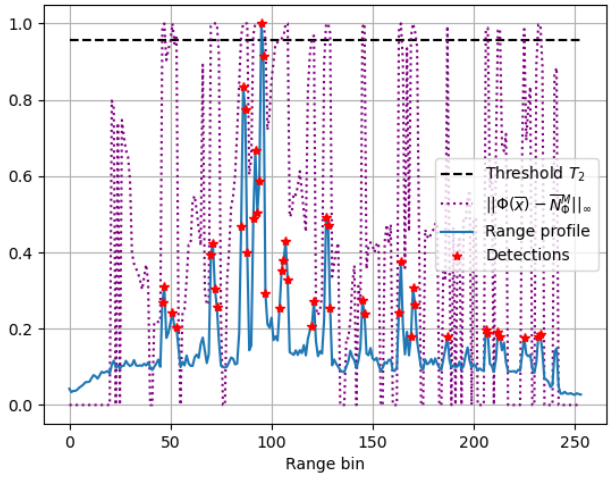

Figure 3.5: *Effect of applying our detector to the RP of Fig. 3.4. The detection is performed as in (3.21) by thresholding* $||\Phi(\bar{x}) - \bar{N}_\Phi^m||_\infty$ *(purple plot) against* $T_2 = 0.95$. *The detections are shown in red. The* D *parameter was set to* 15 *in this example. Some peaks in Fig. 3.4 are not detected leading to false negatives, and some peaks in Fig. 3.4 are detected multiple times (due to the peak tail), leading to false positives.*

of Fig. 3.4, as per (3.21): $||\Phi(\bar{x}) - \bar{N}_\Phi^m||_\infty$ is compared against a threshold level $T_2 = 0.95$ to obtain the subset of detected range bins. Fig. 3.5 shows the processing gain that our detector provides by mapping not only the strong RP peaks, but also the minor but significant peaks of the RP to high values of $||\Phi(\bar{x}) - \bar{N}_\Phi^m||_\infty$ (purple plot), while keeping this value lower for noise bins.

3.5.2 ROC-based results

The standard method used to compare detectors is the generation of the ROC curves, which show the probability of detection P_D versus the probability of false alarm P_{FA}. P_D is defined as the number of ground-truth detection labels found by the detector under test over the total number of ground-truth detection labels. Similarly, P_{FA} is defined as the number of detections returned by the detector under test and not associated to any ground-truth detection label, over the total number of range-bins L. The higher the P_D for a given P_{FA}, the better. We compute P_D and P_{FA} for both our proposed detection method

and OS-CFAR, by applying them to each of the 100 RPs and by comparing their output to the labels. We consider that a detection is a false alarm when it lies at a distance larger than 5 range bins (35 cm) from any label and *vice versa* for a correct detection. We therefore obtain the number of detected labels N_{Di} and the number of false alarms N_{FAi} for each RP_i, with $i = 1, ..., 100$. P_D is then obtained as $\frac{1}{100}\sum_i(N_{Di}/l_i)$ where l_i is the number of labelled indexes for RP_i; while P_{FA} is obtained as $\frac{1}{100}\sum_i(N_{FAi}/L)$, where $L = 256$ is the number of range bins in the RPs. The tuples (P_D, P_{FA}) are computed for different threshold values T_2 for our detector (see section 3.4), different α and k parameters for OS-CFAR (see section 3.3.2) and different numbers of training and guard cells (N_{tc}, N_g) for both methods. Each threshold value leads to a different tuple (P_D, P_{FA}) and the set of tuples is plotted as the ROC curve. For our proposed detector, we set the dimensionality D of the transformation $\Phi(\xi)$ (3.22) to 15, as it gave the best results.

Fig. 3.6 shows the ROC curves for our proposed detector. Fig. 3.7 to 3.9 jointly show the ROC curves for our proposed detector and for OS-CFAR, for different values of order parameter k and for the same combinations of N_{tc} and N_g as in Fig. 3.6. It is visually clear that our method significantly outperforms OS-CFAR without the requirement for computationally expensive *sorting* operations per CUT. Our proposed algorithm always achieves a significantly larger P_D for a given P_{FA}, for the different combinations of k, N_{tc} and N_g, with a measured average gain of more than 19% between ROCs of the same N_{tc} and N_g on the achievable P_D for a given P_{FA} (using P_{FA} values smaller than 1%, as typically used in practice).

In addition, we compare our proposed detector to OR-CFAR, TS-LNCFAR and CHA-CFAR (Fig. 3.10, 3.11 and 3.12) by tuning the truncation parameter γ of OR-CFAR and TS-LNCFAR, and the order parameter m of CHA-CFAR such that each competing detector reaches its best performance on the indoor drone dataset used for the assessment. A trend similar to the OS-CFAR case is observed: our proposed detector systematically results in a higher P_D for a given P_{FA} for the particular indoor scenario used (OR-CFAR, TS-LNCFAR and CHA-CFAR are rather tailored for remote earth observation with higher-resolution SAR images). This experimentally shows the effectiveness of our new detection algorithm when used in indoor settings.

3.5.3 Result confirmation with ground-truth data

As our goal is to detect as many targets as possible for a safe obstacle avoidance, the assessment presented in section 3.5.2 above has been conducted with manually labelled RP data, where each significant peak is marked as a potential

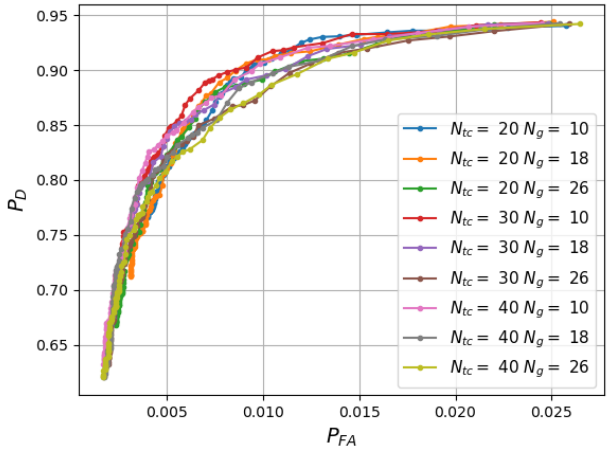

Figure 3.6: *ROC curves for our proposed detector, for different combinations of* N_{tc} *and* N_g*.* $D = 15$*.*

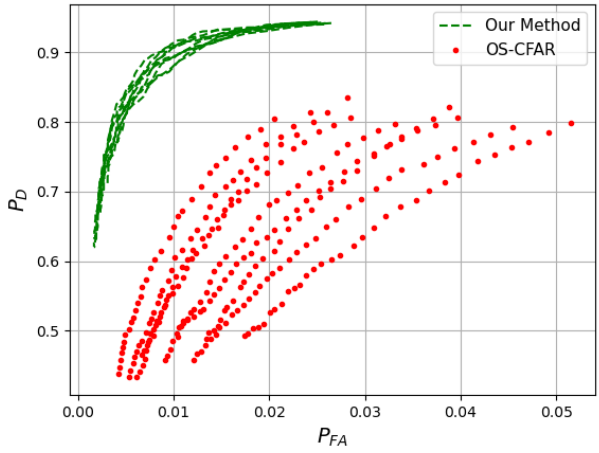

Figure 3.7: *ROC curves for our method compared to OS-CFAR with* $k = 0.5$*. The average gain is 40% on* P_D *compared to OS-CFAR for* $P_{FA} < 1\%$*.*

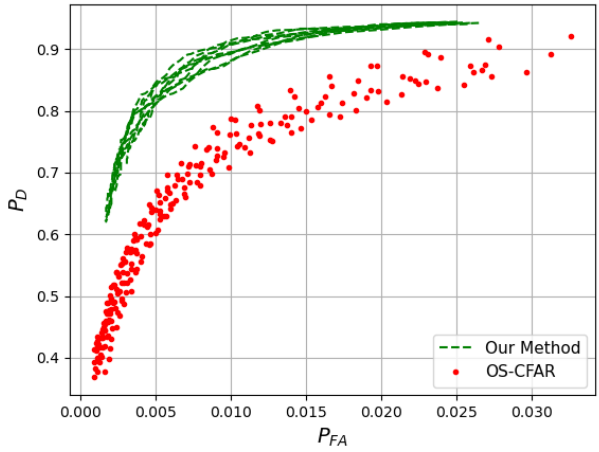

Figure 3.8: *ROC curves for our method compared to OS-CFAR with $k = 0.7$. The average gain is 19% on P_D compared to OS-CFAR for $P_{FA} < 1\%$.*

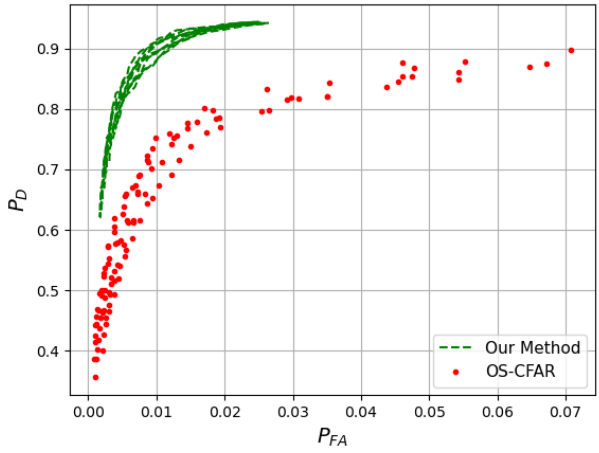

Figure 3.9: *ROC curves for our method compared to OS-CFAR with $k = 0.9$. The average gain is 21% on P_D compared to OS-CFAR for $P_{FA} < 1\%$.*

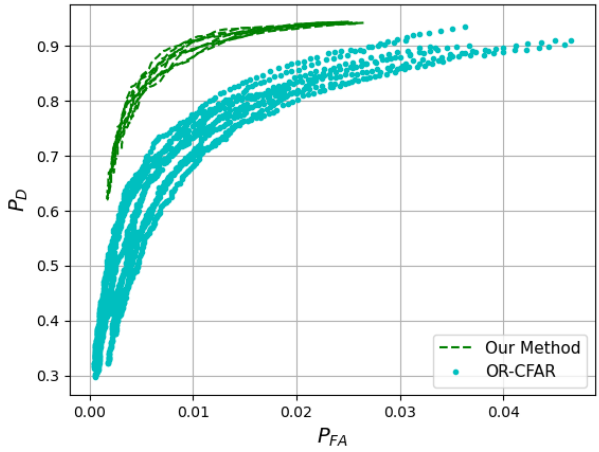

Figure 3.10: *ROC curves for our method compared to OR-CFAR with* $\gamma = 1.3$. *The average gain is 20% on* P_D *compared to CHA-CFAR for* $P_{FA} < 1\%$.

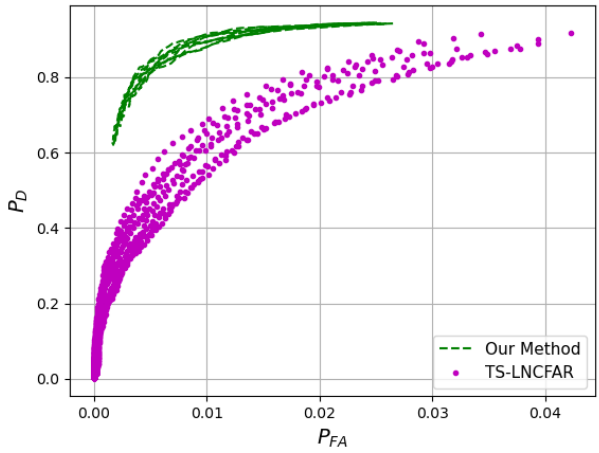

Figure 3.11: *ROC curves for our method compared to TS-LNCFAR with* $\gamma = 1.8$. *The average gain is 35% on* P_D *compared to CHA-CFAR for* $P_{FA} < 1\%$.

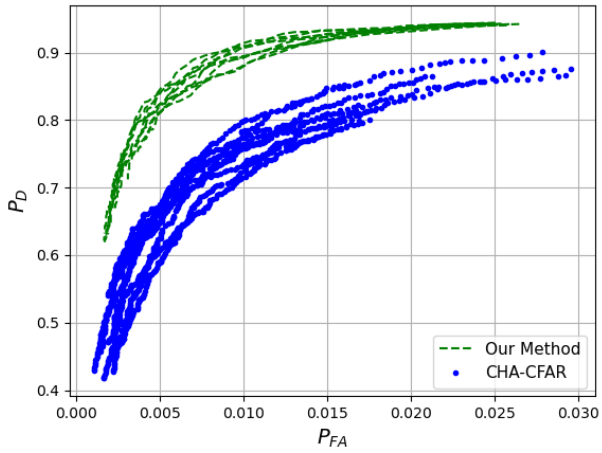

Figure 3.12: *ROC curves for our method compared to CHA-CFAR with* $m = 0.65 \times N_{tc}$. *The average gain is 16% on* P_D *compared to CHA-CFAR for* $P_{FA} < 1\%$.

obstacle. The downside of this visual labelling methodology is that noise and clutter could also lead to peaks in the RP, which, once labelled, would jeopardise the ROC assessment. On the other hand, the scenes acquired by our drone and radar are rich of obstacles. This motivates the assumption that the overwhelming majority of labelled peaks does correspond to actual targets. Still, it is important to confirm this fact using a well-controlled, yet realistic radar dataset with ground-truth measurements. Therefore, we acquired a second RP dataset in an empty room by keeping our drone and radar setup static, with two people walking in front of it. During the radar acquisition, a *Marvermind indoor GPS* device was used to acquire the ground-truth location of the two walking people. This scenario was chosen as it is more realistic than using usual corner reflector beacons (e.g., the scenario simulates a drone waiting for instructions with people walking in its neighborhood). The chosen scenario enables precise ground-truth measurements using the indoor GPS. Fig. 3.13 depicts this acquisition scenario.

Similar to section 3.5.2, a ROC-based comparison of our method against OS-CFAR has been conducted. The results are shown in Fig. 3.14. The OS-CFAR parameter k is set to 0.7, as this value corresponds to the smallest gain between our proposed method and OS-CFAR in the previous assessment of section 3.5.2, Fig. 3.8. The parameter D of our proposed algorithm is set to 4 in order to



Figure 3.13: *Ground-truth and radar acquisition. The two people walking in an empty room hold their indoor GPS beacon on their head during the radar acquisition.*

reach a desired P_{fa} range of $< 1\%$ (the impact of the D parameter will further be detailed in section 3.5.5). Compared to the results of section 3.5.2, Fig. 3.14 shows a similar trend where our proposed detector again provides a significant gain (44%) on P_D compared to OS-CFAR for the different combinations of N_{tc} and N_g (same as in Fig. 3.6).

In addition, Fig. 3.15 shows the performance of CHA-CFAR against our proposed detector (OR-CFAR and TS-LNCFAR do not reach a $P_D > 10\%$ on the tested P_{FA} range and are therefore not shown). The order parameter m of CHA-CFAR is chosen such that the detector performs at its best. Even though CHA-CFAR outperforms OS-CFAR, our proposed method still outperforms CHA-CFAR with an average gain of 32% between the corresponding curves. Again, this validates the effectiveness of our novel method for indoor radar sensing.

3.5.4 Discussion

At this point, we have *experimentally* shown in this indoor experiment that our proposed detector outperforms OS-CFAR, OR-CFAR, TS-LSCFAR and CHA-CFAR in this specific indoor sensing scenario. The reason for this is twofold. First, conventional CFAR methods and their state-of-the-art derivatives (see sections 3.3.2 and 3.3.3) perform detection by *maximum likelihood* estimation of the clutter statistics given a prior clutter distribution model [5, 6]. Therefore, the performance of those conventional techniques degrades when the clutter distribution severely deviates from their prior hypothesis. In outdoor scenarios

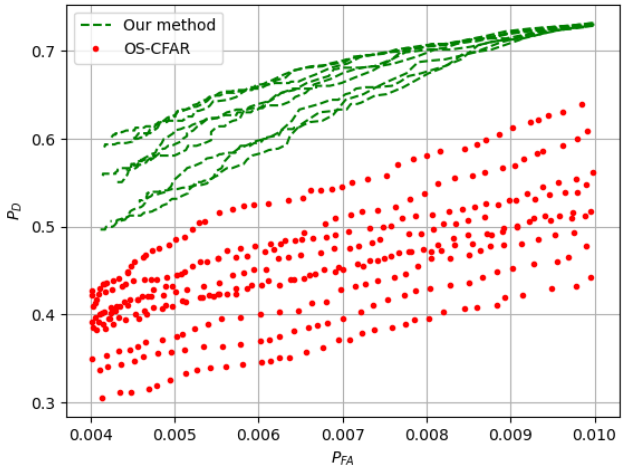

Figure 3.14: *ROC curves for our method compared to OS-CFAR with $k = 0.7$. The average gain is 19 % on P_D compared to OS-CFAR for P_{FA} values smaller than 1%.*

such as remote earth sensing, clutter distribution is often analytically well modelled by distributions such as Gaussian, log-normal and so on [262], as the wave propagation paths are simpler. In contrast, indoor scenarios lead to severe multi-path effects as a large number of reflectors are present. In addition, floor reflectively is not always uniform, which can lead to an increased number of clutter edges. Finally, compared to satellite SAR radars (as used in [5, 6, 237, 278]), our radar setup provides a much more modest resolution, which may lead to residual correlations between the bins in the RP, violating the assumption of inter-bin statistical independence [154]. Fig. 3.16 and 3.17 show the *goodness-of-fit* of an RP data distribution against the log-normal and the exponential models, which resulted in the smallest *Kolmogorov-Smirnov* (KS) distances out of a set of typical hypothesis distributions (log-normal, Gaussian, Exponential, Gamma and Weibull) [154]. For both hypothesis PDFs, the KS distance is above one order of magnitude larger than what is considered to be acceptable in ML-based CFAR estimation [5, 278].

The second reason behind the superiority of our method in this indoor environment is the following. Conventional CFAR methods based on the OS principle (OS-CFAR and CHA-CFAR in this work) reject the outliers within the training cells by sorting the cells and choosing either the nth entry as the clutter

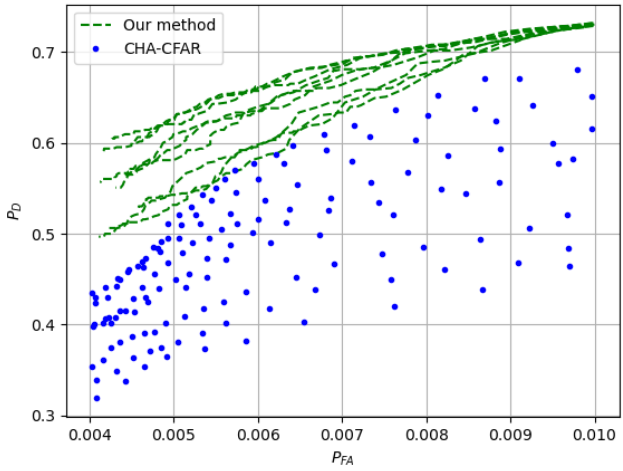

Figure 3.15: *ROC curves for our method compared to CHA-CFAR with* $m = 0.4 \times N_{tc}$. *CHA-CFAR achieves a better performance compared to OS-CFAR (Fig. 3.14) but is still outperformed by our proposed method by an average gain of* 14%.

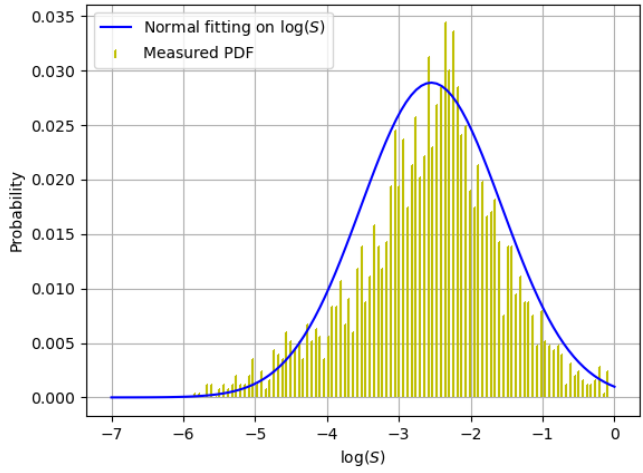

Figure 3.16: *Log-normal fitting of the measured PDF. The KS distance is* 0.048.

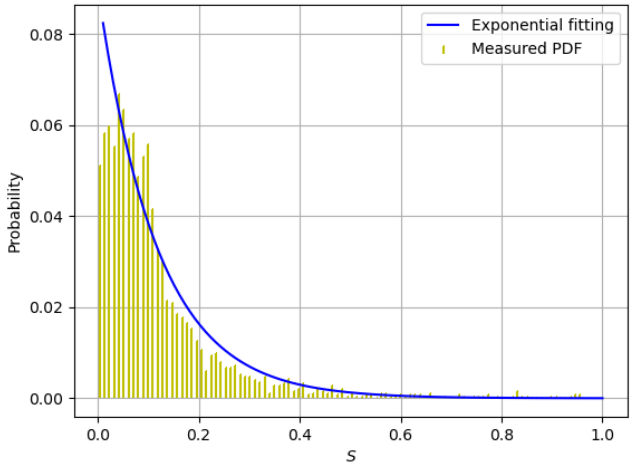

Figure 3.17: *Exponential fitting of the measured PDF. The KS distance is* 0.056.

statistics (as done in OS-CFAR) or either a combination of all entries above n (as done in CHA-CFAR). For OS-CFAR, this means that a maximum of $N_{tc} - n$ outliers can be rejected, making it difficult to find a suitable n in the dense, *indoor* case (the number of outliers per training window is non-homogenous). CHA-CFAR performs slightly better than OS-CFAR in the indoor case (see Fig. 3.14 vs. Fig 3.15) as it estimates the clutter statistics through a combination of the training cell entries above n. This *harmonic averaging* combination is such that entries with large values have a smaller contribution than entries with small values. Consequently, the order n can be chosen smaller than in the OS-CFAR case, which allows an adaptive rejection of an increased number of outliers in each training window (in contrast to the *hard* rejection of OS-CFAR).

On the other hand, modern hyperspectral detection techniques such as KRX [170] (from which our proposed method is derived) do not estimate the clutter statistics *explicitly*, and therefore, must not rely on a prior PDF model for ML estimation. Rather, those techniques solve a binary classification problem by expressing the local data (training cells and CUT) in a high-dimensional feature space $\mathcal{F}$, where the separability between samples is enhanced [170]. Simple decision contours in $\mathcal{F}$ will then correspond to much more complex decision boundaries in the original space, enabling high-performance detection in unknown and complex clutter statistics [170]. As shown experimentally in this chapter, such a detection principle is especially attractive for *indoor* radar

sensing where the *goodness-of-fit* of usual PDF models is not as good as in the *outdoor* case due to the complexity of the sensed environment. Finally, it must be noted that such high-dimensional, nonlinear methods have the downside that finding analytical expressions for setting parameters such as the detection threshold T_2 to meet a target P_{FA} is mathematically intractable. This is not only the case for our proposed detector (which uses (3.22) as a discontinuous nonlinearity function), but this is also the case for the original KRX formulation (which uses a smooth RBF kernel [126]).

3.5.5 Effect of the D parameter

In order to evaluate the effect of the D parameter (the dimensionality of the transformation $\Phi(\xi)$ in (3.22)), we fix $N_{tc} = 20$, $N_g = 10$ and we evaluate the ROC curves of the proposed detector using the same dataset as used in section 3.5.2 for a range of D values. The results are shown in Fig. 3.18.

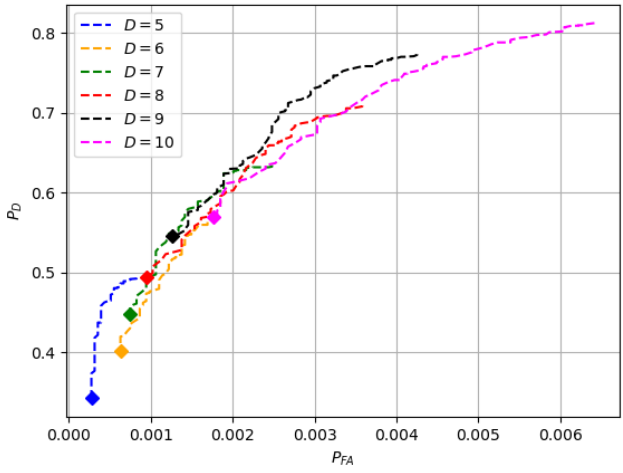

Figure 3.18: *ROC curves of the proposed detector for different D values. The diamond-shaped points denote the* (P_D, P_{FA}) *achieved in the limit of* $T_2 \to 1$ *(which are the limit values).*

As Fig. 3.18 shows, the D parameter affects the minimum reachable P_{FA}. This can be explained as follows. Each element of the vector $\bar{N}_{\Phi}^{m}$ in (3.17) can take a maximum value of 1 because of the normalization by Γ during the *centroid* computation and following the definition of $\Phi(\xi)$ in (3.22), where each element

of the vector $\Phi(\xi)$ is either 0 or 1. Indeed, if all elements of $\Phi(X_{tc}[j]), \forall j$ are equal to 1 (which is the limit case of the $\Phi(\xi)$ encoding), the evaluation of (3.17) results in a vector $\bar{N}_{\Phi}^{m}$ with all of its components equal to 1 because of the normalization by Γ. From (3.21), we can therefore write:

$$||\Phi(\bar{x}) - \bar{N}_{\Phi}^{m}||_{\infty} \leq 1 \tag{3.23}$$

as the entries of $\Phi(\bar{x})$ are either 0 or 1 by the definition of $\Phi(\xi)$ in (3.22). Therefore, only values of T_2 in (3.21) below 1 will affect the P_D and P_{FA} of the proposed detector. The P_{FA} achieved in the limit of $T_2 \to 1$ for a certain D value will thus be the minimal reachable P_{FA} for that D. Then, the smaller D is, the less quantization slots are available during the binary coding induced by $\Phi(\xi)$ in (3.22). As a consequence, the likelihood that the training cells in the vector X_{tc} have a 1 in the same quantization slot (i in (3.22)) grows, leading to larger elements in $\bar{N}_{\Phi}^{m}$, which lowers the evaluation of $||\Phi(\bar{x}) - \bar{N}_{\Phi}^{m}||_{\infty}$, lowering both the P_D and the P_{FA}. This can clearly be observed in Fig. 3.18, where the smaller the value of D, the smaller the minimum reachable P_{FA} is. Naturally, a similar discussion holds concerning the maximum reachable P_D in the limit of $T_2 \to 0$, where, the smaller the value of D, the smaller the maximum reachable P_D is.

3.5.6 Ablation studies

It is now important to verify the impact of the choices made for our detector in section 3.4 on the performance of the proposed detection algorithm. We have introduced a heuristic rule (3.17) that replaced the usual computation of (3.15) by a *centroid*, since we do not expect the projection of the RPs in the feature space $\mathcal{F}$ to be Gaussian, as opposed to the assumption of (3.15). Fig. 3.19 compares our heuristic rule against (3.15) and against OS-CFAR with $k = 0.7$ (as it was the closest to our method in terms of performance). As expected, using (3.15) results in a significant loss of performance compared to (3.17). This experimentally validates the need for our heuristic rule (3.17).

Finally, we have decided to use the L_{∞}-norm in (3.21) instead of the L_2-norm and we have given motivations for this choice is section 3.4. Fig. 3.20 compares the use of the L_{∞}-norm against the L_2-norm. It is interesting to see that OS-CFAR performs better for low P_{FA} values compared to using the L_2-norm in (3.21). Finally, it is visually clear that our method (using the L_{∞}-norm) leads to a significantly better performance compared to the L_2-norm for $P_{FA} < 0.015$.

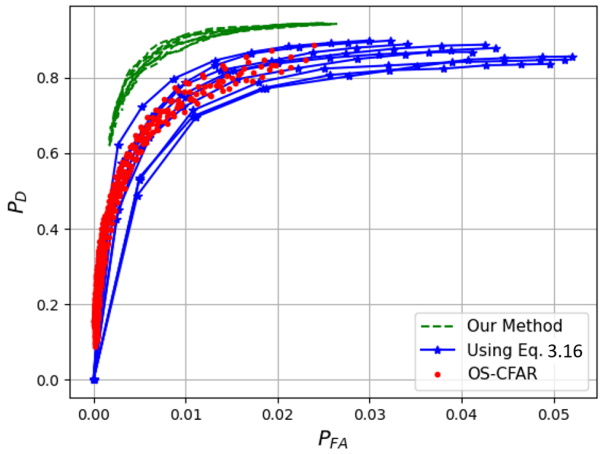

Figure 3.19: *ROC curves for different methods ($k = 0.7$ for OS-CFAR). Our method uses (3.17). The higher ROC performance attained by the proposed detector confirms the use of the heuristic rule (3.17).*

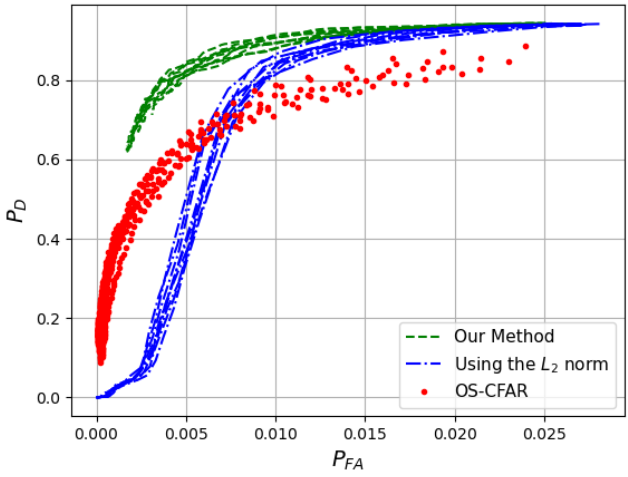

Figure 3.20: *ROC curves for different methods ($k = 0.7$ for OS-CFAR). Our method uses the L_∞ norm. The higher ROC performance attained by the proposed detector confirms the use of the L_∞ norm.*

3.5.7 System implementation and hardware metrics

We have implemented our proposed detector in an *ARM Cortex R4F* processor running at 40 MHz and integrated in the 79-GHz radar chip used during the data acquisitions. Our detector returns a list of populated range bins, which are then passed to a conventional beamformer algorithm [123] to compute the angle of arrival of each detection. As OS-CFAR involves an expensive *search operation* for each cell under test, it is already clear that our method is computationally less expensive. It is, however, more meaningful to compare our method against the standard detection API provided by the chip vendor, which uses a *hardware-accelerated* CA-CFAR applied on the full RD map (2D detection performed along each range and Doppler bin) followed by additional clutter filtering. Compared to our detector (written in C), the commercial built-in detector is not running on the CPU but on a dedicated hardware accelerator specially tailored for their 2D CA-CFAR implementing vectored computation. This hardware acceleration is needed to meet the inter radar frame timing since, unlike our single-pass detection, they perform detection along each range and Doppler bin (256×128 detection passes vs. 1 in our case). Then, as CA-CFAR is subject to a high amount of clutter false alarms, their API performs a final clutter filtering step and the detection list is returned to a conventional beamformer algorithm (identical to the one used with our proposed detector) to compute the angle of arrival of each detection.

We have compared our detection method against the commercial built-in one in term of *average inter-frame processing time margin*, meaning the *time slack* available between the execution of each complete radar frame processing (reception of raw ADC data - range-Doppler processing - detection - angle of arrival estimation). Table 3.2 shows the measured average inter-frame processing time margin, for a radar frame rate of 10 FPS.

Method	**Time Slack [ms]**
Standard	26.7
Ours	**45.7**

Table 3.2: *Measured average inter-frame processing time margin. The radar frame rate is set to 10 FPS.*

Table 3.2 clearly shows that our method executes nearly two times faster, providing a significant increase in the time slack available between the execution of each frame processing step, even though our method does not rely on any dedicated hardware acceleration. This additional time slack enables our system to run at frame rates $\sim$35 FPS before the dead-lock of the MCU (due to the

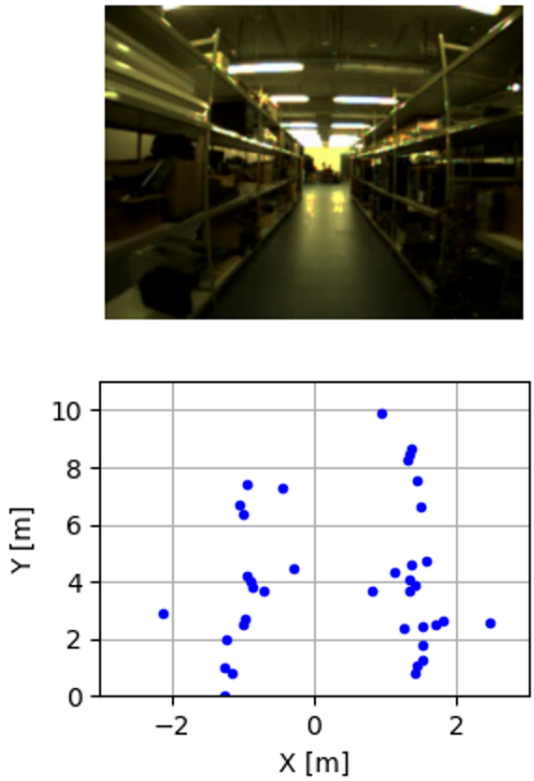

Figure 3.21: *Radar-camera snapshot of our drone flying through indoor industrial aisles with racks on each side. The right and left shelves can easily be identified in the detection point clouds (in blue).*

fact that radar chirps are incoming at speeds larger than what the MCU can process), while the standard method can run at maximum 17 FPS before the dead-lock of the MCU. This larger frame rate allows our radar system to detect obstacles faster, enabling safer drone flights. Furthermore, this additional time slack can be used to execute more complex angle of arrival algorithms after the detection step to enhance the poor angle resolution of the conventional beamformer [123]. Finally, Fig. 3.21 shows a radar-camera snapshot of our drone flying through a room with industrial racks, using our proposed detector running on-line in the MCU of the radar chip at 30FPS, with a conventional beamformer [123] estimating the angle of arrival of the detections. The aisles can clearly be observed thanks to the high P_D over P_{FA} ratio provided by our detector. This shows qualitatively the effectiveness of our detector.

The contributions of the first part of this chapter can be summarized as:

The radar detection technique presented in the first part of this chapter innovates by first projecting the radar data into a higher-dimensional feature space using a hardware-efficient binary mapping and then performing detection in this binary feature space. This leads to significant enhancements ($> +16\%$) in detection probability compared to state-of-the-art automotive-grade detectors while being less computationally expensive.

3.6 Conclusion to the first part

The first part of this chapter has focused on reliable radar target detection in dense indoor environments for drone applications requiring obstacle avoidance for safe navigation. We have presented a novel radar target detection algorithm with low computational cost, suitable for near-sensor implementation in edge devices. It has been shown that the proposed detector outperforms OS-CFAR with a more than 19% higher P_D and outperforms CHA-CFAR with more than 16% higher P_D for the same P_{FA} in our indoor drone navigation scenario, where missed detections can be detrimental, while being computationally less complex than both OS-CFAR and CHA-CFAR (as no sorting operations are needed). To the best of our knowledge, this work significantly improves the state of the art for high-performance yet low-complexity radar detection in very dense indoor sensing applications.

Even though our demonstrations have mainly been provided in two different indoor environments, we expect the detection algorithm presented in the first part of this chapter to be useful as well in other indoor settings and with other types of robot platforms instead of the drone used in this work, since the algorithmic principles behind the proposed detector are not specific to a particular type of room setting (see our future work suggestions in Section 8.2 of the conclusion chapter). On the other hand, we expect a limited gain when using our proposed detector in outdoor scenarios, where conventional CFAR techniques already perform well, due to their careful statistical modelling of the more predictable background clutter found in an outdoor setting.

The radar detector developed in the first part of this chapter will be used in the second part of this chapter for the design of a neuromorphic sensor fusion drone platform.

Part 2: Drone platform for RGB, DVS and radar data acquisition

3.7 Introduction to the second part

The first part of this chapter has presented the design of a low-complexity yet high-performance radar obstacle detection algorithm intended for use in indoor drone navigation scenarios. In this second part, the radar detection algorithm presented in the first part of this chapter will be used to build a first-of-its-kind sensor fusion drone platform combining RGB imaging, a Dynamic Vision Sensor (DVS) event-based camera and a 79-GHz FMCW radar running our custom detector presented in the first part of this chapter above. Our sensor fusion drone platform will then be used to acquire sensory data for the study of Simultaneous Localization and Mapping (SLAM) and People Detection in chapters 5 and 6.

In recent years, a number of datasets featuring DVS cameras for drones and other navigation platforms have been proposed [47, 95, 160] but none of them contain radar data. Yet, radar has extensively been used in automotive applications to increase the system robustness and thus its safety [168, 174]. A number of automotive datasets fusing a standard imaging camera with a FMCW radar have been proposed in literature [155, 183], none of them featuring DVS data. In contrast, our sensor fusion drone platform is the first one fusing RGB, DVS and radar modalities.

Within this context, the major contributions of the second part of this chapter are the following:

1. **Drone platform:** We construct a first-of-its-kind drone platform combining an RGB imager, a DVS event camera and a 79-GHz FMCW radar. In addition, we equip the drone with an altitude ranging sensor and a Raspberry Pi 4B onboard computer for logging the sensory data in real time during the drone flights.

2. **Data acquisition:** We use our drone setup to acquire a diverse range of datasets for studying both SLAM and people detection via SNNs in chapters 5 and 6.

Adapted from: *A. Safa at al., "Fail-Safe Human Detection for Drones Using a Multi-Modal Curriculum Learning Approach," in IEEE Robotics and Automation Letters, 2022.*

The second part of this chapter is organized as follows. Section 3.8 will give an overview of our drone platform and the various sensors mounted on it. Our dataset acquisition campain will be described in Section 3.9. Finally, conclusions will be provided in Section 3.10

3.8 RGB-DVS-radar sensor suite integration on a drone

In this section, we will first briefly introduce the sensing principle of the DVS camera used in our RGB-DVS-radar fusion setup. Then, we will describe the integration of each sensor on the drone.

3.8.1 Background on Dynamic Vision Sensor (DVS) camera

Compared to standard RGB imaging shutters, an event-based camera outputs binary events of polarity p_k for each independent pixel $\vec{x}_k$ in an asynchronous manner whenever the change in log-intensity at time t_k, $\Delta L(\vec{x}_k, t_k)$, crosses a certain threshold C [75]. The polarity p_k is positive if $\Delta L(\vec{x}_k, t_k)$ is increasing and *vice versa* for $\Delta L(\vec{x}_k, t_k)$ decreasing. It can be shown that events are generated by moving edges [75] (since edges have a high contrast $|\Delta L(\vec{x}_k, t_k)|$). Therefore, DVS data conveys information that is complementary to RGB data, which justifies their joint use in fusion settings.

For the sake of illustration, Fig. 3.22 conceptually shows how events are generated by a DVS camera. Fig. 3.22 a) shows that the camera outputs a continuous stream of spikes in both space and time, red spikes denoting positive events and blue spikes denoting negative events. Fig. 3.22 b) shows that each pixel x_{ij} fires a spike when its change in light log-intensity L_{ij} crosses a threshold C. The spike is positive if $\Delta L_{ij} > 0$ and negative vice versa.

3.8.2 Sensor suite description

Our sensor fusion suite is composed of a DAVIS-346 camera [27] simultaneously providing RGB frames at 30 FPS and event data with 1μs resolution. The DAVIS-346 also provides accelerometer and gyroscope data that we log as well during the flights. In addition, we use an AWR1443 79-GHz radar, which provides radar detection frames at 30 FPS using our radar processing and detection pipeline already described in the first part of this chapter. In contrast

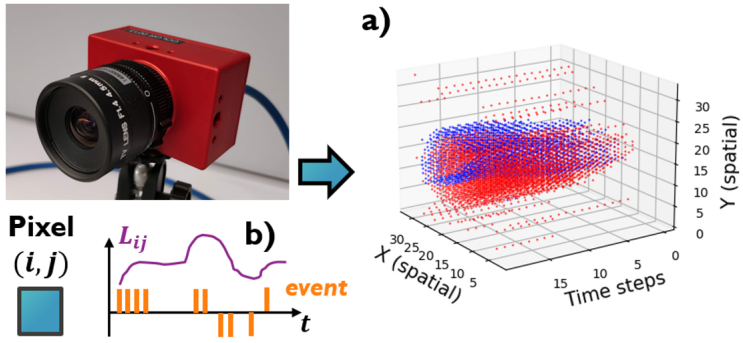

Figure 3.22: *Conceptual illustration of a DVS event-based camera.*

to a number of works that first acquire the raw radar ADC data and post-process them in an external computing unit [77,273], we perform radar detection directly in the embedded radar chip MCU using our custom detector described above, in the first part of this chapter.

By performing detection in the embedded radar MCU rather than in an external computing unit, we 1) drastically reduce the data bandwidth between the radar and the central computing unit, 2) remove the need for additional FPGA-based modules (e.g., DCA1000EVM) used to translate radar signals to gigabit Ethernet, making our setup less bulky, significantly cheaper (saving $\sim$ 500$) and much less energy consuming (saving $>$ 3 W) compared to setups using those FPGA-based modules. Finally, we use a TF-MINI range sensor to log the drone altitude at $\sim$ 20 FPS. The radar is time-synchronized with the DAVIS-346 camera by sending a pulse signal to the *synch input* of the camera when the radar acquisition cycle starts. Finally, a Raspberry Pi 4B is used to log the data coming from all sensors. We mount this sensor suite on a *NXP HoverGames* drone platform. Fig. 3.23 shows the complete drone-sensors setup.

3.9 Dataset acquisition

3.9.1 People detection dataset

With this drone-sensors setup, we have acquired a first dataset intended for the study of robust people detection algorithms, in order to provide safety in environments where humans and drones move around side by side. The data are captured in different locations with different backgrounds, from simple (e.g.,

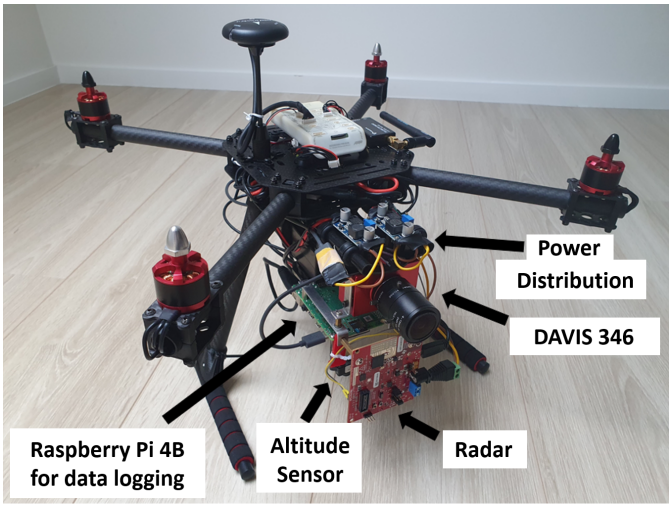

Figure 3.23: *Our drone setup with the sensor suite mounted on it, as indicated with the arrows.*

wall background) to more challenging (e.g., *aisles* background). The dataset features 6 distinct human subjects in order to capture variations in clothes color, shape, walking style and so on. In all acquisitions, the human subject is asked to walk arbitrarily in order to capture a large number of situations (human crossing, coming close, turning, and so on). The drone is either set to hover with light movements (to make the data more challenging) or to fly straight. The radar parameters are given in Table 3.3. Tables 3.4 gives an overview of the acquired data.

θ_{res}	d_{res}	d_{max}	v_{min}	v_{res}	v_{max}
15°	17.8 cm	9.11 m	0.36 m/s	0.12 m/s	3.84 m/s

Table 3.3: *Radar parameters: θ_{res} is the AoA resolution, d_{res} is the range resolution, d_{max} is the maximum range that can be sensed, v_{min} is the Doppler velocity threshold, v_{res} is the velocity resolution, and v_{max} is the maximum velocity that can be sensed.*

During all our acquisitions, we reject detections having a Doppler velocity magnitude smaller than a threshold v_{min}. This enables an automatic, on-line rejection of the static background obstacles when the drone is in a (quasi) hover

Name	Background difficulty	Number of captures
1) h-wall	easy	2
2) h-benches	medium	2
3) h-shelves	medium	2
4) h-aisles	challenging	7
5) f-wall	easy	4
6) f-shelves-benches	medium	2
7) f-gate	medium	2
8) f-aisles	challenging	4
9) f-aisles-gate	challenging	2
10) f-aisles-wall	challenging	3

Table 3.4: *Hover (h) and fly (f) sets with 6 different walking subjects (30 acquisitions in total).*

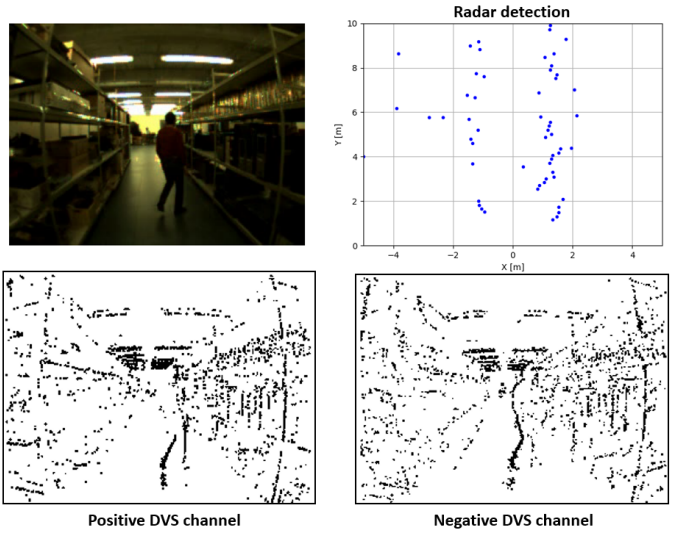

Figure 3.24: *Snippet from the f-aisles-1 acquisition. The static background has a non-zero Doppler velocity with regard to the drone as the drone is flying forward. Detections from the right and left shelves are thus returned and can be used for obstacle avoidance.*

state, which only requires the sensing of moving agents that can collide with the hovering drone (as static obstacles cannot collide with a static drone).

On the other hand, this filtering still allows the detection of static obstacles

when the drone is moving (see Fig. 3.24), as the background has a non-zero Doppler velocity with regard to the drone in this case (moving obstacles with a velocity close to the drone velocity are also detected thanks to the fine-grain velocity resolution of the radar).

The people detection data acquired in this section will be used in chapter 6 to study how an SNN equipped with STDP plasticity can be used to learn to detect people from DVS event data streams.

In addition, Appendix B provides the design of a *baseline* CNN human detection setup using the memory-efficient SqueezeNet architecture [100]. This will be used in chapter 6 to compare the SNN-STDP performance to the CNN performance on the task of DVS-based human detection from drones.

3.9.2 Dataset for Simultaneous Localization and Mapping (SLAM)

The sensor fusion drone setup in Fig. 3.23 has been used to acquire datasets for the study of Simultaneous Localization and Mapping (SLAM). To do so, we fly the drone in a warehouse environment (see Fig. 3.25, where an example flight path is illustrated), passing through storage aisles and open spaces.

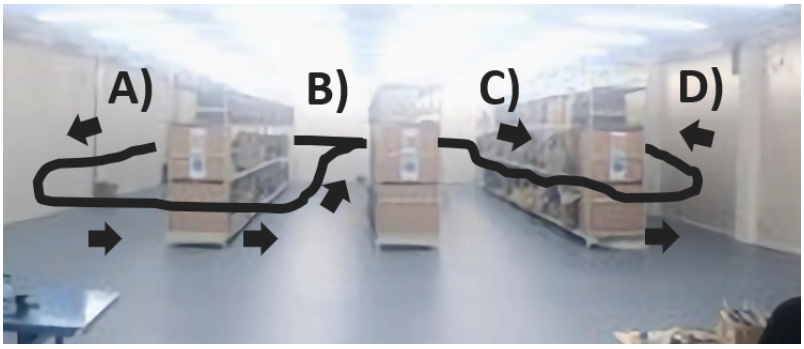

Figure 3.25: *An example drone flight path in the indoor warehouse environment where we collect data for the SLAM experiments.*

In addition, Table 3.5 gives an overview of the acquired flights. Flight 1) was acquired by flying between the wall (A in Fig. 3.25) and the racks (B. in Fig. 3.25), which makes the SLAM problem easier since the visited environment is more diverse and hence, easier to disambiguate. Flight 2) was acquired by flying between the racks only (B and C in Fig. 3.25), which leads to a larger amount

of similar and ambiguous views compared to flight 1). Flight 3) combines the paths of flight 1) and 2). Flight 4) follows a similar path as flight 3) but was acquired while *randomly turning the lights on and off* in the room, which makes the SLAM problem significantly more challenging. Finally, flight 5) covers the complete warehouse (A, B, C, and D in Fig. 3.25).

Flight	Description	Difficulty
1)	Flying between wall and racks, around 1 aisle	easy
2)	Flying between racks only, around 1 aisle	medium
3)	Combining 1) and 2), around 2 aisles	medium
4)	Same path as 3) while *randomly turning lights on/off*	hard
5)	Flying the complete warehouse, around 3 aisles	hard

Table 3.5: *SLAM flight acquisitions.*

The flight sequences of Table 3.5 acquired in this section will be used in chapter 5 to study how an SNN equipped with STDP plasticity can be used to perform SLAM in a continual unsupervised learning setting.

The contributions provided in this Section can be summarized as:

Our first-of-its-kind drone platform, equipped with a DAVIS-346 event camera and with a 79-GHz FMCW radar, is used to acquire data for the study of Simultaneous Localization and Mapping (SLAM), and people detection from drones.

3.10 Conclusion to the second part

The second part of this chapter has described the design of a first-of-its-kind sensor fusion drone platform combining RGB imaging, DVS event camera and a 79-GHz FMCW radar running our custom radar detector algorithm presented before in the first part of this chapter. After explaining the working principle of each sensor modality, the drone platform has been used to acquire a large number of sensory data acquisitions for the study of SLAM and people detection via SNNs. In the next chapter (chapter 4), novel theoretical foundations for the design of SNN-STDP systems will be provided, which will then be used in chapters 5 and 6 for studying SLAM and DVS people detection using the data acquired in this chapter.

Chapter 4

A top-down approach to SNN-STDP networks

4.1 Introduction

The previous chapter has introduced our drone data acquisition platform for acquiring joint RGB, DVS and radar data. This setup will be used to study the applicability of SNN algorithms equipped with Spike-Timing-Dependent Plasticity (STDP) learning for performing useful drone tasks. This chapter will investigate the formulation of a *top-down*, optimisation-based approach to the design of SNN-STDP networks, in order to enhance the SNN-STDP performance compared to prior *bottom-up* approaches (such as LSMs), that mostly rely on the *empirical emulation* of a cortical neural substrate, modelled through *randomly-connected* network designs [42, 78, 104, 142].

Studying how SNNs can learn through the biologically plausible STDP rule (as empirically observed in the brain [252] and as described in chapter 1 before) is important both from a neuroscience research perspective and from an engineering perspective. Indeed, the development of SNN-STDP systems is expected to enable a generation of ultra-low-power and -area artificial neural systems [59],

Adapted from: *A. Safa at al. "Event Camera Data Classification Using Spiking Networks with Spike-Timing-Dependent Plasticity," 2022 IEEE International Joint Conference on Neural Networks (IJCNN).*

integrating unsupervised, on-line learning in the sensor chips themselves, thus reducing the memory access latency (which is crucial for enabling real-time network adaptation to the environment) and preserving the privacy of user data [69].

On the other hand, the bottom-up STDP-driven SNNs reported in literature [42, 78, 104, 142] are currently less competitive (i.e., reaching lower accuracies and harder to train) compared to the use of conventional backprop learning methods:

1. They only have been demonstrated on a significantly simpler and smaller number of applications, often reaching inferior inference performances (e.g., as classification accuracy) compared to non-STDP methods [78, 104, 142];
2. Designing SNN-STDP systems currently involves the careful hand-tuning of a large number of hyper-parameters (STDP parameters, neuron thresholds, time constants and so on) [42];
3. It is often unclear which exact *optimization objective* STDP learning is solving, adding uncertainty to their design and behavior [104].

Here, as described in detail in chapter 1 before, it is argued that the main limiting factor for the design of high-performance SNN-STDP systems is the lack of a top-down, *optimization-based* approach that bridges the gap between conventional feature learning and the biologically-plausible SNN-STDP setting.

Earlier work has shed light on the existence of links between unsupervised SNN-STDP architectures and conventional machine learning methods such as PCA, dictionary learning and auto-encoders [2, 58]. Motivated by those earlier discussions, this chapter provides the following contributions:

1. **Algorithm:** An optimization-based framework is derived for describing SNN-STDP network substrates through the theory of *joint dictionary learning and basis pursuit* (DLBP) [19], widely used in unsupervised learning and sparse coding settings.

   New theoretical grounds will also be derived for SNN-STDP *hyperparameter tuning*, which reduces the need for extensive empirical parameter search traditionally used in prior SNN-STDP systems.

2. **Experiments:** In order to demonstrate the usefulness of the proposed SNN-STDP approach, our methods will be used to form SNN-STDP *feature descriptors* for the classification of DVS camera data. The SNN-STDP system will be assessed on the N-MNIST [178] and IBM DVS128 Gesture [9] datasets for pattern and gesture recognition. Using

the proposed methods, significant accuracy improvements are reported against previously proposed SNN-STDP systems (+9.3% on N-MNIST and +7.74% on IBM DVS128 Gesture).

This chapter is organized as follows. Section 4.2 will provide background information about the joint dictionary learning and basis pursuit (DLBP) method, forming the basis of the proposed SNN-STDP approach. Section 4.3 will present our optimization-based SNN-STDP methods. Section 4.4 will provide new theoretical grounds for STDP hyper-parameter tuning and STDP learning termination. Section 4.5 will show how the proposed SNN-STDP network can be used as feature descriptor for DVS camera data. Finally, section 4.7 will conclude this chapter.

4.2 Background theory

In this thesis, we propose to look at the SNN-STDP problem through the prism of the joint Dictionary Learning and Basis Pursuit (DLBP) theory [185]. This choice is motivated by the fact that usual SNN-STDP ensembles operate by *i)* providing some form of competition between neurons during inference, and *ii)* learning in an unsupervised way a set of feature-bearing weights from the input data (see chapter 1 section 1.2 for a detailed background description of SNNs and STDP learning). Here, we observe that these two points *i) and ii)* qualitatively correspond to what a DLBP system does. Indeed, a DLBP system is also *i)* inferring output vectors through some form of sparsity-defining constraint [43], creating competition between the elements of the output vector, and *ii)* learning a set of filters (i.e., the dictionary) in an unsupervised way. The DLBP optimisation process can be written as [185]:

$$C, \Phi = \arg\min_{C,\Phi} \sum_{k=1}^{K} \frac{1}{2}||\Phi\bar{c}_k - \bar{s}_k||_2^2 + \lambda_1||\bar{c}_k||_1 \tag{4.1}$$

which seeks to jointly learn a dictionary Φ of size $N \times M$ and infer output codes $\bar{c}_k$ of size M from a dataset of N-dimensional vectors $\bar{s}_k, \forall k$ (where $C = [\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_K]$ contains all output vectors $\bar{c}_k$.). In addition, λ_1 is a sparsity-defining hyper-parameter (the higher λ_1, the sparser the output $\bar{c}_k$). The first term in (4.1) seeks to minimize the re-projection error between the output sparse code $\bar{c}_k$ and the input $\bar{s}_k$ through the dictionary Φ. The second term provides l_1 regularization, controlling the sparsity of the codes.

The DLBP problem in (4.1) is classically solved by alternating between a stochastic gradient descent (SGD) step for the learning of Φ [185] and a *proximal*

descent step for the inference of $\bar{c}_k$ [43], with the proximal operator to the l_1 norm defined by:

$$\text{Prox}_{\lambda_1||.||_1}\{\bar{x}\}_i = \text{sign}(x_i)\max(|x_i| - \eta_1\lambda_1, 0) \tag{4.2}$$

Algorithm 3 summarizes the alternating-descent procedure classically used to solve (4.1) [185].

Algorithm 3 joint Dictionary Learning and Basis Pursuit

Input: $\bar{s}_k\ \forall k$: input vector stream, η_c: coding rate, η_d: learning rate, λ_1: regularization, N_{it}: number of iterations.
Init.: $\Phi_{ij} \leftarrow \mathcal{N}(0, \sigma_w)$ (zero-mean normal distribution with std. deviation $\sigma_w \sim 0.01$), $\bar{c} \leftarrow 0$

for $j \in \{1, ..., T_{end}\}$ (feed data sequence) **do**
  for $i = 1, ..., N_{it}$ (local coding iterations) **do**
    $\bar{c}_j \leftarrow \textbf{Prox}_{\eta_c\lambda_1||.||_1}\{\bar{c}_j - \eta_c\Phi^T(\Phi\bar{c}_j - \bar{s}_j)\}$ // see (4.2)
  end for
  for $i = 1, ..., N_{it}$ (local learning iterations) **do**
    $\Phi \leftarrow \Phi - \eta_d(\Phi\bar{c}_j - \bar{s}_j)\bar{c}_j^T$
  end for
end for

Therefore, in order to consolidate an optimisation-based description of STDP learning, we will analyse in section 4.3 the SNN-STDP problem under the light of the DLBP formulation.

4.3 SNN-STDP methods

First, this section will provide a high-level view of the proposed DLBP-based SNN-STDP architecture. Then, it will formally be shown how the proposed SNN-STDP topology is able to solve the DLBP problem, thus acting as a *spiking substrate* for solving DLBP.

4.3.1 High-level view of the proposed SNN-STDP topology

The proposed SNN-STDP architecture shown in Fig. 4.1 is composed of two distinct neural layers:

- A *coding* layer where the spiking activity of the neurons correspond to the DLBP output $\bar{c}_k$ in terms of their local spiking *rate* (i.e., the spike rate defined over a local time window). In addition, the coding layer possesses two distinct sets of weights:

  1. the *input* weight matrix Ω, which converges to the *transposed* DLBP dictionary matrix Φ^T in (4.1) during STDP learning,
  2. the *recurrent* weight matrix W, which converges to the matrix $(\eta_1 \Phi^T \Phi - I_M)$ in the DLBP Algorithm 3 (line 3) during STDP learning (this recurrence is needed for implementing topologically the iterative sparse coding process executed by DLBP).

- An *error* layer where the spiking activity of the neurons corresponds to the DLBP error vector $\bar{e}_k = \Phi \bar{c}_k - \bar{s}_k$ in terms of their local spiking *rate*. In addition, the error layer possesses one set of weights:

  1. the *feedback* weight matrix Ψ which converges to the DLBP dictionary matrix Φ in (4.1) during STDP learning.

Finally, both the coding and error layers receive the spiking input vector $\bar{s}(t)$ (where t denotes the time), corresponding to the DLBP input vector $\bar{s}_k$ in (4.1) (in terms of the local spiking *rate*).

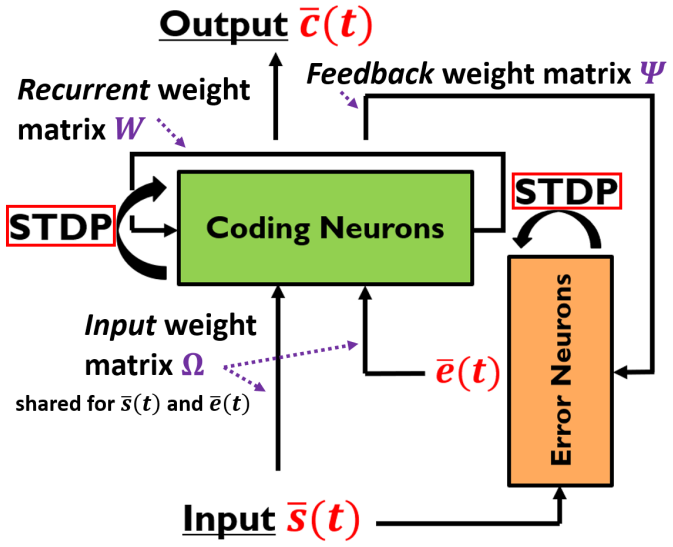

Figure 4.1: *High-level view of the SNN-STDP architecture.*

Furthermore, Algorithm 4 details the operations behind the SNN-STDP network. In general, we make the time dependence explicit as e.g., $\bar{x}(t)$ when we refer to a vector of spike trains, and we denote $\bar{x}$ when we refer to the spiking *rates*.

Algorithm 4 SNN-STDP for solving DLBP (4.1)

Input: $\bar{s}(t)$: input vector of spike trains (from DVS), η_1: SNN output convergence rate, η_2: learning rate, $\mu \leftarrow \eta_1 \times \lambda_1$: neuron threshold, $\tau_{m,s}$: LIF membrane and PSC time constants, $A_{+,-}, \tau_{+,-}$: STDP parameters

Output: $\bar{c}(t)$: output spike trains, Ω, W, Ψ : SNN weight matrices

Init.: Input and feedback weights: $\Omega_{ij}, \Psi_{ij} \leftarrow \mathcal{N}(0, \sigma_w)$ (zero-mean normal distribution with std. deviation $\sigma_w \sim 0.01$)

Recurrent weights: $W \leftarrow \eta_1 \Psi^T \Psi - I_M$ (I_M: identity)

1: **for** $t \in \{0, ..., T_{end}\}$ (feed DVS data in time $\bar{s}(t)$) **do**
2: **for** $i = 1, ..., N$ (neurons in the error layer) **do**
3: // Update error spikes $\bar{e}(t)$ and feedback weights Ψ
4: $J_i^e = \mathcal{PSC}\{\Psi_i \bar{c}(t) - s_i(t)\}$
5: $e_i(t) = LIF_{pp}\{J_i^e\}$ // i in Ψ_i means i^{th} row.
6: $\forall j\ \Psi_{ij} \leftarrow \Psi_{ij} - \eta_2 \Delta w_{\text{STDP}}\{e_i(t), c_j(t)\}$
7: **end for**
8: **for** $j = 1, ..., M$ (neurons in the coding layer) **do**
9: // Update coding spikes $\bar{c}(t)$ and weights Ω, W
10: $J_j^c = \mathcal{PSC}\{\Omega_j \bar{s}(t) - W_j \bar{c}(t)\}$
11: $c_j(t) = LIF_{pp}\{J_j^c\}$ // j in Ω_j, W_j means j^{th} row.
12: $f_j(t) = \tilde{W}_j \bar{c}(t) - \Omega_j \bar{e}(t) - \Omega_j \bar{s}(t)$
13: // Where $W \equiv \eta_1 \tilde{W} - I_M$ following (4.30)
14: $\forall i\ \Omega_{ij} \leftarrow \Phi_{ij} - \eta_2 \Delta w_{\text{STDP}}\{c_j(t), e_i(t)\}$
15: $\forall i\ \tilde{W}_{ij} \leftarrow \tilde{W}_{ij} - \eta_2 \Delta w_{\text{STDP}}\{f_j(t), e_i(t)\}$
16: **end for**
17: **end for**

In Algorithm 4, $\mathcal{PSC}$ denotes the use of a *post-synaptic current* filter (exponential decay) with time constant τ_s (1.3), and LIF_{pp} denotes the application of a *push-pull pair* of Leaky Integrate and Fire (LIF) neurons (1.1) as spiking nonlinearity:

$$LIF_{pp}\{J_{in}\} = LIF\{J_{in}\} - LIF\{-J_{in}\} \tag{4.3}$$

where J_{in} denotes the input current to the push-pull LIF pair. In section 4.3.2, it will be shown that the use of LIF_{pp} as neural units (i.e., positive and negative outputs) allows the network to perform an approximate LASSO regression as in DLBP. In addition, the use of LIF_{pp} is biologically motivated by empirical evidence about the existence of push-pull neural pairs in the cortex [79] and in retinal ganglions used to code light intensities [96].

The proposed SNN-STDP architecture of Fig. 4.1 solves (4.1) by *i)* inferring the next output code $\bar{c}$ using the current weights Ω, W, Ψ, and *ii)* learning the next weights Ω, W, Ψ using the current output $\bar{c}$, by iterating along the time dimension of the event-based data. During the training phase, we feed event-based acquisitions (e.g., from a DVS camera) one by one to the network. Each push-pull neuron LIF_{pp} possesses its own weight vector, which is modified locally during learning by the STDP mechanism.

The input *current* J^c_{in} to the coding neurons is wired according to:

$$\bar{J}^c_{in} = \mathcal{PSC}\{\eta_1 \Omega \bar{s}(t) - W\bar{c}(t)\} \tag{4.4}$$

making the coding neuron output $\bar{c}(t)$ converge to the solution of the LASSO sub-problem *i)* of (4.1), as shown in section 4.3.2.

In addition, the input current $\bar{J}^e_{in}$ to the *error* neurons is wired according to:

$$\bar{J}^e_{in} = \mathcal{PSC}\{\Psi\bar{c}(t) - \bar{s}(t)\} \tag{4.5}$$

and the spiking output of the error neurons $\bar{e}(t)$ is wired back to the *coding* neurons through a copy of the input weights Ω. By applying STDP to all weights, such topology solves the *dictionary learning* sub-problem *ii)* of (4.1), as will be demonstrated in Section 4.3.3.

4.3.2 Solving LASSO via the SNN dynamics

Now, it is shown how the coding neurons of the proposed SNN topology in Fig. 4.1 solve the LASSO part of (4.1). Considering a fixed $N \times M$ dictionary Φ featuring M atoms, the LASSO regression is written as [43]:

$$\bar{c} = \arg\min_{\bar{c}} \frac{1}{2}||\Phi\bar{c} - \bar{s}||_2^2 + \lambda_1||\bar{c}||_1 \tag{4.6}$$

with $\bar{c}$ the M-dimensional sparse code, $\bar{s}$ the N-dimensional input vector and λ_1 a hyper-parameter defining the number of non-zero entries (or *support*) Θ of $\bar{c}$ (the higher λ_1, the smaller Θ). The pursuit (4.6) is classically solved using the *iterative soft-thresholding algorithm* (ISTA) [43]:

$$\bar{c}^{k+1} = \text{Prox}_{\lambda_1||.||_1}\{\bar{c}^k - \eta_1\Phi^T\Phi\bar{c}^k + \eta_1\Phi^T\bar{s}\} \tag{4.7}$$

with η_1 the coding rate, k the iteration index and $\text{Prox}_{\lambda_1||.||_1}$ defined in (4.2).

Eq. (4.7) can be rearranged by identifying the SNN input weights $\Omega \sim \Phi^T$ and the recurrent weights $W \sim \eta_1\Phi^T\Phi - I_M$ in (4.7):

$$\bar{c}^{k+1} = \text{Prox}_{\lambda_1||.||_1}\{-(I_M - \eta_1\Phi^T\Phi)\bar{c}^k + \eta_1\Omega\bar{s}\} = \text{Prox}_{\lambda_1||.||_1}\{\eta_1\Omega\bar{s} - W\bar{c}^k\} \tag{4.8}$$

Hence, this shows why the *specific* SNN wiring pattern that was proposed in (4.4) is needed for making the coding layer solve ISTA (4.7).

Now, let $\bar{J}_{in}^{c}$ be the input current vector to the coding neurons defined in (4.4), which corresponds to the argument of $\text{Prox}_{\lambda_1||.||_1}$ in (4.7). We define the ith pair of LIF neurons in the coding layer as follows [193]:

$$\frac{d}{dt}\begin{bmatrix} V_p^i \\ V_n^i \end{bmatrix} = \frac{1}{\tau_m}\left(\begin{bmatrix} J_{in,i}^{c} \\ -J_{in,i}^{c} \end{bmatrix} - \begin{bmatrix} V_p^i \\ V_n^i \end{bmatrix}\right) \tag{4.9}$$

with V_p^i and V_n^i respectively denoting the ith push or pull *membrane potential* and τ_m denoting the membrane decay constant. Whenever V_p^i or V_n^i exceeds the neuron threshold μ, the corresponding push or pull neuron outputs a spike and the corresponding membrane potential is reset to 0:

$$\begin{bmatrix} \sigma_p^i \\ \sigma_n^i \end{bmatrix} = \mathbb{1}\{\begin{bmatrix} V_p^i \geq \mu \\ V_n^i \geq \mu \end{bmatrix}\} \tag{4.10}$$

where $\mathbb{1}\{.\}$ is the indicator function (equal to 1 if the condition holds true and 0 otherwise) and σ_p^i and σ_n^i respectively denote the ith push or pull output. To show how (4.9) solves (4.6), it follows from the LIF spike rate analysis in [53] that a first-order Taylor approximation around μ between the input $J_{in,i}$ and the push-pull spike rates $r_{p,n}^i$ of the ith *coding* neuron is given by:

$$\begin{bmatrix} r_p^i \\ r_n^i \end{bmatrix} \approx \frac{1}{\tau_m \mu}\begin{bmatrix} J_{in,i}^{c} - \mu \textbf{ if } J_{in,i}^{c} > \mu \textbf{ else } 0 \\ J_{in,i}^{c} + \mu \textbf{ if } J_{in,i}^{c} < -\mu \textbf{ else } 0 \end{bmatrix} \tag{4.11}$$

which is equivalent to $\text{Prox}_{\lambda_1||.||_1}$ (4.2) if:

$$\tau_m = \frac{1}{\mu} \tag{4.12}$$

in order to enforce a unit slope (see Fig. 4.2). To further illustrate this, Fig. 4.2 shows the simulation of a push-pull pair of LIF neurons approximating the proximal operator $\text{Prox}_{\lambda_1||.||_1}$ (4.2) in the *non-saturating* region of the output spike rate. The neuron threshold μ is set to 0.25 in this example and the constraint (4.12) is used to set τ_m.

Therefore, a layer containing M *coding* neurons acts as (4.2) in the spiking rate sense. Using (4.11) and (4.12), the output of the coding layer $\bar{c}$ in the SNN-STDP system can be written as:

$$\bar{c} = \bar{r}_p - \bar{r}_n \approx \text{Prox}_{\mu||.||_1}\{\bar{J}_{in}^{c}\} \tag{4.13}$$

Finally, it directly results from (4.7) that wiring the coding neurons as follows (where $\bar{c}(0)$ is initially null):

$$\bar{J}_{in}^{c} = \mathcal{PSC}\{\eta_1 \Omega \bar{s}(t) - W\bar{c}(t)\} \tag{4.14}$$

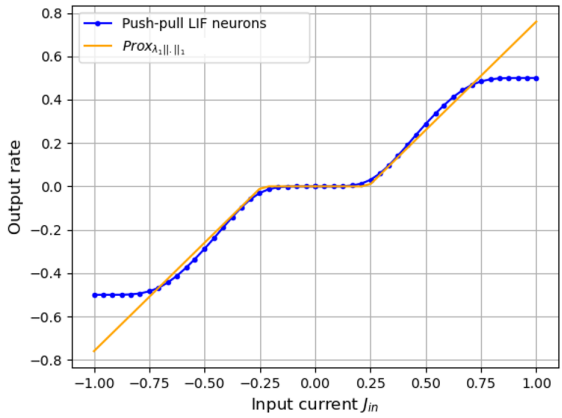

Figure 4.2: *A push-pull pair of LIF neurons approximates* $Prox_{\lambda_1||.||_1}$ *(4.2).*

with W converging to $\eta_1 \Phi^T \Phi - I_M$ during STDP learning (see section 4.3.3):

$$W \sim \eta_1 \Phi^T \Phi - I_M \tag{4.15}$$

and where I_M is the $M \times M$ identity matrix, makes the SNN behave as (4.7) during the iterations through time. This, in turn, makes the push-pull output spiking rates converge to the solution of (4.6).

4.3.3 Learning the network weights via STDP

At this point, it has been shown how the proposed SNN architecture solves LASSO. We still need to show how the SNN topology in Fig. 4.1 solves the dictionary learning part of (4.1). Let $\bar{c}$ be considered to be *fixed*, the dictionary learning problem can then be written as [185]:

$$\Phi = \arg\min_{\Phi} \frac{1}{2} ||\Phi \bar{c} - \bar{s}||_2^2 \tag{4.16}$$

Eq. (4.16) is traditionally solved by stochastic gradient descent (SGD):

$$\Phi^{k+1} = \Phi^k - \eta_2 (\Phi^k \bar{c} - \bar{s}) \bar{c}^T \tag{4.17}$$

where η_2 defines the learning speed. The remainder of this section will show how the descent (4.17) is implemented by the SNN-STDP system.

4.3.4 A rate-based STDP model

Noting $\tau = t_{post} - t_{pre}$ the time difference between a post- and a pre-synaptic spike, the value of a weight subject to STDP is modified following the classical double exponential STDP rule [24]:

$$\kappa(\tau) = \begin{cases} A_+ e^{-\tau/\tau_+}, & \text{if } \tau \geq 0 \\ -A_- e^{\tau/\tau_-}, & \text{if } \tau < 0 \end{cases} \tag{4.18}$$

with τ_+, τ_- the potentiation and depression time constants, and A_+, A_- the potentiation and depression weights. Given a pre- and a post-synaptic spike train $s_{pre}(t)$, $s_{post}(t)$, the weight modification of the synapse at the post-synaptic time instant t can be written as [161]:

$$\Delta w|_t = \eta_2 \int_{-\infty}^{\infty} s_{post}(t) s_{pre}(t-\tau)\kappa(\tau) d\tau \tag{4.19}$$

Under this form, (4.19) is not useful since the post- and pre-synaptic spike trains are not known beforehand. Taking the expected value of (4.19) over t leads to:

$$\mathbb{E}\{\Delta w|_t\} = \eta_2 \int_{-\infty}^{\infty} \mathbb{E}\{s_{post}(t) s_{pre}(t-\tau)\}\kappa(\tau) d\tau \tag{4.20}$$

The right-hand side of (4.20) can be re-written as:

$$\mathbb{E}\{\Delta w|_t\} = \eta_2 \int_{-\infty}^{\infty} \{r_{post} r_{pre} + \mathcal{C}(\tau)\}\kappa(\tau) d\tau \tag{4.21}$$

where $\mathcal{C}(\tau)$ is the covariance between the pre- and post-synaptic spike train and r_{post}, r_{pre} are the average post- and pre-synaptic spiking rates. We further rearrange (4.21) into a rate-only term and a covariance term:

$$\Delta w_{\text{STDP}} \equiv \mathbb{E}\{\Delta w|_t\} = \eta_2(A_+\tau_+ - A_-\tau_-) r_{post} r_{pre} + \eta_2 \int_{-\infty}^{\infty} \mathcal{C}(\tau)\kappa(\tau) d\tau \tag{4.22}$$

At this point, it is interesting to evaluate the contribution of each term on synaptic plasticity. Fig. 4.3 experimentally compares the weight change induced by STDP Δw_{STDP} (4.22) against the weight change due to the rate-based term only $\Delta w_{\text{rate}} = \eta_2(A_+\tau_+ - A_-\tau_-) r_{post} r_{pre}$. Fig. 4.3 was obtained by recording the accumulated weight change of the input synapses Ω to the *coding* layer for 300 time steps using the N-MNIST dataset [178]. It can clearly be noted in Fig. 4.3 that the rate-based term of (4.22) is the main driver of the *long-term* STDP weight change. Indeed, the error of the rate-only model with regard to the true STDP mechanism is $\frac{1}{MN}\sum_{i=1}^{MN} |\Delta w^i_{\text{STDP}} - \Delta w^i_{\text{rate}}| \approx 2 \times 10^{-5}$. This indicates

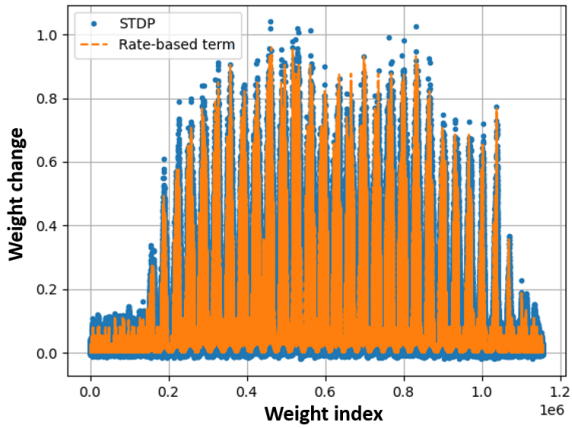

Figure 4.3: *Synaptic modification due to STDP vs. the rate-only weight change model (4.23).*

that the rate-based model can be used for simplifying the analytical study of STDP in the remaining derivations.

Therefore, the full STDP mechanism Δw_{STDP} is approximated as follows in the remainder of the *derivations* (even though full STDP is indeed used in the *actual implementation* of the proposed SNN-STDP network):

$$\Delta w_{\text{STDP}}\{s_{post}, s_{pre}\} \approx \eta_2(A_+\tau_+ - A_-\tau_-)r_{post}r_{pre} \tag{4.23}$$

where it is assumed (without loss of generality) that:

$$A_+\tau_+ - A_-\tau_- = 1 \tag{4.24}$$

in order to not modify the resulting learning speed (setting $A_+\tau_+ - A_-\tau_- > 0$ is sufficient for the learning effect to take place).

Using (4.23), it will now be shown how the SNN weights can learn in order to converge in a related manner to the DLBP dictionary Φ (input weights $\Omega \sim \Phi^T$, recurrent weights $W \sim \eta_1\Phi^T\Phi - I_M$ and feedback weights $\Psi \sim \Phi$).

4.3.5 Using STDP for dictionary learning

Learning the input weights Ω

A *coding* neuron has in its possession one row i of the input weight matrix Ω, noted Ω_i. Each element j of Ω_i is modified using the local STDP update rule:

$$(\Omega_i)_j \leftarrow (\Omega_i)_j - \Delta w_{\text{STDP}}\{c_i(t), e_j(t)\} \tag{4.25}$$

Using (4.23) and (4.24), we can re-write (4.25) as:

$$(\Omega_i)_j \leftarrow (\Omega_i)_j - \eta_2 r\{c_i(t)\} r\{e_j(t)\} \tag{4.26}$$

where $r\{.\}$ denotes the spike rate (similar to r_{post}, r_{pre} used earlier). Since our SNN architecture in Fig. 4.1 ensures:

$$r\{e_j\} = r\{\Phi_j \bar{c}(t) - s_j(t)\} \tag{4.27}$$

it can clearly be seen that (4.25) acts in an equivalent way to the SGD-based dictionary learning step (4.17) under the STDP rate-based approximation (4.23).

Learning the feedback weights Ψ

An *error* neuron has in its possession one row j of the input weight matrix Ψ, noted Ψ_j. Each element i of Ψ_j is modified using the local STDP update rule:

$$(\Psi_j)_i \leftarrow (\Psi_j)_i - \Delta w_{\text{STDP}}\{e_j(t), c_i(t)\} \tag{4.28}$$

It follows directly from the earlier discussion on input synapse modification (4.25) that (4.28) acts in an equivalent way to the SGD-based dictionary learning step (4.17) for the weights in the error layer.

Learning the recurrent weights W

In order to learn the recurrent weights W, we adapt the *consistency-enforcing* method proposed in [2]:

$$\tilde{W} = \arg\min_{\tilde{W}} \frac{1}{2} ||(\tilde{W} - \Phi^T \Phi)\bar{c}||_2^2 \tag{4.29}$$

to our custom use of the STDP rule, where:

$$W \equiv \eta_1 \tilde{W} - I_M \tag{4.30}$$

Eq. (4.29) is traditionally solved by gradient descent as:

$$\tilde{W}^{k+1} = \tilde{W}^k - \eta_2(\tilde{W}^k - \Phi^T\Phi)\bar{c}\bar{c}^T \tag{4.31}$$

Each *coding* neuron i has access to the ith row of W and Ω only. In addition, the coding neuron i receives as inputs the spike train vectors $\bar{s}(t)$, $\bar{c}(t)$ and $\bar{e}(t)$. It is therefore possible to *locally* compute an internal state $f_i(t)$ representing $(\tilde{W} - \Phi^T\Phi)_i\bar{c}$ for each coding neuron i:

$$f_i(t) = \frac{1}{\eta_1}(W + I_M)_i\bar{c}(t) - (\Phi^T)_i\bar{e}(t) - (\Phi^T)_i\bar{s}(t) = (\tilde{W} - \Phi^T\Phi)_i\bar{c}(t) \tag{4.32}$$

since the *error layer* wiring (4.5) ensures (where r denotes the spike rate):

$$r\{\bar{e}(t) + \bar{s}(t)\} = r\{\Phi\bar{c}(t)\} \tag{4.33}$$

It is now clear that the following iterative STDP updates (4.34):

$$(\tilde{W}_i)_l \leftarrow (\tilde{W}_i)_l - \Delta w_{\text{STDP}}\{f_i(t), c_l(t)\} \tag{4.34}$$

with $i = 1, .., M$ spanning the rows and $l = 1, ..., M$ spanning the columns of $\tilde{W}$, act in a similar way as (4.31), making $\tilde{W}$ converge to $\Phi^T\Phi$ (4.29). This in turn makes W converge to $\eta_1\Phi^T\Phi - I_M$ following (4.30), which is needed for solving LASSO via the SNN coding layer (4.8).

In conclusion, the above derivations have shown that, under the local rate-based assumption, the proposed SNN-STDP network *jointly* solves the LASSO basis pursuit (4.6) and the dictionary learning (4.16) problems, making the SNN-STDP system solve the DLBP problem (4.1). It is also important to note that the derivations proposed above were utilizing the local spiking rate (i.e., the rate computed on a small local window of time) instead of pure rate coding or latency coding. As such, the importance of both time and rate are captured by the provided SNN-STDP theory, even though the small time constant (a few milliseconds) used in the PSC filters (4.4) clearly brings these techniques effectively closer to the time-domain paradigm.

In summary, the novel methods provided in this Section can be summarized as:

Compared to prior bottom-up approaches, our top-down approach provides a new look at SNN-STDP networks, as a substrate of computation that provably solves the joint Dictionary Learning and Basis Pursuit problem through its neural and synaptic dynamics via STDP learning.

4.4 Network hyperparameter study and methods

4.4.1 How to tune the STDP parameters?

In prior SNN-STDP networks, tuning the STDP learning parameters A_+, A_-, τ_+, τ_- is typically done in an empirical manner, by resorting to lengthy grid searches or trial-and-error approaches [42]. In this section, a *spectrum matching* method is proposed for accelerating the tuning of the STDP parameters, by finding a constraint linking A_+, A_-, τ_+, τ_- together. By exploiting this constraint, it is possible to determine the value of one of the STDP parameters when the three others are fixed.

Indeed, (4.19) indicates that STDP acts as a *kernel* $\kappa(\tau)$ filtering the pre-synaptic spike train $s_{pre}(t)$ over a certain bandwidth $B\{A_+, A_-, \tau_+, \tau_-\}$ (dependent on A_+, A_-, τ_+, τ_-). Therefore, it must be ensured that the pass-band of the STDP kernel is matching the information-bearing part of the $s_{pre}(t)$ spectrum while filtering noise and spurious frequencies caused by distortion (due to the discontinuous nature of spike trains).

By taking the Fourier transform of (4.18), we obtain:

$$\mathcal{F}\{\kappa(t)\} = \frac{A_+}{\tau_+^{-1} + j\omega} - \frac{A_-}{\tau_-^{-1} - j\omega} \tag{4.35}$$

Taking the magnitude of (4.35), we obtain:

$$|\mathcal{F}\{\kappa(t)\}| = \frac{A_+\sqrt{(\tau_-^{-1} - \alpha\tau_+^{-1})^2 + (1+\alpha)^2\omega^2}}{\sqrt{\tau_+^2 + \omega^2}.\sqrt{\tau_-^2 + \omega^2}} \tag{4.36}$$

with $\alpha = A_-/A_+$. When analysing (4.36), two poles $\omega_{1,2}$ and one zero z_1 can be identified:

$$\omega_1 = \tau_+^{-1} \quad \omega_2 = \tau_-^{-1} \quad z_1 = \frac{\tau_-^{-1} - \alpha\tau_+^{-1}}{1+\alpha} \tag{4.37}$$

Therefore, (4.19) and (4.37) show that the double exponential STDP rule acts as a *first-order bandpass* filter applied to $s_{pre}(t)$. Under the *local spiking rate* assumption, the information carried by a spike train $s(t)$ is considered to be its *mean* spiking rate over a certain time window δT and hence, resides in and near the DC bin $\mathcal{F}\{s(t)\}|_{\omega=0}$ (higher frequency bins containing distortion and noise). Consequently, the STDP kernel must avoid cutting the DC frequency while attenuating AC bins.

This has been verified in Fig. 4.4 when using real-world *event-based camera* data, where the spike train bandwidth covers a narrow information band around

DC. Consequently, the same discussion holds: the STDP kernel must not cut the spectra near DC.

By pole-zero cancellation of ω_1 and z_1 in (4.36) and (4.37), a nearly flat STDP spectrum around DC can be obtained, leading to the constraint:

$$\tau_+^{-1} = \frac{\tau_-^{-1} - \alpha\tau_+^{-1}}{1+\alpha} \tag{4.38}$$

Fig. 4.4 shows both a *matching* STDP kernel, complying with (4.38), and a *non-matching* one, over spectra acquired from DVS camera data. The DVS spectra were acquired by applying the discrete Fourier transform (DFT) to the spike train of each pixel in a data point from the N-MNIST dataset [178] (DFT is applied with Hanning windowing in order to reduce spectral leakage [88]). In order to obtain the *matching* STDP kernel in Fig. 4.4, ω_2 is visually tuned such that the -3dB point covers the information-bearing band (with the example parameters $A_+ = 1, A_- = 0.8, \tau_+ = 0.0208\text{s}, \tau_- = 0.008\text{s}$). On the other hand, the non-matching STDP kernel ($A_+ = 1, A_- = 0.8, \tau_+ = 0.008\text{s}, \tau_- = 0.008\text{s}$) does not comply with the constraint (4.38) and degrades the information-bearing band of the event data.

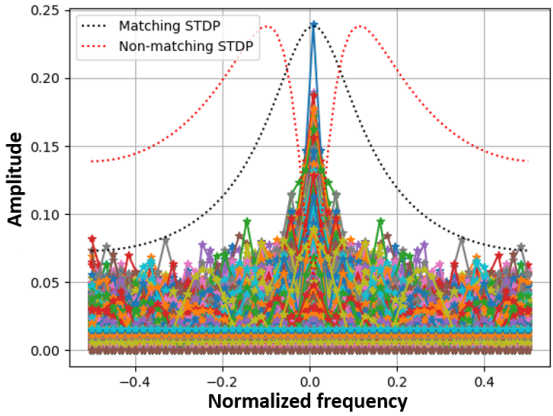

Figure 4.4: *Spectra of the DVS data and STDP kernels. The non-matching STDP kernel strongly attenuates the information-bearing bandwidth of the DVS spectra, leading to the learning of noisier weights (see Fig. 4.5), while this is not the case for the matching kernel.*

Since the non-matching STDP spectrum in Fig. 4.4 significantly attenuates the information-bearing bands while passing noise, we expect the *non-matching* STDP kernel to lead to the learning of *noisier* dictionary weights Φ_i compared

to the *matching* STDP case, degrading the overall quality of the weights learned. Indeed, even though the use of intrinsic noise has been studied in a number of works attempting to explore its impact both from a neuroscience and generative point of view [172], adding noise would not bring additional benefits in the specific case of our DLBP-based SNN-STDP design, since noise will make the dynamics of the proposed SNN-STDP diverge from the DLBP dynamics (4.1) that we seek to solve (as opposed to e.g., DNNs where adding noise during training partially helps overcome the local optimum trapping). On the other hand, it is important to mention that noise plays an important role in biology as it provides a generative capability to the cortical neural ensembles, and can also act as a regularization technique against over-fitting [172].

Fig 4.5 experimentally confirms this discussion and the usefulness of the constraint (4.38) when setting the STDP parameters. Indeed, it can be noted in Fig. 4.5 that the *matching* STDP kernel leads to crisp atoms, mostly capturing feature signals (Fig. 4.5 a) while the *non-matching* STDP kernel leads to fuzzy atoms containing a significant amount of noise (Fig. 4.5 b). The weights shown in Fig 4.5 have been learned by using the matching and non-matching parameters of the kernels shown in Fig. 4.4, while keeping all other learning parameters constant during the experiments.

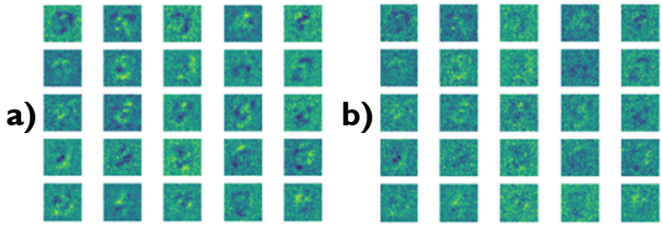

Figure 4.5: *SNN weight subset learnt on N-MNIST: a) matching STDP kernel leads to crisp atoms; b) non-matching STDP kernel leads to noisy atoms.*

4.4.2 How to terminate STDP training?

Detecting when STDP training starts to over-fit and therefore, when it must be stopped, is a key issue rarely mentioned in previous STDP work [142]. Detecting when STDP must be stopped is not only important for avoiding overfitting, but also for reducing latency and energy overheads when implemented in neuromorphic online learning hardware. We believe that this omission has been due to the lack of a link between the SNN dynamics and a clearly formulated objective function describing the quantity that the SNN-STDP system seeks to

optimize. In contrast, the proposed SNN-STDP network in Fig. 4.1 is linked to the DLBP optimization problem (4.1) and enables us to exploit the activity of the neurons in the *error* layer for computing the re-projection error $||\Phi\bar{c} - \bar{s}||_2^2$ of the DLBP objective.

The error layer encodes the DLBP re-projection error as a bipolar spike train vector of push-pull LIF neurons:

$$\begin{cases} \bar{e}^p(t) = LIF\{\bar{J}_{in}^e\} \\ \bar{e}^n(t) = LIF\{-\bar{J}_{in}^e\} \end{cases} \tag{4.39}$$

Hence, the error metric:

$$\mathcal{L}_{\text{in}} = ||r\{\bar{e}^p(t) - \bar{e}^n(t)\}||_2 \tag{4.40}$$

can be considered as a measure of the *model fitting* performance, with $r\{.\}$ denoting the average spiking rate and $\mathcal{L}_{\text{in}}$ referred to as the *inner loss* of the SNN-STDP model.

Thanks to the availability of the inner loss $\mathcal{L}_{\text{in}}$, a conventional cross-validation approach can now be followed in order to detect when STDP training must be terminated (i.e., when the training process is starting to over-fit). In practice, we use a small subset of the training data (~ 10 examples determined heuristically) as the *validation* set that we periodically use to measure the *validation* inner loss $\mathcal{L}_{\text{in}}$ of the model (e.g., by periodically cycling between 9 training examples for 1 validation pass).

As an illustration, Fig. 4.6 shows the inner loss evolution during the training of the proposed SNN-STDP system on the IBM DVS128 gesture dataset [9]. During STDP learning, the activity of the error neurons $\bar{e}(t)$ is tracked as $||r\{\bar{e}(t)\}||_2$ and used to terminate the STDP training process when the inner loss reaches a plateau:

$$\frac{1}{n_\epsilon} \sum_{k=e-n_\epsilon}^{e} |\mathcal{L}_{\text{in}}^k - \mathcal{L}_{\text{in}}^{k-1}| < \epsilon \tag{4.41}$$

where e denotes the current epoch, k is the epoch index, $n_\epsilon \sim 10$ is the number of epochs along which the inner loss convergence is computed, and $\epsilon \sim 10^{-3}$ is the termination threshold.

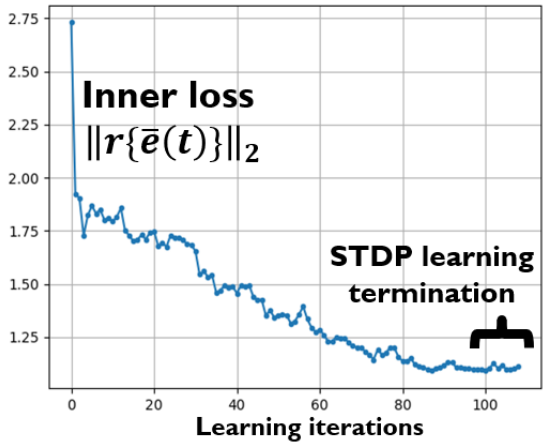

Figure 4.6: *Evolution of the inner loss function during SNN-STDP training on the IBM DVS128 gesture dataset.*

4.5 Using the SNN-STDP as feature descriptor for classification tasks

This section will describe how the SNN-STDP system can be leveraged to build various feature descriptors used to encode the input event data into a *feature space* where *linear separability* between data classes is enhanced. A standard classifier such as a linear *Support Vector Machine* (SVM) can then be used for classifying the *feature vectors* at the output of the SNN.

4.5.1 Global feature descriptor

The SNN-STDP network in Fig. 4.1 can be used to learn *global* feature descriptors for e.g., event camera data by feeding the event data as flattened spiking vectors to the SNN. The descriptor is global in the sense that it translates the whole input vector into a single descriptor vector encoding the input at once (vs. local descriptors such as SIFT, SURF, etc [111]).

The spatio-temporal feature descriptor $\bar{\mathcal{F}}_g$ is obtained during inference by recording the average firing rate of the neurons in the coding layer $r\{\bar{c}(t)\}$, and normalizing the resulting vector to *unit norm* in order to provide *invariance* to

the event density:

$$\bar{\mathcal{F}}_g = \frac{r\{\bar{c}(t)\}}{||r\{\bar{c}(t)\}||_2} \tag{4.42}$$

4.5.2 Action descriptor

It is possible to extend the global feature descriptor described in section 4.5.1 above for recognising *time-varying* signals such as macro actions and gestures performed in front of an event camera. Compared to *saccades* (i.e., vibrations of the event camera used for e.g., the N-MNIST dataset acquisitions [178]), macro actions cause important oscillations in the rate of the output spike trains of the SNN, due to the long-term periodicity of the performed gestures (such as a hand rotation).

Therefore, capturing the *correlations* between the oscillating local spike rates of coding neurons conveys important information about the performed actions. In order to do so, we concatenate (denoted by the $\oplus$ sign) the Pearson correlation matrix ρ (i.e., the normalized covariance matrix) of $r\{\bar{c}(t)\}$ with the matrix P obtained by taking the outer product of the output mean rate vector with itself $P = r\{\bar{c}(t)\}r\{\bar{c}(t)\}^T$ as follows (the correlations in ρ being computed along the time domain, capturing changes in spike rate):

$$\bar{\mathcal{F}}_{ac} = \frac{\rho_{\text{tri}} \oplus P_{\text{tri}}}{||\rho_{\text{tri}} \oplus P_{\text{tri}}||_2} \tag{4.43}$$

where the *tri* sign denotes the upper triangular part of the matrix (since both ρ and P are symmetrical matrices). Doing so, the action descriptor (4.43) captures both the local variations of $r\{\bar{c}(t)\}$ through the Pearson matrix ρ, as well as the DC information in $r\{\bar{c}(t)\}$ through the outer product matrix P.

4.6 Experimental results

In this section, we will assess the performance of our proposed SNN-STDP theory and the feature descriptors of section 4.5 using common benchmark datasets used throughout SNN literature (the use of our SNN-STDP for online learning applications will be presented later in chapter 5). This use of common benchmark datasets enables us to compare our results to the state-of-the-art performances reported by *same-class*, single-layer methods (both with and without STDP learning). Similar to previous works, we consistently use *standard scaling* followed by a *linear* Support Vector Machine (SVM) to classify the feature descriptors obtained at the output of the SNN [194]. The use of a *linear*

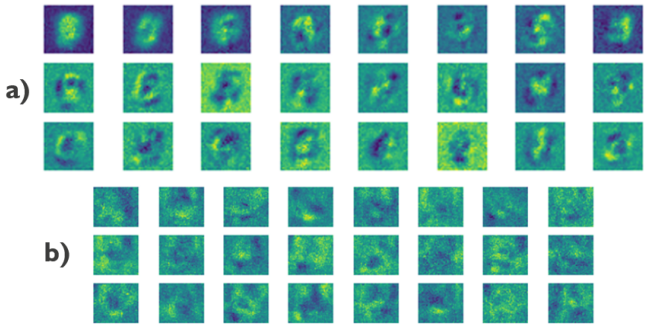

Figure 4.7: *Subset of the learned SNN weights. a) N-MNIST, b) IBM gesture.*

SVM (as opposed to *nonlinear* SVM variants) enables a fair and transparent assessment of the SNN-STDP effect on improving the linear separability of its input data (since when using a non-linear readout, it will not be clear what part of the system is actually doing the useful work of increasing the data separability as the nonlinear readout could be the main cause of the system performance instead of the SNN-STDP).

4.6.1 N-MNIST classification

The 10-class N-MNIST dataset [178] features standard MNIST images, *re-captured* using an event camera. Each acquisition contains a tensor of spike trains (along time and image space). In order to feed the events to our SNN as *global* descriptor (see section 4.5.1), the spiking matrices are flattened to 1-dimensional vectors of spike trains $\bar{s}(t)$. As the N-MNIST matrices are of dimension 34×34, the input dimension is $N = 1156$. We use $M = 4000$ coding neurons for producing over-complete sparse codes in order to enhance the linear separability.

The SNN-STDP weights are learnt by iterating over the first 40 training examples of each class, leading to only $N_{\text{ex}} = 400$ training examples in total. Fig. 4.7 a) shows a subset of weights learned by the SNN-STDP system. Furthermore, Table 4.1 reports the training and SNN-STDP network parameters. The network simulation time step dt is set to 0.005 s. The neuron threshold μ is tuned by grid search for each dataset. The neuron membrane time constant is derived from μ using (4.12). The network weights are initialized following a zero-mean normal distribution with standard deviation 0.01.

In order to perform classification, a linear SVM classifier is trained on the

η_1	η_2	μ	A_+	A_-	τ_+ [s]	τ_- [s]	τ_s [s]
1	0.003	*N-MNIST:* 0.3 *IBM:* 0.15	1	0.8	0.0208	0.008	0.01

Table 4.1: *Learning and network parameters.*

Architecture	N-MNIST [%]
MuST *(STDP)* [142]	89.96
HATS [225]	99.1
DART [194]	97.95
Kostadinov et al. [122]	98.1
DECOLLE [110]	99.04
Ours *(STDP)*	**99.26**

Table 4.2: *N-MNIST performance.*

Architecture	IBM DVS128 gesture [%]
Tempotron *(STDP)* [104]	60.37
Reservoir *(STDP)* [78]	65
SNN-SBP *(STDP)* [279]	84.76
Ours *(STDP)*	**92.5**

Table 4.3: *IBM DVS128 gesture performance.*

complete *SNN-encoded* training dataset. The system accuracy is assessed on the provided N-MNIST test set and reported in Table 4.2. Compared to state-of-the-art DVS feature descriptors and same-class methods proposed in earlier works, Table 4.2 shows that the proposed SNN-STDP system reaches a superior accuracy on the N-MNIST dataset. More specifically, the proposed SNN-STDP system reaches an accuracy gain of 9.3% compared to the state-of-the-art STDP system termed MuST [142]. Compared to conventional feature descriptors, the proposed SNN-STDP reaches a higher accuracy while being biologically-plausible and implementable in neuromorphic hardware.

4.6.2 IBM DVS128 gesture classification

We now assess our SNN-STDP as an *action* descriptor (see section 4.5.2) with $M = 1500$ coding neurons on the 11-class IBM DVS128 gesture dataset [9] following the standard train-test procedure of [9] (where dedicated training and test sets are provided). The spatio-temporal event images are flattened to

spiking vectors $\bar{s}(t)$ before being fed to the SNN-STDP system. The network and learning parameters are given in Table 4.1. The unsupervised SNN learning is conducted by iterating over the first 40 training examples of each class (determined heuristically to reduce training time while not sacrificing learning quality). Learning is terminated by tracking the activity of the error neurons $||r\{\bar{e}(t)\}||_2$, as shown in Fig. 4.6. A subset of weights learnt during STDP training is shown in Fig. 4.7 b).

The SNN outputs are then used to form the action descriptors (see section 4.5.2) which are classified via a linear SVM. Table 4.3 reports the classification accuracy of the proposed SNN-STDP system against state-of-the-art STDP-based methods. The proposed SNN-STDP system outperforms state-of-the-art STDP methods by +7.74% in Table 4.3 (a 2× reduction in error rate compared to SNN-SBP).

4.7 Conclusions

This chapter has presented a *top-down* optimization-based framework explaining how biologically-plausible SNN-STDP ensembles can perform unsupervised feature extraction by linking their dynamics with the theory of joint dictionary learning and LASSO basis pursuit. In addition, theoretical foundations have been provided for tuning the network parameters, reducing the need for empirical parameter search. The proposed methods have been demonstrated on the N-MNIST and the IBM DVS128 gesture datasets, reporting improved performance against the state of the art (+9.3% on N-MNIST and +7.74% on IBM DVS128 gesture compared to state-of-the-art *STDP* systems). Therefore, the significant enhancements on task accuracy shown in this chapter have demonstrated the usefulness of the proposed *top-down* SNN-STDP framework over prior *bottom-up* approaches. In the next chapters, the SNN-STDP theory described here will be used to perform and learn various drone-related tasks such as Simultaneous Localization and Mapping (SLAM) and people detection on drones.

Chapter 5

Sensor-fusion SLAM with continual STDP learning

5.1 Introduction

The previous chapter has presented a top-down optimisation-based approach for designing SNN-STDP networks capable of jointly extracting feature vectors from spiking input data and learning meaningful weights from the input data in an unsupervised manner. This chapter will investigate how the SNN-STDP approach presented in the previous chapter can be applied for *visual encoding* in order to perform Simultaneous Localisation and Mapping (SLAM) [32]. Crucially, visual encoding via SNN-STDP will be investigated in a *Continual Learning* (CL) setting, where no network pre-training is conducted prior to the SNN-STDP deployment for drone SLAM. Rather, the SNN-STDP learns to perform visual encoding *on the fly* in an environment that is a priori unknown and not learned during an offline pre-training phase.

Adapted from: *i) A. Safa et al., "Learning to Encode Vision on the Fly in Unknown Environments: A Continual Learning SLAM Approach for Drones," 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).*

ii) A. Safa et al., "Fusing Event-based Camera and Radar for SLAM Using Spiking Neural Networks with Continual STDP Learning," 2023 IEEE International Conference on Robotics and Automation (ICRA).

SLAM is a fundamental task for autonomous agents such as surveillance and rescue drones [233]. State-of-the-art SLAM systems either rely on handcrafted feature descriptors [159, 167] or on learning-based representations [31, 255] for template matching and loop closure detection. Compared to handcrafted descriptors, learning-based systems perform better in environments with less features or with high amounts of visual ambiguities (i.e., different scenes that look similar) [31, 275].

Still, most learning-based SLAM systems use traditional DNNs, where a dataset of the target environment must be available *a priori* for an offline DNN training phase [31]. Hence, those learning-based systems do not always generalize well to environments unseen during training [255]. This has recently motivated the study of Continual Learning (CL) for SLAM [255].

In the past years, CL has gained much attention and has mostly been investigated in the context of deep learning, with the aim of training deep neural networks on streams of non-shuffled data that cannot be assumed independent and identically distributed (i.i.d.). This poses significant challenges during training, severely jeopardizing the learning performance [46]. Most notably, research in CL has mainly been devoted to *classification* tasks, using non-i.i.d versions of popular datasets such as MNIST, CIFAR10 and ImageNet [46].

More recently, SLAM has also been proposed as an interesting application for CL [255]. Indeed, when adopting a learning-based SLAM system, a robot trained offline on a dataset acquired beforehand could perform poorly in environments that were not captured by the dataset [255], leading to unreliable and unsafe navigation.

Following a number of investigations in the domain of neuroscience that discuss the importance of *surprise* as a modulating signal for continual learning [187], the continual SNN-STDP setup proposed in this thesis makes use of a newly-formulated surprise-driven learning technique, enabling the filtering of the redundant data points. This filtering of redundant data during learning considerably reduces catastrophic forgetting since learning on data points that do not bring any novelty to the learning system could drive the model into over-fitting. Hence, it will be demonstrated that the proposed surprise-modulated technique helps model generalization while reducing the chances of over-fitting on old data.

This chapter will propose methods for continually learning to encode vision and sensory data (RGB, DVS and radar sensing modalities) into feature descriptors that can be integrated by a SLAM back-end, together with odometry information, in order to perform SLAM without any model pre-training. As shown in Fig. 5.1, the sensory input data $\bar{s}_k$ (e.g., RGB) is fed in its natural order into the CL

network that produces feature descriptors $\bar{c}_k$, while learning sensory patterns on the fly. The descriptors $\bar{c}_k$, together with the raw odometry, are fed to a SLAM back-end that performs loop closure detection (when the drone re-visits places visited before) and map correction.

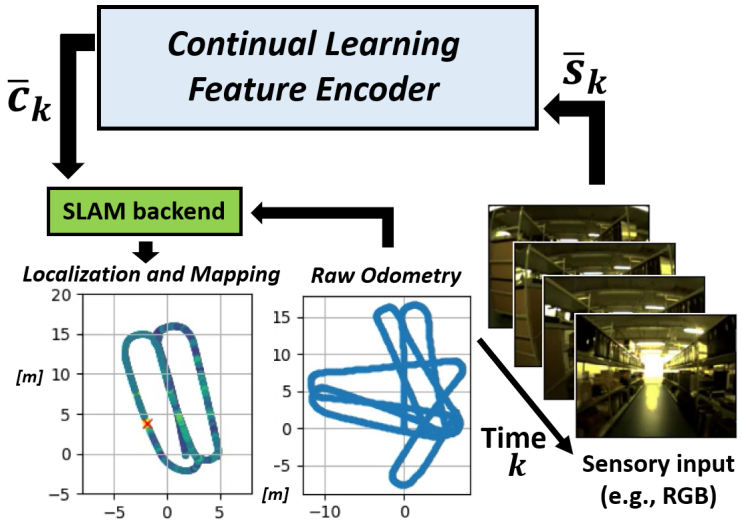

Figure 5.1: *High-level continual-learning SLAM architecture.*

Therefore, the major contributions of this chapter are the following:

1. **Algorithm:** Since it was explained earlier in chapter 4 that joint Dictionary Learning and Basis Pursuit or DLBP (studied up to now from an *offline* training perspective) forms the mechanism underlying the SNN-STDP output and weight dynamics, it will first be studied how CL can be facilitated in the DLBP context through the use of the proposed *Quadratic Bayesian Surprise* (QBS) factor for modulating the learning step in DLBP. Then, the proposed QBS factor will be ported to the SNN-STDP case for facilitating CL with SNN-STDP in order to perform SLAM.

2. **Radar signal processing:** Low-complexity methods for both radar-gyroscope odometry and radar-based obstacle modelling will be proposed as environmentally robust and direct means for both reconstructing walls and obstacles, as well as for raw estimation of the drone trajectory (which is exploited and refined by the SLAM back-end thanks to the feature descriptors provided by the SNN-STDP).

3. **Experiments:** The proposed CL SLAM system will be demonstrated under two settings: a) using non-spiking DLBP with RGB camera input, and b) using the corresponding SNN-STDP with a fusion of DVS and radar data. Localisation and mapping performance will be assessed in a challenging warehouse environment with a high number of visual ambiguities. Furthermore, the proposed CL SNN-STDP system will be compared against an SNN-STDP encoder pre-trained *offline* without continual adaptation and it will remarkably be shown that the CL model significantly outperforms the model trained offline, strongly motivating the effectiveness of our proposed CL SNN-STDP approach for SLAM.

This chapter is organized as follows. Section 5.2 will describe the proposed CL algorithms for encoding the visual sensory data into feature descriptors used by the SLAM back-end for loop closure detection and map correction. The architecture behind the proposed RGB and DVS-Radar SLAM systems, as well as additional methods for radar-based odometry and obstacle modelling will be presented in section 5.3. Experimental results will be detailed in section 5.4. Finally, conclusions will be provided in section 5.7.

5.2 Continual learning to encode vision

5.2.1 What are the challenges posed by continual learning?

In gradient-based learning systems such as DLBP, the input sequence $\bar{s}_k$ fed during a learning procedure (with k the input index) is generally assumed to be *independent and identically distributed* (i.i.d) realizations following the conventional training procedure, where a dataset is assumed to be available *a priori* and shuffled before being fed to the optimisation procedure of DLBP (see Algorithm 3 in section 4.2 of chapter 4). Violating this i.i.d assumption, *as in the CL case*, by feeding video sequences in their natural order, causes significant problems during learning since the stochastic gradients are not representative of the full loss [46]:

$$\mathbb{E}[\frac{\partial l(\bar{s}_k, \Phi)}{\partial \Phi}] \not\to \frac{\partial \frac{1}{K} \sum_{k=1}^{K} l(\bar{s}_k, \Phi)}{\partial \Phi} \tag{5.1}$$

where $\mathbb{E}$ denotes the expected value, K is the total number of data points and $l(\bar{s}_k, \Phi) = \frac{1}{2}||\Phi \bar{c}_k - \bar{s}_k||_2^2 + \lambda_1 ||\bar{c}_k||_1$ is the DLBP loss associated with the data point $\bar{s}_k$ (used by the stochastic gradient descent procedure in DLBP).

In the CL setting, the goal is to enable continual DLBP in non-i.i.d data streams such as videos, where the consecutive image frames are highly correlated [46],

inducing large local *imbalance* in the data stream (i.e., the input stream is not diverse but rather redundant):

$$\bar{s}_{k+1} \approx \bar{s}_k \tag{5.2}$$

Therefore, a CL DLBP algorithm must be proposed to mitigate the effect of this local data stream imbalance (5.2) in order to continuously learn Φ and infer the latent codes $\bar{c}_k$ from a video stream *without any pre-training* and *without re-initializing* $\bar{c}$ between each frame.

Next, a method for alleviating this effect of the local data stream imbalance (5.2) will be derived by taking inspiration from the remarkable learning capability of Support Vector Machines (SVMs) in unbalanced datasets [217].

5.2.2 Proposed method: the Quadratic Bayesian Surprise

In this section, we are going to cast the DLBP problem (4.1) introduced in section 4.2 into the continual learning case, by using the properties of Support Vector Machines [33, 217] (which provide robustness to overfitting) in an online fashion. This will enable the DLBP system to avoid local overfitting on the data currently being processed, reducing catastrophic forgetting issues. In order to cast the DLBP problem (4.1) introduced in section 4.2 into the continual learning case, we use the DLBP reprojection error $||\Phi\bar{c}_k - \bar{s}_k||_2^2$ in (4.1) to reduce the coding problem (inference of $\bar{c}_k$ by LASSO proximal descent) into a 2-class likelihood test. Indeed, the DLBP reprojection is linked to the Gaussian likelihood $p(\bar{s}_k|\Phi)$:

$$-\log p(\bar{s}_k|\Phi) \sim ||\Phi\bar{c}_k - \bar{s}_k||_2^2 \tag{5.3}$$

This likelihood can be used to perform an inlier-outlier test:

$$\begin{cases} \textbf{if } p(\bar{s}_k|\Phi) \geq \theta \textbf{ then } H_0 \\ \textbf{if } p(\bar{s}_k|\Phi) < \theta \textbf{ then } H_1 \end{cases} \tag{5.4}$$

The test (5.4) can be seen as a binary classification problem inferring whether the current input $\bar{s}_k$ corresponds to the model learned in the dictionary Φ (H_0) or not (H_1). This classification is done by testing $p(\bar{s}_k|\Phi)$ against a threshold θ. Therefore, we identify a dual problem: *A)* the original problem that we are seeking to solve (i.e., learning Φ in a continual manner in order to infer $\bar{c}$), and *B)* its associated dual problem as learning Φ in a continual manner in order to classify $\bar{s}$. Hence, by casting the dual problem *B)* to the continual learning case, we implicitly cast the original problem *A)* to the continual learning case as well.

In order to transpose *B)* to the continual case, we must provide robustness to the severe data imbalance caused by the non-stationary stream $\bar{s}_k$ (due to the fact that $\bar{s}_k$ does not originate from a shuffled dataset where realisations are assumed to be i.i.d, see section 5.2.1). Therefore, we propose to solve the binary classification problem *B)* using the well-known Support Vector Classification (SVC) framework [217]:

$$\Phi_{k^*} = \arg\min_{\Phi} \sum_{j=1}^{k^*} \xi_j \ \textbf{subject to:} \begin{cases} ||\Phi\bar{c}_j - \bar{s}_j||_2^2 \leq \log\theta + \xi_j \ \textbf{if} \ \bar{s}_j \in H_0 \\ ||\Phi\bar{c}_j - \bar{s}_j||_2^2 \geq \log\theta - \xi_j \ \textbf{if} \ \bar{s}_j \in H_1 \end{cases} \tag{5.5}$$

where k^* is the index of the current time step in the data stream, θ is the threshold used in (5.4), and ξ_j are the slack variables defining the margin of the separation hyperplane. In general, θ should be determined in a way such that the reprojection error (5.3) falls below θ for signals that were already encountered, and falls above θ in the opposite case.

The SVC formulation in (5.5) is indeed a natural choice when dealing with class imbalance since in an SVC, all data points are *not* contributing to the synthesis of the decision hyperplane [217] (the contribution is only due to a few support vectors). This prevents SVC over-fitting on the redundant data found in the heavily imbalanced video streams.

We can then cast (5.5) into its *hinge loss* formulation [217]:

$$\Phi_{k^*} = \arg\min_{\Phi} \sum_{j\in H_0} \max(0, ||\Phi\bar{c}_j - \bar{s}_j||_2^2 - \log\theta) + \sum_{j\in H_1} \max(0, \log\theta - ||\Phi\bar{c}_j - \bar{s}_j||_2^2) \tag{5.6}$$

where the hinge loss $\max(0, x)$ leads to the sparsity property of SVCs, which in turn leads to a large number of input vectors $\bar{s}_k$ not playing any role in the synthesis of the SVC hyperplane. This provides the robustness to data redundancy.

In our particular case, we seek to continuously maintain a dictionary Φ such that all incoming data points are well modelled by this dictionary. Therefore, the generic 2-class formulation in (5.6) reduces to (5.7) in our case since all data points must be associated to the H_0 class to be inliers [218]:

$$\Phi_{k^*} = \arg\min_{\Phi} \sum_{j=1}^{k^*} \max(0, ||\Phi\bar{c}_j - \bar{s}_j||_2^2 - \log\theta) \tag{5.7}$$

Using (5.7) in a probabilistic context, we can define the *Bayesian surprise factor* $\mathcal{S}_1$ associated with the incoming data point $\bar{s}_{k^*+1}$ as the posterior to prior ratio [15]:

$$\mathcal{S}_1(k^*+1) = -\log \frac{p(\Phi_{k^*+1}|\bar{s}_{k^*+1})}{p(\Phi_{k^*})} \propto -\log p(\bar{s}_{k^*+1}|\Phi_{k^*})$$
$$\propto \max(0, ||\Phi_{k^*}\bar{c}_{k^*+1} - \bar{s}_{k^*+1}||_2^2 - \log\theta) \quad (5.8)$$

where $\mathcal{S}_1$ is proportional to the negative log-likelihood via the Bayes theorem. This surprise factor indicates how much a newly received data point alters the posterior distribution of the model parameters. The larger the surprise, the more the model parameters are to be changed to incorporate this new information. In addition, the formulation in (5.8) assumes that the negative log-likelihood $-\log p(\bar{s}_{k^*+1}|\Phi_{k^*})$ is proportional to the hinge loss energy term in (5.6) under the SVC formulation (5.7). The surprise factor $\mathcal{S}_1$ is zero when the hinge loss associated with the incoming data point $\bar{s}_{k^*+1}$ is zero, denoting that the data point is already well modelled by the dictionary Φ and hence, does not lead to any surprise.

Still, using $\mathcal{S}_1$ in (5.8) is inconvenient since it requires the knowledge of the likelihood threshold θ. This can be an even greater problem in our dynamic CL context since an optimal choice of θ can slowly vary over time due to changes in the environment. Under the generic assumption of a high-enough camera frame rate (5.2), we now show that having an explicit knowledge of θ can be eliminated by introducing a second-order surprise factor $\mathcal{S}_2$:

$$\mathcal{S}_2(k^*+1) \equiv \mathcal{S}_1(k^*+1) - \mathcal{S}_1(k^*)$$
$$= -\log \frac{p(\Phi_{k^*+1}|\bar{s}_{k^*+1})}{p(\Phi_{k^*})} + \log \frac{p(\Phi_{k^*}|\bar{s}_{k^*})}{p(\Phi_{k^*-1})}$$
$$\sim \frac{\delta^2 - \log p(\Phi_k|\bar{s}_k)}{\delta k^2}|_{k^*+1} \quad (5.9)$$

which is related to the discrete-time second derivative ($\frac{\delta^2}{\delta k^2}$) of the posterior density evolution through time. We thus refer to $\mathcal{S}_2$ as a *Quadratic Bayesian Surprise* (QBS) factor.

Intuitively, $\mathcal{S}_2$ can be understood as the *curvature* of the posterior density evolution through time, indicating how much a newly received input vector $\bar{s}_{k^*+1}$ accelerates or decelerates the change in posterior $p(\Phi_k|\bar{s}_k)$.

Dropping proportionality constants and using (5.8) in (5.9):

$$\mathcal{S}_2 \equiv \max(0, \overbrace{||\Phi_{k^*}\bar{c}_{k^*+1} - \bar{s}_{k^*+1}||_2^2}^{e_{k^*+1}} - \log\theta)$$
$$- \max(0, \overbrace{||\Phi_{k^*-1}\bar{c}_{k^*} - \bar{s}_{k^*}||_2^2}^{e_{k^*}} - \log\theta)$$
$$= \begin{cases} 0 \textbf{ if } e_{k^*+1} - \log\theta \leq 0, e_{k^*} - \log\theta \leq 0 & \textbf{a)} \\ e_{k^*+1} - \log\theta \textbf{ only if } e_{k^*} - \log\theta \leq 0 & \textbf{b)} \\ \log\theta - e_{k^*} \textbf{ only if } e_{k^*+1} - \log\theta \leq 0 & \textbf{c)} \\ e_{k^*+1} - e_{k^*} \textbf{ else} & \textbf{d)} \end{cases} \quad (5.10)$$

where e_{k^*+1} and e_{k^*} are the current and past reprojection errors. Four possible outcomes *a),b),c),d)* are identified in (5.10). Under the high frame rate assumption (5.2), the probability of outcomes *b)* and *c)* in (5.10) is very low, since (5.2) induces that if $e_{k^*} - \log\theta \leq 0$ then $e_{k^*+1} - \log\theta \leq 0$ must likely be the case as well, and vice versa. In addition, outcomes *a)* and *c)* in (5.10) can be merged under the assumption (5.2):

a) **if** $(e_{k^*+1} - \log\theta \leq 0, e_{k^*} - \log\theta \leq 0)$

$$\textbf{then } \mathcal{S}_2 = 0 \approx e_{k^*+1} - e_{k^*} \textbf{ using } \bar{s}_{k^*+1} \approx \bar{s}_{k^*} \quad (5.11)$$

while still observing the conditions $e_{k^*+1} - \log\theta \leq 0, e_{k^*} - \log\theta \leq 0$ of *a)* in (5.10). Therefore, the QBS in (5.9) can simply be computed as follows:

$$\mathcal{S}_2(k^*+1) \approx ||\Phi_{k^*}\bar{c}_{k^*+1} - \bar{s}_{k^*+1}||_2^2 - ||\Phi_{k^*-1}\bar{c}_{k^*} - \bar{s}_{k^*}||_2^2 \quad (5.12)$$

In practice, the estimation of the QBS (5.12) can be noisy due to potential noise in the input data. To help alleviate this, it is possible to estimate the QBS (5.12) by low-pass filtering $\mathcal{S}_2$ using a moving average (e.g., of width 5) to attenuate noise.

We can now use the QBS $\mathcal{S}_2$ to modulate stochastic gradient descent (SGD) learning in an online manner using the following dictionary update procedure (with learning rate η_d) in Algorithm 5 (lines 6):

$$\begin{cases} \textbf{if } \mathcal{S}_2(k^*+1) > 0 \textbf{ , } \Phi_{k^*+1} \leftarrow \Phi_{k^*} - \eta_d \frac{\partial e_{k^*+1}}{\partial \Phi} \\ \textbf{else if } \mathcal{S}_2(k^*+1) \leq 0 \textbf{ , } \Phi_{k^*+1} \leftarrow \Phi_{k^*} \end{cases} \quad (5.13)$$

Therefore, by ignoring the effect of redundant data associated to a negative QBS factor, we prevent the over-fitting of the dictionary learning to the local contexts in the video streams. In turn, this leads to the learning of a more diverse set of features in Φ and consequently, to lower reprojection errors during the replay of the input sequence.

Algorithm 5 DLBP-QBS for SLAM (see Fig. 5.3)

Input: $\bar{s}_k \; \forall k$: input vector stream, η_c: coding rate, η_d: learning rate, λ_1: regularization, $N_{c,d}$: number of coding and learning iterations

Init.: $\Phi_{ij} \leftarrow \mathcal{N}(0, \sigma_w)$ (zero-mean normal distribution with std. deviation $\sigma_w \sim 0.01$), $\bar{c}_1 \leftarrow 0$, $\mathcal{S}_2 \leftarrow 1$

for $k \in \{1, ..., T_{end}\}$ (feed data sequence) **do**
  for $i = 1, ..., N_c$ (local coding iterations) **do**
    $\bar{c}_k \leftarrow \mathbf{Prox}_{\eta_c \lambda_1 ||.||_1}\{\bar{c}_k - \eta_c \Phi^T(\Phi \bar{c}_k - \bar{s}_k)\}$ see (4.2)
  end for
  for $i = 1, ..., N_d$ (local learning iterations) **do**
    if $\mathcal{S}_2 > 0$ **then**
      $\Phi_k \leftarrow \Phi_{k-1} - \eta_d(\Phi \bar{c}_k - \bar{s}_k)\bar{c}_k^T$
    else
      $\Phi_k \leftarrow \Phi_{k-1}$
    end if
  end for
  Compute $\mathcal{S}_2$ following (5.12)
  RatSLAM$\{\bar{c}_k, \text{odometry}_k\}$ // Feed to SLAM backend (see section 5.3)
end for

5.2.3 Effect of QBS on DLBP

The effect of the proposed QBS is shown in Fig. 5.2 a), that displays the *reprojection error* obtained after continual dictionary learning on an RGB video sequence captured with the drone setup described in the second part of chapter 3 (images of size $N = 346 \times 260$). The use of the QBS factor leads to a lower loss during sequence replay, indicating a better learning performance (with DLBP parameters: $\eta_c = 5 \times 10^{-3}$, $\lambda_1 = 0.5$, $\eta_d = 2 \times 10^{-3}$, $N_c = 10$, $N_d = 1$, $M = 64$ in Algorithm 5). This clearly indicates that the proposed QBS factor reduces the effects of catastrophic forgetting since it helps achieving a lower loss during the sequence replay. Indeed, achieving a lower loss during the replay is a direct indication that the continual learning procedure has been more successful at retaining past knowledge when learning on new data points. This globally leads to a lower loss during the replay as shown in Fig. 5.2 a), indicating that the early knowledge was not catastrophically forgotten during the future learning steps.

In addition, we observed during our experiments a useful stability property induced by the use of the QBS in our continual dictionary learning scenario. Fig. 5.2 b) shows that without the use of the QBS factor, severe run-away behaviors can happen during the continual learning process (with DLBP parameters:

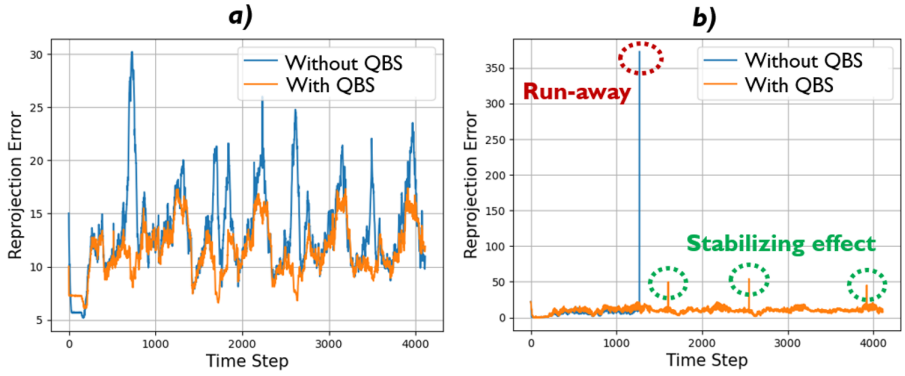

Figure 5.2: *Effect of the QBS on the DLBP performance and stability.*

$\eta_c = 10^{-2}$, $\lambda_1 = 0.25$, $\eta_d = 4 \times 10^{-3}$, $N_c = 10$, $N_d = 1$, $M = 64$ in Algorithm 5). Indeed, it is known that DLBP systems can suffer from instability [268], and we have observed run-away behaviors as well for cases where the QBS was not used. In contrast, we did not observe any instability issue when using our QBS-modulated learning rule (5.13) in Algorithm 5, suggesting a regularizing effect (see Fig. 5.2 b), where using QBS induces stability.

Finally, a potential foreseen limitation of this proposed method would lie in the memory capacity of the DLBP system as a whole. Indeed, when scaling to larger multi-room environments, it can be that the DLBP setup does not have enough memory for capturing features from the larger range of different camera views. To alleviate this, it is possible to use multiple continual-learning DLBP ensembles, and to switch between them as soon as a highly significant change in the environment is detected (e.g., when a robot transitions from one room to another one).

5.2.4 Applying QBS to the SNN-STDP network

The spiking activity of the *error neurons* in the SNN-STDP network architecture described in Fig. 4.1 (section 4.3) represents through its local spiking rate the error signal $\bar{e}(t) = \Phi\bar{c}(t) - \bar{s}(t)$ that can be used to compute the reprojection error $||\Phi\bar{c} - \bar{s}||_2^2$ in QBS (5.12). Hence, the QBS can be estimated in the SNN-STDP case by first averaging the error spike train vector $\bar{e}(t)$ over a time window of length ΔT (set to the simulation time resolution of 5 ms in our experiments) in

order to compute the re-projection error:

$$\epsilon(t) = ||\frac{1}{\Delta T}\int_{t-\Delta T}^{t} \bar{e}(t)dt||_2^2 \tag{5.14}$$

In general, the time window ΔT must be chosen in a way such that a long-enough statistically significant averaging effect takes place, while not causing large delays compared to the expected dynamics of the environment.

Then, the QBS can be estimated as:

$$\mathcal{S}_2(t) = \epsilon(t) - \epsilon(t - \Delta T) \tag{5.15}$$

and can be used to switch STDP learning on or off in a way similar to the DLBP case (5.13).

5.3 Forming the SLAM system

5.3.1 RGB SLAM using DLBP

A schematic view of the components forming the SLAM system is shown in Fig. 5.3. The RGB input stream is fed in its natural order to the QBS-driven CL visual encoding system (either non-spiking DLBP or its equivalent SNN-STDP architecture). The QBS-driven DLBP encoder continuously infers a latent code $\bar{c}_k$ corresponding to the current observation $\bar{s}_k$ and simultaneously learns a dictionary Φ (learning enabled when the QBS factor $\mathcal{S}_2$ is positive)[1]. Finally, the latent codes $\bar{c}_k$ are fed to a RatSLAM back-end [159] to perform loop closure detection through template matching [31]. The RatSLAM back-end also receives as input the raw odometry data providing an instantaneous measurement of the drone heading and rotational velocities. The RatSLAM back-end can then produce a map of the places that the drone is visiting, by correcting the drift in the odometry (due to sensor noise and bias) through the loop closure and map correction procedures.

RatSLAM is a popular *bio-inspired* SLAM system following the navigational processes of the rat's hippocampus [159], and has been shown to be compatible with neuromorphic hardware [76], making it a natural match with the use of

[1]Note that here the QBS surprise factor is disjoint with the concept of surprise found in RatSLAM-based SLAM architecture [31] which can detect novelty in scenes and be sensitive to objects that have been moved, people walking in the scene and so on. In contrast, the QBS indicates the surprise of the incoming data with regard to the model that is being learned online by the system.

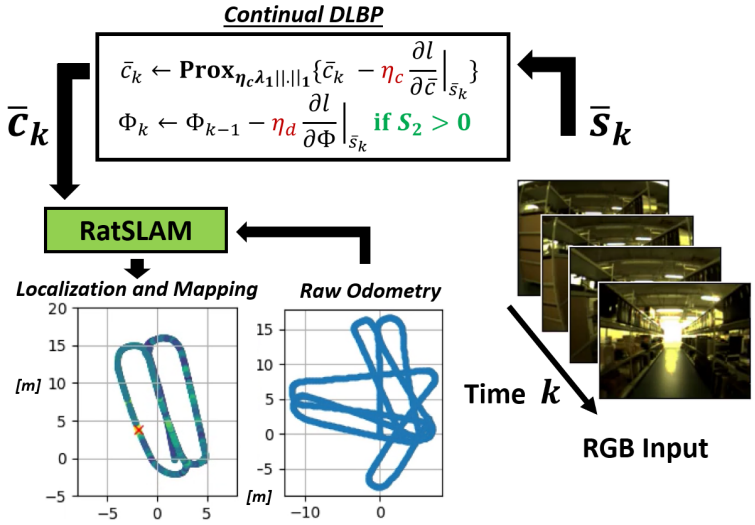

Figure 5.3: *Block diagram of the continual learning (CL) DLBP-based RGB SLAM system, with QBS adaptation ($\mathcal{S}_2$).*

SNN-STDP networks for visual encoding, motivating its use in this work (the RatSLAM back-end used in this work is a slightly modified one with the details behind these modifications described in Appendix C). It is important to note that memory management is a challenging issue when implementing the SLAM back-end in edge hardware. Indeed, as the drone visits new environments, new poses are dynamically registered in the history of poses maintained by the SLAM back-end. As such, the memory consumption is not static and grows with time. Hence, when implementing the SLAM back-end in hardware, special care must be provided to the dynamic memory management required to maintain the history of past SLAM poses (this hardware-centric study is left as future work).

5.3.2 DVS-radar fusion SLAM using SNN-STDP

In order to jointly infer latent codes and continually learn a set of SNN weights capturing the DVS and radar data features, we use the unsupervised SNN-STDP ensemble of Fig. 4.1 in a *fusion* setting, by assigning: *i)* a first SNN-STDP ensemble to the *positive* polarity of the DVS; *ii)* a second SNN-STDP ensemble to the *negative* polarity of the DVS[2], and *iii)* a third SNN-STDP ensemble to

[2]Even though we have separated the positive and negative DVS polarities in this architecture, we expect the use of both polarities merged as one channel to lead to similar SLAM results. This confirmation is left to future work.

the radar detection output (see Fig. 5.4).

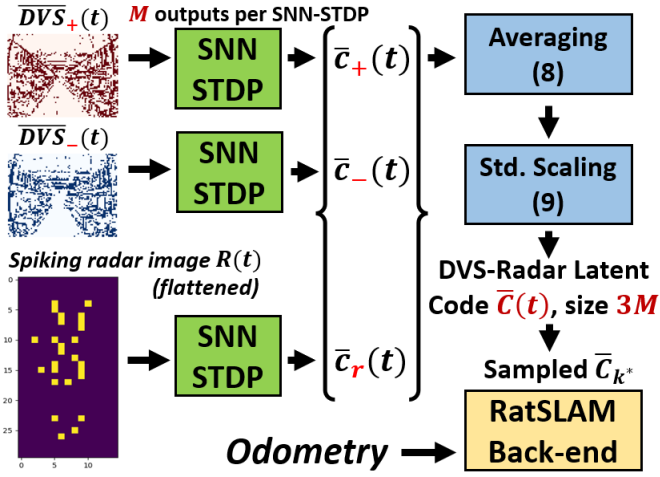

Figure 5.4: *DVS-radar fusion SLAM using continual SNN-STDP networks.*

In our experiments, we degrade the time resolution of the DVS camera to spiking bins of $\delta T_{\text{DVS}} = 5$ ms and the DVS resolution to 65×86, as a good balance between computational time and performance. Each polarity channel of the DVS data is then directly connected to its respective SNN-STDP ensemble by flattening the image plane as a binary spiking vector ($\bar{\text{DVS}}_{+,-}(t)$ in Fig. 5.4).

For the radar data, our drone setup of Fig. 5.6 provides radar detections of the form:

$$\mathcal{D}_k = \{(x_{d,i}, y_{d,i}, v_{d,i}) \;\; \forall i = 1, ..., N_t\} \tag{5.16}$$

where k is the radar frame time index, x_d, y_d are the Cartesian coordinates of the detection (in meter), v_d is the radial (or Doppler) velocity of the detection [212], and N_t is the number of detected targets in frame k (with a frame rate of 20 FPS). Therefore, radar detections can be seen as binary events (or spikes) with their address given by the associated coordinate (x_d, y_d).

We transform the detection set (5.16) into a *spiking image* R of size $W_r \times H_r$ by *a)* quantizing the coordinates to lie between $[0, W_r - 1]$ for x_d and $[0, H_r - 1]$ for y_d via (5.17), and *b)* assigning a value of 1 to the radar image pixels associated with a detection coordinate and 0 everywhere else, via (5.18):

$$Q^x_{d,i} = \lfloor (W_r - 1)\frac{x_{d,i}}{\max_x} \rfloor \;\; Q^y_{d,i} = \lfloor (H_r - 1)\frac{y_{d,i}}{\max_y} \rfloor \tag{5.17}$$

$$R[Q^x_{d,i}, Q^y_{d,i}] = 1 \;\; \forall i = 1, ..., N_t \;\; \textbf{else} \;\; R = 0 \tag{5.18}$$

where $\max_x = 10$ m and $\max_y = 10$ m are the maximum coordinate extents that the radar can detect. We set $W_r = 15, H_r = 30$ as a good balance between radar image resolution and computational time during SNN processing. An example of a *spiking radar image* obtained using (5.17)-(5.18) is shown in Fig. 5.4. Next, we feed the *spiking radar image* R to its SNN-STDP ensemble by flattening R as a spiking vector with an oversampling in time of $10\times$ in order to meet the DVS time resolution of $\delta T_{\text{DVS}} = 5$ ms, i.e., each spiking radar image is fed 10 times to the SNN-STDP input.

As the drone navigates, DVS and radar data are fed to the SNN ensembles which continuously learn features via STDP and infer M-dimensional *spiking* outputs $\bar{c}_+(t), \bar{c}_-(t), \bar{c}_r(t)$ corresponding to the *positive* DVS channel, the *negative* DVS channel and the radar data. The outputs are then concatenated and averaged on a time window of length $\Delta T_{\text{avg}} = 0.1$ s to obtain the fused latent code $\bar{\mathcal{C}} \in \mathbb{R}^{3M}$ (see Fig. 5.4):

$$\bar{\mathcal{C}}(t^*) = \frac{1}{\Delta T_{\text{avg}}} \int_{t^*-\Delta T_{\text{avg}}}^{t^*} \begin{bmatrix} \bar{c}_+(t) \\ \bar{c}_-(t) \\ \bar{c}_r(t) \end{bmatrix} dt \tag{5.19}$$

Finally, we apply *standard scaling* to compensate for the covariate shifts (i.e., the shifts in the SNN-STDP output distribution) during drone navigation [102], as well as for the scale differences between the elements of $\bar{\mathcal{C}}$, greatly helping template matching between the latent codes:

$$\mathcal{C}_l \leftarrow \frac{\mathcal{C}_l - m_l}{std_l} \tag{5.20}$$

where the mean m_l and standard deviation std_l of element $\mathcal{C}_l$ are estimated *on-line* using the algorithm provided in [34].

Fig. 5.4 shows that the *averaged* and *scaled* latent codes $\bar{\mathcal{C}}(t)$, together with the raw *radar-gyroscope* odometry (see section 5.3.3) are fed to the RatSLAM back-end [159] (at a rate $f_s = 10$ FPS). The RatSLAM back-end (which constitutes the more memory and compute-intensive part of the complete SLAM algorithm) detects loop closures by measuring the similarity between the currently sampled latent code $\bar{\mathcal{C}}_{k^*}$ and the previous latent codes $\bar{\mathcal{C}}_k, \forall k < k^*$ associated to the *past* drone poses in order to correct all drone poses $\{X_k, Y_k, \Psi_k\}$ (with location X_k, Y_k and yaw angle Ψ_k), maintaining localisation and mapping.

5.3.3 Radar-gyroscope odometry approach

Since accelerometer data can be too noisy due to the erratic drone vibrations, we use the radar sensor to retrieve the drone heading velocity v_h, while the on-board

gyroscope is used to measure the yaw rotation velocity $\dot{\Psi}$. The odometry is obtained by integrating v_h and $\dot{\Psi}$ following the standard equations of movement:

$$\begin{cases} \Psi_k \leftarrow \Psi_{k-1} + \dot{\Psi}\delta t_g \\ X_k \leftarrow X_{k-1} + v_h \cos(\Psi_k)\delta t_r \\ Y_k \leftarrow Y_{k-1} + v_h \sin(\Psi_k)\delta t_r \end{cases} \tag{5.21}$$

where k denotes the time step or pose index, $\delta t_g = 1$ ms is the gyroscope sampling period, $\delta t_r = 40$ ms is the radar frame period, and X_k, Y_k denote the drone coordinates.

We measure the drone heading velocity v_h from the radar detections in each frame by first projecting the Doppler velocity [212] of each detection back to the heading direction. Then, we take the median value of all the obtained velocities following Algorithm 6 below. We use the median value instead of the mean in order to provide robustness to outliers. In addition, we restrict the detections to lie in a $\pm 60^\circ$ field of view around the drone heading axis in order to further reject unreliable measurements that are caused by the attenuation of the radar antenna gain profile under high azimuth angles.

Algorithm 6 Radar odometry for heading velocity

Input: Radar detections: $\{x_{d,i}, y_{d,i}, v_{d,i} \ \forall i = 1, ..., N_t\}$ for the current frame, Valid angle range $\theta_v = 60^\circ$.
Output: Heading velocity: v_h

$V = \{\}$: empty array
for $i \in \{1, ..., N_d\}$ **do**
  // Loop over all detections in the current radar frame
  $\theta_i = |\arctan(\frac{x_{d,i}}{y_{d,i}})|$
  if $\theta_i < \theta_v$ **then**
    // If detection angle is within the valid range
    Append $\frac{v_{d,i}}{\cos\theta_i}$ to V //projecting radial velocity to
  end if // y-axis (i.e., drone heading axis).
end for
return $v_h = \text{median}(V)$

5.3.4 Radar-based obstacle aggregation

The radar sensor provides a direct and robust mean for the reconstruction of walls and other landmarks observed by the drone during its flight. For each pose k, a set of radar detections $\mathcal{D}_k = \{x_{d,i}, y_{d,i} \ \forall i\}$ is available and can be

aggregated with the previous detections and the previous drone poses in order to reconstruct the walls and obstacles (see Fig. 5.5). For each pose k, the radar detections are rotated by the drone yaw angle Ψ_k and translated by the drone position $[X_k, Y_k]$:

$$\begin{bmatrix} \tilde{x}_{d,i} \\ \tilde{y}_{d,i} \end{bmatrix} \leftarrow \begin{bmatrix} \cos(\Psi_k) & -\sin(\Psi_k) \\ \sin(\Psi_k) & \cos(\Psi_k) \end{bmatrix} \begin{bmatrix} x_{d,i} \\ y_{d,i} \end{bmatrix} + \begin{bmatrix} X_k \\ Y_k \end{bmatrix} \quad \forall i \tag{5.22}$$

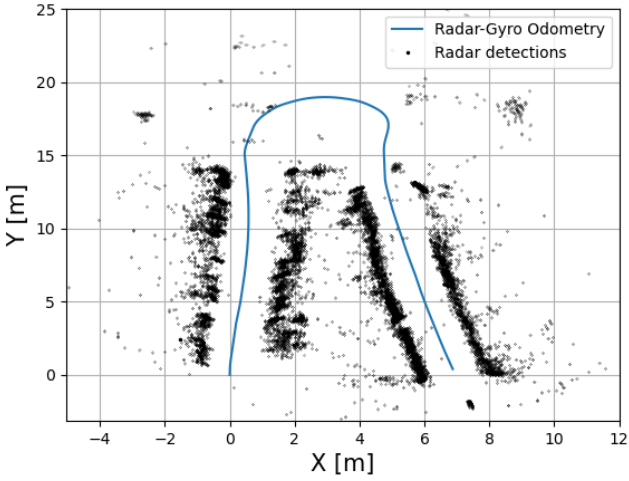

Figure 5.5: Trajectory result of raw radar-gyro odometry and obstacle aggregation.

Therefore, each time a loop closure is detected by the RatSLAM back-end, our pipeline refines not only the past poses $\{X_k, Y_k, \Psi_k\}$, but also the obstacle coordinates $[\tilde{x}_{d,i}, \tilde{y}_{d,i}]$ by applying (5.22) again with the corrected Ψ_k and $[X_k, Y_k]$, thereby correcting the position of the obstacles as well (see Fig. 5.13).

5.4 Experiments

5.4.1 Experimental setup

As a real-world use case, we use the drone setup in Fig. 5.6 to illustrate the applicability of our method for learning to SLAM on the fly in a highly

ambiguous warehouse environment (see Fig. 5.7, where different places in the environment look very close to each other, leading to visual ambiguities). The sensor fusion drone setup shown in Fig. 5.6 can jointly acquire RGB, DVS and radar data. The warehouse environment (detailed in [199] and in the second part of chapter 3) in which the drone navigates is also shown. The warehouse is also equipped with Ultra Wide Band (UWB) localisation for ground truth drone positioning acquisition [199].

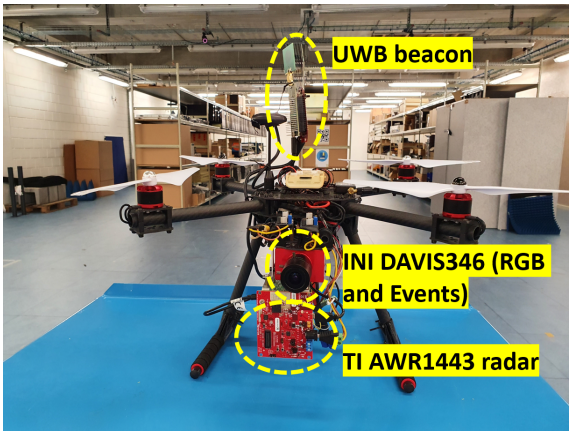

Figure 5.6: *Sensor fusion drone setup.*

The high scene redundancy in the warehouse environment (as shown in Fig. 5.7, see view 1 vs. 4, 2 vs. 3) poses a significant challenge for both:

- the continual learning aspect [46] due to (5.2) inducing the bias (5.1),
- the loop closure detection, due to the high level of aliasing in the environment [275].

5.4.2 Results with RGB and DLBP

We have evaluated the Mean Absolute Error (MAE) of the proposed DLBP-based RGB SLAM approach against the ground truth localisation and mapping, acquired by UWB indoor positioning beacons [199]. As DLBP parameters, we use $\eta_c = 5 \times 10^{-3}$, $\eta_d = 1.4 \times 10^{-3}$, $\lambda_1 = 0.2$, $N_c = 10$, $N_d = 1$, $M = 64$ in Algorithm 5. These parameters were tuned empirically in order to optimise the SLAM performance.

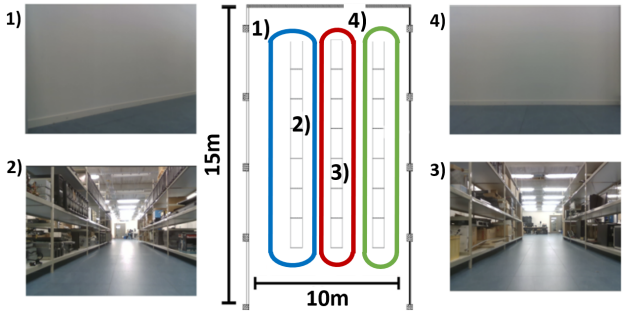

Figure 5.7: *Map of the warehouse and drone flight paths, with pictures taken at each marked locations.*

Three different flight sequences have been used for performance assessment, with increasing difficulty:

- *Flight 1*: Flying between wall and shelves, which makes the continual learning and loop closure detection easier, since the visited environment is more diverse and hence, easier to disambiguate (blue path in Fig. 5.7).
- *Flight 2*: Flying between racks only, which leads to a larger amount of similar and ambiguous views compared to *flight 1* (red path in Fig. 5.7)
- *Flight 3*: Combining *flights 1 and 2* (both red and blue paths in Fig. 5.7 are visited by the drone).

When assessing the performance against the ground truth, since the SLAM result and the ground truth do not share the same coordinate system [159], the ground truth coordinates are projected to the SLAM domain via translation and rotation. The translation vector and the rotation angle are found via grid search, by minimizing the localisation MAE.

The localisation MAE is computed as the average error between the SLAM localisation (x_k, y_k) and the ground truth $(x_k^{\mathrm{gt}}, y_k^{\mathrm{gt}})$ on the flight sequence of length T_{end}:

$$\mathrm{MAE_L} = \frac{1}{T_{end}} \sum_{k=1}^{T_{end}} |x_k - x_k^{\mathrm{gt}}| + |y_k - y_k^{\mathrm{gt}}| \tag{5.23}$$

In addition, the mapping MAE is computed as the *a posteriori* deviation between the map obtained at the end of the SLAM process and all the ground truth

points:

$$\mathrm{MAE_M} = \frac{1}{T_{end}} \sum_{k=1}^{T_{end}} \min_j \{|x_k - x_j^{\mathrm{gt}}| + |y_k - y_j^{\mathrm{gt}}|\} \tag{5.24}$$

Table 5.1 quantitatively compares the performance of our approach against *i)* the original RatSLAM [159]; *ii)* the LatentSLAM, trained specifically for our environment following [31], on a set of 3998 frames acquired independently from flights 1-3; and *iii)* Oriented FAST and Rotated BRIEF (ORB) features for template matching [204] following the standard *Lowe's Ratio Test* [146].

For the ORB features, the similarity between two frames i, j is computed as $s_{ij} = 1 - N_m / \max_m$ where N_m is the number of matches and $\max_m$ the maximum number of possible matches. For both LatentSLAM and our CL method, the similarity is computed as $s_{ij} = \frac{\bar{c}_i^T \bar{c}_j}{||\bar{c}_i||_2 ||\bar{c}_j||_2}$ with $\bar{c}_{i,j}$ denoting the feature descriptor obtained by each method.

The RatSLAM back-end integrates the raw odometry and detects loop closures by sampling latent codes from the data stream each 100 ms and by comparing the similarities s_{ij} against a threshold μ tuned for minimum MAE.

Architecture	Flight 1	Flight 2	Flight 3
RatSLAM [159]	0.72 | 0.22	1.499 | 0.248	2.714 | 1.15
LatentSLAM [31]	0.54 | 0.21	0.438 | 0.239	**1.266** | 0.274
ORB [204]	**0.456** | 0.165	**0.407** | **0.215**	4.258 | 1.595
CL DLBP (Ours)	0.588 | **0.1572**	0.742 | 0.329	1.596 | **0.244**

Table 5.1: *Mean Absolute Errors of Localization and Mapping: MAE_L | MAE_M (the lower the better).*

Fig. 5.8, 5.9 and 5.10 show the SLAM trajectories obtained by each method, next to the proposed DLBP-based CL system. As expected based on prior work [31, 275], the original RatSLAM system does not perform well in our ambiguous warehouse environment since its loop closure detection approach, based on the matching of raw image data, is too sensitive to the ambiguities between different views (see Fig. 5.7).

Latent-SLAM is among the top performers in Table 5.1, but requires the offline training of an 11-layer DNN on a dataset from the specific environment where SLAM must be performed. This makes LatentSLAM not suited for navigation in unknown environments that are not captured in datasets that are available beforehand, which was the goal of the proposed CL system. The use of handcrafted ORB features, extensively used for SLAM [167], performs well

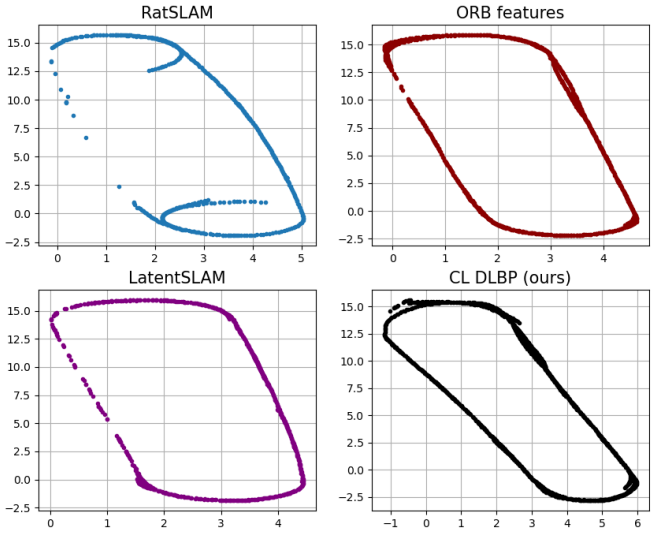

Figure 5.8: *SLAM result for flight 1, blue path in Fig. 5.7.*

in between the storage aisles (e.g., view 2 in Fig. 5.7) but also performs poorly at the end of the aisles, where lighting conditions suddenly change, leading to unreliable feature matching. Still, the SLAM back-end is able to recover the track in most cases (see Fig. 5.8 and Fig. 5.9) but sometimes also fails when, at the same time, the drift in raw odometry becomes too unreliable (see Fig. 5.10). This leads to large errors during *flight 3* in Table 5.1.

Our DLBP-based CL method, on the other hand, either outperforms or reports an MAE performance close to both the pre-trained LatentSLAM and the use of ORB features, while:

1. having a complexity similar to a 1-hidden-layer network;
2. not requiring any pre-training.

Therefore, the proposed CL DLBP approach appears to be a promising avenue for safety-critical navigation in unknown places where a dataset is not available beforehand.

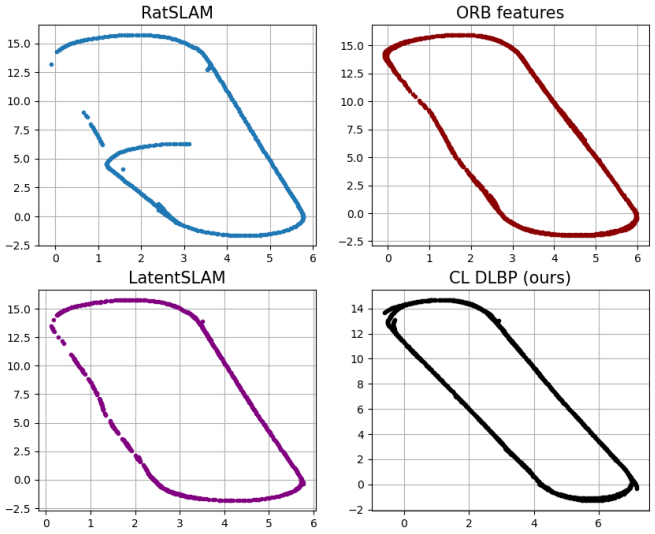

Figure 5.9: *SLAM result for flight 2, red path in Fig. 5.7.*

5.5 Results with DVS-radar fusion via SNN-STDP

We will now compare the *mean absolute error* (MAE) of the proposed **SNN-STDP-based** DVS-radar fusion method against state-of-the-art RGB-based solutions on three different flight sequences:

- *Flight a*: flying through the red path in Fig. 5.7.
- *Flight b*: a combination of all the paths (red, blue and green) in Fig. 5.7.
- *Flight c*: flying through the blue and the red paths *with strong lighting variations* during the flight (by randomly switching the lights on and off).

Table 5.2 reports the SNN-STDP parameter values used in our experiments, tuned manually for optimising the SLAM performance, starting from the values in Table 4.1 (see section 4.6.1 in chapter 4) [209]. In Table 5.2, η_d is the STDP learning rate. η_c is the SNN coding rate [209]. The number of coding neurons M is the same for all SNN ensembles in Fig. 5.4. The number of error neuron $N_{+,-,r}$ is equal to the dimensions of the input image to each SNN, where $+, -, r$

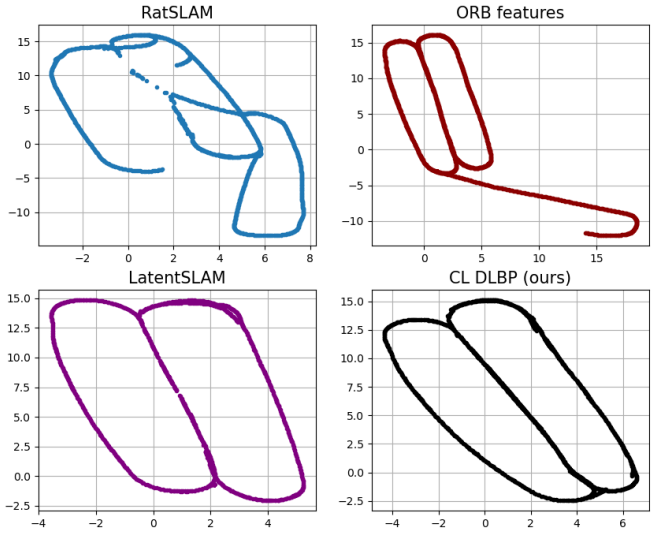

Figure 5.10: *SLAM result for flight 3, both red and blue paths in Fig. 5.7.*

respectively denote the SNN ensemble for the positive DVS, the negative DVS and the radar data in Fig. 5.4. $\mu_{+,-,r}$ and $\tau_m^{+,-,r}$ are the neuron thresholds and membrane constants in (1.1) for the respective SNNs. $A_{p,n}$, $\tau_{p,n}$ are the STDP learning parameters (1.6) in section 1.2.3.

η_d	η_c	M	$N_{+,-}$	N_r	$\mu_{+,-}$	μ_r
3×10^{-4}	1	64	65×86	15×30	0.3	0.03

$\tau_m^{+,-}$	τ_m^r	A_p	A_n	τ_p	τ_n
0.017 s	0.17 s	1	0.8	0.0208 s	0.008 s

Table 5.2: *SNN-STDP network parameters.*

5.5.1 Normal lighting conditions

Table 5.3 reports the MAE_L and MAE_M of our approach against state-of-the-art solutions that make use of standard RGB camera data (same RGB SLAM solutions as in section 5.4.2). Table 5.3 also reports ablation results when using only one modality alone.

The RatSLAM back-end integrates the odometry of section 5.3.3 and detects *loop closures* by sampling the latent codes $\bar{\mathcal{C}}_k$ (see Fig. 5.4) at 100 ms intervals and by testing the similarities $s_{ij}, \forall j$ against a threshold μ tuned for minimum MAE ($\mu = 0.65$ for radar-only, $\mu = 0.6$ for DVS-only and DVS-radar).

Architecture	Flight a (L)	Flight a (M)	Flight b (L)	Flight b (M)
RatSLAM [159]	0.72	0.22	2.49	0.57
LatentSLAM [31]	0.54	0.21	1.68	0.39
ORB features [204]	**0.46**	**0.17**	0.95	**0.39**
Ours (Radar)	0.68	0.28	0.98	0.64
Ours (DVS)	1.65	0.19	0.84	0.51
Ours (DVS-radar)	0.51	**0.17**	**0.81**	0.45

Table 5.3: *MAE_L (Localization, L) and MAE_M (Mapping, M). The lower the value the better.*

In addition to Table 5.3, Fig. 5.11 and 5.12 visually show the SLAM results for both Flight a and b. It can be remarked on both Fig. 5.11, 5.12 and in Table 5.3 that the DVS-Radar fusion setup outperforms the Radar-only and the DVS-only cases, by reaching lower MAE_L and MAE_M values. This clearly shows the advantage of DVS-Radar fusion.

Compared to the previously proposed RGB-based solutions, our proposed system outperforms both the original RatSLAM and the LatentSLAM, which makes use of a 11-layer DNN trained *offline*. In addition, our system either outperforms or reaches MAE_L and MAE_M performances close to ORB features used with RGB.

These findings are of great value given:

1. the small size of our SNN-STDP networks (equivalent to a 1-hidden-layer network in terms of complexity versus 11 layers in LatentSLAM);
2. the fact that our network is not pre-trained offline as in LatentSLAM, but is rather initialized randomly and learns features from the input data *on the fly* via *unsupervised* continual STDP adaptation;
3. the fact that it does not use RGB as in most SLAMs and solely relies on *lower-fidelity* DVS and radar data.

In contrast to the RGB-based solutions of Table 5.3, our DVS-radar setup also enables SLAM under strong lighting variations, where RGB solutions fail, as it will be demonstrated now in section 5.5.2.

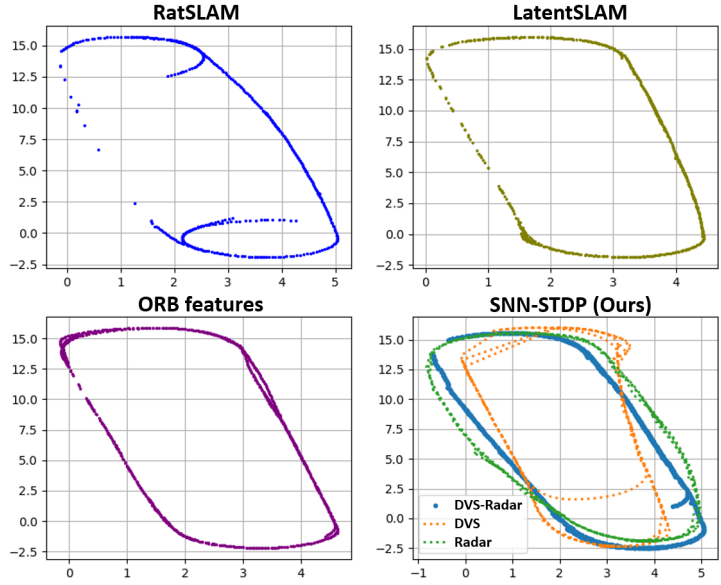

Figure 5.11: *SLAM results for flight a (dimensions in meter); the obstacle modelling of section 5.3.4 is not plotted for clarity.*

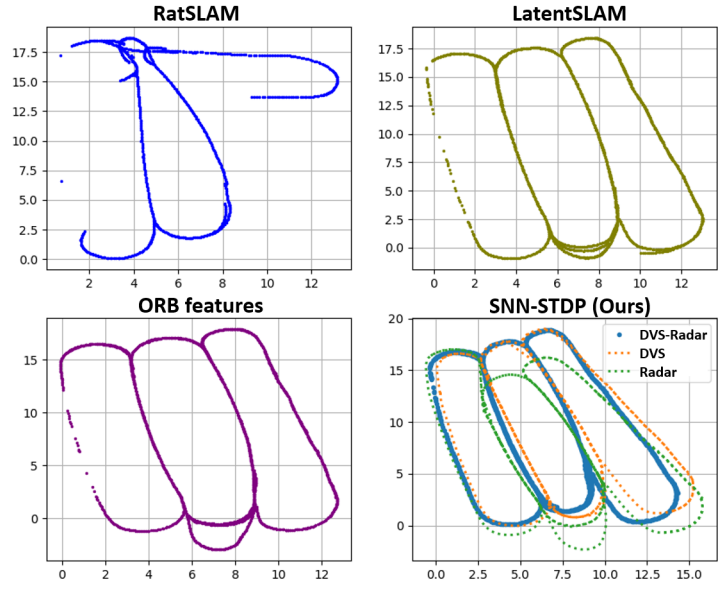

Figure 5.12: *SLAM results for flight b.*

5.5.2 Under strong lighting variations

Fig. 5.13 shows the SLAM results obtained for flight c, where strong lighting variations have been introduced by randomly turning on and off a subset of the neon lighting in the warehouse during the drone flight. The obstacle aggregation of section 5.3.4 is also shown, where walls and racks are modelled by the black dots. Fig. 5.13 clearly demonstrates the *robustness* of our DVS-radar approach in low light, compared to the other methods.

It must be noted that systems fusing RGB with radar have been proposed for having robustness towards environmental conditions [31, 270]. However, the RGB sensor still remains unreliable in low lighting, while in our architecture, both the DVS and the radar stay functional under low light.

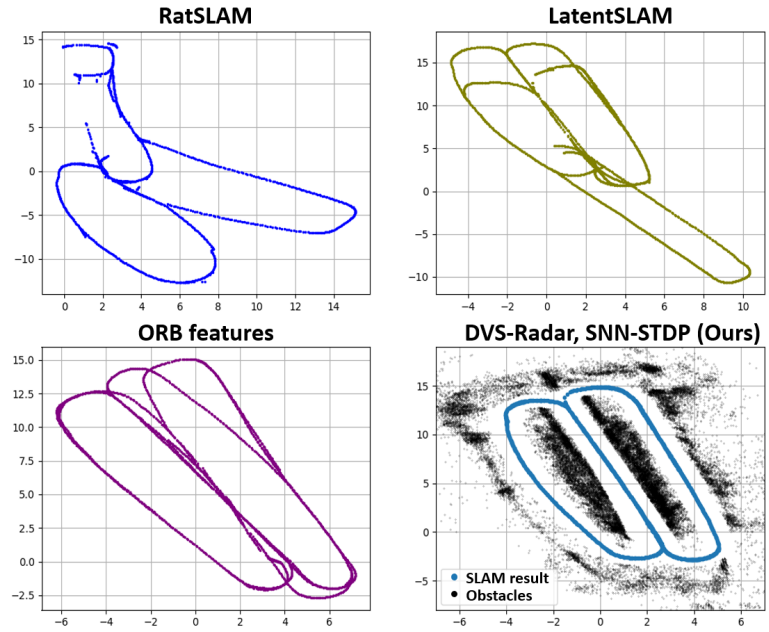

Figure 5.13: *SLAM result for flight c (strong lighting variations). As opposed to the previously-proposed SLAM setups, our proposed system is robust to the strong lighting variations as it can better recover the map of the room.*

5.5.3 Comparing pre-trained SNN-STDP vs. continual-learning SNN-STDP

In this section, we will experimentally compare our *continual learning* SNN-STDP SLAM system against the case where the SNN has been pre-trained *a priori* and then deployed in inference mode only. The goal of this experiment is to demonstrate that the *continual personalization* of the SNN-STDP during SLAM leads to an enhanced localization and mapping performance compared to the conventional pre-trained approach widely used in neural networks.

To this end, we first *pre-train* the DVS-radar fusion SNN-STDP network in Fig. 5.4 using the data acquisition of flight a) in section 5.5 as the train set. Then, we compare the SLAM result obtained using the pre-trained model against the result obtained via the continual-learning SNN-STDP setup on flight b) and flight c) in section 5.5 (where the lights are turned on and off). There results are reported in Fig. 5.14 and 5.15.

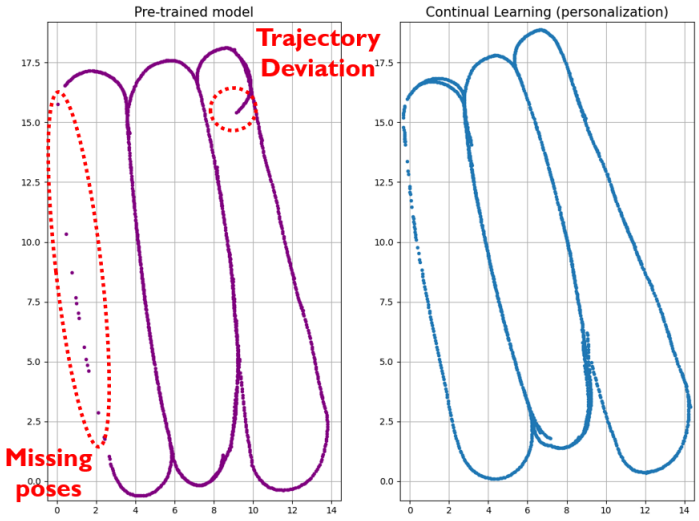

Figure 5.14: *Pre-training vs. continual learning for SLAM on flight b) of section 5.5. The continual-learning setup clearly outperforms the pre-trained method in terms of mapping quality.*

Both Fig. 5.14 and 5.15 show clearly that the continual-learning SNN-STDP outperforms the pre-trained SNN-STDP model since the SLAM results obtained using the pre-trained model contain many missing poses and trajectory deviations. This is due to the fact that the pre-trained model naturally has not



Figure 5.15: *Pre-training vs. continual learning for SLAM on flight c) of section 5.5. The continual-learning setup clearly outperforms the pre-trained method in terms of mapping quality.*

seen all possible scenarios during its training phase and cannot generalize well to places in the warehouse that deviate too much from its training data (e.g., the left aisle in Fig. 5.14). In addition, the random light switching during flight 3) in Fig. 5.15 poses additional generalization challenges to the pre-trained model since light switching was not present during the offline training phase.

Therefore, these results clearly demonstrate that the *continual personalization* of the SNN-STDP system to changes in the environment leads to a better SLAM performance compared to the conventional offline training approach. This strongly motivates the need for more future research in continual learning and unsupervised personalization for robotics, IoT and AI applications in general.

We can summarize the various results of the SLAM demonstrations provided in this chapter as follows:

The continual SNN-STDP SLAM system presented in this chapter significantly innovates by a) not requiring any offline pre-training phase, b) being robust to lighting conditions, c) reaching either better or on-par performance compared to conventional offline pre-trained approaches with much higher complexity, and d) even outperforming the offline training of SNN-STDP (see section 5.5.3).

5.6 Considerations regarding the safety of the SLAM system

Even though our demonstrations have shown the stability of the proposed continual-learning algorithms (see Fig. 5.2), ensuring absolute safety is still a critical but challenging endeavor. To increase the overall safety of the proposed SLAM system, redundancy could be added to the loop closure detection subsystem (which is the most critical mechanism impacting the SLAM system), by fusing the SNN-STDP output with the output of typical feature descriptors such as SIFT, SURF, ORB and so on. Doing so, loop closure detection will be done *through consensus*, by fusing the decision on multiple different feature descriptor approaches.

As interesting future work for further increasing and potentially even guaranteeing the safety of the proposed continual-learning systems, it may be possible to fuse the concept of *dynamic scenarios* proposed in [138] with the algorithms described in this chapter. This will provide additional guidelines for increasing the safety of the system, by adding higher-level system control to guarantee a number of targets (such as not hitting sensitive obstacles like people).

The *dynamic scenario* method proposed in [138] is a general method that can be used to ensure that a system meets a number of requirements and constraints (such as e.g., not hitting walls, pedestrians and so on), while reducing compute and application overheads. The traditional methods for ensuring safety are based on worst-case analysis [138]. But because they focus solely on the worst-case scenario, these traditional methods come with large hardware overheads and ignore the *real behavior* of the system (which exhibit non-chaotic correlations as well). The dynamic scenario method contrasts with this *traditional* method, termed Worst Case Execution Time (WCET) [16], for ensuring constraint requirements based on worst-case scenario analysis. Indeed, complex systems (such as the proposed SNN-STDP SLAM architecture of this chapter) are often non-linear and non-additive, which makes them *chaotic*. This chaotic nature essentially means that for small changes to the system's input, large changes can be observed at the system's output. This chaotic nature makes these complex systems unpredictable in their detailed future evolution.

The dynamic scenario method proposed in [138] exploits the existing correlations in the dynamics of the complex systems that need to be controlled.

To illustrate this, a scenario where a drone needs to navigate from a starting point to an ending point is considered. During the navigation, multiple routes can be taken, with each having *a different trade-off* between safety versus

hardware and navigation overheads (e.g., energy costs and navigation speed). Each of these trade-offs can be modelled as a Pareto curve denoting e.g., the hardware overheads as a function of the system's safety. Within this context, the dynamic scenario method proposed in [138] is capable of clustering the huge multitude of different Pareto curves into a very limited set of main scenarios (see the qualitative illustration in Fig. 5.16), without enumerating one by one each possible Pareto trade-off (i.e., each possible drone path in the case of our example).

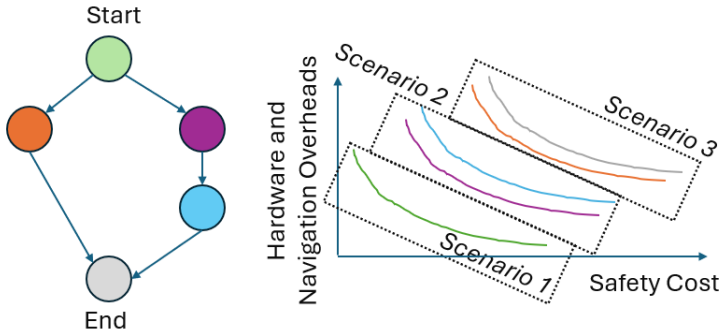

Figure 5.16: *The drone can follow multiple different paths from its starting position to its end goal. Each path comes with a trade-off between hardware and navigational overheads versus safety costs. Each trade-off can be illustrated as a Pareto curve. The dynamic scenario method exploits the correlations inside the complex dynamic system that it controls by clustering the different paths as distinct scenarios.*

Then, using these clustered scenarios and exploiting again the existing correlations, it is possible to precisely define an upper and a lower bound on the system constraint (e.g., on the safety rating). It is important to note that these bounds are not constant, but rather change as the dynamic system evolves (e.g., as the drone navigates in the environment). Using these bounds, it is possible to dynamically trade off safety for hardware efficiency by repeatedly re-evaluating the safety bounds when the drone navigates in the environment. For example, when the drone is far from a walking person, safety bounds are less tight, and hence safety can be traded off for energy consumption or navigation speed. But when a drone comes close to a walking human, the safety bounds become more strict, and hence safety must be ensured by trading off hardware overheads and navigation speed (see the qualitative illustration in Fig. 5.17).

Therefore, it is expected that studying how to fuse the dynamic scenario method [138] with the SNN-STDP SLAM system proposed in this chapter constitutes an attractive path towards ensuring navigation safety while reducing

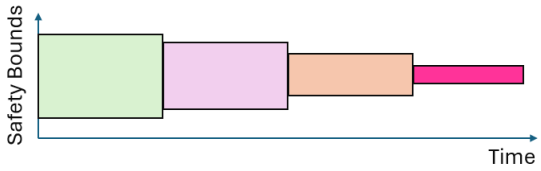

Figure 5.17: *During drone navigation, the lower and upper bounds on the drone safety are estimated by the dynamic scenario algorithm. As the drone gets closer to e.g., a human being, these bounds become tighter, which pushes the drone to have more emphasis on safety by trading related hardware and navigational overheads.*

hardware and navigational overheads. We refer the interested reader to chapter 4 of the thesis in [137] for a detailed description of how the dynamic scenario method works, together with mathematical proofs that provide strong confidence for using this method.

5.7 Conclusions

This chapter has presented both the DLBP-based and SNN-STDP-based methods for continually learning to encode visual sensory data into feature descriptors, used for performing SLAM. Unlike most learning-based SLAM systems, where a DNN is trained offline using a dataset captured beforehand, the proposed methods do not require any pre-training, but rely on the continual and *unsupervised* adaptation of the visual encoder networks (either DLBP or SNN-STDP) when the drone explores the environment, thanks to the use of the QBS factor for modulating learning. Experiments conducted in both an RGB-DLBP SLAM setting and a DVS-radar fusion setting via SNN-STDP have shown that the proposed methods compare favorably to state-of-the-art RGB-based solutions and models that have been pre-trained offline, while even enabling robust navigation and obstacle modelling under strong lighting variations in the DVS-radar fusion case. The presented methods therefore have demonstrated the capability and usefulness of SNN-STDP networks for continually learning to perform SLAM in unknown environments not captured by datasets available beforehand. In the next chapter, the SNN-STDP framework, previously developed in chapter 4 and used in this chapter, will be utilized for performing people detection on drones from DVS data.

Chapter 6

Continual-learning people detection from DVS data

6.1 Introduction

The previous chapter has investigated how the top-down SNN-STDP design approach presented in chapter 4 can be used for continually learning to encode sensory data in order to perform Simultaneous Localisation and Mapping (SLAM). This chapter will show the broader applicability of the proposed SNN-STDP framework, and will explore how our top-down SNN-STDP design approach from chapter 4 can be leveraged for the task of continually learning to detect people from event data acquired via a DVS camera mounted on a drone.

In addition to SLAM, which has been covered in the previous chapter, another fundamental task for autonomous agents such as surveillance and rescue drones is the ability to detect human subjects in order to provide collision avoidance for safety purposes or for localizing people during search and rescue operations [238].

In this chapter, our goal is to investigate the design of a bio-inspired SNN-STDP system that can *continuously learn* attention-based people detection from DVS

Adapted from: *A. Safa et al., "STDP-driven Development of Attention-based People Detection in Spiking Neural Networks," in IEEE Transactions on Cognitive and Developmental Systems, 2022.*

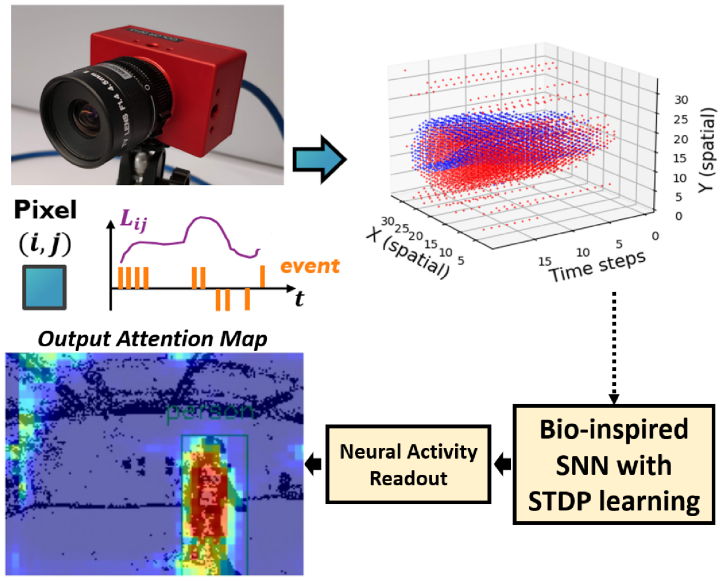

Figure 6.1: *SNN-STDP architecture for continually learning to produce attention maps for people detection.*

data acquired by a flying drone, without any pre-training phase. Fig. 6.1 illustrates the scenario considered here: the drone must explore the environment during about 2 minutes and continuously learn to detect a walking human subject using the current data sample only, without access to the past and the future data. Hence, SNN learning will be done *on the fly*, mimicking the real-time learning observed in biological agents. This also makes our problem fundamentally more challenging since the data sequences are *non-independent and identically distributed* (non-i.i.d) [129] (see a discussion on the challenges posed by continual learning in section 5.2.1). Then, when the learning sequence reaches its end, we will assess the detection performance on a testing sequence in order to characterize the performance of the proposed system.

The major contributions of this chapter are the following:

1. **Algorithm:** We will generalize the previously-proposed *unsupervised* SNN-STDP framework of chapter 4 to the *semi-supervised* learning case where teacher signals are available, and where the network must learn under a continual-learning setting without any offline pre-training.

2. **Bio-plausible learning theory:** We will provide a new interpretation of the *anti-Hebbian*, negative STDP learning rule [39,94] as an over-fitting

control mechanism.

3. **Experiments:** We will experimentally demonstrate our findings by continuously learning to detect human subjects from *non-i.i.d* streams of retina-inspired event data captured by a drone, in order to simulate the *continual* development of *attention-based perception* in biological systems.

This chapter is organized as follows. Section 6.2 will describe the proposed SNN-STDP architecture for continually learning to produce attention map in order to detect walking human subjects. Section 6.3 will provide experimental results and compare the performance of the proposed approach against the use of a CNN with similar memory footprint. Finally, conclusions will be provided in section 6.4.

6.2 Network architecture

In this section, we will first describe how the SNN-STDP architecture previously presented in Chapter 4 can be extended to the *convolutional* case needed for producing attention maps. Then, an overview of the network architecture will be provided to show how the convolutional SNN-STDP system will be integrated with a readout layer in order to localize walking human subjects. Finally, the semi-supervised learning mechanism based on both Hebbian and anti-Hebbian plasticity rules will be presented.

6.2.1 SNN-STDP in a convolutional setting

In order to provide *translational invariance* needed for producing attention maps [185], the fully-connected SNN-STDP network introduced in chapter 4 (see (4.1)) is extended to the *convolutional* setting:

$$\bar{c}, \Phi = \arg\min_{\bar{c},\Phi} \frac{1}{2}||\Phi * \bar{c} - \bar{s}||_2^2 + \lambda_1||\bar{c}||_1 \tag{6.1}$$

where $*$ denotes the convolution operator, $\bar{s}$ is the input vector, $\bar{c}$ the output sparse code, Φ the *convolutional* dictionary and λ_1 the sparsity hyper-parameter.

In order to solve (6.1), the SNN-STDP network in Fig. 4.1 is extended to the *convolutional* setting by sweeping the network input across an $H \times W$ image plane $\mathcal{S}(t)$, with kernel size $[k_H, k_W]$ and stride d. Doing so, a spiking tensor $\mathcal{C}(t)$ of the size $[D_H, D_W, M]$ is then obtained:

$$[D_H, D_W, M] = [\frac{H - k_H}{d} + 1, \frac{W - k_W}{d} + 1, M] \tag{6.2}$$

where M is the number of coding neurons in the SNN-STDP network (see section 4.2 and Fig. 4.1).

For each hop h of the SNN convolution, (6.1) can be writen under the same form as (4.1) by considering $\bar{s}^h$ as the *flattened* $k_H \times k_W$ visual field of hop h and $\bar{c}^h$ as the M-dimensional outputs of hop h:

$$\bar{c}^h = \mathcal{C}(t)[x_h, y_h] \quad \rightarrow \quad \bar{c}^h, \Phi = \arg\min_{\bar{c}^h, \Phi} \frac{1}{2}||\Phi\bar{c}^h - \bar{s}^h||_2^2 + \lambda_1||\bar{c}^h||_1 \tag{6.3}$$

where x_h, y_h are the tensor entries corresponding to hop h. Fig. 6.2 conceptually shows how the SNN-STDP network in Fig. 4.1 can be swept across the DVS image plane $\mathcal{S}(t)$ in order to produce a $D_H \times D_W \times M$ tensor $\mathcal{C}(t)$ as the result of the convolution operation (keeping Φ identical across all hops h).

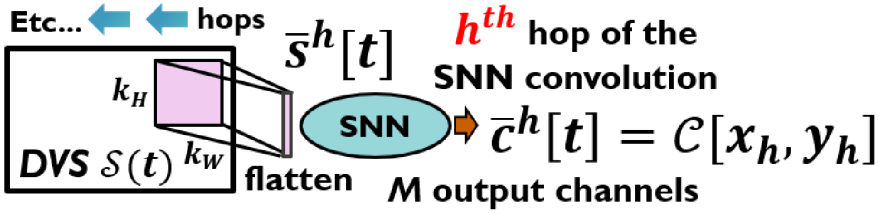

Figure 6.2: *Sweeping the SNN-STDP of Chapter 4 across the image plane in order to produce a convolutional output map.*

Doing so, the fully-connected SNN-STDP originally introduced in chapter 4 is re-formulated under the convolutional case, where the input data are processed by a convolutional filter bank (instead of the fully-connected weights). This convolutional input field enables the SNN-STDP to be invariant to translations, which is important for successfully detecting the location of an object or a person who could be in any place on an image plane [185].

6.2.2 SNN-readout system overview

Fig. 6.3 shows the continual-learning (CL) SNN-STDP system used throughout this work to infer attention maps (noted $\mathcal{A}$). We set up a fully-convolutional network by cascading the convolutional SNN-STDP of Fig. 6.2 with a *convolutional logistic regression* readout in order to infer $\mathcal{A}$. At the output of the convolutional readout, we use *sigmoid* functions (noted σ_s) to constrain the dynamic range of the attention map between 0 and 1.

The *spike rates* of the SNN output $\mathcal{C}(t)$ are estimated using a rolling time window of length $N_s = 20$. Doing so, a tensor $\mathcal{C}$ (see (6.3)) containing the spike

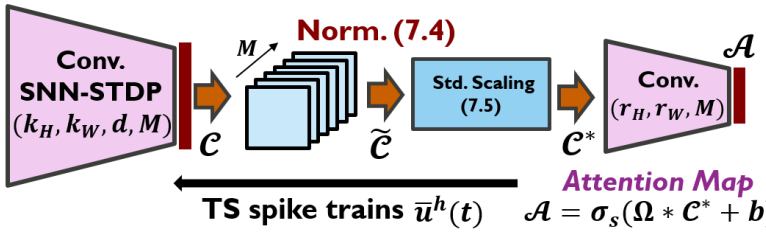

Figure 6.3: *Convolutional SNN-STDP and readout architecture for studying continual SNN-STDP learning with the availability of teacher signals (TS).*

rates for each spatial coordinate $[x, y]$ and each channel $a = 1, ..., M$ in (6.2) is obtained. Then, $\mathcal{C}$ is normalized along a for invariance to the event density:

$$\tilde{\mathcal{C}}[x, y, a] = \frac{|\mathcal{C}[x, y, a]|}{\sum_a |\mathcal{C}[x, y, a]|} \quad \forall x, y \in [D_H, D_W] \tag{6.4}$$

After (6.4), we use *standard scaling* along the channel dimension a to reduce the internal covariate shifts [102] (i.e., the shifts in the SNN-STDP output distribution due to the CL process, similar to (5.20) in section 5.3.2 of chapter 5):

$$\mathcal{C}^*[x, y, a] = \frac{\tilde{\mathcal{C}}[x, y, a] - \mu_a}{\sigma_a} \quad \forall x, y \in [D_H, D_W] \tag{6.5}$$

where the mean μ_a and the standard deviation σ_a are refined on-line by accumulation. Then, $\mathcal{C}^*$ is zero-padded and fed to the readout layer with a convolutional kernel Ω of size $[r_H, r_W, M]$ and bias b:

$$\mathcal{A} = \sigma_s(\Omega * \mathcal{C}^* + b) \tag{6.6}$$

where $\mathcal{A}$ has shape $[D_H, D_W]$ following (6.2). Finally, we interpolate $\mathcal{A}$ back to the input size.

Without the use of the normalization mechanisms (6.4) and (6.5), the CL process fails due to the severe covariate shifts encountered during learning. In addition, using a convolutional readout layer over a fully-connected layer is crucial for avoiding readout *over-fitting* on the local contexts encountered during the CL process (thanks to the weight sharing property of convolutionl layers [182]).

6.2.3 Semi-supervised learning in SNN-STDP

The architecture in Fig. 6.3 also indicates that the readout generates *teacher signal* (TS) spike trains $\bar{u}^h(t)$ that are fed back to the SNN neurons, for each

hop h of the SNN input convolution. The TS spike trains *steer* the STDP-driven weight dynamics in order to *jointly* optimize the SNN weights Φ and the readout Ω, forming a semi-supervised learning system.

Indeed, during our initial investigations, we observed that *independently* optimizing the SNN-STDP system and the readout does not perform well in our specific *continual-learning* setting. Rather, the SNN and the readout must be learned concurrently. Indeed, if the learning process would be independent, the way how the readout should fit the SNN output in the beginning of the STDP learning process is not the same as in the end, since the SNN weights have *independently* changed since then, leading to significant *covariate shifts* at the input of the neural activity readout.

In this work, we use the semi-supervised, *joint SNN and readout* system of Fig. 6.3, where we introduce $\mathcal{L}_s$ as the *supervised* loss function, $\mathcal{Y}$ as the desired attention map (i.e., the label), and $\lambda_{u,s}$ as the hyper-parameters controlling the strength of the unsupervised and supervised contributions in our updated formulation (vs. only unsupervised in (6.1)):

$$\bar{c}, \Phi, \Omega = \arg\min_{\bar{c},\Phi,\Omega} \lambda_u \mathcal{L}_u(\bar{c}, \Phi) + \lambda_s \mathcal{L}_s(\bar{c}, \Phi, \Omega | \mathcal{Y}) \tag{6.7}$$

As loss function for learning the attention maps, we choose the *focal loss* because of its robustness to imbalanced data vs. the standard cross-entropy [139] (where the subscript q denotes the qth tensor element):

$$\mathcal{L}_s = \sum_q -(1-p_q)^\gamma \log(p_q) \tag{6.8}$$

with γ setting the robustness to class imbalance and:

$$p_q = \begin{cases} \mathcal{A}_q \textbf{ if } \mathcal{Y}_q = 1 \\ 1 - \mathcal{A}_q \textbf{ otherwise} \end{cases} \tag{6.9}$$

In order to optimise $\mathcal{L}_s$ as a function of the SNN weights Φ, the link between the SNN-STDP output tensor $\mathcal{C}$ in (6.3) and the SNN weight dictionary Φ must be found. To simplify the notations, we study this problem for the hop h of the SNN input convolution with the corresponding *flattened* input visual field $\bar{s}^h$ and $\bar{c}^h = \mathcal{C}[x_h, y_h]$ the M-dimensional SNN output of hop h in (6.3).

Since $\bar{c}^h$ is the result of an *iterative process* involving the LIF neuron non-linearity, an explicit function $\bar{c}^h(\Phi, \bar{s}^h)$ is hard to find and further approximations must be done. First, we observe in (6.3) that for a small λ_1, $\bar{c}^h_r(\Phi, \bar{s}^h_r)$ is well approximated as:

$$\bar{c}^h_r(\Phi, \bar{s}^h_r) \sim (\Phi^T \Phi)^{-1} \Phi^T \bar{s}^h_r \tag{6.10}$$

Since Φ is initialized following an i.i.d. zero-mean normal distribution with standard deviation σ_w [209], the following relaxation holds during the *early* SNN-STDP learning steps (when most of the learning effect takes place):

$$\bar{c}^h(\Phi, \bar{s}^h) \sim \frac{1}{N\sigma_w^2}\Phi^T \bar{s}^h \tag{6.11}$$

Then, the gradient of $\mathcal{L}_s$ as a function of the SNN-STDP dictionary Φ is:

$$\frac{\partial \mathcal{L}_s}{\partial \Phi} = b\bar{s}^h \underbrace{(\frac{\partial \mathcal{L}_s}{\partial \bar{c}^h})^T}_{\bar{v}^h} \tag{6.12}$$

where $b = 1/(N\sigma^2)$ is a constant (dropped further on), and the gradient of the focal loss with respect to the SNN output $\partial \mathcal{L}_s / \partial \bar{c}^h$ directly follows from (6.4) to (6.8). This gradient can be converted into the TS push-pull *spike trains* using:

$$\bar{u}^h(t) = LIF\{\bar{v}^h\} - LIF\{-\bar{v}^h\} \textbf{ with } \bar{v}^h = \frac{\partial \mathcal{L}_{sup}}{\partial \bar{c}^h} \tag{6.13}$$

where $LIF\{.\}$ denotes the application of the LIF nonlinearity (1.1). Finally, using the rate-based STDP model (4.23), the effect of (6.12) can be incorporated in the learning of the SNN weights by locally applying STDP between the spike trains $\bar{s}^h(t)$ and $\bar{u}^h(t)$. This is repeated for each hop h of the SNN visual field.

The supervised (positive) STDP contributions for the *input* and *feedback* weights are given by (6.14) and (6.15) respectively:

$$\Phi_{ij} \leftarrow \Phi_{ij} - \eta_d \lambda_s \Delta w\{s_i^h(t), u_j^h(t)\} \ \forall h \tag{6.14}$$

$$\Psi_{ij} \leftarrow \Psi_{ij} - \eta_d \lambda_s \Delta w\{u_j^h(t), s_i^h(t)\} \ \forall h \tag{6.15}$$

where η_d is the learning rate, and the *feedback* weight modification (6.15) is *transposed* with regard to the *input* weights (see Fig. 4.1). Under the light of the long-term STDP approximation (4.23), it can be seen that both (6.14) and (6.15) implement an *approximated gradient descent*, with the gradient given by (6.12), thus minimizing the *supervised* objective $\mathcal{L}_s$. In both (6.14) and (6.15), the TS spike trains $\bar{u}^h(t)$ are injected *locally* to each neuron in the coding and error layers in order to steer the weight dynamics according to the supervised contribution.

Finally, the lateral weights W in Fig. 4.1 do not need to be modified explicitly via the TS spike trains, since the STDP mechanism already affecting W in [209] continuously forces W to converge to $\eta_c \Phi^T \Phi - I_M$ in order to keep the consistency needed between all weights for performing an approximate LASSO coding of $\bar{s}(t)$ [209] (see the discussion on the weight symmetries in section 6.2.1 below).

Therefore, we add the *TS-driven* STDP contributions of (6.14) and (6.15) to the *unsupervised* STDP contribution already present in the network of Fig. 4.1.

6.2.4 Anti-Hebbian STDP for reducing over-fitting

In this section, we describe the use of anti-Hebbian STDP for the unsupervised contribution in (6.7), as an important mechanism for alleviating overfitting during the continual learning process, which helps reducing catastrophic forgetting issues (as will be shown experimentally in section 6.3.5). For background on catastrophic forgetting, we refer the reader to section 1.3.1 of the Introduction chapter.

In addition to the use of the standard *positive* STDP rule already described in section 1.2.3, the use of the *anti-Hebbian*, negative STDP rule has also been investigated [39, 94]:

$$w_{ij} \leftarrow \begin{cases} w_{ij} - A_+ e^{-\tau_{ij}/\tau_+}, & \text{if } \tau_{ij} \geq 0 \\ w_{ij} + A_- e^{\tau_{ij}/\tau_-}, & \text{if } \tau_{ij} < 0 \end{cases} \tag{6.16}$$

where the synaptic strength modification is opposite to (1.6). The rate-based approximation of (6.16) directly follows from (4.23):

$$\Delta w_{\text{-}}\{s_{post}, s_{pre}\} \approx -(A_+\tau_+ - A_-\tau_-) r_{post} r_{pre} \tag{6.17}$$

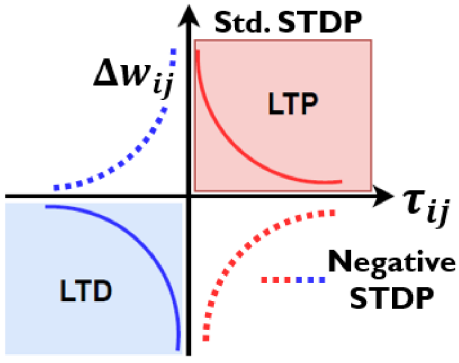

Figure 6.4: *The standard positive (1.6) vs. the negative (6.16) STDP rules as a function of the difference between the post- and the pre-synaptic spike times* $\tau_{ij} = t_{post,i} - t_{pre,j}$.

The optimization-based approach adopted in this work enables us to formulate a new interpretation of the less-investigated *anti-Hebbian* negative STDP rule (6.16) [39,94]. In section 6.2.1, it has been explained that the use of the standard *positive* STDP rule (1.6) *minimizes* the re-projection error $||\Phi\bar{c} - \bar{s}||_2^2$ in (6.3).

For the remainder of this discussion, it is useful to consider a probabilistic, *maximum-likelihood* (ML) framework. In the ML framework, the standard *positive* STDP is learning a dictionary Φ such that the incoming input spikes $\bar{s}(t)$ are associated to a *high probability* of belonging to the SNN likelihood model:

$$\mathcal{P}(\bar{s}|\Phi) \sim \exp\left(-||\Phi\bar{c} - \bar{s}||_2^2\right) \tag{6.18}$$

Under the light of (6.18), a probabilistic interpretation can also be given to *negative* STDP. Negative STDP acts as a mechanism that seeks to modify the SNN weights Φ in order to push some input spike trains $\bar{s}(t)$ towards low likelihood values in (6.18), making them outliers with respect to the likelihood model of the SNN-STDP ensemble. Hence, the use of *anti-Hebbian*, negative STDP could be useful for alleviating the *over-fitting* of the SNN on data streams $\bar{s}(t)$ that are *outliers* with regard to the task that one aims at solving.

For the people detection task considered in this chapter, we want the inputs $\bar{s}(t)$ corresponding to cases where *no person* is present, to be associated with a *low likelihood* value in (6.18) in order to avoid over-fitting on those untargeted *outlier* contexts. Therefore, when *no human* is present in the event camera data, we use the *negative* STDP rule for the *unsupervised* contribution in (6.7) instead of positive STDP (see section 6.3.5).

On the other hand, we *only* use the TS-driven STDP contributions (6.14) and (6.15) when a person *is present* in the learning data sequence since we have observed that adding *positive* STDP in this case introduces antagonistic effects to the *TS-driven* STDP contribution and degrades the results.

6.3 Experimental setup and results

In this section, we assess our theoretical findings and TS-driven STDP framework of section 6.2 for the task of attention-based *people detection* from real-world event camera data captured with a flying drone, in a continual-learning setting (see section 3.9.1 in the second part of chapter 3). This challenging scenario aims at simulating the *continual* development of *attention-based perception* in biological systems, where object detection is related to the activation of specific neurons in a grid-like topology forming an attention map, with regions of high activity relating to the location of the object of interest in the real world [149]. This is in contrast to the regression of bounding-box coordinates used in DNNs [197], where the position of the object of interest is directly regressed as a coordinate on an image plane.

6.3.1 Continual-learning strategy

The event camera dataset previously acquired in section 3.9.1 of chapter 3 and used in this chapter provides labels as *bounding box* coordinates $[b_{x,1}, b_{y,1}, b_{x,2}, b_{y,2}]$ (normalized to the size of the output map), indicating the location of a human subject in the image plane. When a human subject is present, we assign the corresponding *label attention map* $\mathcal{Y}$ in (6.7) with *ones* inside the bounding box region and *zeros* outside. Since there are significantly more zero-valued entries in $\mathcal{Y}$, we use *median frequency balancing* [52] to help mitigate the learning imbalance in space. Therefore, we re-weigh the *supervised* readout loss (6.8) as:

$$\mathcal{L}_s = \sum_q -(1-p_q)^\gamma \log(p_q) \times w_q \textbf{ with } w_q = \frac{F_m}{N_{class,q}} \tag{6.19}$$

where $N_{class,q}$ is the number of entries in $\mathcal{Y}$ of the same class as $\mathcal{Y}_q$ (the class can either be 0 or 1 in our detection case) and F_m is the mean of $N_{class=0}$ and $N_{class=1}$ [52].

When no human subjects are detected, the label map $\mathcal{Y}$ only contains null values for long periods, leading to *over-fitting* on this local context. We alleviate this problem by keeping Ψ *fixed* when no human subjects are present, and by only enabling learning for the SNN weights Φ.

6.3.2 Post-processing for performance evaluation

In order to quantitatively assess the system performance, a set of discrete detections is obtained from the attention map as follows. First, a threshold d_{th} is applied to the output attention map $\mathcal{A}$ and a point cloud is formed. Then, a set of detected clusters $\{C_k\}$ is obtained by applying DBSCAN clustering [56] with *min points* $= 2$ and $\epsilon = 5$ (empirically tuned). This set of detections is then used to measure the precision-recall curves [213] of our SNN-STDP attention-based detector.

6.3.3 System assessment

We assess our SNN-STDP architecture on the dataset for people detection on drones already introduced in the second part of chapter 3 [213]. The dataset features a collection of joint DVS and RGB acquisitions in an *indoor*, warehouse-like environment, where a human subject is walking randomly. During our experiments, we choose the three longest acquisitions (~ 2 min.) from the *f-wall*

collection in [213], where the drone and a human subject are moving in a space surrounded by walls, benches and racks (different human subjects are featured to add variability).

We report the performance of our system via 3-fold cross-validation as follows. First, one of the acquisitions is used as a *learning sequence* and is shown to our SNN-STDP system *only once*, in its natural order (the learning sequence *is not shuffled*). Then, when the end of the learning sequence is reached, *a different acquisition* is used to measure the *precision-recall* curves by sweeping d_{th} from 0 to 1 and measuring the number of false alarms, true positives and false negatives in $\{C_k\}$. This process is repeated three times using different learning and testing sets and the final precision-recall curves are obtained as the average over the three folds.

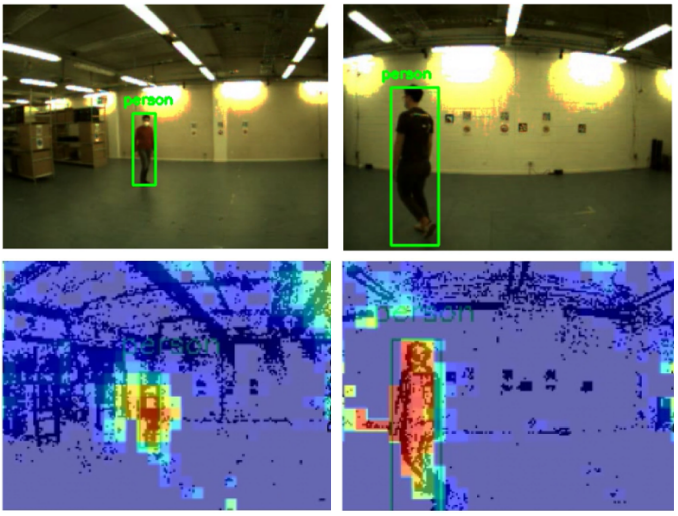

Figure 6.5: *Attention maps inferred from the DVS data by our SNN-STDP system. The label bounding boxes are also shown.*

Fig. 6.5 shows examples of attention maps produced by our system. Table 6.1 reports the SNN-STDP parameters that differ from the ones used previously in Table 4.1 (the other parameters, such as the STDP constants, were kept identical). Table 6.2 reports the parameters used in our system architecture (tuned empirically for maximising the detection rate). For the readout, we use the Adam optimizer [120] with a learning rate $\eta_r = 5 \times 10^{-6}$.

Next, we will study the effect of the focal loss parameter γ (6.8) and the hyper-parameters $\lambda_{u,s}$ (6.7) of the *TS-driven* STDP framework of section 6.2.

η_c	η_d	μ	τ_m
0.07	0.05	0.15	0.07

Table 6.1: *SNN-STDP learning parameters. The simulation time step is* 0.005 *s. The LIF decay τ_m is in seconds. λ_1 in (6.1) is implicitly set by the LIF threshold μ (see section 4.3.2).*

H	W	$k_{H,W}$	d	M	$r_{H,W}$	D_H	D_W
130	173	30	5	64	12	21	29

Table 6.2: *Architectural parameters. $H \times W$: input size. $k_H \times k_W$: SNN convolutional kernel size, d: SNN stride, M: number of SNN kernels (neurons in the coding layer), $r_H \times r_W \times M$: kernel size of the readout. $D_H \times D_W$: shape of the output attention map. The readout has a stride of 1 and zero padding.*

6.3.4 Impact of the focal loss parameter γ

In order to tune the system hyperparameters, we first explore the impact of γ in (6.8) by keeping all other parameters constant with $\lambda_u = 0$, $\lambda_s = 1$. Fig. 6.6 shows the measured precision-recall curves and also reports the peak F_1 score as $\max_i \frac{2P_iR_i}{P_i+R_i}$ along each precision-recall curve (P_i, R_i) [213]. Fig. 6.6 shows that the peak F_1 score is maximised near $\gamma = 0.5$. Therefore, we set $\gamma = 0.5$ in the next experiments, where we will explore the impact of the other hyperparameters to further maximize the detection performance.

6.3.5 Impact of unsupervised and supervised strengths λ_u, λ_s

Fig. 6.7 shows the precision-recall curves obtained by varying the *unsupervised* (negative STDP) and *supervised* contributions $\lambda_{u,s}$. Fig. 6.7 shows that the peak F_1 score is maximised near $\lambda_u = 0.2$, $\lambda_s = 0.8$ (optimal values for these are expected to be task-dependent). During our experiments, we also tried the use of the standard *positive* STDP rule (1.6) for the unsupervised contribution, but always reached lower performances by varying λ_u, due to an increase in false alarms.

In contrast, the use of the *negative* STDP rule (6.16) for the *unsupervised* contribution provides an *opposite weight-steering force* to the *positive* STDP rule used for the *supervised* contribution (6.14),(6.15), preventing the over-fitting of the system to structures in the environment. This leads to $\sim 15\%$ less false alarms at *high recalls* and confirms our analysis of *anti-Hebbian* STDP in section

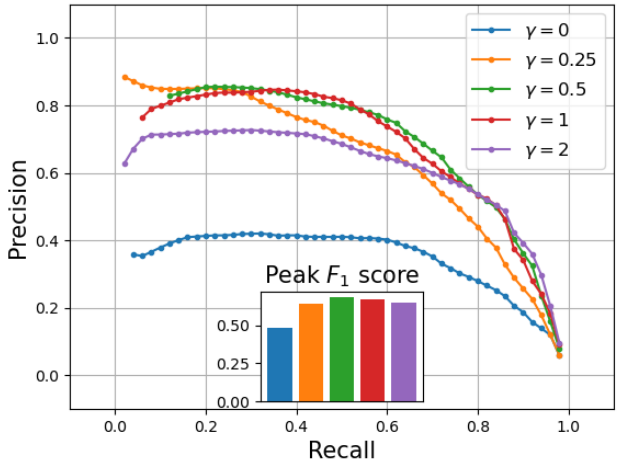

Figure 6.6: *Impact of the focal loss parameter γ on the precision-recall curves. The peak F_1 score is maximized near $\gamma = 0.5$.*

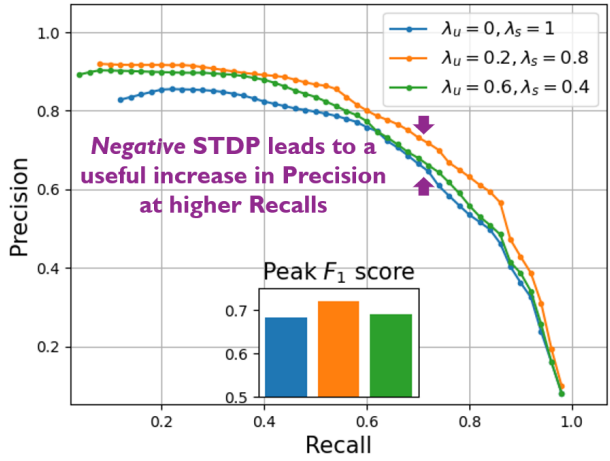

Figure 6.7: *Impact of the semi-supervised hyper-parameters $\lambda_{u,s}$ on the precision-recall curves. The peak F_1 score is maximized near $\lambda_u = 0.2$ and $\lambda_s = 0.8$.*

6.2.4 (see Fig. 6.8 b where the use of negative STDP significantly reduces the false alarms vs. Fig. 6.8 a).

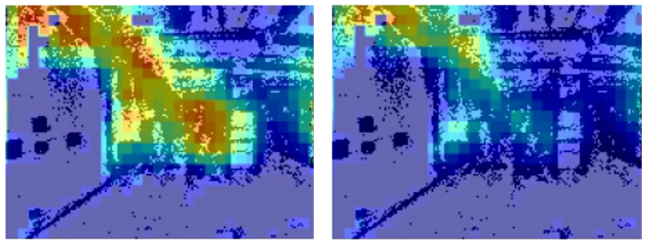

Figure 6.8: *Cluttered scene with no humans. a)* $\lambda_u = 0$ *(negative STDP not used) and* $\lambda_s = 1$*, b)* $\lambda_u = 0.2$ *(negative STDP used) and* $\lambda_s = 0.8$*.*

6.3.6 Continual SNN-STDP vs. CNN with the same memory size

Since CNNs constitute the *standard approach* used in the vision pipelines of drones [145], we compare in Fig. 6.9 the performance of our SNN-STDP architecture to a SqueezeNet-based CNN [100], trained offline with DVS *frames* following Appendix B (DVS data averaged over time into frames at 30 FPS).

Using a SqueezeNet CNN architecture is an appropriate choice for comparison since SqueezeNet aims to be MCU-friendly, using $\sim$ 500KB of weight memory (comparable to our network that uses 514KB) [100].

In order to fairly compare the bounding box output of the CNN to the attention map of our SNN-STDP system, we consider the image region delimited by the bounding box as the active region of an attention map (region inside the bounding box is filled with 1 and outside is filled with 0). We can then use the same precision-recall measurement pipeline as used for the SNN. Interestingly, the CNN-DVS setup is the lowest performer and is outperformed by our SNN-STDP system (+19% on the peak F_1 score). This is mainly due to the fact that the SNN introduces *time recurrence* through its lateral weights, enabling the network to learn *temporal features* that are neglected by the feed-forward CNN-DVS (see [128] for a discussion on temporal features in DVS data).

We now summarize the conclusions from these experiments:

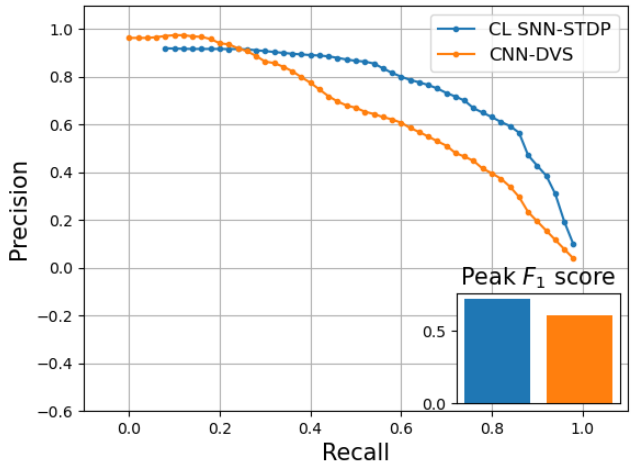

Figure 6.9: *Detection precision vs. recall for the presented SNN-STDP and the conventional CNN with offline training. Our proposed method outperforms the CNN by more than* 19% *on the peak* F_1 *score.*

Even though learning is carried out in a continual manner, the proposed SNN-STDP outperforms by +19% on the peak F_1 *score compared to a SqueezeNet CNN trained offline, with comparable memory footprint as our proposed SNN-STDP system.*

6.4 Conclusions

This chapter has presented a novel study of the *continual development* of attention-based perception in bio-inspired SNN-STDP systems. After the introduction of our semi-supervised SNN-STDP framework, a new interpretation of anti-Hebbian plasticity has been provided, as a bio-plausible mechanism for controlling over-fitting on untargeted background data. Then, numerous experiments using DVS camera data captured from a drone have been conducted to study the applicability of our framework and theoretical derivations to learning attention-based detection from *non-i.i.d* data sequences as found in nature. We have also shown that our bio-inspired system reaches a higher peak F_1 score (+19%) compared to a same-size CNN processing DVS frames. Therefore, to the best of our knowledge, the methods and demonstrations shown in this chapter have provided a first successful account of the capability and usefulness

of SNN-STDP networks for continually learning to detect people on the fly from a DVS camera mounted on a drone. As per future work, we expect the methods provided in this chapter to be applicable to the detection of other types of objects and from different types of robot platforms, under the condition that the background generates a sparser amount of DVS event data compared to the objects to be detected. In the next chapter, we will investigate how continual-learning Hebbian networks can be used for the control of dynamical agents, in order to further extend the application domain of SNN-STDP systems, based on the joint Dictionary Learning and Basis Pursuit (DLBP) theory (discussed in chapter 4).

Chapter 7

Active inference in Hebbian learning networks

7.1 Introduction

The previous chapter investigated how the DLBP-based SNN-STDP framework introduced in chapter 4 can be used for performing people detection from a DVS camera mounted on a drone. In this chapter, we will investigate how Hebbian learning networks (such as SNN-STDP systems) can be utilized in reinforcement learning settings for the control of dynamical agents. In particular, we will uncover ways for unifying joint Dictionary Learning and Basis Pursuit (DLBP) with the theory of Active Inference (AIF) [187] found in neuroscience, which aims at explaining how intelligent (biological) agents can learn and perform actions at the same time.

The study of DLBP [177], [60], [132], [214] and Predictive Coding [73], [72], [277] networks has gained much attention for understanding the mechanisms underlying learning and inference in the brain [176]. In particular, it has been shown that the learning of the *weight dictionary* used to project the input signals into sparse codes can be conducted via the biologically-plausible

Adapted from: *A. Safa et al. (2024). Active Inference in Hebbian Learning Networks. In: Buckley, C.L., et al. Active Inference. IWAI 2023. Communications in Computer and Information Science, vol 1915. Springer, Cham.*

Hebbian learning mechanism [209], with experimental evidence behind this mechanism observed in the brain [23], [195]. Hebbian learning differs from the widely-used back-propagation of error (backprop) technique due to its *local* nature [277], [210], [172], where the weight w_j of neuron i is modified via a combination f of the weight's input x_j and the neuron's output y_i (with η_d the learning rate parameter):

$$w_j \leftarrow w_j + \eta_d f(y_i, x_j) \tag{7.1}$$

When applied to layers that evince some form of competition between their neurons, the Hebbian mechanism in (7.1) leads to the *unsupervised* learning of complementary features from the input signals [124].

At the same time, Active Inference (AIF) has gained huge interest as a *first-principle* theory, explaining how biological agents evolve and perform actions in their environment [187], [103]. In recent years, the use of deep neural networks (DNNs) for parameterizing generative models has gained much attention in AIF research [284], [244], [64]. Deep AIF systems are typically composed of a *posterior* network $q_{\Phi_P}(s_l|o_{l-1}, a_{l-1})$, inferring the latent state s_l given an incoming observation-action pair $\{o_{l-1}, a_{l-1}\}$, and a *state-transition* network $p_{\Phi_S}(s_l|s_{l-1}, a_{l-1})$, predicting the next latent state s_l given the current state-action pair $\{s_{l-1}, a_{l-1}\}$ [284]. The state-transition network is used to generate the agent's roll-outs for different policies in order to compute the Expected Free Energy associated to each policy [284]. Finally, a *likelihood* network $p_{\Phi_L}(o_l|s_l)$ reconstructing the input observation o_l from the latent state s_l can also optionally be implemented [245] (not utilized in this work). Each network parameterizes its respective density function through weight tensors Φ_P, Φ_S and Φ_L.

In this chapter, we aim to study how AIF can be performed in Hebbian learning networks *without resorting to backprop* (as typically used in deep AIF systems). Experiments conducted in the OpenAI Mountain Car environment [29] show that the proposed Hebbian AIF approach outperforms the use of Q-learning and compares favorably to the backprop-trained Deep AIF system of [284], while *not requiring* any replay buffer, as in typical reinforcement learning systems [234]. Our derivations and experiments add to a growing number of works addressing the study of Hebbian Active Inference [180], [181].

This chapter is organized as follows. Background theory about Hebbian learning networks will be provided in section 7.2. Our Hebbian AIF methods will be covered in section 7.3. Experimental results will be shown in section 7.4. Conclusions will be provided in section 5.7.

7.2 Background theory on Hebbian learning networks

Inspired by previous works that model the neural activity of biological agents through DLBP [73], [209] (such as in the mushroom body of an insect's brain [134]), we model each individual Hebbian Ensemble layer of our networks as an identically-distributed Gaussian likelihood model with a Laplacian prior on the neural activity c:

$$p(c|o,\Phi) \sim \exp\left(-||\Phi c - o||_2^2\right) \exp\left(-\lambda||c||_1\right) \tag{7.2}$$

where o is the input of dimension N, c is the output of dimension M, Φ is the $N \times M$ weight matrix of the layer (also called *dictionary*), and λ is a hyper-parameter setting the scale of the Laplacian prior. Choosing a Laplacian prior is motivated by the fact that it promotes sparsity in the output neural code in a way similar to how sparsity is induced in networks of Spiking Leaky Integrate-and-Fire neurons, modelling cortical neural activity [209].

Under DLBP (7.2), inference of c and learning of Φ is carried out via [209]:

$$C, \Phi = \arg\min_{C,\Phi} \sum_l ||\Phi c_l - o_l||_2^2 + \lambda||c_l||_1 \textbf{ with } C = \{c_l, \forall l\} \tag{7.3}$$

which can be solved via Proximal Stochastic Gradient Descent [4], by *alternating* between: *a)* the inference of c_l, given the current input o_l and the weight Φ, and *b)* the learning of Φ, given the current c_l and o_l.

Hence, we instantiate Hebbian layers as the dynamical system given in (7.4), where T denotes the transpose, η_c is the coding rate, η_d is the learning rate and $\textbf{Prox}_{\lambda||.||_1}$ is the proximal operator to the l_1 norm (non-linearity) [2]. For each input o_i, the neural and weight dynamics of the Hebbian network follows the update rules in (7.4) for an arbitrary number of iterations (set to 100 in this work, as a good balance between the speed of convergence and the convergence quality), in order to infer the corresponding c_l and learn Φ [209].

$$\begin{cases} c_l \leftarrow \textbf{Prox}_{\lambda||.||_1}\{c_l - \eta_c \Phi^T(\Phi c_l - o_l)\} \\ \Phi \leftarrow \Phi - \eta_d(\Phi c_l - o_l)c_l^T \end{cases} \tag{7.4}$$

with $\textbf{Prox}_{\lambda||.||_1}$ acting as the neural non-linearity:

$$\textbf{Prox}_{\lambda||.||_1}(c_i) = \text{sign}(c_i)\max(0, |c_i| - \eta_c\lambda), \forall i \tag{7.5}$$

From a neural point of view, the dynamical system of (7.4) can be implemented as the network architecture in Fig. 7.1, where all weight updates follow the standard Hebbian rule (7.1) [209].

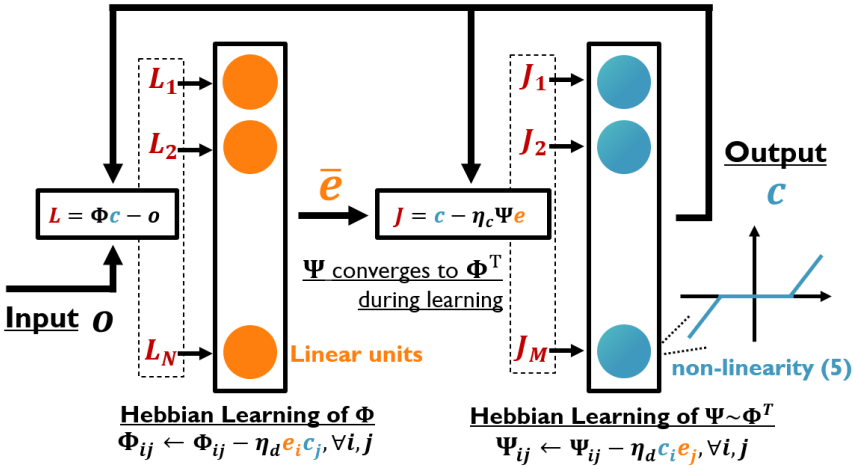

Figure 7.1: *Baseline Hebbian network architecture used in this work. The dynamics of the network follow (7.4) and minimize (7.3), given subsequent input vectors o. Each layer possesses its own weight matrix* Φ, Ψ *which evolve through Hebbian plasticity (*$\Psi \sim \Phi^T$ *in (7.4), as an independent, local set of weights).*

7.3 Active inference in Hebbian learning networks

In this section, we show how the Hebbian network described above in section 7.2 is utilized in order to build an Active Inference (AIF) system. First, we describe how Variational Free Energy minimization can be performed by a cascade of two Hebbian networks: a *state-transition* network predicting the next latent states given the previous ones, and a *posterior* network providing latent states given the input observations. Crucially, it will be shown that Free Energy minimization necessitates top-down Hebbian learning connections from the *state-transition* network towards the *posterior* network, steering the posterior output activity towards the state-transition output during learning. Then, we will show how the Expected Free Energy is computed by generating state transition roll-outs. In summary, the ensuing scheme showcased below can be regarded as learning to plan; in which the requisite inferences are amortised by Hebbian learning. Note that this learning is effectively a local scheme that eschews need for back propagation.

7.3.1 Minimizing the Variational Free Energy

The Variational Free Energy can be decomposed as [284] [71] (where $\mathbb{E}$ denotes the expected value):

$$\mathcal{F} = \underbrace{D_{KL}[q_{\Phi_P}(s_l|o_{l-1}, a_{l-1})||p_{\Phi_S}(s_l|s_{l-1}, a_{l-1})]}_{\text{expected complexity (ambiguity)}} - \underbrace{\mathbb{E}_q[\log(p_{\Phi_L}(o_l|s_l))]}_{\text{expected accuracy (risk)}} \quad (7.6)$$

with the parametrized densities $q_{\Phi_P}, p_{\Phi_S}, p_{\Phi_L}$ described above in section 7.1. Since the Hebbian network architecture used here intrinsically provides a means to reconstruct its input x_l in (7.3) (i.e., *likelihood* modelling) from its produced latent code c_l in (7.3) (i.e., *posterior* modelling), using the *same dictionary* parameter matrix Φ in (7.3) that was used to generate c_l via (7.4), we have $\Phi_L = \Phi_P$ in (7.6) and we will solely use Φ_P below to denote the *posterior* weight matrix.

Under the assumption of Gaussian likelihood with identity covariance in (7.2), the KL divergence D_{KL} in $\mathcal{F}$ can be simplified to [89]:

$$\mathcal{F} \sim ||\Phi_S c_{S,l} - \{s_l(\Phi_P), a_l\}||_2^2 + ||\Phi_P s_l - \{o_l, a_l\}||_2^2 \quad (7.7)$$

where $s_l(\Phi_P)$ explicits the dependency of s_l on Φ_P (s_l being the posterior network output activity), and c_S denotes the output activity of the state-transition network given s_l. The Free Energy in (7.7) must be minimized with regard to the state transition weights Φ_S and the posterior weights Φ_P during learning:

$$\Phi_S, \Phi_P = \arg\min_{\Phi_S, \Phi_P} ||\Phi_S c_{S,l} - \{s_l(\Phi_P), a_l\}||_2^2 + ||\Phi_P s_l - \{o_l, a_l\}||_2^2, \forall l \quad (7.8)$$

This indicates that it is not only the state transition model that must be steered towards the posterior model, but also, the posterior model must be steered towards the output of the state-transition network. This effect can be achieved by re-formulating the optimization in (7.8) as:

$$\begin{cases} \Phi_S = \arg\min_{\Phi_S} ||\Phi_S c_{S,l} - \{s_l, a_l\}||_2^2, \forall l & \text{(a)} \\ \Phi_P = \arg\min_{\Phi_P} ||\Phi_P s_l - \{o_l, a_l\}||_2^2 + ||\Phi_P(\Phi_S c_{S,l}) - \{o_l, a_l\}||_2^2 & \text{(b)} \end{cases} \quad (7.9)$$

Intuitively, the right-hand term in (7.9 b) steers the *posterior* model towards the *state-transition* model by first re-projecting the output activity of the state-transition network c_S into the latent space as $\Phi_S c_S$ (considering Φ_S fixed). Then, minimizing $||\Phi_P(\Phi_S c_{S,l}) - \{o_l, a_l\}||_2^2$ modifies Φ_P in order to steer its posterior output s_l towards the re-projected state-transition activity $\Phi_S c_S$ (considering $\{o_l, a_l\}$ fixed).

State-transition model

Inspired by prior work on dictionary-based sequence modeling [115], we implement the transition model $p_{\Phi_S}(s_l|\tilde{s}_{l-1}, \tilde{a}_{l-1})$ as an *auto-regressive* Hebbian network (see Fig. 7.2 a), taking as input a sequence of state-and-action *history* $\tilde{s}_{l-1} = [s_{l-1}, \ldots, s_{l-L_{buf}}]$, $\tilde{a}_{l-1} = [a_{l-1}, \ldots, a_{l-L_{buf}}]$, and inferring the next state $\tilde{s}_l = [s_l, \ldots, s_{l-L_{buf}}]$ as the re-projection of its internal sparse code c_S in the input space through the network weights Φ_S:

$$\tilde{s}_l = \Phi_S c_{S,l} \tag{7.10}$$

where $\Phi_{S,j}$ and $c_{S,j}$ respectively denote the weight vector and the sparse code of each layer j in the state transition network. Therefore, the state-transition network effectively projects the L_{buf} previous states (noted $\tilde{s}_{l-1}$) into a common internal sparse code c_S and reconstructs the next states $\tilde{s}_l$ by re-projection of c_S into the input space.

The state-transition network learns its weights Φ_S following (7.9 a) and infers its output activity c_S via sparse coding (see section 7.2):

$$\Phi_S, c_{S,l} = \arg\min_{\Phi_S, c_{S,l}} ||\Phi_S c_{S,l} - \tilde{s}_l||_2^2 + \lambda_P ||c_{S,l}||_1, \forall l \tag{7.11}$$

where λ_P is a parameter that sets the strength of the *sparsity* of the *state-transition* output activity. (7.11) can therefore be implemented via the DLBP-based Hebbian learning ensemble described in section 7.2. This auto-regressive strategy enables the network to learn state predictions using Hebbian learning, *without* the need for non-bio-plausible back-propagation through time (BPTT) [115], [263].

In order to prevent the vanishing or exploding of the state transition model when producing roll-outs further in time, we regularize the norm of the reconstructed states s_l to an arbitrary magnitude α using (7.12). We keep $\alpha = 5$ in our experiments in section 7.4, giving a good balance for the dynamic range of the network output activity (adjusted empirically).

$$s_l = \alpha \frac{s_l}{||s_l||_2} \tag{7.12}$$

Posterior model

Similar to the state-transition model, we use a Hebbian ensemble as posterior model, where the internal sparse code c_P (see Fig. 7.2 b) is identified as the

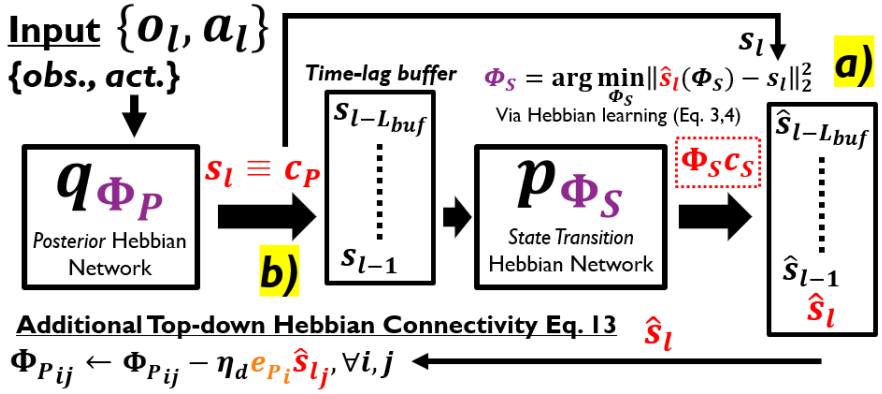

Figure 7.2: *Hebbian Active Inference Architecture. a) The state-transition network takes as input the L_{buf} previous latent states produced by the posterior network, and projects them onto its internal representation c_S via the learned Φ_S. When producing roll-outs, the state-transition network estimates the next state $\hat{s}_l$ by re-projecting the output activity c_S due to $[s_{l-L_{buf}}, \ldots, s_{l-1}]$ back onto the input space via (7.10). b) The posterior network takes observation-action pairs as input and produces latent states s (corresponding to the output in Fig. 7.1). In addition to the Hebbian mechanisms depicted in Fig. 7.1, the weights Φ_P of the posterior network are also subject to a top-down Hebbian learning mechanism for minimizing the second term in (7.9 b).*

hidden state $s_l \equiv c_{P,l}$ inferred by the posterior network $q_\nu(s_l|o_{l-1}, a_{l-1})$, given the observation and action pair $\{o_{l-1}, a_{l-1}\}$ in (7.2).

Therefore, the posterior network learns its weights Φ_P following (7.9 b) and infers its output activity $s_l = c_{P,l}$ via sparse coding (see section 7.2):

$$\Phi_P, c_{P,l} = \arg\min_{\Phi_P, c_{P,l}} \underbrace{||\Phi_P c_{P,l} - \{o_l, a_l\}||_2^2 + \lambda_Q ||c_{P,l}||_1}_{\text{Standard Sparse Coding}} + \overbrace{||\Phi_P(\Phi_S c_{S,l}) - \{o_l, a_l\}||_2^2}^{\textbf{Top-down Connection}} \tag{7.13}$$

for all l, where λ_Q sets the strength of the *sparsity* of the *posterior* output activity. The left-hand *standard sparse coding* term in (7.13) can be implemented via the Hebbian learning ensemble described in section 7.2, while the right-hand term in (7.13) can be implemented using *top-down* connections from the state-transition output activity c_S towards the posterior network, via the state-transition weight matrix Φ_S (see Fig. 7.2 b).

Finally, here again, we apply the regularization rule (7.12) to the inferred posterior state, effectively constraining s_l to lie on the α-sphere manifold.

7.3.2 Minimizing the Expected Free Energy

Given a policy π, the Expected Free Energy $G(\pi)$ (EFE) can be written as [220]:

$$G(\pi) = \sum_l \mathbb{E}_{q(o_l, s_l|\pi)}[\log q(s_l|\pi) - \log p(s_l, o_l|\pi)]$$

$$= \sum_l -H\{q(s_l|\pi)\} - \mathbb{E}_{q(o_l, s_l|\pi)}[\log p(s_l, o_l|\pi)] \quad (7.14)$$

where H denotes the Shannon entropy. It can be seen in (7.14) that selecting a policy that minimizes the EFE entails the maximization of the posterior entropy (promoting exploration) and the joint posterior over the states and observations (reaching the desired goal) [220].

To reach the desired goal, we produce roll-outs of states s_l given a certain policy and approximate the risk term $-\mathbb{E}_q[\log p(s_l, o_l|\pi)]$ to be minimized as:

$$-\mathbb{E}_{q(o_l, s_l|\pi)}[\log p(s_l, o_l|\pi)] \sim ||s_l - s^*||_2^2 \quad (7.15)$$

where s^* is the desired state that the agent must reach, corresponding to a desired observation (e.g., the agent's position). Since the observation o_l can encompass more than just the goal to be reached (i.e., the observation o_l could be both the position and the velocity of an agent, even though the desired goal is to reach a specific position regardless of the velocity), we compute the *goal* state s^* as:

$$s^* = \arg\max_s \int_{\omega \in D_\omega} q(s|\Omega^*, \omega) d\omega \quad (7.16)$$

where Ω^* contains all observations that must be reached in order to attain the desired goal, and ω designates all observation modalities that are *not* taking part in defining the goal that must be reached, with D_ω their domain of definition. In practice, (7.16) is estimated by averaging the output of the posterior network, while sweeping ω for a grid of possible values and keeping Ω^* fixed.

Regarding the exploration term in (7.14), our Hebbian network does not directly allow the estimation of the entropy $H\{q(o_l, s_l|\pi)\}$, since the network does not infer standard deviations as in a variational auto-encoder (VAE) [284]. We propose to replace the maximization of the entropy $H\{q(o_l, s_l|\pi)\}$ with a surrogate term, crafted to promote exploration as well. As a surrogate for

$H\{q(o_l, s_l|\pi)\}$, we choose to maximize the variance (noted Var) of the state trajectory $s_l, \forall l = 1, ..., L$ along time during the roll-outs. Intuitively, a state trajectory that presents lots of variation in time will promote the exploration of new states, providing a similar qualitative effect as maximizing $H\{q(o_l, s_l|\pi)\}$. Therefore, we select the policy π such that the distance to the desired state is minimized, while achieving a state trajectory variance larger than a certain threshold t_v:

$$\pi^* = \arg\min_{\pi} \mathcal{G}(\pi) = \sum_{l=1}^{L} ||s_l - s^*||_2^2 \quad \text{s.t.} \quad \text{Var}(||s_l - s^*||_2^2, l = 1, ..., L) \geq t_v \tag{7.17}$$

Given a set of N_p policies to try, t_v can be determined in an adaptive way as follows, such that the divergence from the desired state is minimized, while ensuring the variance of counterfactual state trajectories exceeds a certain threshold:

$$t_v = \beta \times \frac{1}{2}[\mathbf{max}_{\boldsymbol{\pi}}(\text{Var}(||s_l(\pi) - s^*||_2^2, \forall l)) + \mathbf{min}_{\boldsymbol{\pi}}(\text{Var}(||s_l(\pi) - s^*||_2^2, \forall l))] \tag{7.18}$$

where β is the strength hyper-parameter (empirically set to 0.5 in our experiments reported below). β acts as the precision or inverse temperature parameter associated with prior preferences (i.e., the precision of the prediction error between predicted and desired states). It must be noted that this approach might have some limitations since it could promote a presence of noisy perturbation during state transition (as the variance in (7.17) represents the entropy of the *environment* vs. the entropy of the *agent's brain* in (7.14))

7.4 Experimental results

The aim of our experimental studies is to determine *i)* how the main network hyper-parameters (number of neurons, sparsity in output activity, etc.) impact the success rate of the proposed Hebbian AIF system; *ii)* to what extent Hebbian AIF is robust when learning without using a replay buffer, and *iii)* how Hebbian AIF compares to Q-learning (which uses *dense rewards* versus *unsupervised learning* in Hebbian AIF).

7.4.1 Mountain car environment

We perform experiments in the *Mountain Car* environment from the OpenAI gym suite [29]. In this task, a car starts at a *random* position at the bottom

of a hill and is expected to reach the top of a mountain within 200 time steps. The agent is subject to gravity and cannot reach the goal trivially, just by accelerating towards it. Rather, the agent must learn to gain momentum before accelerating towards the goal.

In this environment, the x-axis position x and the velocity v_x of the car constitute the input observations to the Hebbian AIF network. Before feeding the observation tuple (x, v_x) to our Hebbian network, we normalize (x, v_x) using (7.19) in order to equalize the dynamic range of the position and velocity signals:

$$\begin{cases} x \leftarrow \frac{x-\mu_x}{\sigma_x} \\ v_x \leftarrow \frac{v_x-\mu_{v_x}}{\sigma_{v_x}} \end{cases} \tag{7.19}$$

where (μ_x, σ_x) and $(\mu_{v_x}, \sigma_{v_x})$ denote the mean and standard deviation of the position and velocity signals, respectively (estimated during random environment runs).

We use an action space constituted by two discrete actions: *accelerate to the left* and *accelerate to the right*. In addition, each action is repeated for 10 consecutive time steps once selected during the Expected Free Energy minimization in (7.17).

In order to compute the Expected Free Energy, we generate roll-outs of $L = 200$ time step predictions for 100 different random policies $\pi^j, j = 1, ...100$ with equal probability of selecting the *accelerate to the left* or the *accelerate to the right* actions.

As learning rate for the Hebbian learning mechanism (7.4), we use $\eta_d = 10^{-4}$ with a *decay rate* of 0.8 applied at the end of each *successful* episode, i.e. if the episode terminates successfully, $\eta_d \leftarrow \eta_d \times 0.8$ (else no decay is applied on η_d). All weights are initialized randomly from a normal distribution with standard deviation 0.01.

In the remainder of this section, we perform all our experiments using a 10-fold validation approach, by reporting the success rate curves as averages over 10 different runs (with 35 episodes per runs), with different random network initializations. For each run, we compute the success rate curve using a moving average window of size 5, and report the mean success rate curve by averaging over the 10 runs, alongside with its standard deviation (see e.g. Fig. 7.3). We will now study the impact of the various network hyper-parameters on the achieved success rates, so as to give a *complete account* of their effect during model tuning.

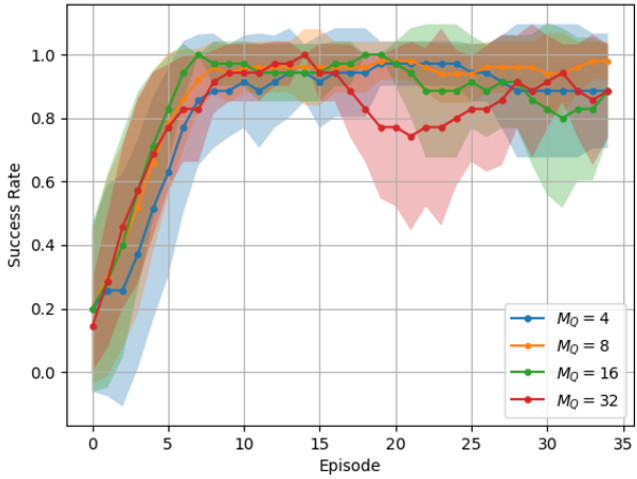

Figure 7.3: *Impact on the success rate when changing the number of neurons M_Q in the posterior network.*

7.4.2 Impact of the number of neurons in the posterior and state transition networks

Fig. 7.3 and 7.4 show the effect of sweeping the number of coding neurons M_Q and M_P in both the *posterior* and *state-transition* networks. Fig. 7.3 shows that for $M_Q < 8$, the success rate is sub-optimal, but reaches a steady plateau around $M_Q = 8$ (orange curve in Fig. 7.3). Then, as M_Q is increased for $M_Q > 8$, the success rate becomes sub-optimal again, with dips in the performance along the episodes (e.g., red curve in Fig. 7.3). This phenomenon can be explained as follows: for $M_Q < 8$, the posterior network does not have enough parameters to capture the input dynamics into its latent space and *under-fits*, while for $M_Q > 8$, the posterior network starts over-fitting, reducing the success rate again[1]. Regarding the *state-transition* network, Fig. 7.4 shows that the higher the number of neurons M_P, the flatter the success rate curves become, leading

[1]Note that we are using our Hebbian scheme to amortize variational inference. This means we are optimizing amortization (encoding) parameters, not the (decoding) parameters of the generative model. An alternative approach would be to treat the model parameters as random variables and derive the update rules for minimizing variational free energy, in the form of a Hebbian update. In this instance, over-fitting would be precluded because of the complexity term in (7.6). However, this would not be amortization; this would be an implementation of AIF as described in [74].

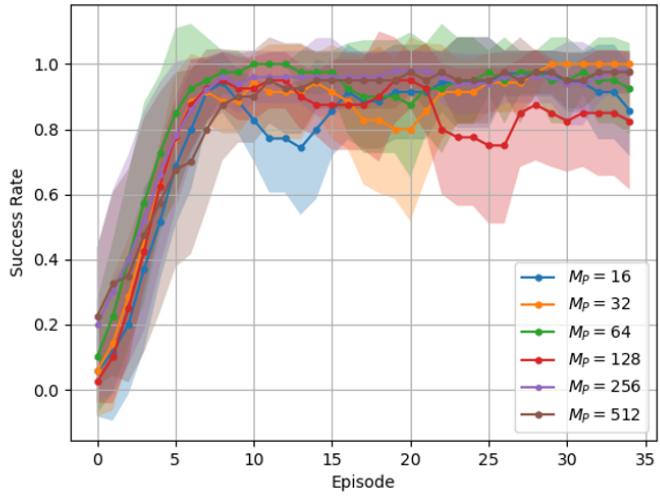

Figure 7.4: *Impact on the success rate when changing the number of neurons M_P in the state transition network.*

to higher performance. The state transition network does not seem to over-fit as M_P is increased (for $\lambda_P = 10^{-4}$ kept fixed). Rather, Fig. 7.4 indicates that a higher state-transition network capacity is beneficial for capturing important dynamics in the latent space, at the output of the posterior network.

7.4.3 Impact of the sparsity of the output activity in the posterior and state transition networks

Fig. 7.5 and 7.6 show the effect of sweeping the *sparsity-defining* hyper-parameters λ_Q and λ_P in both the *posterior* and *state-transition* networks. For the posterior network, Fig. 7.5 shows that the success rate performance initially grows as λ_Q is increased from $\lambda_Q = 10^{-6}$ to $\lambda_Q = 10^{-5}$. Doing so, the non-linearity of the posterior network is increased, better capturing observation features into its latent space. Then, as λ_Q grows past $\lambda_Q = 10^{-4}$, the success rate degrades again, indicating a too strong posterior network non-linearity.

Regarding the *state-transition* network, Fig. 7.6 a) shows that the lower λ_P, the higher the success rate becomes. This suggests that making the state-transition

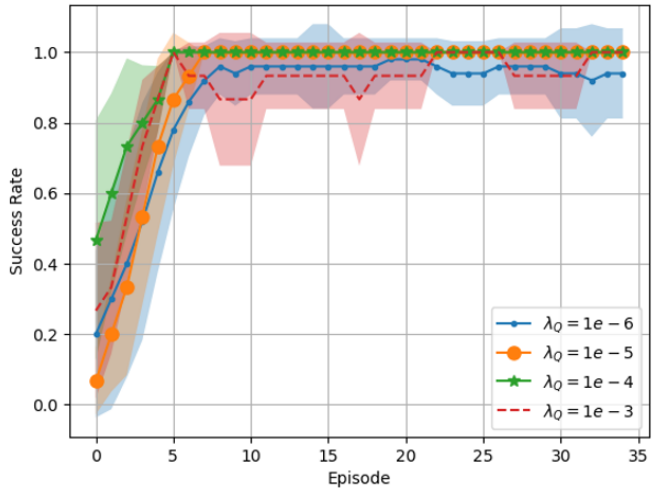

Figure 7.5: *Impact on the success rate when changing the sparsity hyper-parameter λ_Q in the posterior network.*

network *more linear* (i.e., lower λ_P) better captures the dynamics of the latent space produced by the posterior network (other parameters kept fixed).

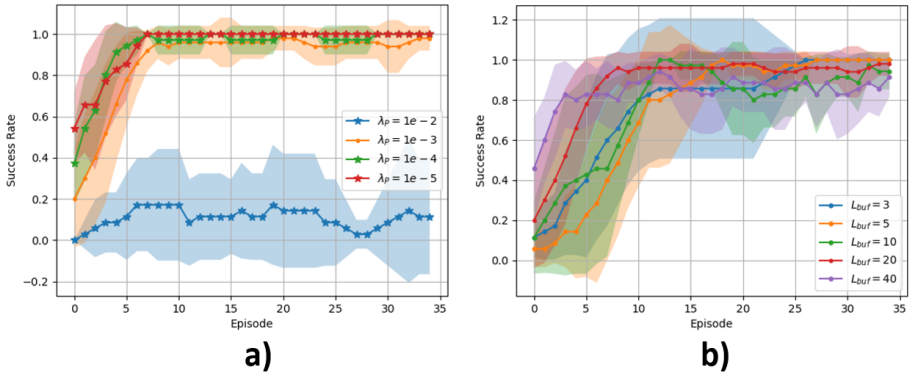

Figure 7.6: *a) Impact on the success rate when changing the sparsity λ_P in the state transition network. b) Success rate when changing the time-lag buffer length L_{buf}.*

7.4.4 Impact of the time-lag buffer length on task performance

Fig. 7.6 b) shows how the length L_{buf} of the time-lag buffer impacts the achieved success rate. Initially, as L_{buf} increases, the success rate increases as well, due to an increased availability of past latent states used by the state-transition network to estimate the next expected state. Then, as L_{buf} is further increased for $L_{buf} > 20$, the success rate drops again due to the addition of latent states *from deep in the past* that are less useful for estimating the present dynamics.

7.4.5 Comparing Hebbian active inference against the use of a replay buffer and against Q-learning

Fig. 7.7 compares the success rate obtained using the presented Hebbian AIF system against *a)* the use of a replay buffer during learning and *b)* the use of a Q-learning agent. Experience replay is done by saving the history of observation-action pairs in a buffer after each episode. After the end of the episode, a past experience is randomly selected and used to train the Hebbian AIF system for one episode.

Regarding the Q-learning setup, we use a standard Q-table learning approach [234], with an open python implementation[1].

Fig. 7.7 shows that our Hebbian AIF system converges much faster than the Q-learning system and behaves in a comparable manner to the Hebbian AIF setup with a *replay buffer*. Indeed, the Q-learning agent needs two orders of magnitude more training episodes in order to converge, despite the fact that it utilizes the *dense rewards* provided by the Mountain Car environment [29] (vs. *unsupervised* learning for Hebbian AIF)[2]. This confirms prior observations about the efficient convergence of AIF systems, due to their ability to learn a generative model of environment dynamics used to select actions during learning (vs. supervised learning of a Q-table) [284].

Finally, it is interesting to note that, compared to the Deep AIF results *reported in* [284] (using a fully-connected 2-hidden-layer network trained through backprop), the Hebbian AIF system proposed in this work eventually reaches

[1] `https://gist.github.com/gkhayes/3d154e0505e31d6367be22ed3da2e955`

[2] Although we have referred to the AIF scheme as unsupervised, there is an implicit constraint on behavior that is implemented, in this instance, by the goal states in (7.16). In AIF, goal-directed behavior emerges from inferring the right courses of action that lead to preferred outcomes. In amortized AIF, this planning as inference is learned; as we have demonstrated. In contrast, reinforcement learning ignores inference and simply learns rewarded behaviors, which can take a very long time — because there is no learning of a generative model, or the constraints that it affords.

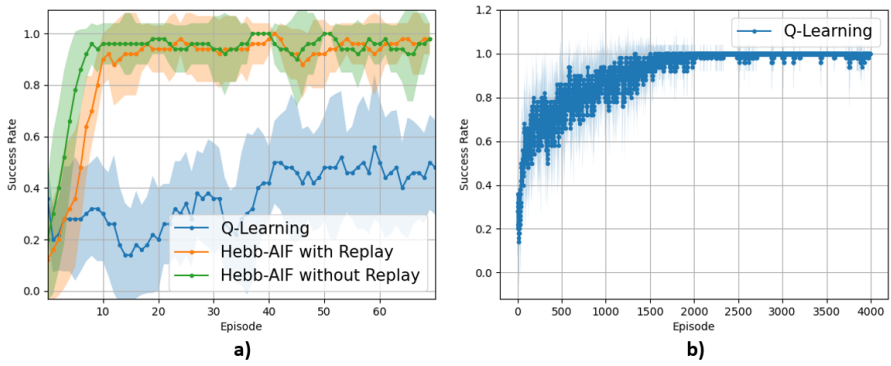

Figure 7.7: *Hebbian AIF versus the use of a replay buffer and Q-learning (a). Q-learning needs two orders of magnitude more episodes in order to converge (b).*

$\sim 100\%$ success rate (see the red curve in Fig. 7.6 a), while the system in [284] reaches $\sim 95\%$, motivating further investigations of Hebbian learning for AIF systems.

We now summarize the conclusions from these experiments:

The demonstrations provided in this chapter show that the proposed continual-learning Hebbian Active Inference (AIF) system successfully solves the mountain car problem while not requiring any replay buffer, in contrast to previously-proposed DNN solutions. In addition, the proposed Hebbian AIF system outperforms the use of Q-learning by a more than two orders of magnitude faster convergence rate.

7.5 Conclusions

This chapter has investigated how neural ensembles equipped with local Hebbian plasticity can perform active inference for the control of dynamical agents. First, a Hebbian network architecture performing joint Dictionary Learning and Basis Pursuit (DLBP) has been introduced for implementing both the posterior and the state-transition models forming our generative Active Inference (AIF) system. Then, it has been shown how Free Energy minimization can be performed by the proposed Hebbian AIF system. Finally, experiments for parameter exploration and benchmarking have been performed to study the impact of the network parameters on the task performance. Experimental results on the example mountain car environment have shown that the proposed system outperforms

the use of Q-learning, while not requiring the use of a replay buffer during learning, motivating future investigations of using Hebbian learning for designing active inference systems. As per future work, it will be of interest to extend the *non-spiking* DLBP-based AIF framework developed in this chapter towards the use of SNNs with STDP learning, by using the DLBP-based SNN-STDP framework developed earlier in chapter 4. In addition, it will be interesting to investigate how the Hebbian AIF framework presented in this chapter can be applied successfully to a range of other agent control tasks, beyond the mountain car environment used in this chapter.

Chapter 8

Conclusions and future work

The general aim of this thesis has been to significantly extend the current domain of application of neuromorphic learning techniques (SNNs and STDP) towards novel sensory use cases and emerging paradigms (continual STDP learning for drone SLAM using DVS-radar sensors, radar-SNN gesture recognition, etc). To do so, this thesis has provided extensive studies on how to efficiently encode radar signals into spike trains so as to match SNN performance to DNNs. This thesis has also integrated a drone platform with radar and DVS camera sensing capability, and used it to collect various datasets for SLAM and people detection in a warehouse environment. Crucially, this thesis has provided a novel top-down optimization-based theory of SNN-STDP learning and has demonstrated the proposed SNN-STDP framework for SLAM and people detection from drones, using a continual learning approach. The various experimental results and large-scale, real-world demonstrations on tasks never tackled before via SNN-STDP have clearly shown the usefulness of our methods for bridging the gap between SNNs and conventional methods such as DNNs in terms of accuracy and task capability.

8.1 Main contributions

The main contributions realized in this thesis have enabled us to address the open questions first listed in section 1.4 of the Introduction in chapter 1.

Question 1 (Q1): What radar pre-processing and event encoding steps are best suited for SNN-based radar signal recognition? Chapter

2 has presented what is, to the best of our knowledge, the most detailed analysis on the use of SNNs for radar gesture signal classification, and has extensively compared the behavior of the proposed SNN architecture against an equivalent CNN topology. It has been shown that resource-constrained SNNs using only 4-bit precision weights can reach the accuracy of conventional same-class CNNs (93% in our experiments) that use full-bit weights and ReLU activations, under the important condition that a carefully-designed data pre-processing and spike encoding queue is being used prior to feeding the sensory data into the SNN. Indeed, in the absence of a well-conceived pre-processing, the SNN accuracy was observed to be much lower: as low as 36% in the μDoppler-only case. After comparing more than 10 different radar pre-processing and event-encoding approaches (range-Doppler, μDoppler and their variants), it has been found that the impact of pre-processing on the SNN and CNN accuracy is not identical and even antagonistic in some cases. Therefore, conclusions drawn for radar processing using CNNs cannot be generalized to the SNN case, which required the specific pre-processing steps elaborated in chapter 2 in order to reach competitive accuracy with the SNN for the gesture recognition use case.

Question 2 (Q2): How can we derive a top-down optimization-based theory of SNN-STDP learning? Previously proposed SNN-STDP works mainly focus on bottom-up network design using random Liquid State Machine-like topologies tuned heuristically. Compared to SNNs with backprop and DNNs, these previous works often reached a lower accuracy and were usually applied on simpler tasks. Crucially, it has been argued in chapter 4, that in order to enhance SNN-STDP performance, there is a need for formulating a top-down, optimization-based theory for SNN-STDP learning. Chapter 4 has presented the design of an SNN-STDP architecture that provably solves the unsupervised dictionary learning and basis pursuit (DLBP) problem via its neural and synaptic dynamics. Using common benchmark datasets, it has been shown that this top-down SNN-STDP framework results in significant boosts of +9.3% in accuracy on *neuromorphic*-MNIST (N-MNIST) and +7.7% on IBM DVS128 gesture compared to previously proposed STDP-based works. Therefore, these large improvements in accuracy obtained compared to previous STDP works have supported our claim for the need of a top-down approach when designing SNN-STDP systems.

Question 3 (Q3): How can SNN-STDP be used for performing useful drone navigation tasks such as Simultaneous Localization and Mapping (SLAM)? In order to significantly extend the domain of application of STDP-based systems, usually limited in prior works to pattern recognition problems, this thesis has applied the top-down SNN-STDP theory formulated in chapter 4 to the challenging case of performing real-world drone tasks (vs. using simulation data). In addition, since SNN-STDP systems

have been identified as a well-suited technique for on-device learning and personalization, this thesis has studied the application of SNN-STDP to drones under a continual-learning setting. Chapter 5 has presented a first-of-its-kind Simultaneous Localization and Mapping (SLAM) system using continual SNN-STDP learning to fuse DVS camera and radar data. The SNN-STDP network is initialized randomly and has been deployed in a warehouse environment without any pre-training. Then, as the drone navigates, the SNN-STDP network continually learns to extract features from the sensory data on the fly and provides the inferred feature descriptors to a RatSLAM back-end for loop closure detection and map correction. Experimental results on various flights have shown that the proposed system either outperforms or reaches a similar precision compared to conventional SLAM setups using CNN encoders. Finally, chapter 5 has provided a comparison between the continual-learning SNN-STDP setup proposed and a pre-trained SNN-STDP used in inference-only mode. It has been demonstrated that the pre-trained model performs significantly worse than the SNN-STDP with continual personalization (due to the inferior generalization ability), showing the usefulness of emerging continual-learning SNN-STDP systems from an algorithmic performance point of view.

Question 4 (Q4): How can SNN-STDP be further extended for performing other tasks in a continual-learning manner, such as for people detection on drones? To further extend the application domain of continual SNN-STDP within the drone context, chapter 6 has applied the top-down SNN-STDP framework presented in chapter 4 to the task of people detection from DVS data on drones. In order to successfully perform people detection, the SNN-STDP network used in chapter 6 has been augmented with additional supervised STDP contributions originating from teacher signals, forming a semi-supervised system. After about 2 minutes of learning phase on a continual stream of DVS event camera data, the network has been tested in inference mode on other independently captured flight acquisitions. Experimental results have shown that the proposed SNN-STDP system achieves a boost of +19% on the peak F_1 score compared to a CNN with comparable memory footprint processing the DVS data. Together with the SLAM application realized in chapter 5, this demonstration of human detection on drones via SNN-STDP constitutes true demonstrations of what is, to the best of our knowledge, amongst the most challenging real-world applications of continual SNN-STDP learning. In addition, chapter 7 has further extended the investigations on the possible application domains for continual-learning Hebbian systems by uncovering ways to unify the theory of joint Dictionary Learning and Basis Pursuit (DLBP), which forms the basis for SNN-STDP in chapter 4, with the theory of Active Inference (AIF), which describes how bio-inspired agents perform actions. Using the OpenAI mountain car environment, it has been shown that the proposed continual-learning Hebbian AIF system

outperforms the use of Q-learning by more than two orders of magnitude faster convergence rate, while not requiring the use of a replay buffer during learning. These findings and demonstrations open the door towards a more general application of continual-learning SNN-STDP systems.

8.2 Suggestions for future work

Future work that builds on the contributions made in this thesis can be directed towards the following points.

- In chapter 2, radar gesture recognition via SNNs has been demonstrated using surrogate-gradient back-propagation to enable a close comparison of the SNN behavior against the CNN behavior (since CNNs are usually trained using backprop). In addition, Appendix A has shown the feasibility of using STDP learning instead of surrogate gradient backprop in order to learn the SNN weights. This feasibility study could be extended further by investigating which radar pre-processing method would optimize the SNN-STDP accuracy even more and derive a methodology for such design. Furthermore, it might be possible to explore how the proposed pre-processing techniques apply to other sensory modalities such as audio signals.
- In addition to extending the investigation of radar pre-processing in the SNN-STDP case of Appendix A, it might be interesting to study how one can get rid of the *supervised* readout classifier system, used at the output of the SNN-STDP network for classifying the feature descriptors returned by the SNN-STDP model, and replace this supervised classifier (linear SVM) with an *unsupervised* clustering readout, which seeks to group the SNN-STDP feature descriptors into clusters corresponding to each of the considered gesture classes.
- Since many of the pre-processing techniques used in chapter 2 are also applicable to other sensor modalities such as, for example, audio signals and FMCW LIDAR, it might be valuable to study how the observations provided in chapter 2 about the impact of data pre-processing on the SNN accuracy generalize to the use of these other sensing modalities.
- In the first part of chapter 3, a novel radar detector has been proposed which, compared to conventional CFAR detectors, performs detection after a hardware-efficient feature mapping into a higher-dimensional binary space where linear separability between the true detection and the noise statistics is expected to be enhanced. Even though chapter 3 has provided a

thorough experimental characterization of the proposed detector, it might be possible to extend the theoretical study of the proposed detector further by attempting at finding analytical links between the detector parameters and the probability of detection and false alarm, given certain background clutter statistics. In addition, it might be possible to extend the domain of applicability of this method further by exploring other hardware-efficient feature encodings and by linking the binary feature encoding proposed in chapter 3 to the field of *Hyperdimensional Computing* [211], where feature mapping is also carried out in a binary space.

- It might also be interesting to experimentally assess the performance of the proposed radar detector of chapter 3 in a larger number of different indoor environments, so as to provide additional motivation for its general use in indoor settings.

- In the second part of chapter 3, data has been acquired for two different application domains: SLAM and people detection. It might be interesting to acquire data in a mixed scenario, where the drone must perform SLAM in the presence of walking people in its path. This can be useful for studying how to perform SLAM in unstructured environments prone to variability (since people are walking in the environment) and can be useful to build a complete system, unifying both SLAM and people detection and avoidance. This will also require the extension of the drone platform to incorporate any potentially co-designed digital processor and memory management support.

- In chapter 4, a top-down optimization-based theory has been presented for modelling the dynamics of an SNN-STDP system with a single coding (hidden) layer. It might be interesting to extend the SNN-STDP theory proposed in chapter 4 to the multi-layer case, where multiple SNN-STDP ensembles are cascaded. As a starting point, an interesting approach would be to feed the modelling error of each layer (i.e., the spikes returned by the error layer) into the next layer instead of simply the coding output of the previous SNN-STDP ensemble, or alternatively, to feed a combination of the both coding and error signals. This might enable a wavelet-style learning of *detail* coefficients, where the next SNN-STDP ensemble is trying to compensate for the error made by the previous SNN-STDP ensemble [112]. Furthermore, it can be interesting to study how to extend the proposed SNN-STDP theory to incorporate long-term homeostasis and multiple time scale dynamics.

- Chapter 5 has presented two continual-learning SLAM setups: a first one using RGB data with a non-spiking model and a second one fusing DVS and radar data with SNN-STDP networks. A low-hanging-fruit extension

to this is to merge the RGB and the DVS-radar setup so as to obtain an RGB-DVS-radar fusion SLAM. Furthermore, the SLAM system developed in chapter 5 could be used on other types of autonomous platforms such as wheeled robots.

- Even though our demonstrations in chapter 5 have shown the stability of the proposed continual-learning algorithms (see Fig. 5.2), ensuring absolute safety is still a critical but challenging endeavor. As interesting future work for further increasing and potentially even guaranteeing the safety of the proposed continual-learning systems, it may be possible to fuse the concept of *dynamic scenarios* proposed in [138] with the continual-learning SNN-STDP algorithms provided in chapter 5. This will provide additional guidelines for increasing the safety of the overall system, by adding higher-level system control to guarantee certain design targets (such as not hitting sensitive obstacles like people).

- In addition, a challenging aspect of SLAM covered in chapter 5 is the need for dynamic memory allocation to store the information about the new environments that the drone will visit in the future (memory consumption is not static but rather grows with time). This can become an issue from a hardware implementation perspective, impacting the energy and area consumption. Hence, interesting future work could address the memory management aspect of SLAM in order to improve the hardware efficiency.

- The continual SNN-STDP learning system for people detection presented in chapter 6 has been formulated as a semi-supervised learning problem where supervised teacher signals are also provided as additional STDP contributions to the SNN-STDP network. In this case, the supervised contribution consisted of *label* attention maps indicating where a walking human subject is present in the scene. Instead of using such explicit labelling system, it may be possible to re-cast the SNN-STDP people detection problem under a reinforcement learning setting, where a binary alarm signal (reward) indicates if the drone is approaching a human subject too closely. Using this reward, the system can learn to perceive human subjects and avoid them, closing the loop between perception and action. In addition, the continual SNN-STDP people detection algorithm of chapter 6 could be extended to other applications beyond people detection and using other types of robotics platforms.

- Chapter 7 has presented a promising approach for unifying DLBP-based Hebbian Learning network theory (which forms the basis of SNN-STDP in chapter 4) with the field of Active Inference, which studies how biological agents can learn and perform action at the same time. As future work, it might be interesting to study how the non-spiking Hebbian network

used in chapter 7 could be extended to the *spiking* SNN-STDP case. In addition, it is of interest to investigate how the Hebbian AIF framework presented in this chapter can be successfully applied to a range of other agent control tasks, beyond the Mountain Car environment used in chapter 7.

- Regarding continual learning in general, it might be interesting to further explore Bayesian techniques since iterative Bayesian updates provide a systematic way to achieve continual learning from first principles, where the posterior density at time step k is used as the prior density at time step $k+1$ [208]. Bayesian techniques are notorious though for being extremely compute intensive since the estimation of multi-dimensional probability density functions is very computationally demanding [152]. Hence, future research could be directed towards the simplification of the Bayesian update and its implementation with SNNs and local learning rules such as STDP.

- Finally, as a general future research direction overarching the various neuromorphic algorithmic concepts presented in this work, it could be of interest to unify the unsupervised continual SNN-STDP learning techniques presented in this thesis with the significant body of research conducted within the European MeMScales project[1], which studied the use of novel memory technologies with multi-scale time constants at both the algorithmic and hardware levels, for their use in neuromorphic computing architectures [105]. Indeed, the current SNN-STDP models presented in this thesis do not feature neurons and synapses with varying time constants, even though the extensive body of work produced by the MeMScales project has clearly shown the usefulness of multi-time-scale computing in neuromorphic systems. In addition, homeostasis techniques developed by the MeMScales consortium could also be fused with the SNN-STDP approach presented in this thesis, potentially leading to higher performances for even more complex and dynamic tasks.

8.3 Perspective on continual SNN-STDP learning

Chapter 5 clearly has demonstrated the advantages of continual SNN-STDP for SLAM in unknown environments. In the SLAM situation, the problem formulation was *i)* fully unsupervised and *ii)* about feature extraction, for projecting high-dimensional sensory data into lower-dimensional feature descriptors (clustering). In contrast, the problem formulation for the people

[1] https://memscales.eu/

detection SNN-STDP system of chapter 6 was *i) not* fully unsupervised (semi-supervised due to the addition of teacher signals) and *ii)* about the systematic detection of a specific type of object in the scene. These experimental investigations coupled with the general practical experience gathered in this thesis enable us to conclude the following.

We expect continual SNN-STDP learning systems to bring significant added value compared to SNNs or CNNs trained offline in applications where the learning process can be formulated as a fully unsupervised problem, and where the problem is about feature description in a continuous stream of multi-sensor data rather than the recognition of a clear systematic pattern (e.g., an object shape, a specific sound,...). Indeed, if the problem has a supervised nature, it is unclear how labels or teaching reference signals could be provided on chip during the continual-learning process. By definition, continual learning seeks to tune models on a stream of continuous sensory data and labels are not usually provided alongside this data stream. In addition, when a problem is about the systematic recognition of a certain pattern (with some level of repeatability in this pattern), it is well-known that performing offline training using a large dataset containing many instances of the object of interest is highly effective. Since pattern recognition is usually considered under a supervised setting, this might indicate that tasks such as people detection could be tackled better via conventional CNNs and offline training implemented in existing highly energy-efficient CNN accelerators such as [18, 93, 163].

In addition to pattern recognition involving well-specified object shapes, other deep learning tasks that are supervised in nature, such as during the training of transformers used in Large Language Models (LLMs), are also expected to currently be out of the domain of applicability of the proposed SNN-STDP approach, notwithstanding the fact that the unsupervised continual-learning processes covered in this thesis could bring additional functionalities to these larger-scale and supervised neural networks when faced with very non-stationary data streams.

On the other hand, when a task is not about recognizing a repeatable pattern that does not drift (like the shape of human bodies) but is rather operating in an environment that is subject to change or that can be part of a very heterogeneous set in a (multi-)sensor stream of data (e.g., there can be many visual differences between different warehouses), continual learning via hardware-efficient SNN-STDP is expected to provide a competitive advantage over the offline training of CNNs and SNNs. Indeed, as opposed the applications running in the cloud, extreme edge applications exhibit the above nature where processing must be done in a continual learning manner using a heterogeneous set of (multi-)sensor data streams without resorting to large models stored in the cloud. For example, SNN-STDP systems could be used to track the data features stemming

from a sensory array, subject to sensor drift due to changes in environmental conditions. Then, in order to perform the recognition of these output features, unsupervised cluster identification readouts could be used. This would lead to a fully unsupervised setup with the added benefit that sensor drifts would be compensated continually by the SNN-STDP system.

Still, it is important to list some of the foreseen limitations with the current SNN-STDP framework proposed in this thesis. The thesis has shown the applicability of the proposed SNN-STDP system to SLAM and people detection, which can be considered as significantly more challenging tasks compared to tasks tackled by prior SNN-STDP systems. On the other hand, these newly tackled tasks may still be seen as generic with regard to the larger-scale achievements of deep learning, often reaching extremely high task precisions. In the future, it might be possible to engineer more complex SNN-STDP-based systems through domain specialization and design expertise, in order to reach higher precisions in more complex task scenarios. This is unlike DNNs where the networks can gradually be made larger and learn to pre-process the heterogeneous data in a relatively task-agnostic manner. Hence, it is believed that with proper domain knowledge and sufficient resources, it is possible to scale the proposed SNN-STDP system even more in terms of task complexity and precision. This could also be done by cascading SNN-STDP systems to form multi-layer networks in order to tackle more complex tasks.

A final interesting point regarding SNN-STDP is its potential for low energy consumption and compactness in terms of implementation footprint. Indeed, chapter 5 has remarkably contrasted the proposed 1-hidden-layer SNN-STDP to the 11-layer LatentSLAM CNN as shown in Table 5.3. It can be seen that the SNN-STDP outperforms this 11-layer CNN, while being significantly smaller and hence, less computationally complex. This indicates that continual SNN-STDP systems might be a well-suited solution for reducing the energy and memory consumption compared to current CNN implementations which usually requiring a large memory and large data buffers for the intermediate data inside each layer of the CNN model.

8.4 Closing remarks

In recent years, emerging neuromorphic computing techniques such as SNNs and STDP learning have attracted considerable attention for extreme-edge AI applications, where neural network models must be executed with the tightest of energy and area budgets. But since SNNs and STDP learning have been studied less from an algorithm point of view, compared to the ubiquitous use

of DNNs, SNNs have often been lagging behind DNNs in terms of the task accuracies they can achieve, and the complexity of tasks they can tackle. As such, there is a need for considerable research effort at the algorithmic and theoretical level in order to help make neuromorphic techniques such as SNNs and STDP learning more successful on a broader set of sensory applications and task paradigms. This thesis has focused on these algorithmic aspects of SNNs and STDP, by providing SNN demonstrations with new sensor modalities such as radar, and by developing a top-down, optimization-based theory of continual STDP learning (vs. bottom-up in most previous works), which then has successfully been used to perform complex real-world drone navigation tasks. To the best of our knowledge, the drone demonstrations provided in this thesis constitute some of the most complex tasks successfully tackled by SNN-STDP systems in literature. As such, this thesis has demonstrated that adopting a top-down approach for the study of SNNs and STDP learning indeed has led to higher performances on more complex tasks than ever before.

Appendix A

Using SNN-STDP for radar gesture recognition

In this appendix, we investigate the feasibility of using the SNN-STDP architecture proposed in chapter 4 for the task of radar gesture recognition, *previously addressed* in chapter 2 using a convolutional SNN trained via backpropagation through time (BPTT) and the surrogate-gradient descent method described in section 2.5.2.

To this end, we apply the fully-connected SNN-STDP network in Fig. 4.1 to the 5-class 8-GHz radar gesture recognition dataset described in section 2.3.2, as follows. We use the μDoppler database obtained via the pre-processing steps corresponding to entry 6) in Table 2.3, as it led to the highest accuracy in Table 2.3. The database is randomly shuffled and partitioned into a training set and a test set using a 80%-20% split. A 2D μDoppler patch Θ is converted into a vector of spike trains by first scaling the patch to values between 0 and 1:

$$\Theta \leftarrow \frac{\Theta - \min(\Theta)}{\max(\Theta) - \min(\Theta)} \tag{A.1}$$

Then, the re-scaled patch is transformed into a binary image by using a threshold of 0.5 (i.e., all values values above 0.5 are mapped to a spike value of 1 and the remaining values are mapped to 0). Doing so, we can shift the column of the obtained binary patch one by one in time to the SNN-STDP input (as spiking vector of dimension 100 in the case of pre-processing 6) in Table 2.3). The spiking vectors from the train set are then fed to the SNN-STDP model,

which learns to extract features via STDP learning. Fig. A.1 illustrates the spike conversion steps described above.

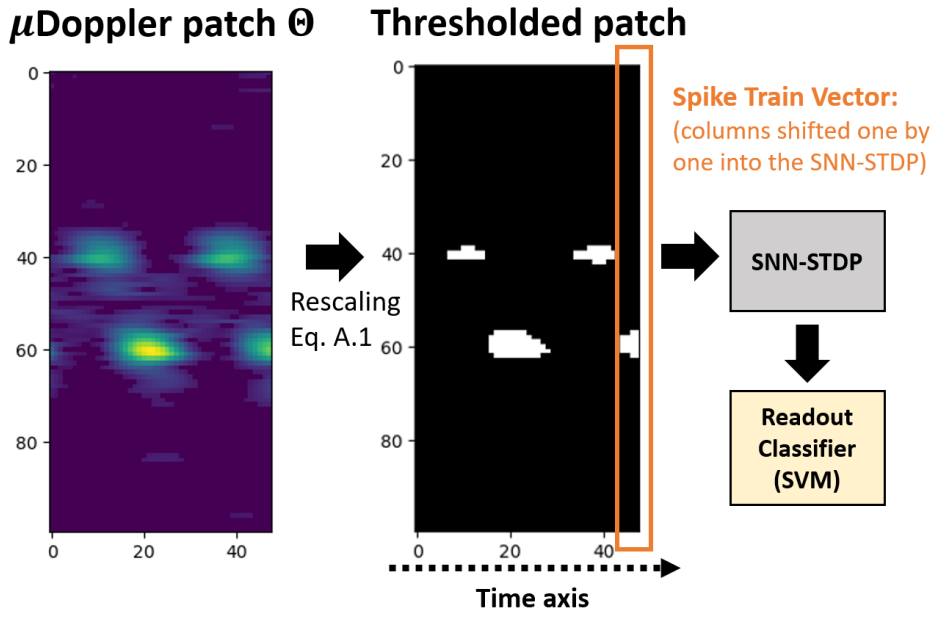

Figure A.1: *Converting the μDoppler patches to a spike train as input to the SNN-STDP network.*

After the STDP learning step, the SNN weights are fixed and the model is used to encode the input spike train vectors found in the training set into corresponding *output* spiking vectors $\bar{c}(t)$. Then, the average spiking rate of each output element $r\{\bar{c}(t)\}$ is derived by averaging $\bar{c}(t)$ along the time dimension. After normalization of all $\bar{c}$ into feature descriptors $\bar{\mathcal{F}}_g = \frac{r\{\bar{c}(t)\}}{||r\{\bar{c}(t)\}||_2}$ (following (4.42) in section 4.42), a linear Support Vector Machine or SVM (readout classifier) is trained on the feature descriptors obtained via the SNN-STDP.

During the test time, the data samples in the test set are first encoded into their feature descriptor representation $\bar{\mathcal{F}}_g$ using the learned SNN-STDP, and then classified via the SVM in order to derive the final test accuracy.

Fig. A.2 reports the test accuracy of the SNN-STDP system as a function of the number of *coding neurons*. The accuracy is averaged over 10 random train-test folds and the standard deviation is also shown in Fig. A.2. It can be seen in Fig. A.2 that, after an initial increase in accuracy, increasing the number of coding neurons from ~ 1000 onward does not significantly increase

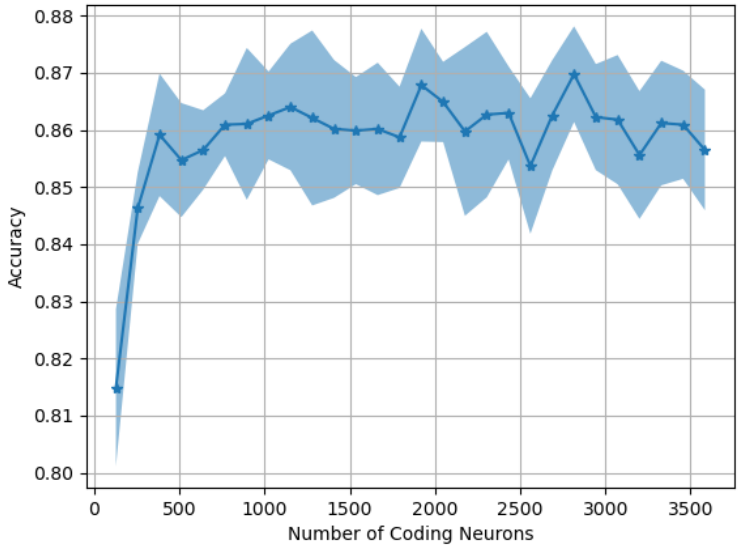

Figure A.2: *Radar gesture recognition accuracy and standard deviation as a function of the number of coding neurons.*

the accuracy anymore. The average accuracy along the plateau (from 1000 coding neurons onward) is ~ 86 %, which clearly demonstrates the feasibility of the proposed SNN-STDP system for the task of radar gesture recognition. It must be noted that the short SNN-STDP investigation provided above reaches 7% less accuracy compared to the convolutional SNN trained via backprop and gradient descent in chapter 4, which reaches 93% vs. 86% in this SNN-STDP case. This is likely due to the use of a convolutional layer in the convolutional SNN setup of chapter 4, which provides additional *translational invariance* compared to the fully-connected SNN-STDP setup of this appendix.

Still, since our experiments in chapter 4 have shown the crucial importance of pre-processing towards reaching a high SNN accuracy, it might be possible to increase the accuracy of the SNN-STDP system even more by further finetuning the radar pre-processing steps. This is left for further investigation as future work.

Appendix B

Baseline CNN design for people detection on drones

This appendix presents the CNN architecture we use as *baseline* deep learning model to compare the performance of our SNN-STDP system against a CNN with similar memory footprint in chapter 6 (see Fig. B.1). In this appendix, we will first detail the input data pre-processing prior to feeding the RGB, DVS and radar data to the CNN in section B.1. Then, we will describe our baseline CNN architecture together with our CNN training approach in section B.2. Finally, we will report experimental results on the precision-recall performance of the proposed CNN detector in section B.4 and we will provide conclusions in section B.5.

B.1 Input pre-processing

The input to our CNN is a concatenation of the RGB, DVS and radar modalities, forming a $(356 \times 260 \times 6)$ tensor (see the input of the network in Fig. B.1). The first three channels are the RGB channels, rescaled between 0 and 1. The fourth and fifth channels correspond to the positive and the negative polarity of the DVS frames obtained by taking the mean of the event time stamps $t_{k,n}$

Adapted from: *A. Safa at al., "Fail-Safe Human Detection for Drones Using a Multi-Modal Curriculum Learning Approach," in IEEE Robotics and Automation Letters, 2022.*

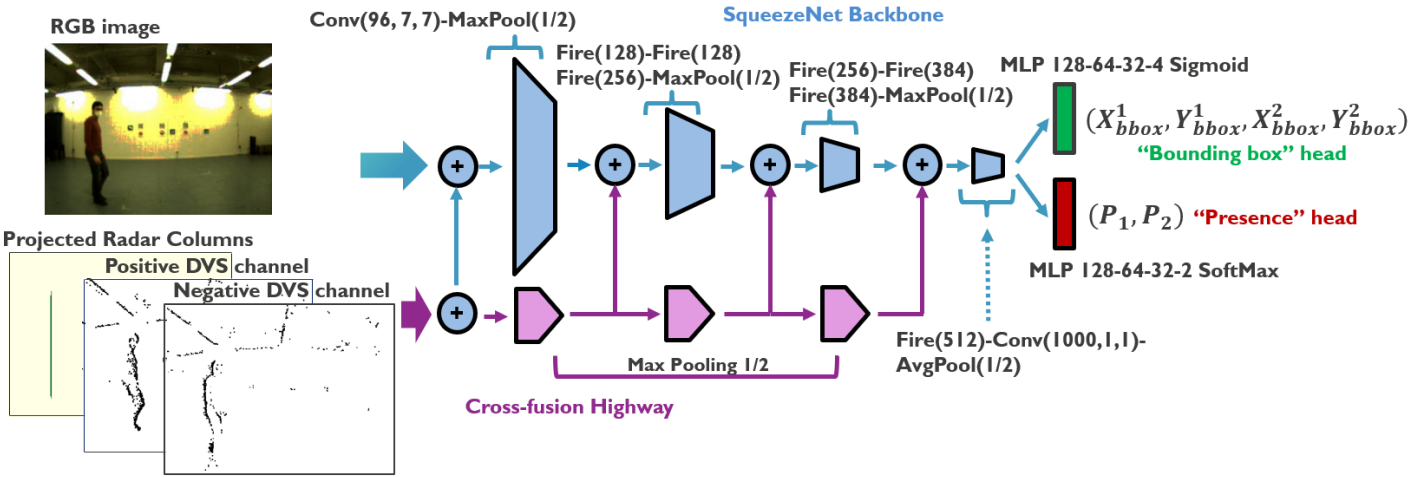

Figure B.1: *Our CNN architecture with cross-fusion highways. The RGB, DVS and radar inputs are concatenated and fed to the input of the SqueezeNet backbone. In addition, the DVS and radar inputs are concatenated and fed to the input of the cross-fusion highway. During training, the network learns at which scale fusion must happen the most along the backbone. Finally, two MLP heads are used for bounding box regression and for human presence detection.*

for each pixel $\vec{x}_k$ during a time window $\Delta T = 10$ ms, as follows:

$$D(\vec{x}_k, t_0) = \frac{\sum_{n=1}^{N_k(\Delta T)} t_{k,n} - t_0}{N_k(\Delta T)} \tag{B.1}$$

where, D denotes the final pixel value of the DVS frame, $N_k(\Delta T)$ is the number of events generated at pixel location $\vec{x}_k$ during ΔT, and t_0 is the initial time of the accumulation window. Eq. (B.1) is applied separately to the positive and the negative channels, obtaining one DVS frame for each polarity. Then, the DVS frames are normalized between 0 and 1 and concatenated with the RGB. Finally, the radar data must be projected from the range-azimuth view to the same perspective domain as the RGB and DVS images. As the drone altitude is not fixed, computing a homography between the radar range-azimuth plane and the cameras directly as in [14] cannot be done since the radar is not a *projective* sensor. This means that, as the drone altitude changes (for fixed X, Y location), the radar detection map does not change as the radar cannot distinguish the objects' elevation extents.

Instead, we use the TF-MINI range sensor mounted on our drone to measure the drone altitude h and we augment the radar range-azimuth detections (r_i, θ_i) with this elevation information to obtain tuples of the form $(r_i, \theta_i, -h)$ for each detection i. Thus, we assume that the detections are generated from objects that are anchored to the ground, which is a very generic assumption (shelves, benches and walking humans are all tied to the ground floor). Using these augmented radar detections, we construct an intermediate image-like representation of the radar data as follows:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \frac{f}{-h} \begin{bmatrix} r_i \cos\theta_i \\ r_i \sin\theta_i \end{bmatrix} \tag{B.2}$$

where (x_i, y_i) represent the coordinates of the radar detection i in the intermediate space and f is a focal length factor, which we chose equal to the focal length of our DAVIS-356 camera (measured through checkerboard calibration). Then it is sufficient to compute the homography matrix M using more than eight detection coordinates (x_i, y_i) and their corresponding locations on the image planes, for different drone heights.

We perform this step offline as an extrinsic calibration procedure using a corner reflector as in [14]. Then, we project the intermediate point coordinates (x_i, y_i) on the image plane, and we extend their height vertically as done in [174] to better cover their spatial context, assuming a prior height of 1.5 m for all detections. We set the pixel values of the resulting *radar columns* as the normalized depth value $d_i = \frac{r_i \sin\theta_i}{d_{max}}$ measured by the radar. Fig. B.2 shows an example of the *projected radar columns* obtained through this process (input to the CNN). It must be noted that slight calibration errors may occur. But, as our cross-fusion CNN can learn to mitigate them during training, we did not

observe any problem linked to such errors during our experiments. Finally, we



Figure B.2: *Radar columns obtained by projecting the range-azimuth radar detections on the image plane by homography (star-shaped markers) and extending them in height to cover their spatial context.*

concatenate the *radar columns* with the RGB and DVS channels as the last channel of the input tensors T_n:

$$T_n = (I_n^3, D_n^2, R_n^1) \tag{B.3}$$

where n is the frame index, I_n^3 is the RGB image, D_n^2 is the DVS frame with its two polarity channels, and R_n^1 is the single radar channel.

In order to obtain training labels, we use a YOLOv3 detector [197] to automatically estimate the bounding box coordinates and the presence or absence of a human subject. In the case of occlusion, the YOLOv3 still returns a bounding box covering the visible part of the walking subject. The bounding box coordinates are encoded as:

$$\underline{b} = (X_{bbox}^1, Y_{bbox}^1, X_{bbox}^2, Y_{bbox}^2) \tag{B.4}$$

where X_{bbox}^1, Y_{bbox}^1 is the normalized top-left coordinate and X_{bbox}^2, Y_{bbox}^2 is the normalized bottom-right coordinate. The normalization is done by dividing the coordinates by the length of their respective axis to obtain values between 0 and 1. Finally, we encode the presence or the absence of a human subject as a one-hot encoded vector of two elements (P_1, P_2). Naturally, it should be noted that the YOLOv3 annotation is not perfect, but still, the learning is very robust towards annotation noise since the number of mistakes in automated annotation is very small compared to the correct annotations. The successive tensors and their corresponding labels are saved at the frame rate of the RGB camera (30 FPS) to form the final labelled datasets for each acquisition.

B.2 CNN design

Fig. B.1 shows the baseline CNN architecture used in this work. The network backbone is a SqueezeNet [100], chosen for its advantageous balance between high accuracy, low inference speed on typical edge-TPUs ($\sim$ 2 ms) [221] and low memory footprint ($<$ 50MB). This performance is achieved thanks to the use of *Fire* modules instead of the typical *Convolutional* layers (see Fig. B.1) [100]. We augment the SqueezeNet backbone by adding a *cross-fusion highway* [174], which successively applies *Max Pooling* on its input data and concatenates the intermediate results with the corresponding feature maps of the backbone, for each resolution level (see Fig. B.1). In contrast to traditional *early fusion*, which only fuses data at maximal resolution, our *cross-fusion* setting enables the network to learn *at which resolution level* fusion must best take place [174] (see the ablation experiments in section B.4.2). Finally, we connect two distinct *Multi-Layer Perceptron* (MLP) heads to the backbone. The *bounding box* head (see the green box in Fig. B.1) features four sigmoid outputs that are used for the regression of the normalized bounding box coordinates (B.4). The *presence* head (see the red box in Fig. B.1) features a two-class *SoftMax* output for predicting the presence or the absence of the human subject. The input to the SqueezeNet backbone is the concatenation of the RGB, DVS and radar maps (B.3), while the input to the cross-fusion highway is the concatenation of only the DVS and the radar maps (D_n^2, R_n^1).

As the people detection dataset captured in section 3.9.1 of Chapter 3.7 is composed of a single walking subject and as our aim is to analyse the sensor fusion encoding and learning strategies, we have used a *single* bounding box regression output at the head of our CNN. Thus, our backbone can be extended naturally to a *multi* bounding box case by adding e.g., the output layers of a Tiny-YOLO [61] or of a *RetinaNet* [139], as done in [174].

B.3 Choice of the loss function

In a general way, we can write the CNN learning objective as the minimization of the following loss function:

$$\mathcal{L}_{tot} = \lambda \mathcal{L}_{bbox} + (1 - \lambda)\mathcal{L}_{pres} \tag{B.5}$$

where $\mathcal{L}_{bbox}$ is the bounding box loss, measuring the distance between the predicted bounding box and the ground truth, $\mathcal{L}_{pres}$ is the presence loss associated with the human presence detection head in Fig. B.1, and λ is a hyper-parameter controlling the balance between the two losses. As usual for

classification, we choose $\mathcal{L}_{pres}$ to be the *cross-entropy* loss. For bounding box regression, the learning task is to minimize the distance $\mathcal{L}_{bbox}(\hat{\underline{b}}, \underline{b})$ between the estimated top-left and bottom-right coordinates of the bounding box $\hat{\underline{b}}$ in Fig. B.1 and the ground-truth coordinates $\underline{b}$ (B.4). In a number of works [140], $\mathcal{L}_{bbox}$ is chosen as the *Huber* (or *smooth* L_1) loss for its robustness towards outlier coordinates. Instead, we use the inverted Huber (or *berHu*) loss (B.6) in order to focus learning on the difficult examples [125]. During our experiments, this choice was confirmed by observing that the *berHu* loss gave significantly better results than the *Huber* loss, qualitatively leading to *less fuzzy behavior* in the bounding box estimates during testing.

$$\mathcal{L}_{berHu}(\hat{b}_i, b_i) = \begin{cases} |\hat{b}_i - b_i|, & \text{if } |\hat{b}_i - b_i| \leq c \\ \frac{(\hat{b}_i - b_i)^2 + c^2}{2c}, & \text{else} \end{cases} \tag{B.6}$$

Similar to [125], we adaptively set the c parameter of (B.6) as $c = 0.2 \times \max_i |\hat{b}_i - b_i|$ where the i index denotes the ith element of the bounding box coordinate vector.

Regarding the setting of the hyper-parameter λ in (B.5), we have observed during our experiments that we systematically achieved better results by first training our CNN for bounding box regression only, and then freezing the weights of the network, adding a second MLP head for human presence detection (see Fig. B.1) and training this second head independently. Indeed, optimizing for the detection and bounding box regression objectives at the same time may introduce antagonistic effects for our specific architecture, which may harm learning. Instead, training for the bounding boxes first enables the backbone CNN to learn robust human feature representations, well suited for the presence detection head as well. Then, fine-tuning the detection head, in turn, results in a high-accuracy human detection output.

B.3.1 Training approach: a curriculum learning strategy

Still one question remains: how should we *present* the data during training to learn a system which does not predominantly rely on one of the sensing modalities only? Indeed, a naive learning strategy might result in a system that has learned to rely on e.g., the RGB data only, not taking advantage of the other modalities, which is fatal in the case of a hard RGB sensor fault. To address this issue, the authors in [174] introduced a procedure called *BlackIn*, where each modality j of the input data has a probability p_j of being zeroed out during training. In our case, the input tensor is the concatenation of the three RGB channels, the two DVS channels and the radar channel (B.3). Using

BlackIn, the nth input training example can be written as:

$$T_n^b = (\mathcal{B}_{p_1} I_n^3, \mathcal{B}_{p_2} D_n^2, \mathcal{B}_{p_3} R_n^1) \tag{B.7}$$

where $\mathcal{B}_{p_j}$ is a Bernoulli-distributed random variable with probability p_j. Here, we note that the stochasticity introduced by *BlackIn* is *stationary* during training as the $p_j, \forall j$ are fixed during the training procedure. In contrast, we use *curriculum learning* [21], which relies on a changing dataset during training (a non-stationary process). Formally, let Q_λ be a sequence of distributions modelling the target training distribution from which the function of interest should be learned, with a content difficulty parametrized through $\lambda \in [0, 1]$. Then, the sequence Q_λ is defined as a *curriculum* if [21]:

$$H(Q_\lambda) < H(Q_{\lambda+\epsilon}) \quad \forall \epsilon > 0 \tag{B.8}$$

where H denotes the entropy of the distribution. It has been shown that such learning strategy can significantly enhance the model performance on uni-modal tasks such as image recognition, where gradually increasing the difficulty of the images to be recognized (or equivalently, the entropy of the training distribution) helps to obtain a better model [21].

In our case, instead of defining a curriculum over *data examples*, from easy to hard, why not defining it over *sensors*, from low to high-entropy *sensory* data, and combine this principle with *BlackIn*? In order to test this idea, we define the learning procedure of Algorithm 1 by combining *BlackIn* [174] with *curriculum learning* to *shake up* the basins of attraction of the loss landscape in order to ease convergence towards a better local minimum [21]. We call this new method *Shake-up Learning* (SAUL). The algorithm is described below.

Algorithm 7 SAUL procedure

Input: $\mathbf{T} = T_n^b, \forall n$: training data (B.7), $\vec{p}_a, \mathcal{E}_1, \vec{p}_b, \mathcal{E}_2$: probability vectors and corresponding epochs

1: **for** epochs in 1 to N_e **do**
2: **if** epochs $\in \mathcal{E}_1$ **then**
3: $\vec{p} \leftarrow \vec{p}_a$ {//initial probabilities of (B.7)}
4: **else if** epochs $\in \mathcal{E}_2$ **then**
5: $\vec{p} \leftarrow \vec{p}_b$ {//abrupt change in probabilities of (B.7)}
6: **end if**
7: Train($\mathbf{T}, \vec{p}$) {//training with pruned dataset according to (B.7) where p_j are the components of the $\vec{p}$ vector.}
8: **end for**

In the case of our sensor suite, we initialise the probabilities in (B.7) such that the RGB data is the most pruned (high p_1), followed by the DVS data

($p_2 < p_1$). The radar data are never pruned ($p_3 = 0$). Then, we decrease the pruning rate of both the RGB and the DVS data abruptly in epochs $\mathcal{E}_2$, half way through the training procedure. By doing so, we impose a sudden jump in the entropy of the training data distribution. This complies with the curriculum definition of (B.8) as the RGB data distribution has a higher entropy than the DVS distribution (edge image with an important number of zero-valued pixels compared to non-zero pixels), which has a higher entropy compared to radar (sparse set of detections with even more null pixels than the DVS). In the next section, we will compare the performance of our model trained via *BlackIn* against the performance obtained via SAUL, and we will report a significant boost in detection and bounding box regression performance.

B.4 Experimental results

B.4.1 Precision-recall analysis

We first train our model for bounding box regression during 300 epochs. We then fine-tune the presence detection head during 40 epochs. For training and testing, we use 5-fold leave-one-out cross-validation as follows. In each cross-validation fold, we keep as test set one fly acquisition with challenging background in Table 3.4 and use the remaining acquisitions as the training set. We use *Adam* with batch size 32 and learning rate 5×10^{-5} for both training phases. We report the performance of our proposed model by measuring the peak F_1 score (i.e., maximum over i of $F_{1,i} = 2\frac{P_i R_i}{P_i + R_i}$) along each precision-recall curve (P_i, R_i) in Fig. B.3 (the higher, the better) [246].

During our experiments, we train two models. One model is trained using the *BlackIn* procedure of [174] with $p_1 = 0.3$ for the RGB and $p_2 = 0.09$ for the DVS, while keeping $p_3 = 0$ for the radar in (B.7). The other model is trained with our proposed SAUL procedure (Algorithm 7) with $\vec{p}_a = [0.7, 0.2, 0]$ (for the first 150 epochs during bounding box learning and the first 20 epochs during detection learning) and $\vec{p}_b = [0.1, 0, 0]$ (for the last 150 epochs and the last 20 epochs of each learning phase), in order to first severely prune the RGB and to moderately prune the DVS, and then, to moderately prune the RGB only. We found those parameter values by testing different combinations and choosing the best ones for both *BlackIn* and SAUL. Doing so, the model is trained to rely mostly on DVS and radar, and to be resilient to an RGB fault since RGB is pruned the most during training.

The precision-recall curves are generated for an Intersection over Union (IoU) threshold of 0.5, by sweeping the threshold of the *presence* output of our CNN

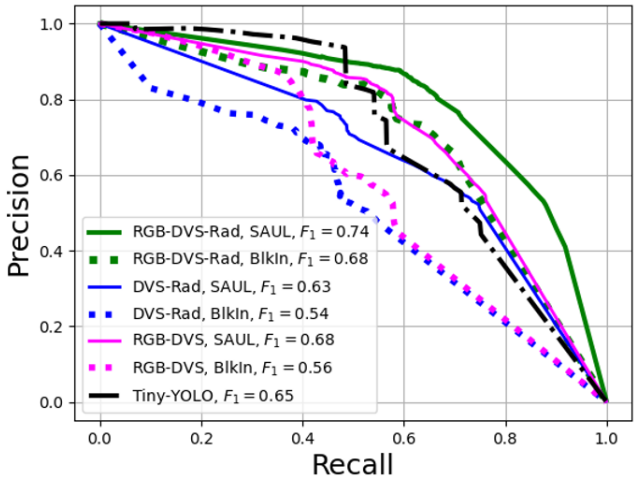

Figure B.3: *Precision-recall curves of the CNN detector with an IoU threshold of* 0.5.

(see Fig. B.1) from 0 to 1 [246]. Compared to a pre-trained tiny-YOLO [61], our cross-fusion system provides a gain of 13% on the peak F_1 score. Compared to *BlackIn*, our proposed SAUL procedure provides an average gain of 15% across the different sensor combinations. This clearly shows that using SAUL, our *cross-fusion* model can learn to use the sensing modalities in a significantly better way than through the use of *BlackIn*. Fig. B.3 also shows the robustness of the system against an RGB fault. Under such RGB sensor fault, a gain of 17% is obtained against *BlackIn* when using our SAUL procedure. In addition, Fig. B.3 shows the impact of using the radar on the peak F_1 score. Furthermore, Fig. B.3 also show that the losses on the precision-recall trade-off due to sensor ablation is more sudden using our proposed SAUL procedure compared to the losses obtained using *BlackIn*. It should be noted that the models are very sensitive to a DVS ablation, which is expected since the training procedure mostly relies on DVS and radar at the beginning of the curriculum.

B.4.2 Ablation studies

Cross-fusion vs. early-fusion

In section B.2, we have motivated our use of a cross-fusion strategy in order to enable the network to learn *at which feature scales* fusion should happen the most. It is therefore important to demonstrate the gain induced by cross-fusion

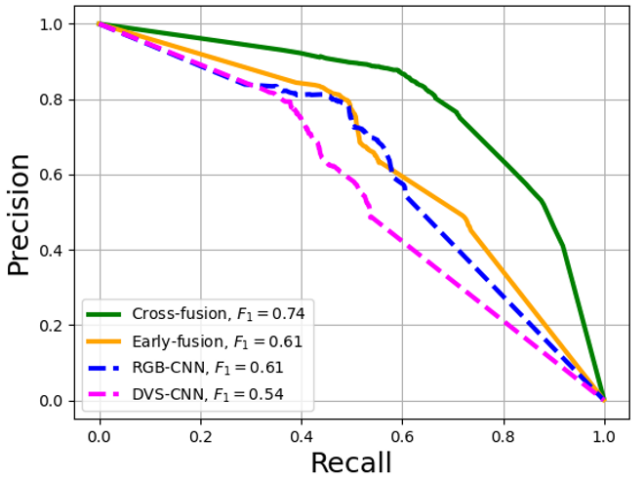

Figure B.4: *Ablation studies showing precision-recall curves of a) cross-fusion vs. early-fusion of RGB, DVS and radar modalities, and b) fusion vs. single modality networks.*

over a traditional *early-fusion* strategy, where all modalities are concatenated once and given as input to the CNN (no cross-fusion highway). Fig. B.4 compares the precision-recall curves obtained using our cross-fusion network against early-fusion. The cross-fusion strategy provides a gain of 21% on the peak F_1 score. This is expected since the cross-fusion strategy enables the network to learn at which scales the different sensing modalities should optimally be fused.

Sensor fusion vs. single modality

In addition, it is important to quantify the robustness gained by the fusion of the modalities versus single-modality networks. We compare our cross-fusion CNN to both an RGB-CNN and a DVS-CNN. We do not report a radar-trained CNN system since radar detections alone do not convey any features that can be learned efficiently by the CNN (they only indicate the position of a reflector object). Fig. B.4 shows the precision-recall curves comparing the single modality networks to our cross-fusion CNN. The cross-fusion strategy provides respective gains of 21% and 37% on the peak F_1 score compared to the RGB-only and the DVS-only CNNs.

B.5 Conclusion

This appendix has provided the design of a baseline CNN architecture for people detection. After introducing the topology of our baseline CNN based on the SqueezeNet architecture and the input data pre-processing steps, our innovative multi-modal training approach termed SAUL has been proposed, based on a curriculum learning strategy. Finally, experiments have been described to study the performance of the CNN baseline, and a number of ablation studies have been conducted to compare the proposed cross-fusion approach and the SAUL training strategy with early fusion and the previously proposed *BalckIn* training method for multi-modal learning. It has been shown that our cross-fusion approach with the SAUL training strategy outperforms both early fusion and the use of the *BlackIn* method. Chapter 6 compares the performance of our newly proposed SNN-STDP DVS people detection network against the baseline CNN (with similar memory footprint) described in this appendix.

Appendix C

Enhancements added to the RatSLAM back-end

C.1 Graph-based map correction

Our graph-based SLAM back-end takes its roots in the popular *RatSLAM* architecture [159], but deviates from it through the use of the graph formulation in [84], as well as the custom data association and link information management techniques detailed in the rest of this appendix.

C.1.1 New node registration

For each new drone pose, the back-end receives a tuple $\{\Psi_{k^*}, X_{k^*}, Y_{k^*}, \dot{\Psi}, v_h, \mathcal{R}_{k^*}, \mathcal{O}_{k^*}\}$ with $\Psi_{k^*}, X_{k^*}, Y_{k^*}, \dot{\Psi}, v_h$, the raw odometry information in (5.21), $\mathcal{R}_{k^*}$ the radar detections and $\mathcal{O}_{k^*}$ the observations (RGB or event data) at pose index k^*.

Then, global feature descriptors $\bar{\mathcal{F}}$ are inferred from the observation $\mathcal{O}_k$ by using the CL DLBP system on the RGB data (see section 5.3.1) or the CL SNN-STDP network on the DVS data (see section 5.3.2):

$$\bar{\mathcal{F}}_{k^*} = \mathcal{G}(\mathcal{O}_{k^*}) \tag{C.1}$$

where $\mathcal{G}(.)$ denotes the application of the CL system to the observation.

The feature descriptors are then compared to the previous descriptors associated with the previous drone poses $k < k^* - k_g$ using the Pearson correlation coefficient, where k_g is a guarding constant preventing erroneous association with the most recent poses:

$$s_k = \frac{\bar{\mathcal{F}}_k^T \bar{\mathcal{F}}_{k^*}}{||\bar{\mathcal{F}}_k||_2 ||\bar{\mathcal{F}}_{k^*}||_2}, \ \forall k < k^* - k_g \tag{C.2}$$

If all s_k, $\forall k < k^* - k_g$ are smaller than a threshold T_1, the current pose k^* is registered as a *new node* in the graph, and is connected by an *edge* $l(k^* \leftrightarrow k^* - 1)$ to the previous node $k^* - 1$ (corresponding to the previous drone pose). In addition, the *edge* $l(k^* \leftrightarrow k^* - 1)$ is assigned with the odometry distance d, the yaw angle difference ψ between the two nodes, and an *information factor* Ω indicating the *confidence level* of the link [84]:

$$\begin{cases} d(k^* \leftrightarrow k^* - 1) = \sqrt{(X_{k^*} - X_{k^*-1})^2 + (Y_{k^*} - Y_{k^*-1})^2} \\ \psi(k^* \leftrightarrow k^* - 1) = \text{wrap}_{2\pi}\{\Psi_{k^*} - \arctan(\frac{Y_{k^*} - Y_{k^*-1}}{X_{k^*} - X_{k^*-1}})\} \\ \Omega(k^* \leftrightarrow k^* - 1) = \omega_o \end{cases} \tag{C.3}$$

where $\text{wrap}_{2\pi}\{.\}$ denotes the wrapping of the angle to the $[0, 2\pi]$ range and ω_o is a tuning parameter.

C.1.2 Loop closure detection

If the similarity with a previous pose s_k is larger than T_1 and the distance between this pose k, as well as the yaw angle difference with regard to the current pose k^* is smaller than a threshold R_ψ, a *loop closure* is detected:

$$\begin{cases} s_k \geq T_1 \\ \sqrt{(X_{k^*} - X_k)^2 + (Y_{k^*} - Y_k)^2} < R_d[k^*] \\ \text{wrap}_{2\pi}\{\Psi_{k^*} - \Psi_k\} < R_\psi \end{cases} \Rightarrow l(k^* - 1 \leftrightarrow k) \tag{C.4}$$

where $R_d[k^*]$ is the radius of association for the distance between poses, and R_ψ is the radius of association for the yaw angle difference. The Euclidean association radius $R_d[k^*]$ evolves throughout the iterations k following the data association strategy detailed in Section C.1.4. If multiple nodes are eligible for loop closure, we choose the one with the highest Pearson correlation s_k.

In addition, we define a second condition for loop closure as follows:

$$\begin{cases} s_k \geq T_2 \\ \sqrt{(X_{k^*} - X_k)^2 + (Y_{k^*} - Y_k)^2} < R_{sp} \end{cases} \Rightarrow l(k^* - 1 \leftrightarrow k) \tag{C.5}$$

where $T_2 > T_1$ is a *larger* second threshold and R_{sp} is a fixed radius. This second condition helps recovering the drone track when the first condition (C.4) is stalled.

When a loop closure is detected, a new node is not added to the graph, but instead the previous node $k^* - 1$ is linked with the targeted node k through an edge $l(k^* - 1 \leftrightarrow k)$ with the following edge attributes: [84]:

$$\begin{cases} d(k^* \leftrightarrow k^* - 1) = \sqrt{(X_{k^*} - X_{k^*-1})^2 + (Y_{k^*} - Y_{k^*-1})^2} \\ \psi(k^* \leftrightarrow k^* - 1) = \text{wrap}_{2\pi}\{\Psi_{k^*} - \arctan(\frac{Y_{k^*} - Y_{k^*-1}}{X_{k^*} - X_{k^*-1}})\} \\ \Omega(k^* \leftrightarrow k^* - 1) = \omega[k^*] \end{cases} \tag{C.6}$$

where X_{k^*}, Y_{k^*} and Ψ_{k^*} are the output of the raw odometry (5.21), and $\omega[k^*]$ is the information factor of the edge that evolves throughout the iterations k following the data association strategy detailed in section C.1.4.

After the edge assignment (C.6), the drone position and heading are moved to the node k (with which the loop closure happened) by promoting the node k to be the *current node* $k^* \leftarrow k$ (a new node is therefore *not* created). Doing so, the current position of the drone and its current heading are corrected by assigning them to the position and heading of the target node k:

$$X_{k^*} \leftarrow X_k \quad Y_{k^*} \leftarrow Y_k \quad \Psi_{k^*} \leftarrow \Psi_k \tag{C.7}$$

Consequently, the odometry integration in (5.21) is also corrected for the upcoming time steps.

Each time a loop closure happens, the pose returned by the odometry integration and the detected pose are different, leading to an error between the pose expected by odometry $\hat{P}(k^* \leftrightarrow k^* - 1)$:

$$\begin{aligned} \hat{P}(k^* \leftrightarrow k^* - 1) = &\begin{bmatrix} X_{k^*-1} \\ Y_{k^*-1} \end{bmatrix} \\ &+ \begin{bmatrix} d(k^* \leftrightarrow k^* - 1)\cos(\Psi_{k^*-1} + \psi(k^* \leftrightarrow k^* - 1)) \\ d(k^* \leftrightarrow k^* - 1)\sin(\Psi_{k^*-1} + \psi(k^* \leftrightarrow k^* - 1)) \end{bmatrix} \end{aligned} \tag{C.8}$$

and the actual pose $\begin{bmatrix} X_{k^*} \\ Y_{k^*} \end{bmatrix}$ after the correction (C.7):

$$\bar{e}_d(k^* \leftrightarrow k^* - 1) = \begin{bmatrix} X_{k^*} \\ Y_{k^*} \end{bmatrix} - \hat{P}(k^* \leftrightarrow k^* - 1) \tag{C.9}$$

In addition, a heading angle error is also made:

$$e_\psi(k^* \leftrightarrow k^* - 1) = \text{wrap}_{2\pi}\{\Psi_{k^*} - (\Psi_{k^*-1} + \psi(k^* \leftrightarrow k^* - 1))\} \tag{C.10}$$

As the drone hovers through the environment, errors (C.9, C.10) will be distributed along the edges of the pose graph and must be corrected by a pose correction algorithm in order to refine the pose map [84], as will be described next.

C.1.3 Graph correction

The graph correction algorithm seeks to minimize the errors along the graph, weighted by the information factors [84]:

$$X_k, Y_k, \Psi_k = \arg \min_{X_k, Y_k, \Psi_k} \sum_{ij} \Omega(k_i \leftrightarrow k_j) ||\bar{e}_d(k_i \leftrightarrow k_j)||_2^2 + \sum_{ij} \Omega(k_i \leftrightarrow k_j) e_\psi^2(k_i \leftrightarrow k_j) \;\; \forall k \leq k^* \quad \text{(C.11)}$$

As conventionally done [31, 159], we minimize (C.11) by applying *gradient descent* for N_l number of loops with convergence rate η_g each time a new loop closure is executed.

C.1.4 Data association and edge information

Radius of association

Instead of using a fixed radius of association as in [159], we allow the radius $R_d[k]$ in (C.4) to change along the iterations k. Inspired by the evolution of the state error covariance matrix of a *Kalman filter*, which is typically used to synthesize an ellipsoidal data association gate in tasks such as radar tracking [191], we use the following scheme to adapt $R_d[k]$:

$$\begin{cases} R_d[k] = R_{init}, k_{init} \leftarrow k \;\; \textbf{if} \;\; \textit{loop closure} \\ R_d[k] = R_{init} + \nu_1 (k - k_{init}) \;\; \textbf{if} \;\; (k - k_{init}) < K \\ \textbf{else} \;\; R_d[k] = R_{init} + \nu_1 K \end{cases} \quad \text{(C.12)}$$

where R_{init} is an initial gate radius value, k_{init} keeps track of the time from the *last* loop closure event, ν_1 is a slope governing the speed of the radius growth and K is the maximum number of time steps during which the radius grows.

By using (C.12), the data association becomes more restrictive when a loop closure happens, denoting the high confidence of the system in its current

position, which prevents spurious associations with possible aliasing nodes. Then, this confidence diminishes as long as new loop closures are not validated and hence, the radius of association grows towards a maximum.

Edge information factor

Complementary to the growth of the radius of association detailed in section C.1.4, we also modulate in a similar manner the edge information $\omega[k]$ in (C.6):

$$\begin{cases} \omega[k] = \omega_{init}, k_{init} \leftarrow k \ \textbf{if} \ \textit{loop closure} \\ \omega[k] = \omega_{init} + \nu_2(k - k_{init}) \ \textbf{if} \ (k - k_{init}) < K \\ \omega[k] = \omega_{init} + \nu_2 K \ \textbf{else} \end{cases} \tag{C.13}$$

where ω_{init} is the initial information factor and ν_2 is a slope governing the speed of the information growth.

The motivation behind our use of (C.13) stems from the fact that right after a loop closure, the odometry integration (5.21) is well corrected, and hence edges initialized via raw odometry integration (C.3) are of *high confidence* right after the execution of the loop closure and correction (C.7). Therefore, the confidence of edges initialized via the *template matching* (C.6) (vs. via the odometry) must be reduced to give more importance to the edge initialized via odometry (C.3). Conversely, when a certain time has elapsed without any detected loop closures, it indicates that the raw odometry information has been accumulating erroneous deviations. In this case, if a loop closure happens, its information content can be considered of high importance. This is why we make $\omega[k]$ in (C.13) grow in time in the absence of any loop closure.

C.2 Conclusion

This appendix has provided a description of the modifications done to the original RatSLAM back-end in order to improve its localization and mapping precision further. These modifications were done by fusing the map correction algorithm found in the original RatSLAM [159] with the more recently proposed enhancement techniques provided in [84]. This modified RatSLAM back-end has been used in the SLAM experiments of chapter 5.

Bibliography

[1] FFT implementation on the TMS320VC5505, TMS320C5505, and TMS320C5515 DSPs. *Texas Instruments application report* (2013).

[2] Sparse dictionary learning by dynamical neural networks. In *International Conference on Learning Representations*, T.-H. Lin and P. T. P. Tang, Eds. 2019.

[3] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Ablin, P., Moreau, T., Massias, M., and Gramfort, A. Learning step sizes for unfolded sparse coding, 2019.

[5] Ai, J., Luo, Q., Yang, X., Yin, Z., and Xu, H. Outliers-robust cfar detector of gaussian clutter based on the truncated-maximum-likelihood-estimator in sar imagery. *IEEE Transactions on Intelligent Transportation Systems 21*, 5 (2020), 2039–2049.

[6] Ai, J., Yang, X., Song, J., Dong, Z., Jia, L., and Zhou, F. An adaptively truncated clutter-statistics-based two-parameter cfar detector in sar imagery. *IEEE Journal of Oceanic Engineering 43*, 1 (2018), 267–279.

[7] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget.

In *Computer Vision – ECCV 2018* (Cham, 2018), V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Springer International Publishing, pp. 144–161.

[8] Aljundi, R., Rohrbach, M., and Tuytelaars, T. Selfless sequential learning. In *International Conference on Learning Representations* (2019).

[9] Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., and Modha, D. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), IEEE.

[10] Angelopoulos, A. N., Martel, J. N., Kohli, A. P., Conradt, J., and Wetzstein, G. Event based, near-eye gaze tracking beyond 10,000 hz. *IEEE Transactions on Visualization and Computer Graphics (Proc. VR)* (2021).

[11] Arnon, Taba, B., Berg, D., Melano, T., Mckinstry, J., Nolfo, C., Di, Tapan, Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Michael, Esser, S., Delbruck, T., Flickner, M., and Modha, D. A low power, fully Event-Based gesture recognition system. In *Computer Vision and Pattern Recognition (CVPR) 2017*. Honolulu, 2017, pp. 7243–7252.

[12] Arsalan, M., Santra, A., and Issakov, V. Radarsnn: A resource efficient gesture sensing system based on mm-wave radar. *IEEE Transactions on Microwave Theory and Techniques 70*, 4 (2022), 2451–2461.

[13] Arsalan, M., Santra, A., and Issakov, V. Spiking neural network-based radar gesture recognition system using raw adc data. *IEEE Sensors Letters 6*, 6 (2022), 1–4.

[14] Aziz, K., De Greef, E., Rykunov, M., Bourdoux, A., and Sahli, H. Radar-camera fusion for road target classification. In *2020 IEEE Radar Conference (RadarConf20)* (Sept. 2020), IEEE.

[15] Baldi, P., and Itti, L. Of bits and wows: A bayesian theory of surprise with applications to attention. *Neural networks : the official journal of the International Neural Network Society 23* (2010), 649–666.

[16] Bambagini, M., Marinoni, M., Aydin, H., and Buttazzo, G. Energy-aware scheduling for real-time systems: A survey. *ACM Trans. Embed. Comput. Syst. 15*, 1 (jan 2016).

[17] Banerjee, D., Rani, S., George, A. M., Chowdhury, A., Dey, S., Mukherjee, A., Chakravarty, T., and Pal, A. Application of spiking neural networks for action recognition from radar data. In *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020), IEEE.

[18] Bankman, D., Yang, L., Moons, B., Verhelst, M., and Murmann, B. An always-on 3.8 μ j/86 *IEEE Journal of Solid-State Circuits 54*, 1 (2019), 158–172.

[19] Bao, C., Ji, H., Quan, Y., and Shen, Z. Dictionary learning for sparse coding: Algorithms and convergence analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 38*, 7 (July 2016), 1356–1369.

[20] Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications 11* (07 2020).

[21] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, June 2009), ACM.

[22] Bengio, Y., Mesnard, T., Fischer, A., Zhang, S., and Wu, Y. STDP-Compatible Approximation of Backpropagation in an Energy-Based Model. *Neural Computation 29*, 3 (03 2017), 555–577.

[23] Bi, G., and Poo, M.-m. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience 18* (01 1999), 10464–72.

[24] Bi, G. Q., and Poo, M. M. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci. 18*, 24 (Dec. 1998), 10464–10472.

[25] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. YOLOv4: Optimal speed and accuracy of object detection.

[26] Bolsee, Q., and Munteanu, A. Cnn-based denoising of time-of-flight depth images. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (Oct. 2018), IEEE.

[27] Brandli, C., Berner, R., Yang, M., Liu, S.-C., and Delbruck, T. A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor. *IEEE J. Solid-State Circuits 49*, 10 (Oct. 2014), 2333–2341.

[28] Brette, R. Philosophy of the spike: Rate-Based vs. Spike-Based theories of the brain. *Frontiers in Systems Neuroscience 9* (2015).

[29] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

[30] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot. 32*, 6 (Dec. 2016), 1309–1332.

[31] Catal, O., Jansen, W., Verbelen, T., Dhoedt, B., and Steckel, J. LatentSLAM: unsupervised multi-sensor representation learning for localization and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (May 2021), IEEE.

[32] Catal, O., Verbelen, T., Wang, N., Hartmann, M., and Dhoedt, B. Bio-inspired monocular drone slam. In *System Engineering for Constrained Embedded Systems* (New York, NY, USA, 2022), DroneSE and RAPIDO, Association for Computing Machinery, p. 21–26.

[33] Cauwenberghs, G., and Poggio, T. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems* (2000), T. Leen, T. Dietterich, and V. Tresp, Eds., vol. 13, MIT Press.

[34] Chan, T. F., Gene, H., and Golub, R. J. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician 37*, 3 (1983), 242–247.

[35] Charlotte, Frenkel, J.-D., and Legat, D. *A 28-nm Convolutional Neuromorphic Processor Enabling Online Learning with Spike-Based Retinas*. 2020.

[36] Chen, J.-Y., Lonjers, P., Lee, C., Chistiakova, M., Volgushev, M., and Bazhenov, M. Heterosynaptic plasticity prevents runaway synaptic dynamics. *Journal of Neuroscience 33*, 40 (2013), 15915–15929.

[37] Chen, Y.-H., Krishna, T., Emer, J. S., and Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-State Circuits 52*, 1 (Jan. 2017), 127–138.

[38] Chen, Y.-H., Yang, T.-J., Emer, J., and Sze, V. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9*, 2 (2019), 292–308.

[39] Choe, Y. Anti-Hebbian learning. In *Encyclopedia of Computational Neuroscience*, D. Jaeger and R. Jung, Eds. Springer, New York, NY, 2014.

[40] Corradi, F., Adriaans, G., and Stuijk, S. Gyro: A digital spiking neural network architecture for multi-sensory data analytics. In *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings* (New York, NY, USA, 2021), DroneSE and RAPIDO '21, Association for Computing Machinery, p. 9–15.

[41] Covi, E., George, R., Frascaroli, J., Brivio, S., Mayr, C., Mostafa, H., Indiveri, G., and Spiga, S. Spike-driven threshold-based learning with memristive synapses and neuromorphic silicon neurons. *Journal of Physics D: Applied Physics 51*, 34 (jul 2018), 344003.

[42] Das, A., Pradhapan, P., Groenendaal, W., Adiraju, P., Rajan, R. T., Catthoor, F., Schaafsma, S., Krichmar, J. L., Dutt, N., and Van Hoof, C. Unsupervised heart-rate estimation in wearables with liquid states and a probabilistic readout. *Neural Networks 99* (2018), 134–147.

[43] Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math. 57*, 11 (Nov. 2004), 1413–1457.

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

[45] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 7 (2022), 3366–3385.

[46] De Lange, M., and Tuytelaars, T. Continual prototype evolution: Learning online from non-stationary data streams. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (Oct. 2021), IEEE.

[47] Delmerico, J., Cieslewski, T., Rebecq, H., Faessler, M., and Scaramuzza, D. Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset. In *2019 International Conference on Robotics and Automation (ICRA)* (May 2019), IEEE.

[48] Dietsche, A., Cioffi, G., Hidalgo-Carrió, J., and Scaramuzza, D. Powerline tracking with event cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), pp. 6990–6997.

[49] Doutsi, E., Fillatre, L., Antonini, M., and Tsakalides, P. Dynamic image quantization using leaky integrate-and-fire neurons. *IEEE Trans. Image Process. 30* (Apr. 2021), 4305–4315.

[50] Dupeyroux, J., Hagenaars, J. J., Paredes-Vallés, F., and de Croon, G. C. H. E. Neuromorphic control for optic-flow-based landing of mavs using the loihi processor. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), pp. 96–102.

[51] Dzvonkovskaya, A. L., and Rohling, H. Target detection with adaptive power regression thresholding for HF radar. In *2006 CIE International Conference on Radar* (Oct. 2006), IEEE.

[52] Eigen, D., and Fergus, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec. 2015), IEEE.

[53] Eliasmith, C., and Anderson, C. *Neural Engineering: Computation, Representation and Dynamics in Neurobiological Systems*. MIT press, 2003.

[54] Eshraghian, J. K., Ward, M., Neftci, E. O., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D. S., and Lu, W. D. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE 111*, 9 (2023), 1016–1054.

[55] Espino, H., Bain, R., and Krichmar, J. L. Selective memory replay improves exploration in a spiking wavefront planner. In *2023 International Joint Conference on Neural Networks (IJCNN)* (2023), pp. 1–8.

[56] Ester, M., Kriegel, H. P., Sander, J., and Xu, X. A Density-Based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[57] Falanga, D., Kleber, K., and Scaramuzza, D. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics 5*, 40 (2020), eaaz9712.

[58] Falez, P., Tirilly, P., Bilasco, I., Devienne, P., and Boulet, P. *Unsupervised Visual Feature Learning with Spike-timing-dependent Plasticity: How Far are we from Traditional Feature Learning Approaches?* 2019.

[59] Falez, P., Tirilly, P., Marius Bilasco, I., Devienne, P., and Boulet, P. Multi-layered spiking neural network with target timestamp threshold adaptation and STDP. In *2019 International Joint Conference on Neural Networks (IJCNN)* (July 2019), IEEE.

[60] Fang, M. Y.-S., Mudigonda, M., Zarcone, R., Khosrowshahi, A., and Olshausen, B. A. Learning and Inference in Sparse Coding Models With Langevin Dynamics. *Neural Computation 34*, 8 (07 2022), 1676–1700.

[61] Fang, W., Wang, L., and Ren, P. Tinier-YOLO: A real-time object detection method for constrained environments. *IEEE Access 8* (2020), 1935–1944.

[62] Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Trans. Intell. Transp. Syst. 22*, 3 (Mar. 2021), 1341–1360.

[63] Forno, E., Fra, V., Pignari, R., Macii, E., and Urgese, G. Spike encoding techniques for iot time-varying signals benchmarked on a neuromorphic classification task. *Frontiers in Neuroscience 16* (2022).

[64] Fountas, Z., Sajid, N., Mediano, P., and Friston, K. Deep active inference agents using monte-carlo methods.

[65] Frenkel, C. *Bottom-up and top-down neuromorphic processor design: Unveiling roads to embedded cognition.* PhD thesis, 01 2020.

[66] Frenkel, C., Bol, D., and Indiveri, G. Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence. *Proceedings of the IEEE 111*, 6 (2023), 623–652.

[67] Frenkel, C., and Indiveri, G. Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 1–3.

[68] Frenkel, C., Lefebvre, M., and Bol, D. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in Neuroscience 15* (2021).

[69] Frenkel, C., Lefebvre, M., Legat, J.-D., and Bol, D. A 0.086-mm ^{2} 12.7-pJ/SOP 64k-synapse 256-neuron Online-Learning digital spiking neuromorphic processor in 28nm CMOS. *IEEE Trans. Biomed. Circuits Syst.* (2018), 1–1.

[70] Frenkel, C., Lefebvre, M., Legat, J.-D., and Bol, D. A 0.086-mm ^{2} 12.7-pJ/SOP 64k-synapse 256-neuron Online-Learning digital spiking neuromorphic processor in 28nm CMOS. *IEEE Trans. Biomed. Circuits Syst.* (2018), 1–1.

[71] Friston, K. Friston, k.j.: The free-energy principle: a unified brain theory? nat. rev. neurosci. 11, 127-138. *Nature reviews. Neuroscience 11* (02 2010), 127–38.

[72] Friston, K. Does predictive coding have a future? *Nature Neuroscience 21* (07 2018).

[73] Friston, K., and Kiebel, S. Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences 364* (05 2009), 1211–21.

[74] Friston, K. J. Hierarchical models in the brain. *PLoS Computational Biology 4* (2008).

[75] Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., and Scaramuzza, D. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell. 44*, 1 (Jan. 2022), 154–180.

[76] Galluppi, F., Conradt, J., Stewart, T., Eliasmith, C., Horiuchi, T., Tapson, J., Tripp, B., Furber, S., and Etienne-Cummings, R. Live demo: Spiking ratslam: Rat hippocampus cells in spiking neural hardware. In *2012 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (2012), pp. 91–91.

[77] Gao, X., Xing, G., Roy, S., and Liu, H. RAMP-CNN: A novel neural network for enhanced automotive radar object recognition. *IEEE Sens. J. 21*, 4 (Feb. 2021), 5119–5132.

[78] George, A. M., Banerjee, D., Dey, S., Mukherjee, A., and Balamurali, P. A reservoir-based convolutional spiking neural network for gesture recognition from DVS input. In *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020), IEEE.

[79] Georgopoulos, A. P., Taira, M., and Lukashin, A. Cognitive neurophysiology of the motor cortex. *Science 260*, 5104 (Apr. 1993), 47–52.

[80] Gerhards, P., Kreutz, F., Knobloch, K., and Mayr, C. G. Radar-based gesture recognition with spiking neural networks. In *2022 7th International Conference on Frontiers of Signal Processing (ICFSP)* (2022), pp. 40–44.

[81] Ghosh, S., and Sen, D. An inclusive survey on array antenna design for millimeter-wave communications. *IEEE Access 7* (2019), 83137–83161.

[82] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track 9* (2010), 249–256.

[83] Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. *In Journal of Machine Learning Research* (2010).

[84] Grisetti, G., Kümmerle, R., Stachniss, C., and Burgard, W. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine 2*, 4 (2010), 31–43.

[85] Gupta, A., and Saurabh, S. On-chip unsupervised learning using stdp in a spiking neural network. *IEEE Transactions on Nanotechnology 22* (2023), 365–376.

[86] Hamdioui, S., Kvatinsky, S., Cauwenberghs, G., Xie, L., Wald, N., Joshi, S., Elsayed, H. M., Corporaal, H., and Bertels, K. Memristor for computing: Myth or reality? In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017* (2017), pp. 722–731.

[87] Harris, F. J. On the use of windows for harmonic analysis with the discrete fourier transform. *Proc. IEEE Inst. Electr. Electron. Eng. 66*, 1 (1978), 51–83.

[88] Harris, F. J. On the use of windows for harmonic analysis with the discrete fourier transform. *Proc. IEEE Inst. Electr. Electron. Eng. 66*, 1 (1978), 51–83.

[89] Hershey, J. R., and Olsen, P. A. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07* (2007), vol. 4, pp. IV–317–IV–320.

[90] Hidalgo-Carrio, J., Gehrig, D., and Scaramuzza, D. Learning monocular dense depth from events. In *2020 International Conference on 3D Vision (3DV)* (Nov. 2020), IEEE.

[91] Hines, A. D., Stratton, P. G., Milford, M., and Fischer, T. Vprtempo: A fast temporally encoded spiking neural network for visual place recognition, 2023.

[92] Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (Feb. 2014), IEEE.

[93] Houshmand, P., Sarda, G. M., Jain, V., Ueyoshi, K., Papistas, I. A., Shi, M., Zheng, Q., Bhattacharjee, D., Mallik, A., Debacker, P., Verkest, D., and Verhelst, M. Diana: An end-to-end hybrid digital and analog neural network soc for the edge. *IEEE Journal of Solid-State Circuits 58*, 1 (2023), 203–215.

[94] Hu, T., Pehlevan, C., and Chklovskii, D. B. A Hebbian/Anti-Hebbian network for online sparse dictionary learning derived from symmetric matrix factorization. In *2014 48th Asilomar Conference on Signals, Systems and Computers* (Nov. 2014), IEEE.

[95] Hu, Y., Liu, H., Pfeiffer, M., and Delbruck, T. DVS benchmark datasets for object tracking, action recognition, and object recognition. *Front. Neurosci. 10* (Aug. 2016), 405.

[96] Hubel, D. H., and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol. 160*, 1 (Jan. 1962), 106–154.

[97] Hunsberger, E., and Eliasmith, C. Spiking deep networks with LIF neurons.

[98] Hussaini, S., Milford, M., and Fischer, T. Ensembles of compact, region-specific & regularized spiking neural networks for scalable place recognition. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), pp. 4200–4207.

[99] Hyun, E., and Lee, J. A new OS-CFAR detector design. In *textit2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering.* Jeju Island, 2011, pp. 133–136.

[100] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.

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

[102] Ioffe, S. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 448–456.

[103] Isomura, T., Shimazaki, H., and Friston, K. Canonical neural networks perform active inference. *Communications Biology 5* (01 2022).

[104] Iyer, L. R., and Chua, Y. Classifying neuromorphic datasets with tempotron and spike timing dependent plasticity. In *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020), IEEE.

[105] Jaeger, H., Doorakkers, D., Lawrence, C., and Indiveri, G. Dimensions of timescales in neuromorphic computing systems. *CoRR abs/2102.10648* (2021).

[106] Jain, V., and Verhelst, M. *Scoping the Landscape of (Extreme) Edge Machine Learning Processors*. Springer Nature Switzerland, Cham, 2024, pp. 43–57.

[107] Jeong, N., Hwang, H., and Matson, E. T. Evaluation of low-cost LiDAR sensor for application in indoor UAV navigation. In *2018 IEEE Sensors Applications Symposium (SAS)* (Mar. 2018), IEEE.

[108] Jin, B., Milling, M., Plaza, M. P., Brunner, J. O., Traidl-Hoffmann, C., Schuller, B. W., and Damialis, A. Airborne pollen grain detection from partially labelled data utilising semi-supervised learning. *Sci. Total Environ. 891*, 164295 (Sept. 2023), 164295.

[109] Kaiser, J., Mostafa, H., and Neftci, E. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci. 14* (May 2020), 424.

[110] Kaiser, J., Mostafa, H., and Neftci, E. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci. 14* (May 2020), 424.

[111] Karami, E., Prasad, S., and Shehata, M. S. Image matching using sift, surf, BRIEF and ORB: performance comparison for distorted images. *CoRR abs/1710.02726* (2017).

[112] Keuninckx, L., Danckaert, J., and Van der Sande, G. Real-time audio processing with a cascade of discrete-time delay line-based reservoir computers. *Cognitive Computation 9* (06 2017).

[113] Khacef, L., Klein, P., Cartiglia, M., Rubino, A., Indiveri, G., and Chicca, E. Spike-based local synaptic plasticity: A survey of computational models and neuromorphic circuits. *Neuromorphic Computing and Engineering* (2023).

[114] Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks 99* (2018), 56–67.

[115] Kim, E., Lawson, E., Sullivan, K., and Kenyon, G. T. Spatiotemporal sequence memory for prediction using deep sparse coding. In *Proceedings of the 7th Annual Neuro-Inspired Computational Elements Workshop* (New York, NY, USA, 2019), NICE '19, Association for Computing Machinery.

[116] Kim, S., Kim, S., Hong, S., Kim, S., Han, D., and Yoo, H.-J. C-dnn: A 24.5-85.8tops/w complementary-deep-neural-network processor with heterogeneous cnn/snn core architecture and forward-gradient-based sparsity generation. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 334–336.

[117] Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: Spiking neural network for energy-efficient object detection. *Proceedings of the AAAI Conference on Artificial Intelligence 34*, 07 (Apr. 2020), 11270–11277.

[118] Kim, W., Cho, H., Kim, J., Kim, B., and Lee, S. YOLO-based simultaneous target detection and classification in automotive FMCW radar systems. *Sensors (Basel) 20*, 10 (May 2020), 2897.

[119] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*. 2014.

[120] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization.

[121] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences 114* (12 2016).

[122] Kostadinov, D., and Scaramuzza, D. Unsupervised feature learning for event data: Direct vs inverse problem formulation. In *2020 25th*

International Conference on Pattern Recognition (ICPR) (Jan. 2021), IEEE.

[123] Krim, H., and Viberg, M. Two decades of array signal processing research: the parametric approach. *IEEE Signal Process. Mag. 13*, 4 (July 1996), 67–94.

[124] Krotov, D., and Hopfield, J. J. Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences 116*, 16 (2019), 7723–7731.

[125] Kuznietsov, Y., Stuckler, J., and Leibe, B. Semi-supervised deep learning for monocular depth map prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), IEEE.

[126] Kwon, H., and Nasrabadi, N. M. Kernel RX-algorithm: a nonlinear anomaly detector for hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens. 43*, 2 (Feb. 2005), 388–397.

[127] Kwon, H., and Nasrabadi, N. M. A comparative analysis of kernel subspace target detectors for hyperspectral imagery. *EURASIP J. Adv. Signal Process. 2007*, 1 (Dec. 2006).

[128] Lagorce, X., Orchard, G., Galluppi, F., Shi, B. E., and Benosman, R. B. HOTS: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell. 39*, 7 (July 2017), 1346–1359.

[129] Lange, M. D. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44* (2022), 3366–3385.

[130] Lazar, A., Pipa, G., and Triesch, J. Sorn: a self-organizing recurrent neural network. *Frontiers in Computational Neuroscience 3* (2009).

[131] Lee, C., Panda, P., Srinivasan, G., and Roy, K. Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning. *Frontiers in Neuroscience 12* (2018).

[132] Lee, H., Battle, A., Raina, R., and Ng, A. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems* (2006), B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19, MIT Press.

[133] Li, S., Bi, X., Huang, L., and Tan, B. 2-D CFAR procedure of multiple target detection for automotive radar. *SAE Int. J. Passeng. Cars Electron. Electr. Syst. 11*, 1 (Sept. 2017).

[134] Liang, Y., Ryali, C. K., Hoover, B., Grinberg, L., Navlakha, S., Zaki, M. J., and Krotov, D. Can a fruit fly learn word embeddings?, 2021.

[135] Lien, J., Gillian, N., Karagozler, M., Amihood, P., Schwesig, C., Olson, E., Raja, H., and Poupyrev, I. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Trans. Graph 35*, 4 (2016).

[136] Lin, J.-J., Li, Y.-P., Hsu, W.-C., and Lee, T.-S. Design of an FMCW radar baseband signal processing system for automotive application. *Springerplus 5*, 1 (Dec. 2016).

[137] Lin, J.-Y. *Workload Dependent Mitigation Approaches for Performance Variability: Ensuring Timing Guarantees of Integrated Circuits.* PhD thesis, 01 2024.

[138] Lin, J. Y., Weckx, P., Mishra, S., Spessot, A., and Catthoor, F. Proactive run-time mitigation for time-critical applications using dynamic scenario methodology. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2022), pp. 616–621.

[139] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell. 42*, 2 (Feb. 2020), 318–327.

[140] Liu, A. SSD: Single shot MultiBox detector. In *Computer Vision - ECCV 2016*. Springer International Publishing, 2016, pp. 21–37.

[141] Liu, C., and Szirányi, T. Real-time human detection and gesture recognition for on-board uav rescue. *Sensors 21*, 6 (2021).

[142] Liu, Q., Pan, G., Ruan, H., Xing, D., Xu, Q., and Tang, H. Unsupervised AER object recognition based on multiscale spatio-temporal features and spiking neurons. *IEEE Trans. Neural Netw. Learn. Syst. 31*, 12 (Dec. 2020), 5300–5311.

[143] Liu, Y.-H., Sheelavant, S., Mercuri, M., Mateman, P., Dijkhuis, J., Zomagboguelou, W., Breeschoten, A., Traferro, S., Zhan, Y., Torf, T., Bachmann, C., Harpe, P., and Babaie, M. 9.3 A680 μW burst-chirp UWB radar transceiver for vital signs and occupancy sensing up to 15m distance. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)* (Feb. 2019), IEEE.

[144] Liu, Y.-H., Sheelavant, S., Mercuri, M., Mateman, P., Dijkhuis, J., Zomagboguelou, W., Breeschoten, A., Traferro, S., Zhan, Y., Torf, T., Bachmann, C., Harpe, P., and Babaie, M. 9.3 A680

μW burst-chirp UWB radar transceiver for vital signs and occupancy sensing up to 15m distance. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)* (Feb. 2019), IEEE.

[145] Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics 36*, 1 (2020), 1–14.

[146] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110.

[147] Loyez, C., Carpentier, K., Sourikopoulos, I., and Danneville, F. Subthreshold neuromorphic devices for spiking neural networks applied to embedded a.i. In *2021 19th IEEE International New Circuits and Systems Conference (NEWCAS)* (2021), pp. 1–4.

[148] Lubecke, V. M., Boric-Lubecke, O., Host-Madsen, A., and Fathy, A. E. Through-the-wall radar life detection and monitoring. In *2007 IEEE/MTT-S International Microwave Symposium* (June 2007), IEEE.

[149] Luo, T. Z., and Maunsell, J. H. R. Attention can be subdivided into neurobiological components corresponding to distinct behavioral effects. *Proceedings of the National Academy of Sciences 116*, 52 (2019), 26187–26194.

[150] Maass, W. Liquid state machines: Motivation, theory, and applications.

[151] Maass, W., Natschläger, T., and Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput. 14*, 11 (nov 2002), 2531–2560.

[152] MacKay, D. J. C. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[153] Mark, A. *Fundamentals of Radar Signal Processing*. McGraw-Hill Professional, 2005.

[154] Mark, A. *Fundamentals of Radar Signal Processing*. McGraw-Hill Professional, 2005.

[155] Mark, A. *Fundamentals of Radar Signal Processing*. McGraw-Hill Professional, 2005.

[156] Maro, J. M., Ieng, S. H., and Benosman, R. Event-Based gesture recognition with dynamic background suppression using smartphone computational capabilities. *Frontiers in Neuroscience 14* (2020).

[157] Massa, R., Marchisio, A., Martina, M., and Shafique, M. An efficient spiking neural network for recognizing gestures with a DVS camera on the loihi neuromorphic processor. In *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020), IEEE.

[158] Mazzaglia, P., Verbelen, T., Çatal, O., and Dhoedt, B. The free energy principle for perception and action: A deep learning perspective. *Entropy 24*, 2 (2022).

[159] Milford, M. J., Wyeth, G. F., and Prasser, D. RatSLAM: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004* (2004), IEEE.

[160] Mitrokhin, A., Fermuller, C., Parameshwara, C., and Aloimonos, Y. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct. 2018), IEEE.

[161] Montangie, L., Miehl, C., and Gjorgjieva, J. Autonomous emergence of connectivity assemblies via spike triplet interactions. *PLoS Comput. Biol. 16*, 5 (May 2020), e1007835.

[162] Moons, B., Bankman, D., Yang, L., Murmann, B., and Verhelst, M. Binareye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm cmos. In *2018 IEEE Custom Integrated Circuits Conference (CICC)* (2018), pp. 1–4.

[163] Moons, B., Uytterhoeven, R., Dehaene, W., and Verhelst, M. 14.5 envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)* (Feb. 2017), IEEE.

[164] Moons, B., and Verhelst, M. An energy-efficient precision-scalable convnet processor in 40-nm cmos. *IEEE Journal of Solid-State Circuits 52*, 4 (2017), 903–914.

[165] Moradi, S., Qiao, N., Stefanini, F., and Indiveri, G. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Trans. Biomed. Circuits Syst. 12*, 1 (Feb. 2018), 106–122.

[166] Mozafari, M., Kheradpisheh, S. R., Masquelier, T., Nowzari-Dalini, A., and Ganjtabesh, M. First-spike-based visual categorization using reward-modulated stdp. *IEEE Transactions on Neural Networks and Learning Systems 29*, 12 (2018), 6178–6190.

[167] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot. 31*, 5 (Oct. 2015), 1147–1163.

[168] Nabati, R., and Qi, H. Radar-camera sensor fusion for joint object detection and distance estimation in autonomous vehicles.

[169] Naidoo, Y., Stopforth, R., and Bright, G. Development of an UAV for search & rescue applications. In *IEEE Africon '11* (Sept. 2011), IEEE.

[170] Nasrabadi, N. M. Hyperspectral target detection : An overview of current and future challenges. *IEEE Signal Process. Mag. 31*, 1 (Jan. 2014), 34–44.

[171] Nazaré, M. Deep convolutional neural networks and noisy images. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing, 2018, pp. 416–424.

[172] Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience 7* (2014).

[173] Neftci, E. O., Mostafa, H., and Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag. 36*, 6 (Nov. 2019), 51–63.

[174] Nobis, F., Geisslinger, M., Weber, M., Betz, J., and Lienkamp, M. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)* (Oct. 2019), IEEE.

[175] Noureldin, A., Karamat, T., and Georgy, J. *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. 2013.

[176] Olshausen, B., and Field, D. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381* (07 1996), 607–9.

[177] Olshausen, B. A., and Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research 37*, 23 (1997), 3311–3325.

[178] Orchard, G., Jayawant, A., Cohen, G., and Thakor, N. Converting static image datasets to spiking neuromorphic datasets using saccades.

[179] Ororbia, A. Spiking neural predictive coding for continually learning from data streams. *Neurocomputing 544* (2023), 126292.

[180] Ororbia, A., and Mali, A. Backprop-free reinforcement learning with active neural generative coding, 2021.

[181] Ororbia, A., and Mali, A. Active predictive coding: Brain-inspired reinforcement learning for sparse reward robotic control problems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), pp. 3015–3021.

[182] Ott, J., Linstead, E., LaHaye, N., and Baldi, P. Learning in the machine: To share or not to share? *Neural Networks 126* (2020), 235–249.

[183] Ouaknine, A., Newson, A., Rebut, J., Tupin, F., and Pérez, P. CARRADA dataset: Camera and automotive radar with range-angle-doppler annotations.

[184] Pan, X. Expressiveness of rectifier networks. *CoRR* (2015).

[185] Papyan, V., Romano, Y., Elad, M., and Sulam, J. Convolutional dictionary learning via local processing. In *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017), IEEE.

[186] Park, S., Kankuppe, A., Renukaswamy, P., Guermandi, D., Visweswaran, A., Garcia, J. C., Sinha, S., Wambacq, P., and Craninckx, J. A 62mw 60GHz FMCW radar in 28nm CMOS. In *2020 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)* (Aug. 2020), IEEE.

[187] Parr, T., Pezzulo, G., and Friston, K. J. *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press, 03 2022.

[188] Partzsch, J., Hoppner, S., Eberlein, M., Schuffny, R., Mayr, C., Lester, D. R., and Furber, S. A fixed point exponential function accelerator for a neuromorphic many-core system. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (May 2017), IEEE.

[189] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff* (2017).

[190] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[191] Pearson, J. B., and Stear, E. B. Kalman filter applications in airborne radar tracking. *IEEE Transactions on Aerospace and Electronic Systems AES-10*, 3 (1974), 319–329.

[192] Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., and Indiveri, G. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience 9* (2015).

[193] Rajendran, B., Sebastian, A., Schmuker, M., Srinivasa, N., and Eleftheriou, E. Low-power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches. *IEEE Signal Process. Mag. 36*, 6 (Nov. 2019), 97–110.

[194] Ramesh, B., Yang, H., Orchard, G. M., Le Thi, N. A., Zhang, S., and Xiang, C. DART: Distribution aware retinal transform for event-based cameras. *IEEE Trans. Pattern Anal. Mach. Intell.* (2019), 1–1.

[195] Rao, R., and Ballard, D. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience 2* (02 1999), 79–87.

[196] Rebecq, H., Gallego, G., Mueggler, E., and Scaramuzza, D. Emvs: Event-based multi-view stereo–3d reconstruction with an event camera in real-time. *Int. J. Comput. Vision 126*, 12 (dec 2018), 1394–1414.

[197] Redmon, J., and Farhadi, A. YOLOv3: An incremental improvement.

[198] Reed, I. S., and Yu, X. Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution. *IEEE Trans. Acoust. 38*, 10 (1990), 1760–1770.

[199] Ridolfi, M., Macoir, N., Gerwen, J. V.-V., Rossey, J., Hoebeke, J., and de Poorter, E. Testbed for warehouse automation experiments using mobile AGVs and drones. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (Apr. 2019), IEEE.

[200] Ritchie, M., Capraru, R., and Fioranelli, F. Dop-net: a micro-doppler radar data challenge. *Electronics Letters 56*, 11 (2020), 568–570.

[201] Rohan Ghosh, A., and Gupta, A. Spatiotemporal filtering for Event-Based action recognition. *Alcimar Soares, Nitish Thakor* (2019).

[202] Rohling, H., and Mende, R. OS CFAR performance in a 77 GHz radar sensor for car application. In *Proceedings of International Radar Conference* (2002), Publishing House of Electron. Ind.

[203] Rohman, B. P. A., Kurniawan, D., and Miftahushudur, M. T. Switching CA/OS CFAR using neural network for radar target detection in non-homogeneous environment. In *2015 International Electronics Symposium (IES)* (Sept. 2015), IEEE.

[204] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision* (2011), pp. 2564–2571.

[205] Rueckauer, B., Lungu, I. A., Hu, Y., Pfeiffer, M., and Liu, S. C. Conversion of Continuous-Valued deep networks to efficient Event-Driven networks for image classification. *Frontiers in Neuroscience 11* (2017).

[206] Rueckauer, B., Lungu, I. A., Hu, Y., Pfeiffer, M., and Liu, S. C. Conversion of Continuous-Valued deep networks to efficient Event-Driven networks for image classification. *Frontiers in Neuroscience 11* (2017).

[207] Safa, A., Bourdoux, A., Ocket, I., Catthoor, F., and Gielen, G. G. E. On the use of spiking neural networks for ultralow-power radar gesture recognition. *IEEE Microw. Wirel. Compon. Lett. 32*, 3 (Mar. 2022), 222–225.

[208] Safa, A., Jaltare, V., Sebt, S., Gano, K., Leugering, J., Gielen, G., and Cauwenberghs, G. Towards chip-in-the-loop spiking neural network training via metropolis-hastings sampling, 2024.

[209] Safa, A., Ocket, I., Bourdoux, A., Sahli, H., Catthoor, F., and Gielen, G. G. Event camera data classification using spiking networks with spike-timing-dependent plasticity. In *2022 International Joint Conference on Neural Networks (IJCNN)* (2022), pp. 1–8.

[210] Safa, A., Ocket, I., Bourdoux, A., Sahli, H., Catthoor, F., and Gielen, G. G. Stdp-driven development of attention-based people detection in spiking neural networks. *IEEE Transactions on Cognitive and Developmental Systems* (2022), 1–1.

[211] Safa, A., Ocket, I., Catthoor, F., and Gielen, G. Supporthdc: Hyperdimensional computing with scalable hypervector sparsity. In *Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference* (New York, NY, USA, 2023), NICE '23, Association for Computing Machinery, p. 20–25.

[212] Safa, A., Verbelen, T., Keuninckx, L., Ocket, I., Hartmann, M., Bourdoux, A., Catthoor, F., and Gielen, G. G. E. A low-complexity radar detector outperforming OS-CFAR for indoor drone obstacle avoidance. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 14* (2021), 9162–9175.

[213] Safa, A., Verbelen, T., Ocket, I., Bourdoux, A., Catthoor, F., and Gielen, G. G. E. Fail-safe human detection for drones using a multi-modal curriculum learning approach. *IEEE Robot. Autom. Lett. 7*, 1 (Jan. 2022), 303–310.

[214] Safa, A., Verbelen, T., Ocket, I., Bourdoux, A., Sahli, H., Catthoor, F., and Gielen, G. Fusing event-based camera and radar for slam using spiking neural networks with continual stdp learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), pp. 2782–2788.

[215] Safa, A., Verbelen, T., Çatal, O., Van de Maele, T., Hartmann, M., Dhoedt, B., and Bourdoux, A. Fmcw radar sensing for indoor drones using variational auto-encoders. In *2023 IEEE Radar Conference (RadarConf23)* (2023), pp. 1–6.

[216] Scannapieco, A. F., Renga, A., Fasano, G., and Moccia, A. Ultralight radar for small and micro-uav navigation. *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. XLII-2/W6* (Aug. 2017), 333–338.

[217] Schölkopf, B., and Smola, A. J. *Learning with kernels : support vector machines, regularization, optimization, and beyond.* Adaptive computation and machine learning. MIT Press, 2002.

[218] Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., and Platt, J. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*. MIT Press, 1999.

[219] Schrauwen, B., Verstraeten, D., and Campenhout, J. M. V. An overview of reservoir computing: theory, applications and implementations. In *The European Symposium on Artificial Neural Networks* (2007).

[220] Schwartenbeck, P., FitzGerald, T., Dolan, R., and Friston, K. Exploration, novelty, surprise and free energy minimisation. *Frontiers in psychology 4* (10 2013), 710.

[221] Seshadri, K., Akin, B., Laudon, J., Narayanaswami, R., and Yazdanbakhsh, A. An evaluation of edge TPU accelerators for convolutional neural networks. In *2022 IEEE International Symposium on Workload Characterization (IISWC)* (Nov. 2022), IEEE.

[222] Shouval, H., Wang, S., and Wittenberg, G. Spike timing dependent plasticity: A consequence of more fundamental learning rules. *Frontiers in Computational Neuroscience 4* (2010).

[223] Shrestha, A., Fang, H., Mei, Z., Rider, D. P., Wu, Q., and Qiu, Q. A survey on neuromorphic computing: Models and hardware. *IEEE Circuits and Systems Magazine 22*, 2 (2022), 6–35.

[224] Shrestha, S. B., and Orchard, G. SLAYER: Spike layer error reassignment in time. In *Neural Information Processing Systems*. Montreal, Canada, 2018.

[225] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., and Benosman, R. HATS: Histograms of averaged time surfaces for robust event-based object classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (June 2018), IEEE.

[226] Strang, G. *Introduction to Linear Algebra*, fourth ed. Wellesley-Cambridge Press, Wellesley, MA, 2009.

[227] Stuijt, J., Sifalakis, M., Yousefzadeh, A., and Corradi, F. *mu*Brain: An Event-Driven and fully synthesizable architecture for spiking neural networks. *Frontiers in Neuroscience 15* (2021).

[228] Su, Q., Chou, Y., Hu, Y., Li, J., Mei, S., Zhang, Z., and Li, G. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 6555–6565.

[229] Suh, J. S., Ryu, S., Han, B., Choi, J., Kim, J.-H., and Hong, S. 24 GHz FMCW radar system for real-time hand gesture recognition using LSTM. In *2018 Asia-Pacific Microwave Conference (APMC)* (Nov. 2018), IEEE.

[230] Suleiman, A., Zhang, Z., Carlone, L., Karaman, S., and Sze, V. Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE J. Solid-State Circuits 54*, 4 (Apr. 2019), 1106–1119.

[231] Sun, Y., Fei, T., Li, X., Warnecke, A., Warsitz, E., and Pohl, N. Real-time radar-based gesture detection and recognition built in an edge-computing platform. *IEEE Sens. J. 20*, 18 (Sept. 2020), 10706–10716.

[232] Sun, Y., Fei, T., Li, X., Warnecke, A., Warsitz, E., and Pohl, N. Real-time radar-based gesture detection and recognition built in an edge-computing platform. *IEEE Sens. J. 20*, 18 (Sept. 2020), 10706–10716.

[233] Surmann, H., Slomma, D., Grobelny, S., and Grafe, R. Deployment of aerial robots after a major fire of an industrial hall with hazardous substances, a report. In *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (Oct. 2021), IEEE.

[234] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[235] Tang, G., and Michmizos, K. P. Gridbot: An autonomous robot controlled by a spiking neural network mimicking the brain's navigational system. In *Proceedings of the International Conference on Neuromorphic Systems* (New York, NY, USA, 2018), ICONS '18, Association for Computing Machinery.

[236] Tang, G., Shah, A., and Michmizos, K. P. Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), pp. 4176–4181.

[237] Tao, D., Anfinsen, S. N., and Brekke, C. Robust CFAR detector based on truncated statistics in multiple-target situations. *IEEE Trans. Geosci. Remote Sens. 54*, 1 (Jan. 2016), 117–134.

[238] Tariq, R., Rahim, M., Aslam, N., Bawany, N., and Faseeha, U. Dronaid : A smart human detection drone for rescue. In *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT and IoT (HONET-ICT)* (2018), pp. 33–37.

[239] Tavanaei, A., and Maida, A. Bp-stdp: Approximating backpropagation using spike timing dependent plasticity. *Neurocomputing 330* (2019), 39–47.

[240] Tiemann, J., and Wietfeld, C. Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization. In *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (Sept. 2017), IEEE.

[241] Tostado, P. M., Pedroni, B. U., and Cauwenberghs, G. Performance trade-offs in weight quantization for memory-efficient inference. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (2019), pp. 246–250.

[242] Tsang, I. J., Corradi, F., Sifalakis, M., Van Leekwijck, W., and Latré, S. Radar-based hand gesture recognition using spiking neural networks. *Electronics 10*, 12 (2021).

[243] Tulyakov, S., Gehrig, D., Georgoulis, S., Erbach, J., Gehrig, M., Li, Y., and Scaramuzza, D. TimeLens: Event-based video frame interpolation. *IEEE Conference on Computer Vision and Pattern Recognition* (2021).

[244] Ueltzhöffer, K. Deep active inference. *Biological Cybernetics 112*, 6 (Oct. 2018), 547–573.

[245] Van de Maele, T., Verbelen, T., Çatal, O., De Boom, C., and Dhoedt, B. Active vision for robot manipulators using the free energy principle. *Frontiers in Neurorobotics 15* (2021).

[246] Van Etten, A. Satellite imagery multiscale rapid detection with windowed networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (Jan. 2019), IEEE.

[247] Vandersmissen, B., Knudde, N., Jalalvand, A., Couckuyt, I., Bourdoux, A., De Neve, W., and Dhaene, T. Indoor person identification using a low-power FMCW radar. *IEEE Trans. Geosci. Remote Sens. 56*, 7 (July 2018), 3941–3952.

[248] Vandersmissen, B., Knudde, N., Jalalvand, A., Couckuyt, I., Bourdoux, A., De Neve, W., and Dhaene, T. Indoor person identification using a low-power FMCW radar. *IEEE Trans. Geosci. Remote Sens. 56*, 7 (July 2018), 3941–3952.

[249] Verhelst, M., and Moons, B. Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices. *IEEE Solid-State Circuits Magazine 9*, 4 (2017), 55–65.

[250] Victor, C. *Radar Micro-Doppler Signatures: Processing and Applications*. Institution of Engineering and Technology, 2014.

[251] Vidal, A. R., Rebecq, H., Horstschaefer, T., and Scaramuzza, D. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters 3*, 2 (2018), 994–1001.

[252] Vigneron, A., and Martinet, J. A critical survey of STDP in spiking neural networks for pattern recognition. In *2020 International Joint Conference on Neural Networks (IJCNN)* (July 2020), IEEE.

[253] Visweswaran, A., Vaesen, K., Sinha, S., Ocket, I., Glassee, M., Desset, C., Bourdoux, A., and Wambacq, P. 9.4 a 145GHz FMCW-Radar transceiver in 28nm CMOS. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)* (Feb. 2019), IEEE.

[254] Vitale, A., Renner, A., Nauer, C., Scaramuzza, D., and Sandamirskaya, Y. Event-driven vision and control for uavs on a neuromorphic chip. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), pp. 103–109.

[255] Vödisch, N., Cattaneo, D., Burgard, W., and Valada, A. *Continual SLAM: Beyond Lifelong Simultaneous Localization and Mapping through Continual Learning.* 2022.

[256] Wadhwani, P. Edge ai market size by component (hardware [gpu, asic, cpu, fpga], software, services), by application (video surveillance, remote monitoring, predictive maintenance) by end-use, & forecast, 2023 – 2032.

[257] Wan, W., Kubendran, R., Schaefer, C., Eryilmaz, S. B., Zhang, W., Wu, D., Deiss, S., Raina, P., Qian, H., Gao, B., Joshi, S., Wu, H., Wong, H. S. P., and Cauwenberghs, G. Edge ai without compromise: Efficient, versatile and accurate neurocomputing in resistive random-access memory, 2021.

[258] Wang, E., Wang, J., and Ponsford, A. M. An adaptive hierarchal CFAR for optimal target detection in mixed clutter environments. In *2011 IEEE RadarCon (RADAR)* (May 2011), IEEE.

[259] Wang, G., Munoz-Ferreras, J.-M., Gu, C., Li, C., and Gomez-Garcia, R. Application of linear-frequency-modulated continuous-wave (LFMCW) radars for tracking of vital signs. *IEEE Trans. Microw. Theory Tech. 62*, 6 (June 2014), 1387–1399.

[260] Wang, L., Wang, D., and Hao, C. Intelligent CFAR detector based on support vector machine. *IEEE Access 5* (2017), 26965–26972.

[261] Wang, S., Song, J., Lien, J., Poupyrev, I., and Hilliges, O. Interacting with soli: Exploring Fine-Grained dynamic gesture recognition in the Radio-Frequency spectrum. 851–860.

[262] Wang, X., Li, G., Zhang, X.-P., and He, Y. A fast CFAR algorithm based on density-censoring operation for ship detection in SAR images. *IEEE Signal Process. Lett. 28* (2021), 1085–1089.

[263] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE Inst. Electr. Electron. Eng. 78*, 10 (1990), 1550–1560.

[264] Wessendorp, N., Dinaux, R., Dupeyroux, J., and de Croon, G. C. H. E. Obstacle avoidance onboard mavs using a fmcw radar. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), pp. 117–122.

[265] Wong, M. M., Shrestha, S. B., Nambiar, V. P., Mani, A., Lee, Y. K., Koh, E. K., Jiang, W., Chai, K. T. C., and Do, A. T. A 2.1 pj/sop 40nm snn accelerator featuring on-chip transfer learning using delta stdp. In *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)* (2021), pp. 95–98.

[266] Wu, J., Yilmaz, E., Zhang, M., Li, H., and Tan, K. C. Deep spiking neural networks for large vocabulary automatic speech recognition. *Frontiers in Neuroscience 14* (2020).

[267] Wu, S., Sakamoto, T., Oishi, K., Sato, T., Inoue, K., Fukuda, T., Mizutani, K., and Sakai, H. Person-specific heart rate estimation with ultra-wideband radar using convolutional neural networks. *IEEE Access 7* (2019), 168484–168494.

[268] Xu, H., Caramanis, C., and Mannor, S. Sparse algorithms are not stable: A no-free-lunch theorem. *IEEE Trans. Pattern Anal. Mach. Intell. 34*, 1 (Jan. 2012), 187–193.

[269] Xu, H., Wang, Z., Yang, H., Liu, D., and Liu, J. Learning simple thresholded features with sparse support recovery. *IEEE Trans. Circuits Syst. Video Technol. 30*, 4 (Apr. 2020), 970–982.

[270] Yadav, R., Vierling, A., and Berns, K. Radar + RGB fusion for robust object detection in autonomous vehicle. In *2020 IEEE International Conference on Image Processing (ICIP)* (Oct. 2020), IEEE.

[271] Yan, J., Hu, J., Sun, L., Hong, H., Gu, C., Li, C., and Zhu, X. Full-coverage indoor SAR imaging with a vehicle-based FMCW radar system. In *2018 IEEE/MTT-S International Microwave Symposium - IMS* (June 2018), IEEE.

[272] Yang, Z., Huang, Y., Zhu, J., and Ye, T. T. Analog circuit implementation of lif and stdp models for spiking neural networks. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI* (New York, NY, USA, 2020), GLSVLSI '20, Association for Computing Machinery, p. 469–474.

[273] Yanik, M. E., Wang, D., and Torlak, M. Development and demonstration of MIMO-SAR mmwave imaging testbeds. *IEEE Access 8* (2020), 126019–126038.

[274] Youngseok, and Sangdong Lee, J. A Low-Complexity FMCW surveillance radar algorithm using two random beat signals. *Sensors 19* (2019).

[275] Yu, S., Wu, J., Xu, H., Sun, R., and Sun, L. Robustness improvement of visual templates matching based on Frequency-Tuned model in RatSLAM. *Frontiers in Neurorobotics* (2020).

[276] Zahariadis, T., Voulkidis, A., Karkazis, P., and Trakadas, P. Preventive maintenance of critical infrastructures using 5G networks & drones. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (Aug. 2017), IEEE.

[277] Zahid, U., Guo, Q., and Fountas, Z. Predictive Coding as a Neuromorphic Alternative to Backpropagation: A Critical Evaluation. *Neural Computation 35*, 12 (11 2023), 1881–1909.

[278] Zefreh, R. G., Taban, M. R., Naghsh, M. M., and Gazor, S. Robust CFAR detector based on censored harmonic averaging in heterogeneous clutter. *IEEE Trans. Aerosp. Electron. Syst. 57*, 3 (June 2021), 1956–1963.

[279] Zhang, T., Cheng, X., Jia, S., ming Poo, M., Zeng, Y., and Xu, B. Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks. *Science Advances 7*, 43 (2021), eabh0146.

[280] Zhang, X., Wu, Q., and Zhao, D. Dynamic hand gesture recognition using FMCW radar sensor for driving assistance. In *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)* (Oct. 2018), IEEE.

[281] Zhong, Y., Cui, X., Kuang, Y., Liu, K., Wang, Y., and Huang, R. A spike-event-based neuromorphic processor with enhanced on-chip stdp learning in 28nm cmos. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)* (2021), pp. 1–5.

[282] Zhou, J., Zhu, R., Li, L., and Fu, Q. Comparision between the RMA and BPA for indoor SAR imaging applications. In *2019 6th Asia-Pacific Conference on Synthetic Aperture Radar (APSAR)* (Nov. 2019), IEEE.

[283] Zubić, N., Gehrig, D., Gehrig, M., and Scaramuzza, D. From chaos comes order: Ordering event representations for object recognition and detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2023), pp. 12846–12856.

[284] Çatal, O., Wauthier, S., De Boom, C., Verbelen, T., and Dhoedt, B. Learning generative state space models for active inference. *Frontiers in Computational Neuroscience 14* (2020).

Curriculum Vitae

Ali Safa is a doctoral researcher in Electrical Engineering at imec and Katholieke Universiteit Leuven, Belgium, advised by Prof. Francky Catthoor and Prof. Georges Gielen. His research focuses on the study of bio-inspired neuromorphic AI algorithms for energy-efficient sensor processing and drone navigation. He has published more than 25 articles in international journals, conferences and workshop proceedings.

During his doctoral studies, he was a visiting researcher during Spring 2023 in Prof. Gert Cauwenberghs' Integrated Systems Neuroengineering Laboratory (ISNL) at the University of California in San Diego, La Jolla, USA, supported through a Long Stay Abroad grant awarded by the Scientific Research Fund (Fonds Wetenschappelijk Onderzoek) of the Flemish Government, Belgium.

Prior to his doctoral studies, he received a MSc degree (*summa cum laude*) in Electrical Engineering from the Brussels Faculty of Engineering, Université Libre de Bruxelles. He was an intern with imec during the summer of 2019 and an intern at the Institute for High Energies (IIHE, VUB-ULB) during the summer of 2017 working for the CMS detector at CERN.

He is the recipient of the Best Paper Award second place at the 2023 Neuro-Inspired Computing Element (NICE) conference and the co-recipient of the Best Paper Award honorary mention at the 2023 IEEE International Symposium on Circuits and Systems (ISCAS).

List of publications

Articles in Letters, Journals and Magazines

A. Safa, T. Verbelen, L. Keuninckx, I. Ocket, M. Hartmann, A. Bourdoux, F. Catthoor, G. Gielen, "A Low-Complexity Radar Detector Outperforming OS-CFAR for Indoor Drone Obstacle Avoidance," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 9162-9175, August 2021.

Ali Safa, Francky Catthoor, Georges Gielen, "ConvSNN: A surrogate gradient spiking neural framework for radar gesture recognition," in Elsevier Software Impacts, Volume 10, 100131, ISSN 2665-9638, November 2021.

A. Safa, T. Verbelen, I. Ocket, A. Bourdoux, F. Catthoor and G. G. E. Gielen, "Fail-Safe Human Detection for Drones Using a Multi-Modal Curriculum Learning Approach," in IEEE Robotics and Automation Letters, vol. 7, no. 1, pp. 303-310, Jan. 2022.

A. Safa, A. Bourdoux, I. Ocket, F. Catthoor and G. G. E. Gielen, "On the Use of Spiking Neural Networks for Ultralow-Power Radar Gesture Recognition," in IEEE Microwave and Wireless Components Letters, vol. 32, no. 3, pp. 222-225, March 2022.

A. Safa, J. Van Assche, M. D. Alea, F. Catthoor and G. G. E. Gielen, "Neuromorphic Near-Sensor Computing: From Event-Based Sensing to Edge Learning," in IEEE Micro, vol. 42, no. 6, pp. 88-95, 1 Nov.-Dec. 2022.

A. Safa, F. Corradi, L. Keuninckx, I. Ocket, A. Bourdoux, F. Catthoor, G. Gielen, "Improving the Accuracy of Spiking Neural Networks for Radar Gesture Recognition Through Preprocessing," in IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 6, pp. 2869-2881, June 2023.

A. Safa, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor and G. G. E. Gielen,

"STDP-driven Development of Attention-based People Detection in Spiking Neural Networks," in IEEE Transactions on Cognitive and Developmental Systems, February 2024.

Articles in International Conferences and Workshops

A. Safa, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor and G. G. E. Gielen, "Event Camera Data Classification Using Spiking Networks with Spike-Timing-Dependent Plasticity," 2022 IEEE International Joint Conference on Neural Networks (IJCNN), Padua, Italy, pp. 1-8, 2022.

A. Safa, T. Verbelen, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor and G. Gielen, "Learning to Encode Vision on the Fly in Unknown Environments: A Continual Learning SLAM Approach for Drones," 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Sevilla, Spain, pp. 373-378, 2022.

A. Safa, I. Ocket, F. Catthoor, G. Gielen. 2022. "Exploring Cross-fusion and Curriculum Learning for Multi-modal Human Detection on Drones." In System Engineering for constrained embedded systems (DroneSE and RAPIDO). Association for Computing Machinery, New York, NY, USA, pp. 1–7, 2022.

M. D. Alea, **A. Safa**, J. V. Assche and G. G. E. Gielen, "Power-Efficient and Accurate Texture Sensing Using Spiking Readouts for High-Density e-Skins," 2022 IEEE Biomedical Circuits and Systems Conference (BioCAS), Taipei, Taiwan, pp. 359-363, 2022.

A. Safa, T. Verbelen, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor, G. Gielen, "Fusing Event-based Camera and Radar for SLAM Using Spiking Neural Networks with Continual STDP Learning," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 2023.

A. Safa, T. Verbelen, O. Catal, T. Van de Maele, M. Hartmann, B. Dhoedt, A. Bourdoux, "FMCW Radar Sensing for Indoor Drones Using Variational Auto-Encoders," 2023 IEEE Radar Conference (RadarConf23), San Antonio, TX, USA, pp. 1-6, 2023.

A. Safa, J. Van Assche, C. Frenkel, A. Bourdoux, F. Catthoor, and G. Gielen. 2023. "Exploring Information-Theoretic Criteria to Accelerate the Tuning of Neuromorphic Level-Crossing ADCs." In Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference (NICE '23). Association for Computing Machinery, New York, NY, USA, pp. 63–70, 2023.

A. Safa, I. Ocket, F. Catthoor, and G. Gielen. 2023. "SupportHDC: Hyperdimensional Computing with Scalable Hypervector Sparsity." In Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference (NICE '23). Association for Computing Machinery, New York, NY, USA, pp. 20–25, 2023.

A. Safa, T. Verbelen, L. Keuninckx, I. Ocket, A. Bourdoux, F. Catthoor, G. Gielen, G. Cauwenberghs, "Active Inference in Hebbian Learning Networks," In: Buckley, C.L., et al. Active Inference. IWAI 2023. Communications in Computer and Information Science, vol 1915. Springer, Cham., 2023.

M. D. Alea, **A. Safa**, F. Giacomozzi, A. Adami, I. R. Temel, R. J. Maestro, L. Lorenzelli, G. Gielen, "A 12x16 Fingertip-Mimicking per-Taxel e-Skin Spiking Readout Chip with 200 μm-Spatial Resolution," 2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits), Kyoto, Japan, pp. 1-2, 2023.

G. Tang, **A. Safa**, K. Shidqi, P. Detterer, S. Traferro, M. Konijnenburg, M. Sifalakis, G. van Schaik, A. Yousefzadeh, "Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design," 2023 IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, CA, USA, pp. 1-5, 2023.

Seminar and Poster Presentations

A. Safa, T. Verbelen, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor, G. Gielen, "Learning to SLAM on the fly using surprise," *poster contribution at* 2022 IEEE International Conference on Development and Learning (ICDL), London, UK, 2022.

A. Safa, "Neuromorphic Sensor Fusion with Applications to Drone Navigation," *contributed talk* in the Chalk Talk series at the Institute for Neural Computation (INC), University of California in San Diego, Gilman Drive, La Jolla, USA. `https://youtu.be/aE56Byw5zQk` 2023.

A. Safa, "SupportHDC: Hyperdimensional Computing with Scalable Hypervector Sparsity," *contributed talk* in the VSAONLINE workshop series, organized by Lulea University of Technology, Finland and UC Berkeley, USA. `https://youtu.be/9HS1oE-0URQ` 2023.



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
MICAS-INSYS
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
ali.safa@imec.be