University of Lille - University of Kent

Submitted in fulfilment of the degree of

Doctor of Philosophy

in Computer Science

by Nicolas Thuylie

Visual tracking with spiking neural networks and its application to aerosol event tracking

---

Thesis defended on 31/03/2026 before the committee composed of:

Mme Sylvie CHAMBON	Professeure des universités Toulouse INP	*Reviewer*
M. Sergio DAVIES	Senior lecturer Manchester Metropolitan University	*Reviewer*
Mme Rokia MISSAOUI	Professeure University of Quebec in Outaouais	*Examiner*
M. Daniel SORIA	Senior Lecturer University of Kent	*Examiner*
M. Jean MARTINET	Professeur des universités Université Côte d'Azur	*Examiner (President)*
M. Chaabane DJERABA	Professeur des universités Université de Lille	*Supervisor*
Mme Ioanna GIORGI	Lecturer University of Kent	*Co-supervisor*
M. Pierre TIRILLY	Maître de conférences Université de Lille	*Co-supervisor*
M. Jérôme RIEDI	Professeur des universités Université de Lille	*Invited Member*

School of Computing

École Doctorale Mathématiques, sciences du numérique et de leurs interactions (MADIS)

Université de Lille - University of Kent

Thèse présentée pour obtenir le grade de

Docteur en Informatique

par Nicolas Thuylie

Suivi visuel basé sur les réseaux de neurones à impulsions et son application pour le suivi d'événements atmosphériques

---

Thèse soutenue le 31/03/2026 devant le jury composé de :

Mme Sylvie CHAMBON	Professeure des universités Toulouse INP	*Reviewer*
M. Sergio DAVIES	Senior lecturer Manchester Metropolitan University	*Reviewer*
Mme Rokia MISSAOUI	Professeure University of Quebec in Outaouais	*Examiner*
M. Daniel SORIA	Senior Lecturer University of Kent	*Examiner*
M. Jean MARTINET	Professeur des universités Université Côte d'Azur	*Examiner (President)*
M. Chaabane DJERABA	Professeur des universités Université de Lille	*Supervisor*
Mme Ioanna GIORGI	Lecturer University of Kent	*Co-supervisor*
M. Pierre TIRILLY	Maître de conférences Université de Lille	*Co-supervisor*
M. Jérôme RIEDI	Professeur des universités Université de Lille	*Invited Member*

School of Computing

École Doctorale Mathématiques, sciences du numérique et de leurs interactions (MADIS)

Abstract

Aerosol plumes, such as intense smoke or dust, affect population health by causing pollution peaks. These plumes are responsible for millions of deaths per year and are mainly due to human activities. Therefore, we need an efficient and robust automatic plume tracking system that enables faster air quality prediction and supports environmental disaster response (e.g., volcanic eruptions or pollution peaks). To achieve this, satellites should be able to move their instruments directly to areas of interest to obtain the maximum amount of data from specific events. Furthermore, due to the limited energy resources onboard satellites, such tracking models should consume as little energy as possible. Unlike standard artificial neural networks, Spiking Neural Networks (SNNs) consume very little energy on suitable hardware, making them good candidates for this task. However, no atmospheric plume datasets are available for Visual Object Tracking (VOT). In addition, deep learning-based VOT methods have never been applied to atmospheric plumes, let alone low-energy SNN trackers. This absence of trackers for such tasks raises the question of the capacity of a tracker to extract relevant optical signatures from aerosol plumes. To that end, we first introduce a new VOT dataset designed for atmospheric aerosol plume tracking. This dataset is based on the Copernicus Atmosphere Monitoring Service European Air Composition reanalysis 4 (CAMS EAC4), a multi-year aerosol dataset produced by the European Centre for Medium-Range Weather Forecasts (ECMWF). This dataset focuses on sea salt and dust aerosol plumes. Then, inspired by the work of Bertinetto et al., we adapt the Fully-Convolutional Siamese Network (SiamFC) tracker to SNNs. We introduce a spiking SiamFC architecture, trained end-to-end with backpropagation and the surrogate gradient method. In this model, we introduce new spiking mechanisms: synchronised threshold adaptation, and a head-asymmetry mechanism that accumulates spikes in the instance branch. Our results show that our spiking SiamFC yields higher or similar performances to comparable SNN trackers, on generic object benchmarks. Finally, we evaluate our spiking SiamFC model on our aerosol plume tracking dataset. Results show that our spiking tracker can learn plume features, paving the way for future developments in VOT applied to aerosol plumes.

Résumé

Les panaches d'aérosols, tels que les fumées intenses ou les poussières, affectent la santé des populations en provoquant des pics de pollution. Ces panaches sont responsables de millions de décès chaque année et sont principalement dus aux activités humaines. Il est donc nécessaire de disposer d'un système de suivi automatique des panaches efficace et robuste, permettant un suivi plus rapide de la qualité de l'air afin de contribuer à la gestion des catastrophes environnementales (par exemple, les éruptions volcaniques ou les pics de pollution). Pour atteindre cet objectif, les satellites doivent être capables de diriger leurs instruments directement vers les zones d'intérêt afin d'obtenir un maximum de données sur des événements spécifiques. De plus, en raison des ressources énergétiques limitées à bord des satellites, ces modèles de suivi doivent consommer le moins d'énergie possible. Contrairement aux réseaux de neurones artificiels classiques, les réseaux de neurones impulsionnels (Spiking Neural Networks, SNN) consomment très peu d'énergie sur du matériel adapté, ce qui en fait de bons candidats pour cette tâche. Cependant, aucun jeu de données atmosphérique n'existe pour le suivi d'objets visuels (Visual Object Tracking, VOT). De plus, les méthodes de VOT basées sur l'apprentissage profond n'ont jamais été appliquées aux panaches atmosphériques, et encore moins les systèmes de suivi visuels SNN à faible consommation. Cette absence de systèmes de suivi visuels adaptés soulève la question de la capacité d'un modèle de suivi à extraire des signatures optiques pertinentes des panaches d'aérosols. À cette fin, nous introduisons d'abord un nouveau jeu de données VOT conçu pour le suivi des panaches d'aérosols atmosphériques. Ce jeu de données repose sur la réanalyse européenne de la composition de l'air du service Copernicus Atmosphere Monitoring Service (CAMS EAC4), un jeu de données multiannuel sur les aérosols produit par le Centre européen pour les prévisions météorologiques à moyen terme (ECMWF). Ce jeu de données met principalement l'accent sur les panaches d'aérosols de sels marins et de poussières. De plus, en nous inspirant des travaux de Bertinetto et al., nous adaptons le système de suivi visuel Fully-Convolutional Siamese Network (SiamFC) aux réseaux SNN. Nous proposons une architecture impulsionnelle de SiamFC, entraînée de bout en bout par rétropropagation à l'aide de la méthode du substitut du gradient. Dans ce modèle, nous introduisons de nouveaux mécanismes im-

pulsionnels : une adaptation synchronisée du seuil et un mécanisme d'asymétrie de tête accumulant les impulsions dans la branche d'instance. Nos résultats montrent que notre SiamFC impulsionnel obtient des performances supérieures ou équivalentes à celles de systèmes de suivi visuels SNN comparables sur des jeux de données génériques. Enfin, nous évaluons notre modèle SiamFC impulsionnel sur notre jeu de données de suivi de panaches d'aérosols. Les résultats montrent que notre système de suivi visuel impulsionnel parvient à apprendre des caractéristiques de panaches, ouvrant ainsi la voie à de futurs développements du VOT appliqué aux panaches d'aérosols.

Acknowledgements

I would first like to express my deep gratitude to my family, particularly my brother, mother, father, and grandmother, as well as to my dog Nubia and my Cat Uranie. Their constant support, encouragement and understanding have helped me throughout the years of this thesis. Their patience in the difficult times and their confidence in me have been invaluable.

I am sincerely grateful to my supervisor, Pierre Tirilly, for his guidance, insightful feedback and continued support during this work and the writing of this thesis. I would also like to thank Jérôme Riedi in particular for his unwavering support, both human and scientific, as well as for his availability, which were decisive at many stages of this thesis. Their scientific rigour and trust have greatly shaped the direction and quality of this work.

I would also like to give special thanks to Ioanna Giorgi, who took over my thesis supervision following Dominique Chu's departure.

Finally, I would like to thank Chaabane Djeraba for getting me the opportunity to do this thesis.

My thanks also go to my friends and colleagues from LOA, Alexis, Elise, Paul, Louis, Clara, Perla, and Gabriela, with whom I had numerous discussions, constructive comments, and a friendly atmosphere in the lab. This thesis would not have been possible without their unconditional support. Many thanks also to my friend Henri and Sixtyne, who have supported me since engineering school.

I would also like to thank the matcha for its contribution and help in staying

alert during this thesis, and bouldering for helping me clear my mind and relax during the hard moments.

Finally, I wish to acknowledge all those, in and outside academia, who have contributed in various ways to the completion of this thesis, whether through collaborations, technical assistance, or simply by providing much-needed encouragement.

Contents

1 Introduction **23**

1.1 Problem 23

1.2 Motivation 26

1.3 Research Questions 28

1.4 Contributions 29

1.5 Outline 31

I Background **33**

2 Existing Aerosol Tracking Models **35**

2.1 Fundamental Notions and Concepts 35

2.1.1 Aerosol Species and Life Cycle 35

2.1.2 Aerosol Optical Depth 36

2.1.3 Type of Products 37

2.2 Existing Models for Aerosol Analysis 37

2.2.1 VOLCAT 37

2.2.2 VolcPlume 38

2.2.3 ECMWF - CAMS 39

2.3 Discussion 43

3 Visual Object Tracking (VOT) **45**

3.1 General Concepts of VOT 45

3.2 Challenges & Datasets in VOT 47
3.3 Appearance Models 53
3.3.1 Generative Methods 53
3.3.2 Discriminative Methods 55
3.3.3 Feature-Based Methods 56
3.4 Motion Models 58
3.5 Deep Learning-based Methods 60
3.5.1 MDNet 60
3.5.2 DeepSort 62
3.5.3 LSTM and CNN 62
3.5.4 SiamFC and Siamese-based Trackers 64
3.5.5 Transformer-based Trackers 67
3.6 Conclusion 68

4 Spiking Neural Networks **73**
4.1 Core Concepts of Spiking Neurons 73
4.1.1 Spiking Neuron Models 74
4.1.2 Neural Coding 76
4.2 Training Spiking Neural Networks 78
4.2.1 ANN to SNN Conversion 79
4.2.2 Hebbian Learning 80
4.2.3 Feedback Alignment 81
4.2.4 Backpropagation with Surrogate Gradient 82
4.2.5 Energy Consumption of SNNs 83
4.3 SNNs for VOT 85
4.3.1 Previous SNN Trackers for SOT 85
4.3.2 Spiking Siamese Networks for Tracking 87
4.4 Conclusion 88

II Contributions **91**

5 Aerosol tracking dataset **93**

5.1 Aerosol Data Acquisition 94

5.1.1 ECMWF CAMS EAC4 Data 96

5.1.2 Preprocessing Steps 98

5.2 Annotation Process 101

5.3 Dataset Analysis 102

5.3.1 Dataset Properties 103

5.3.2 Quality Control 103

5.4 Limitations and Future Work 109

5.5 Conclusion 112

6 Spiking SiamFC **115**

6.1 Baseline: SiamFC ANN 117

6.2 Proposed Spiking SiamFC 119

6.2.1 Neural Coding 120

6.2.2 Synchronised Threshold Adaptation for Siamese Networks . 121

6.2.3 Backbone Architecture 122

6.2.4 Spike Rate Pooling 122

6.2.5 Similarity Function 124

6.2.6 Training 125

6.3 Experiments 131

6.3.1 Datasets 131

6.3.2 Evaluation Metrics 132

6.3.3 Evaluation Protocol 133

6.3.4 Implementation Details 134

6.3.5 Benchmark Evaluation 135

6.3.6 Ablation Study 139

6.4 Discussion 142
6.5 Conclusion 144

7 Spiking SiamFC on Atmospheric Products 147

7.1 Dataset Properties 149
7.2 Preprocessing 149
7.3 Experimental Protocol 150
7.4 Implementation Details 152
7.5 Results 153
7.6 Discussion 159

8 Conclusion and Future Work 163

8.1 Summary of Contributions 163
8.1.1 Construction of a Dataset for the VOT of Atmospheric Plumes 164
8.1.2 Spiking SiamFC Tracker 164
8.1.3 Evaluation of Spiking SiamFC on Atmospheric Plumes . . . 165
8.2 Dissemination 166
8.3 Future Work 166
8.3.1 Annotation of other AOD Species from CAMS EAC4 167
8.3.1.1 Training and Evaluation on Data from Real Instruments 167
8.3.2 Motion Model and Physics Informed Neural Networks . . . 168
8.3.3 MOT with Spiking SiamFC 168

Appendices 171

A Aerosol Tracking Dataset 173

A.1 Data Format and Structure 173

List of Figures

2.1 Basic steps of aerosol position prediction from ECMWF. 40

2.2 Computation process in a single outer loop for 4D-Var. It minimizes the cost function J through the assimilation window [1]. 41

3.1 Schematic representation of an object tracker [2]. The tracker is initialised by selecting the targeted object in the first frame, which defines the initial bounding box. From this initial bounding box, the motion model estimates a region of interest. From this region of interest, the tracker extracts features to refine the position given by the motion model with a specific appearance model (e.g., a similarity measure on features). This newly given position is then used to update the motion model, and the cycle starts over for the next frame. 47

3.2 Taxonomy of VOT methods used in this thesis. Methods are grouped by appearance (generative or discriminative) and motion (filtering or probabilistic) models. 54

3.3 MDNet architecture [3]. MDNet is composed of a feed-forward convolutional backbone that learns general features from large-scale visual tracking data. Then, domain-specific layers are trained for specific tracking tasks. This figure comes from Nam et al.'s paper [3] . 61

3.4 Workflow of SiamFC architecture [4]. This figures has been used from the paper of Bertinetto et al. [4]. 65

5.1 Illustration of the AOD for each species of CAMS EAC4. CAMS provides five categories of aerosols: dust, sea salt, organic matter, black carbon, and sulphate. 96

5.2 CAMS reanalysis maps for 1st September, 2020, showing the dust AOD at 550nm (top), and the corresponding dust aerosol fraction (bottom). The dust fraction map highlights the relative contribution of dust to the overall aerosol content. 99

5.3 CAMS AOD maps of dust aerosols for 1st September, 2020, showing the dust AOD fraction map (top), and the same map after min-max normalisation (bottom). The normalised map enhances local contrasts and the sharpness of dust plume boundaries. 100

5.4 Heatmap of the distribution of the sizes of bounding boxes annotated on the dust AOD maps over the Gobi and Saharan desert regions. The distribution peaks correspond to the typical spatial scales of dust plumes observed in CAMS EAC4 dust aerosol data for these areas [5]. The annotations were realised from the 1st of September, 2020 to 31st of December, 2022, reflecting realistic plume size variability over multiple seasons for dust aerosols. 104

5.5 Heatmap of the size distribution of bounding boxes annotated on AOD maps of sea salt. Most annotated plumes have moderate horizontal extents, consistent with the short lifetime and marine origin of sea salt aerosol maps discussed by Bian et al. [6]. 105

5.6 By-species statistics of the areas of bounding boxes of groundtruth plumes. Red crosses represent the mean area, and black crosses represent the outlier bounding boxes. The statistics for dust agree with the tables provided by Silva et al. [5]. 106

5.7 Class distribution of the annotated tracking aerosol plume dataset. The number of sequences (blue bars, left axis) and frames (green bars, right axis) are shown for dust plumes (28 months) and sea salt plumes (5 months). Percentages indicate the relative contribution of each class for frames and sequences. 107

5.8 Intensity distribution of the by-species AOD within bounding box annotations for dust (blue) and sea salt (orange). The distribution follows an inverse exponential law, expected from aerosols behavior [6, 5]. Most of the values have low intensities and come from the background. Higher intensities represent events that are less frequent. The sea salt density curve shows a slight perturbation compared to dust and gets higher intensity values. This reflects more episodic events and fewer cyclic and continuous emissions of aerosols. 108

5.9 Box plot of the intensity distribution of dust and sea salt by-species AOD. This figure shows a median of 0.1 for dust AOD and 0.04 for sea salt AOD. Red crosses represent the mean AOD value, and black crosses represent outliers. 109

5.10 Seasonal trends of the mean by-species AOD for dust plumes from the 1st of September, 2020 to 31st of December, 2022. The graph shows the mean (curve) and standard deviation (bars) for each month. These trends agree with the climate statistics provided by Kunchala et al. [7]. 110

5.11 Seasonal trends of the mean by-species AOD for sea salt plumes from the 1st of September, 2020 to 31st of January, 2021. The graph shows the mean (curve) and standard deviation (bars) by month. These trends align with those of Aldhaif et al. [8]. 111

6.1 Overview of our spiking SiamFC model. x is the instance image, z is the examplar image, and ψ is the backbone. 119

6.2 Direct coding of an input image. 120

6.3 Backbone ψ of our model. This backbone is an adaptation of AlexNet [9] used in SiamFC ANN from Bertinetto et al. [4]. 122

6.4 Unfolding and chain rule computation of our model for surrogate gradient backpropagation, showing the error gradients across multiple time steps. The left-hand part represents the unfolded computational graph with the output gradients. The right-hand part shows the computation of the similarity with its gradient. 128

6.5 **Success plot and precision plot for OPE of spiking SiamFC on the OTB-2015 benchmark (values in legends are AUC scores).** Values of the SiamFC ANN have been obtained by training our own SiamFC ANN, following the protocol provided by Bertinetto et al. [4] (except for the conversion of 25% of the training set to grayscale). (a): Comparison of the success rate across the whole overlap threshold range between our spiking model and the original SiamFC from Bertinetto et al. [4]. (b): Comparison of the precision rate across the whole overlap threshold range between our spiking model and the original SiamFC from Bertinetto et al. [4]. 136

6.6 **Success plot and precision plot for OPE of spiking SiamFC on the OTB-2015 benchmark.** (a): Success curve of each specific ablation setting in Table 6.3. Values in the legend represent the AUCs. (b): Precision curve of each specific ablation setting in Table 6.3. 138

6.7 **Success plot for OPE of spiking SiamFC on the GOT-10k benchmark (values in legends are AUC scores).** Each curve represents the success rate of a specific ablation setting with respect to the overlap threshold. The orange curve shows our model without ablation. 139

6.8 IoU on the validation set with and without synchronized threshold adaptation (blue curve and green curve, respectively). 141

7.1 Illustration of the spiking SiamFC for each setting from Table 7.1. For each setting, we display the frames at time steps 5, 10, 15 and 20. Each time step is equivalent to 3 hours. Green rectangles are the ground truth bounding boxes, while red rectangles represent the predicted bounding boxes. 151

7.2 IoU learning curves for aerosol tracking with total AOD input. Top panels compare the tracking of sea salt only (a) and dust only (b), while the bottom panel presents combined the tracking of dust and sea salt (c). 154

7.3 IoU learning curves for aerosol tracking with by-species AOD. Top panels compare the tracking of sea salt only (a) and dust only (b), while the bottom panel presents the tracking of combined dust and sea salt (c). The results show the improved convergence with by-species for sea salt, but not for dust. The combined species case shows similar results for validation as total AOD. 155

7.4 Success plot following the OPE evaluation protocol for tracking plumes using CAMS EAC4 total AOD. Each line represents a different aerosol class: dust and sea salt combined (blue), dust (orange), and sea salt (green). The success rate at 50% for each species is given in the legend. The dust and sea salt class and the dust class show similar tracking performance, while sea salt plumes are notably more difficult to track, reflected by the lower success rates across thresholds. The results highlight the challenge of accurately following diffuse and low-contrast aerosol plumes from reanalysis data. 157

7.5 Success plot following the OPE evaluation protocol for plume tracking using CAMS EAC4 by-species AOD. The blue curve shows combined dust and sea salt performance, the orange one shows dust only, and the green one shows sea salt only. The success rate at 50% indicates that tracking is most successful for combined plumes, with lower performance for individual dust and especially sea salt plumes. This highlights the increased difficulty of localising and following diffuse or low-contrast events. 158

7.6 Cross-correlation of the mean AOD over ground truth regions by the IoU through time. Figures (a) and (b) show this cross-correlation for the E1 and E4 settings, respectively (see Table 7.1). (c) and (d) show the ones for settings E2 and E5, respectively. (e) and (f) show the cross-correlation from settings E3 and E6, respectively. Each cross-correlation has been computed over at least 25 samples to ensure statistical robustness. 160

List of Tables

3.1 Overview of commonly used VOT benchmarks. 50

3.2 Description of commonly used VOT benchmarks. 51

3.3 Representative VOT trackers reviewed in this thesis, from classical correlation filters to Siamese CNNs and transformer-based architectures. 71

4.1 Energy consumption estimations of SNNs and ANNs counterparts on CIFAR10, GSC, and N-CARS from Lemaire et al. [10]. 85

6.1 Details of the backbone ψ. Convolution layers are two-dimensional spiking convolutional layers equiped with our synchronized threshold adaptation (see Section 6.2.2, Page 121), and pooling layers use our spike rate pooling layer (see Section 6.2.4, Page 122). * same in both dimensions. ** threshold is $\theta_{\mathbf{base}}$ for convolutional layers and θ_p for pooling layers. 135

6.2 Success rates at 50% of SiamSNN and spiking SiamFC on OTB-2015 and GOT-10k. 137

6.3 Results of the ablation study on OTB-2015 and GOT-10k. 140

7.1 Experimental settings for the tracking of dust and sea salt aerosols. Each setting corresponds to the type of aerosols we want to track and the product on which we track them. 152

List of Acronyms

4D-Var Four-Dimensional Variational. 40

Adam Adaptive Moment Estimation. 134

ANN Artificial Neural Network. 25

AOD Aerosol Optical Depth. 36

AUC Area Under the Curve. 49

BCE Binary Cross-Entropy. 126

BPTT Backpropagation Through Time. 82

CAMS Copernicus Atmosphere Monitoring Service. 29

CNN Convolutional Neural Network. 60

CPU Central Processing Unit. 25

CSNN Convolutional Spiking Neural Network. 76

DeepSORT Deep Simple Online and Realtime Tracking. 60

DVS Dynamic Vision Sensors. 85

EAC4 Atmospheric Composition Reanalysis 4. 94

ECMWF European Centre for Medium-Range Weather Forecasts. 29

EPSP Excitatory Postsynaptic Potential. 74

Faster R-CNN Faster Region-based Convolutional Neural Network. 66

GOES Geostationary Operational Environmental Satellites. 39

GOT-10k Generic Object Tracking. 31

GPU Graphics Processing Unit. 25

GSC Google Speech Commands V2. 84

HH Hodgkin-Huxley. 74

HOG Histogram of Oriented Gradients. 57

IASI Infrared Atmospheric Sounding Interferometer. 39

ICARE Interaction Clouds Aerosols Radiations. 38

IF Integrate-and-Fire. 74

IFS Integrated Forecasting System. 39

IFS-AER Integrated Forecasting System AERosol scheme. 39

IoU Intersection-over-Union. 48

IPSP Inhibitory Postsynaptic Potential. 74

KCF Kernelised Correlation Filter. 24

LaSOT Large-scale Single Object Tracking. 49

LIDAR LIght Detection and RAnging. 39

LIF Leaky Integrate-and-Fire. 74

LOA Laboratoire d'Optique Atmosphérique. 38

LSTM Long Short-Term Memory. 60

MACC Monitoring Atmospheric Composition and Climate. 95

MDNet Multi-Domain Network. 60

MODIS MODerate resolution Imaging Spectroradiometer. 110

MOSSE Minimum Output Sum of Squared Error. 24

MOT Multiple Object Tracking. 46

MSG Meteosat Second Generation. 39

MVTrack Multi-View object Tracking. 49

N-CARS Neuromorphic CARS. 84

OMI Ozone Monitoring Instrument. 39

OMPS Ozone Mapping and Profiler Suite. 39

OPE One-Pass Evaluation. 133

OTB Object Tracking Benchmark. 31

PINN Physics-Informed Neural Network. 166

PM Particle Matter. 23

ReLU Rectified Linear Unit. 83

ResNet Residual Neural Network. 79

RGB Red-Green-Blue. 30

RNN Recurrent Neural Networks. 62

ROLO Recurrent you Only Look Once. 63

SBT Single Branch Tracking. 68

SConv Convolutional Spiking layer. 120

SCTN Spiking Convolutional Tracking Network. 85

SiamFC Fully-Convolutional Siamese Network. 24

SiamMT Siamese Multi-Task. 65

SiamRPN Siamese Region Proposal Network. 65

SNNs Spiking Neural Networks. 25

SORT Simple Online and Realtime Tracking. 59

SOT Single Object Tracking. 45

SRM Spike Response Model. 74

STDP Spike Timing-Dependent Plasticity. 78

SUOMI-NPP Suomi National Polar-orbiting Partnership. 39

SVM Support Vector Machines. 55

TET Temporal Efficient Training. 116

TROPOMI TROPOspheric Monitoring Instrument. 39

TWSC Temporally Weighted Spike Count. 116

UV Ultra-Violet. 36

VGG Visual Geometry Group. 79

VOLCAT Volcanic Cloud Analysis Toolkit. 37

VOT Visual Object Tracking. 24

YOLO You Only Look Once. 63

Chapter 1

Introduction

1.1 Problem

Global anthropogenic aerosol emissions[1] increased dramatically during the industrial period. They reached their highest levels in the late 20th century, then declined or stabilised in most regions over the past two decades. However, these emissions are still increasing in some industrialising regions [11]. These aerosols and gas plumes (i.e., a visible area that contains gas or solid particles) could have a significant impact on the health and well-being of populations, regardless of whether they are natural or anthropogenic in origin. Even more concerning are the anthropogenic aerosol emissions (such as small particles like Particle Matter (PM) below 2.5μm). These aerosols, responsible for millions of deaths per year worldwide [12], are mainly linked to fossil energy exploitation and other human activities. Efficient and robust automatic plume tracking could enable better, faster air quality prediction and support environmental disaster response (e.g., volcanic ash, forest fire or pollution peaks).

Furthermore, these aerosols also play a crucial role in climate change by being part of most atmospheric phenomena. Therefore, automating tracking in atmo-

[1] Anthropogenic emissions are those made by humans.

spheric science could help researchers better understand cloud formation, patterns, and spatial distributions of rainfall, and atmospheric chemistry in general, by providing comprehensive spatial and temporal data on plumes for specific events. Coupled with a plume detector, such trackers could help follow even small events (e.g, local pollutant emissions). Extensive research exists for detecting these plumes [13]. However, existing tracking methods (with automatic re-identification of plumes) remain, to the best of our knowledge, very limited in the literature. These methods are either manual, with an operator identifying and re-identifying the plumes in VolcPlume [14], or based on a thresholding method [15]. In the former case, each plume must be tracked manually; in the latter, pixels are first filtered using thresholding, then a machine learning model classifies them as being a plume or not. These methods are effective, but lack automation in the former case, or the target plume could be integrated with adjacent plumes to form a single large plume in the latter.

In recent years, Visual Object Tracking (VOT) [16] (defined as the localisation and re-localisation of the same object across a sequence of frames) and, especially, tracking by detection [17] have made significant progress. This type of tracker has known many improvements, passing from popular correlation-filter-based methods, such as the Minimum Output Sum of Squared Error (MOSSE) [17] or Kernelised Correlation Filter (KCF) [18] methods, to Siamese-based methods such as the Fully-Convolutional Siamese Network (SiamFC) [4].

Compared to tracking generic objects, tracking plumes poses unique challenges, as the plumes have no definite shapes, and require the tracker to verify an initial hypothesis suggested by Francis et al. [19]: this hypothesis states that a significantly unique optical signature can be extracted from plume textures. However, no VOT methods have been applied to such atmospheric data, and this hypothesis has not been validated for VOT in the literature. Therefore, tracking plumes requires models that can verify the hypothesis of Francis et al., while remaining sufficiently

energy-efficient to be embedded in instruments.

In recent years, VOT models based on deep learning have demonstrated superior performance compared to traditional approaches. However, the choice of a deep learning-based approach is generally a trade-off between tracking accuracy and inference speed. Among these models, SiamFC is an efficient Artificial Neural Network (ANN) model that learns a universal correlation function between the targeted object and the scene, thanks to a network composed of two branches sharing their weights. SiamFC and its variants became popular for their offline training (no further training is required for objects with similar distributions) and their near-real-time performance. At the time when we started this work, these were the state-of-the-art models for VOT. Thus, trackers of the Siamese type are used as a basis for our work.

However, SiamFC and ANNs in general consume a massive amount of energy both for training and inference [20]. In particular, computer vision models are often highly resource-intensive. As a consequence, they consume a tremendous amount of energy, which not only contributes to climate change, as shown in [20], but also prevents them from being embedded in instruments, such as those on satellites, due to rapid battery depletion and the limits on battery capacity. Spiking Neural Networks (SNNs), considered the third generation of neural networks [21], are promising candidates to replace ANN in deep learning trackers, thanks to their spike-based information processing [22]. This information processing could allow SNNs to consume much less energy than ANNs when dedicated hardware is used [23]. Recently, researchers have developed such neuromorphic hardware designed for spike-based computations from SNNs (such as Intel's Loihi [24] or IBM's TrueNorth [25]). This hardware has demonstrated significant gains in energy efficiency compared to the conventional Graphics Processing Units (GPUs) and Central Processing Units (CPUs) used for computer vision models [26]. SNNs run on these neuromorphic platforms can consume orders-of-magnitude less energy, due to their sparse signal

processing and hardware-level architectural optimisations [27, 28]. The evaluation of performances also shows that SNNs running on these hardware platforms not only reduce energy consumption, but can also achieve real-time inference with minimal latency. These results are promising; however, to the best of our knowledge, very little work [29, 30] has been conducted on VOT with SNN architectures: the authors of existing spiking Siamese models either use an ANN-to-SNN conversion [29], or do not provide enough experimental details to reproduce their work [30]. Implementing such trackers with directly-trained SNNs still poses many challenges, such as the propagation of error through the network due to the discrete nature of SNNs and weight sharing within the Siamese network, which hampers the backpropagation of the gradient.

In this manuscript, we aim to evaluate whether a low-energy model of VOT is able to track atmospheric plumes with enough accuracy and efficiency.

1.2 Motivation

With many available ground-based instruments and instruments onboard satellites, atmospheric science provides a huge and growing amount of multispectral and multidimensional data at multiple resolutions. As previously stated, analysing this data is necessary for studying the atmosphere and the climate, which impact everyone's health and well-being, especially in the context of climate change driven by anthropogenic emissions. Given this amount of data, analysing plumes at multiple resolutions across different spectral bands is time-consuming for researchers. Consequently, most of the effort is devoted to tracking only a few significant events.

The manual annotation of atmospheric data prior to analysis slows down the study of these events. This process can overshadow the contributions of small events to atmospheric phenomena and their impact on health. A study by Manavi et al. [31] shows a significant underestimation of these small plumes, which can

lead to failures to detect pollution peaks at the local scale. This misdiagnosis can shift the responsibility for poor air quality away from local industry. Furthermore, the understanding of interactions between small and larger events is poorly understood. Thus, the automated tracking of these events is essential to advance our understanding of these phenomena.

Existing methods for aerosol plume tracking are either manual annotations [14] or detection [15] (which are not VOT methods) or tracking that operate at the air parcel level [32] (i.e., they simulate the motion of a sub-volume of air at a specific altitude) to produce individual trajectories. However, for this last method, an air parcel does not contain enough information to represent or track an entire plume. Accurately tracking a plume would require generating trajectories for all its constituent air parcels. Even then, these models fail to capture the changes in size of aerosol plumes, which occur continuously over time. So, plumes must be processed as a whole to be tracked successfully. Since the last decade, deep learning trackers have emerged as the best methods for VOT [33]. They could offer a solution to the tracking of aerosol plumes. However, ANN trackers are not energy-efficient enough to be embedded on satellites [34, 35].

One way to mitigate these problems would be to create a low-energy tracker. This solution would enable researchers to automatically move instruments to focus on specific events, and facilitate the process of obtaining data for these specific events (e.g., forest fires). One promising solution is to adapt existing ANN trackers with SNNs and embed this module directly on instruments. However, little work on SNN trackers is available. The authors of [29] convert a pre-trained SiamFC ANN [4] to an SNN; however, ANN-to-SNN conversion sacrifices a part of the energy savings expected from SNNs. Xiang et al [30] present interesting results on a directly-trained SNN tracker, spiking SiamFC++, but do not provide an analysis of their more complex model, nor sufficient details for reproducibility. In addition, they do not follow the same experimental protocol as the community [4, 36, 37].

In this thesis, we chose to focus on SiamFC from Bertinetto et al. [4] because of its maturity and because subsequent improvements to it and transformer-based trackers still rely on Siamese architectures. Consequently, SiamFC provides an ideal baseline for building SNN Siamese trackers from scratch.

1.3 Research Questions

To achieve the elaboration of this low-energy plume tracker, we face two research questions, which are at the core of this thesis. The first question addressed in this thesis is how a Siamese tracker based on a directly trained SNN can be developed. The second question is how effectively this SNN Siamese tracker would perform on aerosol plume tracking. In addition, to the best of our knowledge, no VOT dataset of aerosol plumes exists, which brings a third research question: how to design a VOT dataset for aerosol plume tracking. Thus, the following challenges have to be considered.

- The quantity of sequences to annotate: creating a VOT dataset requires to annotate a large quantity of atmospheric data featuring different types of aerosol plumes. Given the hundreds of sequences of dozens of frames needed to train trackers, the annotation process represents a tremendous effort.

- Input representation: standard trackers process images as continuous pixel intensity values. In SNNs, information is represented as spike trains. It raises the problem of representing input data as spikes. Moreover, atmospheric data is not made of regular colour pixels, as in generic object tracking datasets, but rather physical values defined over unbounded ranges [38], making their conversion to spikes more complex. Therefore, multiple spike encoding strategies will have to be explored during the design of the tracker.

- The adaptation of the SiamFC architecture: SiamFC is an effective, two-

branch tracking model that learns a universal correlation function. In its spiking counterpart, this function will require perfect synchronisation between the binary signals from its two branches. Such synchronisation is not guaranteed in SNNs. The correlation function must therefore be redefined for spike trains. Training the tracker adds another challenge, since SiamFC learns from pairs of frames randomly sampled from the training sequences; this training procedure is already unstable in standard ANNs. This issue is likely to worsen when using SNNs, which often cause unstable loss dynamics in the early stages of training [39].

- Tracking of deformable objects: atmospheric plumes (aerosols or gases) are shapeless objects prone to deformation, making it difficult for the model to rely on classical features. Instead, the model has to extract visual features from the texture of the input data to learn an optical signature of the plume.

- The physical nature of plumes: atmospheric plumes can undergo heavy changes over time, making them more challenging to re-identify due to their altered appearance. In addition, depending on the type of particles to be tracked, plumes will exhibit different types of behaviour, that the tracker must accommodate.

1.4 Contributions

This thesis proposes contributions in order to address the above challenges. These contributions advance both the theoretical understanding of spiking Siamese trackers and their practical implementation for atmospheric plume tracking.

- The annotation of the first dataset to track aerosols from the Copernicus Atmosphere Monitoring Service (CAMS) [38]. We introduce a new dataset for VOT based on the reanalysis dataset from the European Centre for Medium-

Range Weather Forecasts (ECMWF) [38]. We prepared the data and annotated it for dust and sea salt aerosols, covering an easy use case for the former, and one of the most difficult ones for the latter. This strategy is used to evaluate trackers on representative instances of the physical problem.

- Design of a new spiking Siamese tracker. We develop a spiking version of the SiamFC model of Bertinetto et al. [4]. This model is trained end-to-end with backpropagation equipped with a surrogate gradient function [40]. We also propose a full derivation of the error for backpropagation in spiking Siamese networks.

- Comparison of spiking encoding methods on generic object datasets. Since spike coding strongly influences SNN performance, we compare various spike encoding strategies. Our results show that direct coding provides the best performance for grayscale and Red-Green-Blue (RGB) data, confirming findings from detection models with SNN architectures [41].

- Temporally-weighted spike correlation. Siamese SNN trackers (trained as ANNs, then converted into SNNs) accumulate the spikes in the branch of the searched object, consuming more computational power. We show that we can achieve the same level of accuracy by accumulating spikes along the branch where we search for this object. Doing so reduces the number of spikes used by the tracker.

- Adaptive threshold synchronisation: the spiking neurons used in this thesis rely on a threshold. This value decides whether a neuron will emit a spike; adaptating this value during training and inference has been shown to yield better performance [40]. However, this adaptation of the threshold in Siamese network has not been extensively studied in the literature. We propose, in our spiking SiamFC model, an evolution of threshold adaptation to synchronise

thresholds across branches after each time step. We show that this method stabilises training.

- Validation of our spiking SiamFC. We conduct an ablation study on Object Tracking Benchmark (OTB) [42] and Generic Object Tracking (GOT-10k) [43], and compare its performance to SiamFC ANN [4] and SiamSNN [29].

- Evaluation on CAMS data. We show the limitations of the proposed spiking tracker across the dataset developed in Chapter 5, for atmospheric plume tracking, and identify directions for future improvements. Finally, we confirm our initial assumption that the model extracts an optical signature for each plume.

1.5 Outline

This document is structured into two main parts: a background section reviewing the state of the art relevant to the topics addressed, and a second section presenting our contributions. Overall, the manuscript comprises eight chapters, designed to guide the reader from the general context of the thesis to the specific contributions and experimental findings.

- In Chapter 2, we provide some theoretical and technical foundations necessary for understanding physical concepts and introduce the problem of tracking aerosols, emphasising existing approaches and their limitations in the context of current atmospheric science;

- In Chapter 3, we review existing VOT methods, detailing how computer vision methods have been developed and evaluated to follow dynamic targets across sequences. In this part, we review classical methods such as generative, discriminative, and probabilistic trackers before presenting more advanced deep learning-based approaches based on those methods. We focus particularly on

Siamese architectures, leading to Siamese trackers, explaining the core concept of each one, with their advantages and limits;

- Chapter 4 is an introduction to SNNs covering neuron models, spike coding methods, training algorithms, and existing spiking trackers;

- In Chapter 5, we describe the design of a dataset of annotated aerosol plumes. We provide an annotation protocol, the preprocessing steps on the CAMS reanalyses dataset used to facilitate annotation, and the final annotations of aerosol plumes. Finally, we propose a statistical study and validate the dataset to show its relevance to atmospheric science.

- In Chapter 6, we present our SNN model, spiking SiamFC, including its direct coding, synchronised threshold adaptation, and temporally-weighted spike count similarity measure, as well as its validation on the OTB-2015 and GOT-10k datasets.

- In Chapter 7, we evaluate the spiking Siamese-based model designed in Chapter 6 on the atmospheric plume tracking dataset of Chapter 5, and analyse the results.

- In Chapter 8, we summarise our main findings, discuss the limitations of our contributions, and propose directions for future research. These directions aim at enhancing the efficiency, robustness, and applicability of our spiking SiamFC on atmospheric data.

Part I

Background

Chapter 2

Existing Aerosol Tracking Models

Atmospheric aerosols play a key role in air quality, climate, and health. As discussed in Section 1.1 (Page 23), providing automation of tracking of such atmospheric plumes can improve early warning for exposed populations and support atmospheric scientists in analysing specific small events. However, existing approaches to aerosol and gas plume tracking remain limited. In this chapter, we first introduce several fundamental notions and concepts related to aerosols. We then review representative existing systems for aerosol plume monitoring. This review will include dedicated platforms and forecasting and reanalysis models.

2.1 Fundamental Notions and Concepts

2.1.1 Aerosol Species and Life Cycle

Aerosols are small solid or liquid particles suspended in the atmosphere [44]. They originate from both natural sources (e.g., dust, sea salt, volcanic ash emissions) and anthropogenic activities (e.g., combustion, industrial pollution). Aerosols are usually categorised into species such as dust, sea salt, sulphate, organic matter, and black carbon. Each of these species has distinct sources, composition, and optical properties. In this thesis, these particles are not considered individually but as

aerosol plumes, defined as coherent atmospheric structures of concentrated aerosol particles.

The life cycle of an aerosol in the atmosphere starts with its emission, due to events like forest fires or dust storms in the Sahara. Then, aerosols are transported across the atmosphere by a complex motion system of air masses [44]. During transport, aerosols undergo chemical and microphysical transformation (e.g., coagulation, condensation) [44]. During this phase, aerosol plumes rarely follow a simple, coherent trajectory. Instead, they can split into multiple sub-plumes, or merge with other nearby plumes, under the effect of movements of air masses [44]. They are finally removed from the atmosphere by dry deposition (i.e., due to the action of gravity), and wet deposition (i.e., due to precipitations) [44]. These processes shape the spatial and temporal evolution of aerosol plumes, which is central to their detection and tracking.

2.1.2 Aerosol Optical Depth

Aerosol Optical Depth (AOD) is a dimensionless quantity that describes how strongly aerosols reduce the intensity of sunlight (a phenomenon called extinction) over a vertical column of the atmosphere from the surface to its top.

AOD depends on the wavelength at which it is measured, since the scattering and absorption of light by aerosols are sensitive to particle size and composition. In general, shorter wavelengths, such as Ultra-Violet (UV), are more strongly affected by small particles, while longer wavelengths, such as infrared, are more influenced by coarse aerosol particles.

The total AOD integrates all aerosol species present in the air column, whereas the by-species AOD considers only the contribution from a given species within the air column (e.g., dust AOD or sulphate AOD) [45]. Contrary to total AOD, which can be obtained by measurement, by-species AOD needs to be inferred from other data. AOD can reflect transient aerosol events or more persistent events through

time. Background AOD denotes the baseline AOD of aerosols normally present in the atmosphere.

2.1.3 Type of Products

Atmospheric reanalysis products are datasets that describe how physical variables, such as AOD, evolved in the past. They are produced by computational models that are regularly adjusted using measurements from different types of sensors. These models also help fill in missing data. In contrast, forecast products use similar models to predict how these variables will evolve in the near future.

2.2 Existing Models for Aerosol Analysis

As discussed in Section 1.1 (Page 23), the automated tracking of atmospheric plumes can bring health benefits for populations, and help researchers in atmospheric science further understand small atmospheric events. However, existing methods are limited and rely on a a threshold-based method together with a classifier predicting whether a pixel belongs to a plume or not, or manual methods. In this chapter, we introduce the methods related to VOT existing in the atmospheric literature.

2.2.1 VOLCAT

Monitoring volcanic aerosols is critical in atmospheric science to quantify their effects on climate and to predict air quality. For instance, volcanic eruptions release large amounts of sulfur dioxide (SO_2) gas and ash particles into the atmosphere. Thus, among many physical values, researchers are particularly interested in tracking SO_2 and ash clouds. One of the most advanced systems developed to that end is the Volcanic Cloud Analysis Toolkit (VOLCAT) [15]. VOLCAT is a model from the National Oceanic and Atmospheric Administration that aims to alert and monitor plumes. VOLCAT is one of the significant existing satellite-based aerosol trackers,

and provides a near-real-time detection and monitoring of volcanic ash and aerosol clouds. It uses data from geostationary and polar-orbiting satellites.

VOLCAT uses multispectral satellite imaging to detect volcanic ash clouds by analysing their distinct spectral profiles. The toolkit utilises brightness temperatures from geostationary satellites around the world and low-orbiting satellites to separate volcanic aerosols from other atmospheric components. Once detected, researchers use VOLCAT retrieval models to estimate aerosol properties, such as AOD and particle radius.

A novelty of VOLCAT is its ability to track aerosols automatically. However, its tracking ability is, in fact, based only on the detection of plumes with maximum intensity in each specific region specified by the operator [15]. VOLCAT achieves this by processing multi-spectral satellite observations (particularly in the infrared band) to identify pixels where signals, such as brightness temperature or SO_2 concentration, exceed certain thresholds. These threshold-exceeding pixels are grouped into connected regions that each represent one plume. The model then evaluates the spatiotemporal evolution of these plumes by tracking the location of the maximum intensity within the area defined by the operator. This location is provided using a threshold to define pixels of interest, which are then classified as pixels belonging to the target plume. Since VOLCAT uses threshold-based methods to estimate the location of these plumes, the model quickly tracks the dominant plume but can sometimes merge multiple nearby plumes into a single entity, or switch tracking to a new, more intense plume if one appears, depending on which feature exhibits the strongest measured value at each time step.

2.2.2 VolcPlume

Like VOLCAT, the VolcPlume platform [14] is a toolkit made for detecting volcanic aerosol plumes in the atmosphere. VolcPlume is maintained by the Laboratoire d'Optique Atmosphérique (LOA) and the AERIS/Interaction Clouds Aerosols

Radiations (ICARE) Data Center at University of Lille. VolcPlume delivers both near-real-time and back monitoring of the physical and chemical properties of volcanic aerosol plumes locally at multiple pressure levels.

A novelty of VolcPlume is the simultaneous analysis of satellite and ground-based data for both SO_2 gas and aerosols (such as ash and sulfate particles). The platform gathers data from the following satellites (with their embedded instruments): Sentinel-5P/TROPOspheric Monitoring Instrument (TROPOMI) [46], MetOp/Infrared Atmospheric Sounding Interferometer (IASI) [47], Suomi National Polar-orbiting Partnership (SUOMI-NPP)/Ozone Mapping and Profiler Suite (OMPS) [48], Aura/Ozone Monitoring Instrument (OMI) [49], and geostationary imagers (Geostationary Operational Environmental Satellites (GOES) [50], Meteosat Second Generation (MSG) [51], and HIMAWARI [52]). VolcPlume ground-based data comes from AERONET photometers, LIght Detection and RAnging (LIDAR) networks, and air quality measurement devices. VolcPlume aggregates sensor data from these satellites and ground-based data into a single sequence of maps. To track SO_2 plumes, an operator typically starts by selecting a volcano and a time range. The platform then loads the SO_2 maps, which are collocated with the physical parameters provided by the sensors. Geostationary data helps to separate the volcanic plumes from meteorological phenomena, but the annotation process remains entirely manual: by iteratively marking the position of a target plume of SO_2 from frame to frame, the user defines the plume trajectory.

2.2.3 ECMWF - CAMS

ECMWF has implemented the Integrated Forecasting System (IFS) [53], which includes an aerosol module called Integrated Forecasting System AERosol scheme (IFS-AER) [54], through CAMS [45]. This module provides forecasts of different aerosol species (e.g., dust, sea salt, black carbon) by simulating their emissions, transport, and transformations. Figure 2.1 shows the pipeline of ECMWF for data

acquisition and processing. The first step in this pipeline is data acquisition using passive instruments onboard satellites. These radiance measurements are then fed into a radiative transfer model to obtain physical parameters, which the IFS model (shown in the red dashed box in Figure 2.1) then uses to track the evolution of the physical parameters of the plumes over time. The IFS model relies on two major components: an assimilation model and a forecast model.

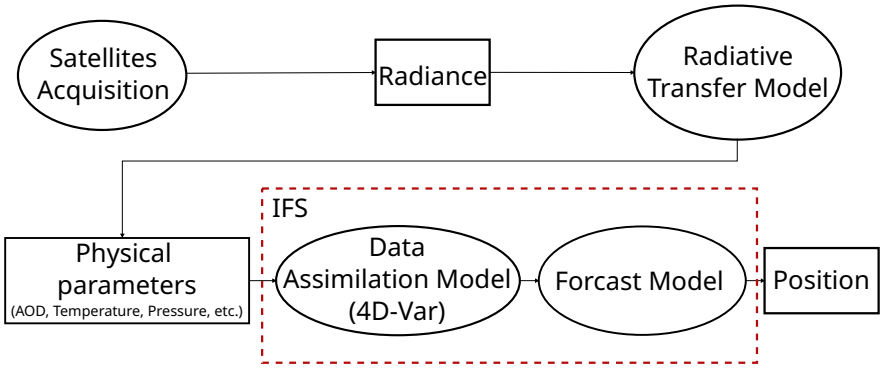


Figure 2.1: Basic steps of aerosol position prediction from ECMWF.

First, IFS relies on a data assimilation model that gathers these observations within a Four-Dimensional Variational (4D-Var) [1] framework to compute the most precise initial conditions, thereby avoiding noise and filling gaps in the acquired data. Figure 2.2 shows the assimilation process inside 4D-Var. This method consists in refining parameters (e.g., density, size distribution parameters, hygroscopic growth, or refractive index) in the four dimensions (longitude, latitude, altitude, and time) by adjusting the background guess $\boldsymbol{x}_b$ with the available observations $\boldsymbol{y}_i$ (acquired or derived from multiple sensors coming from satellites and ground devices) and prediction $\boldsymbol{x}_i$, through a time window.

The 4D-Var model takes as input an initial vector of parameters $\boldsymbol{x}_b$, called the background guess, which generally comes from the previous forecasting iteration. At each iteration, this model compares the predicted state $\boldsymbol{x}_i$ from the forecast model to the observations $\boldsymbol{y}_i$ through an observation operator $\mathcal{H}$, by computing the gradient of the cost function J via the adjoint. This gradient provides a search

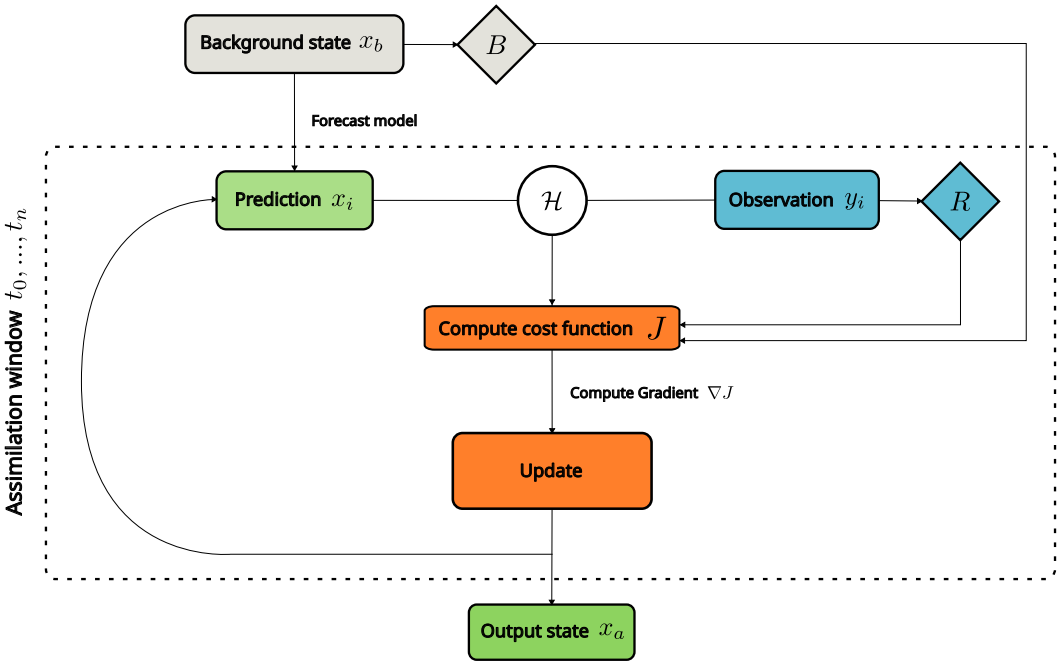


Figure 2.2: Computation process in a single outer loop for 4D-Var. It minimizes the cost function J through the assimilation window [1].

direction in the minimisation space. It is used to compute a refined state $\boldsymbol{x}_a$. This step is repeated, taking $\boldsymbol{x}_a$ as the new background guess, until convergence. In that sense, 4D-Var is a gradient-based method (the adjoint provides ∇J), but the actual minimisation is done with the conjugate-gradient [55] rather than simple gradient descent. The cost J function of this model is defined by Equation 2.1:

$$J(\boldsymbol{x}_a) = \frac{1}{2}\sum_{i=0}^{n}[\mathcal{H}(\boldsymbol{x}_i) - \boldsymbol{y}_i]^T\mathbf{R}_i^{-1}[\mathcal{H}(\boldsymbol{x}_i) - \boldsymbol{y}_i] + \frac{1}{2}(\boldsymbol{x}_a - \boldsymbol{x}_b)^T\mathbf{B}^{-1}(\boldsymbol{x}_a - \boldsymbol{x}_b) \quad (2.1)$$

where i indexes the observation times t_i within the assimilation window, $\boldsymbol{x}_a$ is the parameter vector refined from $\boldsymbol{x}_b$, and to be used as input for the forecasting model, $\boldsymbol{x}_i$ denotes the forecast of the IFS forecasting model at time t_i. $\mathcal{H}$ is the observation operator that maps the model space to the observation space, $\boldsymbol{y}_i$ is the vector of observations at time t_i, $\mathbf{B}$ is the background-error covariance matrix associated with the background state $\boldsymbol{x}_b$, and $\mathbf{R}_i$ is the observation-error covariance

matrix for $\boldsymbol{y}_i$. Matrices $\mathbf{B}$ and $\mathbf{R}_i$ are obtained through statistics of past forecast errors for $\mathbf{B}$, and from documented instrument uncertainties for $\mathbf{R}_i$.

Then, IFS uses a forecast model using $\boldsymbol{x}_a$, the output of 4D-Var, as input to predict these values at the next time step. These initial conditions are represented as a vector of physical properties within a three-dimensional Eulerian grid (i.e., a grid composed of three-dimensional cells defined along latitude, longitude, and altitude, where each cell stores the state of the physical properties). In this grid, air and aerosol motion are computed as fluxes within a bounded, stationary cell rather than following individual air parcels. This method enables forecasts of aerosol species presence and concentration worldwide with high spatial coherence and numerical stability. This forecasting model is defined by Equation 2.2:

$$\boldsymbol{x}_{\boldsymbol{t_{k+1}}} = \mathcal{M}_{t_k \to t_{k+1}}\big(\boldsymbol{x}_{\boldsymbol{t_k}}\big), \qquad \text{with } \boldsymbol{x}_{\boldsymbol{t_0}} = \boldsymbol{x}_a \tag{2.2}$$

where, $\boldsymbol{x}_{\boldsymbol{t_k}}$ is the model state vector at time t_k, $\boldsymbol{x}_{\boldsymbol{t_{k+1}}}$ is the model state at the next time level t_{k+1}, $\mathcal{M}_{t_k \to t_{k+1}}$ is the non-linear forecast model operator from the IFS forecast model that advances the state from time t_k to t_{k+1}. $\boldsymbol{x}_{\boldsymbol{t_0}}$ is the state at the initial forecast time t_0, $\boldsymbol{x}_a$ is the first guess provided by 4D-Var (the optimized initial conditions).

For instance, to predict dust AOD fields, ECMWF first builds a background state x_b at the beginning of the assimilation window, from a previous IFS forecast. Satellite and ground-based AOD measurements are then assimilated over this window using the 4D-Var model, which then produce the state x_a, which contains 3-dimensional physical parameters (e.g., dust mass, mixing ratios, and other aerosol variables). Finally, this state x_a is used as the initial condition in the IFS forecast model.

2.3 Discussion

In this chapter, we have described models related to the tracking of atmospheric plumes. These methods illustrate various use cases that will be treated later with tracking methods in this thesis. These methods range from volcanic gas plume monitoring to aerosol forecasting. We first presented VOLCAT and VolcPlume platforms. These two platforms aggregate data from multiple sensors to characterise plumes manually. We also presented the IFS model from the ECMWF CAMS service. This model is used to predict physical parameters directly with a data assimilation model.

Despite the efficiency of these models, none of them provides automated object tracking of individual aerosol plumes in the sense used in computer vision, where coherent plume objects are followed over time with well-defined trajectories and identity preservation.

VOLCAT detects volcanic plumes with a thresholding method and a machine learning classifier to classify pixels as a volcanic plume or not. This method does not maintain the identity of a full plume over time and may mix up nearby plumes with the target plume. This problem hampers the localization of the target plume and increases the risk of losing it. VolcPlume, on the other hand, integrates multiple instruments for analysing volcanic plumes. However, the platform does not integrate any automated tracking methods. The user needs to manually mark the position of the volcanic plumes at each frame. Finally, IFS provides a precise evolution of physical parameters over time through the combination of 4D-Var and forecast models, but plumes are treated as fields of physical parameters rather than as object entities. Therefore, IFS cannot maintain the identity of a specific plume in the VOT sense.

This gap motivates the approach developed in the rest of this thesis, to bring modern VOT concepts to aerosol and gas plume tracking. Our work will be to

develop a tracker that maintains the identity of a given plume as a plain object across frames rather than following the evolution of a field of values.

Chapter 3

Visual Object Tracking (VOT)

3.1 General Concepts of VOT

VOT is a long-studied field of computer vision that focuses on finding the position of a target object inside a series of frames. Smeulders et al. [16] defined object tracking as "*the analysis of video sequences for the purpose of establishing the location of the target over a sequence of frames (time)*". More formally, let I_t denote the frame (image) at time t and b_t the bounding box of the target. Given an initial box b_0 in I_0, an object tracker estimates a sequence $\{b_t\}_{t=1}^{T}$ such that each b_t encloses the same object across the sequence until the frame I_T. VOT keeps the identity and the trajectory of objects across time. While tracking and detection share some aspects, object detection identifies a specific class of objects in an image without re-identifying them afterward. On the other hand, tracking needs an initial, manually defined bounding box of the object and re-locates it on every frame of the sequence.

VOT has evolved quickly over the last two decades. It passed from handcrafted trackers [56, 57, 58, 59], which fail in complex object sequences involving appearance changes of the target object, to more robust, deep learning-based methods [60], which extract features to create a latent representation of the target object. VOT task has been classified by researchers into two categories. The first one is Single

Object Tracking (SOT), which is defined as the process of following a specific target object through a sequence of images. This type of tracker follows exclusively the object specified by the bounding box and cannot track new objects entering the frames over time. On the other hand, Multiple Object Tracking (MOT) can process several targets at the same time, which requires assigning a unique identifier to each target object, making this task more complex.

As introduced before, tracking is viewed as an iterative process that estimates the position of a target object over time in a sequence of frames. Figure 3.1 shows the general workflow of VOT. This process starts from an initial object area in frame I_t. The motion model first predicts where the target is most likely to be in the next frame, defining a region of interest in frame I_{t+1}. Within this region, the appearance model evaluates candidate locations using a representation of the visual appearance of the target, then selects the one that best matches this representation as the new object location. The tracker then updates its internal representation of the target, so that gradual changes in appearance, scale, or illumination can be taken into account. This algorithm is repeated with the same prediction, matching, and update cycle from $t+1$ to $t+2$, and so on. Depending on the tracking paradigm used, the tracker can be based on an appearance model only, solely on a motion model, or on both.

As the field of tracking evolved, many trackers have been derived from this general framework. The first approaches focused on detection-free methods [16, 57], but could not handle common challenges such as changes in appearance (e.g., motion blur). Then, feature-based [16, 61] and estimation-based methods [16, 62] has been developed. These methods focus on particular features (like corners and edges) to track the target object with better inference time and robustness. Then, correlation filter-based trackers [17, 18] have been proposed: they learn a discriminative filter from the appearance of the object in the first frame and, for each new frame, apply this filter over a search region around the previous position to obtain a response

map whose maximum gives the new target location. Afterward, researchers have done extensive research on tracking-by-detection [16, 63], which does not use any kind of motion model. In this paradigm, the dynamics of the object do not need to be specified explicitly, which makes the tracker more generic and applicable to a wider range of motion patterns. From this principle have been derived multiple deep-learning-based models, capable of learning more effective appearance models (see Section 3.5 for further information on these models).

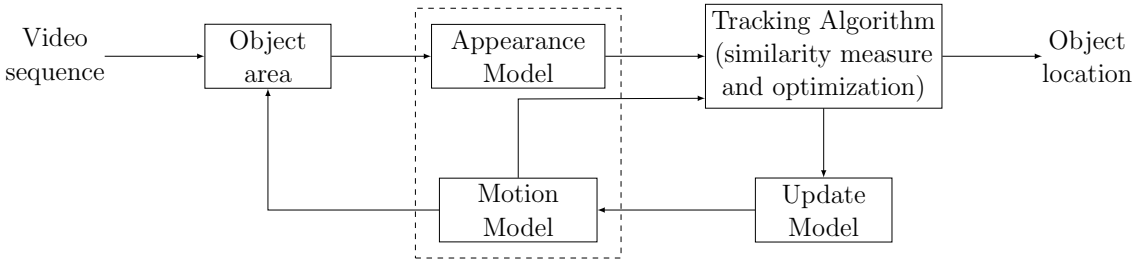


Figure 3.1: Schematic representation of an object tracker [2]. The tracker is initialised by selecting the targeted object in the first frame, which defines the initial bounding box. From this initial bounding box, the motion model estimates a region of interest. From this region of interest, the tracker extracts features to refine the position given by the motion model with a specific appearance model (e.g., a similarity measure on features). This newly given position is then used to update the motion model, and the cycle starts over for the next frame.

3.2 Challenges & Datasets in VOT

A tracker has to explore scenes to localize objects; however, the nature of these scenes highly varies in terms of composition. The appearance of the target can change due to illumination variations, deformations, scale shifts, or partial occlusions. Moreover, some noise due to the acquisition device or acquisition conditions can appear, such as motion blur (i.e., when the object makes an important displacement between frame I_t and I_{t+1}, depending on the matching between the velocity of the target and the temporal resolution of the device). Furthermore, the frame may contain background clutter, where objects similar to the target can mislead trackers. Finally, the length of sequences can be challenging for trackers. In long-

term tracking, the target is more likely to get fully occluded over a subsequence of frames. In this configuration, the tracker has to be able to detect such disappearances and successfully re-detect them when they reappear. In contrast, short-term tracking assumes that the target remains mostly visible (even though there can be partial or brief disappearances). In this setting, once the target object is lost, the sequence is considered failed, and no re-detection is usually performed.

Lastly, VOT methods must balance a trade-off between accuracy and speed. More complex models (e.g., using a large model, high-dimensional features for the appearance model, or heavy optimization schemes) generally provide more accurate localisation and are more robust to the challenges introduced before. However, their higher computational cost reduces the number of frames per second that can be processed if run on a given device. On the other hand, small models can run in near-realtime, but often at a lower localisation accuracy.

Trackers are evaluated by matching their bounding boxes predictions with ground-truth annotations, frame by frame. From these matches, several metrics are derived to quantify different aspects of performance at the sequence or dataset level. For evaluation, the metrics mainly used in the literature are:

- localisation error: This metric is derived from the bounding box misalignment between the predicted and the ground-truth. It is measured either as the Euclidean distance between bounding-box centers or as the Intersection-over-Union (IoU) between predicted and ground-truth boxes. This metric is generally not used directly for comparison between trackers since it works at the frame level, but serves as a basis for the global metrics described below;

- precision rate: precision evaluates how often the tracker predicts bounding boxes sufficiently close to the ground-truth. It is defined as the percentage of frames where the center-location error is below a fixed pixel threshold (e.g., 20 pixels);

- success rate: success rate counts the number of frames in which the IoU between predicted and ground-truth bounding boxes exceeds a threshold; by varying this threshold, one can obtain a success plot, and compute its Area Under the Curve (AUC). AUC is one of the most used metrics to compare trackers over a complete dataset.

In addition, some evaluation protocols include a failure metric, which determines when the overlaps between predicted and ground-truth bounding boxes drop below some small threshold, indicating that the tracker needs to be re-initialised with the ground-truth bounding box of the current frame. The failure rate is expressed as the number of failures per sequence. It focuses mostly on the robustness of the tracker over long sequences.

Precision rate and success rate will be used in our experiments, and will be detailed in Chapter 6. Together, these metrics quantify how accurately and how consistently a tracker can localize the target over time.

Benchmark datasets are required to evaluate trackers and compare their performance in a consistent way. Classic datasets like ImageNet Video 2015 [64], OTB [42] and Visual Object Tracking (VOT) [65] make a solid point of comparison, while modern benchmarks such as Large-scale Single Object Tracking (LaSOT) [66] or the GOT-10k [43], and the recently introduced Multi-View object Tracking (MV-Track) [67] and QuadTrack [68] datasets provide large scale, diverse, and richly annotated sequences that better reflect real-world challenges. Table 3.1 summarizes the task type (e.g., SOT, MOT), the number of sequences and frames, and the available ground-truth annotations for each dataset. Table 3.2 describes the use cases of these datasets. These datasets include a large range of scenarios, including long-term tracking, SOT, MOT, and omnidirectional video, allowing a complete analysis of model performance in diverse settings. We selected these datasets because they serve as the standard benchmarks in the recent VOT literature.

OTB is one of the first benchmarking datasets used for tracker evaluation. It

Table 3.1: Overview of commonly used VOT benchmarks.

Dataset	Task type	# seq. / frames	Annotations
ImageNet VID 2015 [64]	SOT	3,862 / ≈1.1M	Bounding boxes
OTB-2013/2015 [42]	SOT	51 / 100 seq.	Bounding boxes + attributes
VOT (annual) [65]	SOT / MOT	≈60–400 seq. (per year)	Bounding boxes / masks
LaSOT [66]	SOT	1,400 / ≈3.5M	Bounding boxes + attributes
GOT-10k [43]	SOT	>10,000 / ≈1.5M	Bounding boxes + motion, visibility
MVTrack [67]	Multi-view MOT	234,000 frames	Bounding boxes + occlusion labels
QuadTrack [68]	Panoramic MOT	32 / 19,200 frames	Bounding boxes + IDs

consists of annotated video sequences where the position of only one target object is annotated at each frame, allowing for a fair comparison of tracker performance. The most popular versions are OTB-2013 and OTB-2015, which contain 51 and 100 sequences, respectively[1]. Sequences are quite short, with a size from a few hundred up to a thousand frames. These sequences are annotated with bounding boxes and the types of challenge represented in the sequence. OTB has been designed for SOT on short sequences and a focus on challenges such as occlusions, deformations, and fast motion.

The VOT challenge is an annual benchmarking competition during which various tracking models are compared based on a precisely constructed dataset. VOT provides datasets, evaluation protocols, and toolkits for comparing the performance of short-term and long-term trackers. VOT has been designed to include both SOT and MOT challenges, as well as segmentation-based localisation. The format of the VOT challenge varies significantly from year to year. For instance, the VOT short-term 2020 dataset provides 60 sequences annotated with segmentation masks, while

[1]OTB-2015 includes the sequences of OTB-2013.

Table 3.2: Description of commonly used VOT benchmarks.

Dataset	Description
ImageNet VID 2015 [64]	Large-scale video detection and tracking benchmark built on ImageNet object classes.
OTB-2013/2015 [42]	Short single-target sequences with annotated challenges such as occlusion, deformation and fast motion.
VOT (annual) [65]	Challenge-style benchmark with standardized toolkit for short- and long-term tracking evaluation.
LaSOT [66]	High-quality long sequences with dense annotations and diverse object categories for large-scale SOT.
GOT-10k [43]	Generic tracking in the wild with non-overlapping train/test classes to evaluate generalization.
MVTrack [67]	Multi-camera synchronized views of 27 object classes, with identity consistency across viewpoints.
QuadTrack [68]	Panoramic videos (around 190,000 boxes) targeting fast motion, long-term tracking and geometric distortions.

the VOT long-term 2019 dataset comprises 50 sequences with an average length of over 4,000 frames.

LaSOT [66] is a large-scale benchmark specifically designed for single object tracking in challenging real-world conditions. It contains 1,400 long sequences, averaging more than 2,500 frames per sequence, for a total of over 3.5 million annotated frames. Each sequence focuses on a single target and is annotated with bounding boxes. LaSOT also includes attributes such as the type of challenge (e.g., occlusion). LaSOT is used to evaluate the long-term robustness and large-scale performance of SOT trackers, emphasizing long-term target disappearance and tracking across multiple object categories.

GOT-10k is a highly diverse dataset designed for generic object tracking in

the wild. It contains over 10,000 sequences and more than 1.5 million annotated bounding boxes, covering around 560 object classes and around 80 patterns. Its training and test sets are non-overlapping in terms of object classes, which makes it a strong dataset to evaluate the generalisation capability of trackers. The dataset provides additional annotations of object visibility ratios and types of motion (e.g., jogging, swimming, flying, jumping) described by the classes. GOT-10k is used to evaluate the generalisation capacity of trackers on non-overlapping test classes.

MVTrack is a dataset built to fulfill the need for multi-view object tracking (i.e., tracking the same target simultaneously in videos acquired from multiple synchronized cameras with different viewpoints, while maintaining a consistent object identity across views and time). It is made of 234,000 annotated frames acquired from multiple camera angles and positions. This dataset consists of 27 object classes and multiple sequences per class. Each frame is annotated with bounding boxes and occlusion labels, which allow for evaluating the trackers across multiple occlusion configurations. MVTrack is used for the benchmarking of trackers that can handle multiple viewpoints of the same object.

QuadTrack is a recent benchmark dataset used for MOT in complex environments. Images are collected using a panoramic camera to get each angle of the tracked object. QuadTrack offers multiple challenges, such as fast-moving objects and long-term tracking scenarios. It also introduces a new challenge by tracking a target object on panoramic frames. This dataset is made of 32 video sequences containing 19,200 frames and around 190,000 bounding boxes, with a duration of 60 seconds for each sequence. QuadTrack focuses on the new challenge of tracking in panoramic frame, which induces geometric distortions.

Overall, these benchmarks cover complementary challenges of VOT, from early OTB to LaSOT, GOT-10k, and more specialized settings such as MVTrack and QuadTrack. In the rest of this thesis, OTB and GOT-10k are used as the primary evaluation datasets for generic objects, given their widespread adoption in the SOT

literature.

3.3 Appearance Models

Appearance models can be grouped into several families of methods, according to how they model the target object and its motion. Generative methods [69] build a model of the appearance of the target and locate the target in each new frame by searching for the image region that best fits this model (without explicitly learning to discriminate the target object from the background). Discriminative methods [63], on the other hand, treat tracking as a classification task and learn a decision function that separates the target object from background regions, often relying on hand-crafted or deep feature representations. Filtering and probabilistic methods [62] define tracking as a recursive state-estimation problem. They predict and update a state, such as the position and velocity of the target, over time. These methods use motion and observation models like Kalman or particle filters. Finally, feature-based methods [70, 57, 56, 58, 59] focus on constructing handcrafted visual properties, such as gradient, texture, colour, and shape. These properties are used to match the target object across frames under various conditions. This taxonomy is based on the surveys by Soleimanitaleb et al. [71] and Smeulders et al. [16]. Figure 3.2 illustrates this taxonomy.

3.3.1 Generative Methods

Generative appearance models aim to describe how target objects are generated by approximating a likelihood. Precisely, the tracker builds a target model and, in each frame, searches for the candidate region that maximises this likelihood approximation. In practice, generative trackers often rely on simple intensity templates [69], colour histograms [57], or more sophisticated subspace representations [72] to capture the target appearance. Tracking then becomes a template-matching prob-

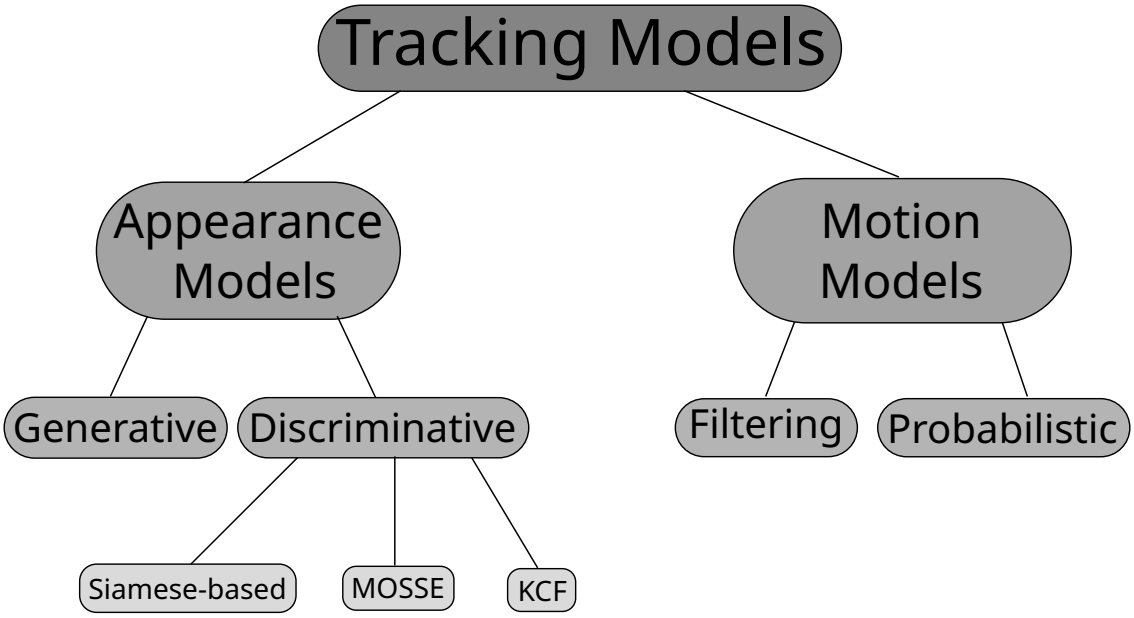


Figure 3.2: Taxonomy of VOT methods used in this thesis. Methods are grouped by appearance (generative or discriminative) and motion (filtering or probabilistic) models.

lem [16] in which the algorithm scans the scene and selects the region that best fits the learned target model, without explicitly learning a decision boundary between target and background, in contrast to discriminative methods (which will be introduced in Section 3.3.2).

Early methods relied on fixed templates and struggled to handle strong appearance changes over time. Later work on template-matching generative methods [72] introduced models with greater generalisation capacity, allowing the template to adapt to the appearance evolution of the target object. For example, expectation–maximization schemes [16] introduce learning to update the appearance model, improving robustness to gradual variations of appearance.

More sophisticated subspace-based generative trackers [16, 72] project the target appearance into a lower-dimensional subspace, often via principal component analysis [16]. These methods capture a wider range of appearance variations and are more robust to noise and minor deformations.

Recently, some generative trackers [73] have integrated learning mechanisms to

estimate the appearance model from large-scale data. This improves robustness in complex sequences while retaining the generative matching principle.

In summary, generative methods have the strength of providing an explicit model of the target appearance, which makes them easy to implement. However, these approaches often struggle in cluttered scenes where background regions can accidentally fit the target model, and they tend to be less discriminative than methods that directly learn a boundary between target and background. Moreover, maintaining and updating a generative model that remains reliable across long sequences is challenging, limiting its performance in the presence of severe occlusions, strong deformations, or illumination changes.

3.3.2 Discriminative Methods

Discriminative methods [74] have become the major methods in VOT. They frame object tracking as a classification problem. These methods use classification models (with handcrafted or learned features) to separate the target from the background and produce a decision threshold for each region of interest, rather than only measuring the likelihood of a target to an appearance model as in generative approaches. The classifier can be updated to adapt to changes in the object appearance, the illumination, and the scene context.

The most common discriminative trackers include correlation filter-based methods [17], where the decision function is implemented as a learned filter that produces higher responses at the location of the target than on the background, and more general classifier-based trackers that rely on models such as Support Vector Machines (SVM) [75]; both categories have demonstrated strong performance in challenging datasets. Benchmark studies show that discriminative trackers generally outperform classical generative methods on standard datasets such as OTB-2013 and OTB-2015. On OTB-2015, discriminative correlation-filter trackers typically achieve success AUC scores in the range of 0.55–0.60 on average [76], whereas clas-

sical generative template and subspace-based trackers reported in the same studies remain around 0.45–0.50 AUC on average [76], confirming the advantage of discriminative appearance models on these benchmarks. Especially, discriminative methods are more effective than generative ones in cluttered scenes. Indeed, generative approaches can confuse the target with the background, as they do not model the latter explicitly. By modelling the difference between the target and the background, discriminative methods can perform better in the presence of occlusions, illumination changes, and similar-looking objects. Moreover, discriminative models have a lower inference time, making them suitable for near-real-time applications.

However, the effectiveness of discriminative methods depends critically on the quality and representativeness of the model features. Handcrafted or shallow features, in particular, may struggle when there are significant appearance changes or non-rigid deformations [77]. More recent work [60] integrates the learning of deep feature representations.

Another form of discriminative methods is correlation filter-based trackers. They learn, from the first frame, a filter such that the correlation between this filter and a search region produces a maximum of response at the location of the target object. Correlation-based methods such as the MOSSE introduced by Bolme et al. [17], or KCF from Henriques et al. [18] achieve high inference speeds (often hundreds of frames per second on CPU [18]) while maintaining accuracy on benchmarks like OTB-2013 [78].

3.3.3 Feature-Based Methods

In this section, we describe handcrafted and machine learning appearance models used in generative and discriminative models. Handcrafted feature-based [74] methods offered effective object representation before deep learning was adopted. These methods use designed features [56, 70, 58, 57], based on texture, colour, shape, or edges. This design allows tracking under various conditions. Handcrafted feature-

based tracking methods aim to extract descriptive features from the target object, which are used to identify and localize the object in subsequent frames. Common feature descriptors include:

- Histogram of Oriented Gradients (HOG) [56]: they extract local edge features by computing the distribution of gradient orientations within localized regions. Histograms of oriented gradients are particularly effective for tracking objects with sharp contours.
- Local binary patterns [70]: they encodes local texture by thresholding pixel neighborhoods, providing robustness to changes in illumination, as well as minor deformations.
- Colour histograms [57]: they represent the distribution of colours within the target and local regions. This method is robust to partial occlusions and small appearance changes.
- Gabor filters [58]: they extract texture features. This model performs well for distinguishing objects with complex surface patterns.
- Hough transform [59]: it detects geometric shapes such as lines or circles, and is suited for tracking objects with regular structures.

Feature-based methods use similarity measures (e.g., the Euclidean distance, correlation measures) or classifiers (such as SVM) to perform object matching or discrimination.

These methods are computationally efficient, making them suitable for near-real-time applications. They are particularly effective when the target object is very distinct from the background, when it has stable properties (e.g. strong colour, clear edges or texture patterns), and when training data is scarce or unavailable. However, these methods can struggle with significant appearance changes, non-rigid deformations, and complex backgrounds, as their performance relies strongly on the

quality of the chosen representation of the descriptors. Designing effective features often requires extensive experimentation.

Despite these advantages, classical correlation filter-based trackers still suffer from important limitations. Their linear formulation and reliance on relatively simple handcrafted features (e.g., grayscale intensities, HOG, colour names) restrict their discriminative power in cluttered scenes or in the presence of similar distractors, and they are particularly sensitive to significant appearance changes, heavy occlusions, and complex background motion.

While deep learning has surpassed handcrafted features in state-of-the-art trackers, recent work [77, 79] has explored hybrid approaches that combine handcrafted and learned features. Moreover, handcrafted features remain relevant in applications where interpretability, computational efficiency, or data scarcity are essential.

3.4 Motion Models

Estimation-based [16] methods form a specific class of visual object tracking methods. These models focus on the prediction and update of the object state (i.e., its position, velocity and acceleration) through a sequence using motion models. These methods use the temporal consistency over sequences, extracting specific physical features (e.g., displacement between consecutive positions, direction, or centre of mass) from previous frames to maintain an accurate tracking even under challenging conditions like occlusions.

Estimation-based trackers cast VOT into a recursive process. At each frame, the algorithm predicts the next state of the target object using a motion model, then adjusts this prediction based on observed position of the target. This process allows the tracker to handle occlusions and missing detections by estimating the direction of the target object. Among motion models, Kalman filters [80] and particle filters [62] show effective results for VOT.

Kalman filters are among the most used techniques in VOT. These methods assume that the state of the object (such as its position and velocity) follows a simple linear motion model, while both the motion and the measurements are affected by Gaussian noise. At each time step, the Kalman filter first predicts the new state from the previous one using the motion model, then recursively adapts this prediction by combining it with the current noisy position of the target given by a detector (learned or not). The covariances of the prediction and the measurement quantify how uncertain each source of information is; the filter computes a Kalman gain that automatically gives more weight to the source with lower uncertainty and less weight to the source with higher uncertainty. Kalman filters are efficient and have a short inference time for targeted objects having a smooth motion. They are particularly efficient when the target follows a smooth, approximately linear motion between frames and when the process and measurement noise can be reasonably modelled as Gaussian.

An example of an estimation-based tracker is Simple Online and Realtime Tracking (SORT) [81], which combines a Kalman filter with the Hungarian algorithm [82] for data association for MOT. In this framework, detections provided by an external object detector are treated as noisy measurements, while the Kalman filter predicts each future state of the target object based on its past motion. By linking detections to predicted positions, SORT achieves fast, online tracking in frames with several targets.

Particle filters extend this framework to non-linear sequences. These methods represent the state distribution as a set of weighted particles, each particle corresponding to a possible object state. Particle filters are particularly useful for tracking objects with sudden motion changes or in cluttered backgrounds, as they can represent complex, multi-modal state distributions.

While improving the robustness of the tracker compared to generative or discriminative methods, estimation-based methods depend highly on the accuracy of the

underlying motion and appearance models. More recently, researchers have explored some hybrid approaches that combine the strengths of different estimation-based techniques [62]. For example, some authors [83] have integrated deep learning-based methods with state estimation methods, using deep learning features to create complex motion models.

3.5 Deep Learning-based Methods

Deep learning has significantly advanced VOT by providing powerful appearance models and learned motion representations. In this section, we review representative deep learning-based trackers that illustrate the main design choices adopted in modern VOT. We first present the Multi-Domain Network (MDNet) [3], which formulates tracking by detection with a multi-domain Convolutional Neural Network (CNN) classifier, then Deep Simple Online and Realtime Tracking (DeepSORT) [83], which combines a deep appearance embedding with an estimation-based motion model for MOT. We then discuss the use of Long Short-Term Memory (LSTM) [84] and CNN architectures [85] together [86] to explicitly model temporal dependencies. Finally, we focus on SiamFC from Bertinetto et al. [4] and Siamese-based trackers in general [4, 87, 88].

3.5.1 MDNet

MDNet is a tracking by detection model that uses a CNN as an appearance model for tracking targeted objects in sequences. The network architecture consists of a single backbone of convolutional layers followed by domain-specific branches (see Figure 3.3). MDNet is trained using a multi-domain learning method: it includes several specialised neural networks trained for specific sequences. However, before training any of these specialized parts, the convolutional layers of backbone are pretrained to capture a generalized representation of features shared among all

sequences. This shared feature space makes MDNet able to represent a wide range of object appearances.

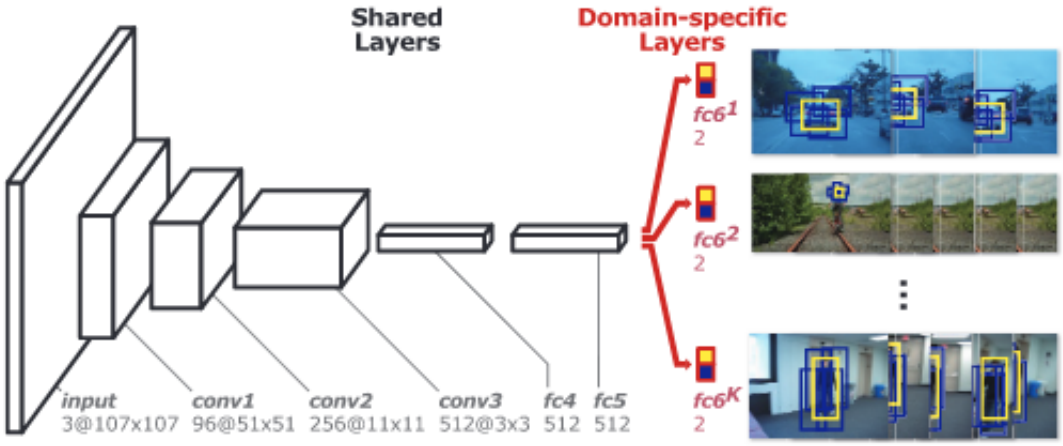


Figure 3.3: MDNet architecture [3]. MDNet is composed of a feed-forward convolutional backbone that learns general features from large-scale visual tracking data. Then, domain-specific layers are trained for specific tracking tasks. This figure comes from Nam et al.'s paper [3]

For new sequences containing a new target object, MDNet removes the domain-specific layers and updates the shared representation online using a binary classification layer. This online adaptation allows the tracker to quickly specialise to the appearance of the new target in the current sequence, making it able to separate it from the background. Such a mechanism makes MDNet robust to occlusions, deformations, or illumination changes.

MDNet has outperformed traditional trackers on commonly used datasets such as OTB and VOT [3]. However, its gain in robustness comes at the cost of a low computational speed, as the need for frequent online updates makes MDNet slower than some real-time trackers.

Research has since enhanced MDNet [89] making it faster for feature extraction, with a more discriminative loss function, and an introduced region-of-interest alignment [90] (a precise way to extract fixed-size features from candidate regions in the feature map) to increase its speed and accuracy.

3.5.2 DeepSort

DeepSORT [83] is one of the most used architectures for MOT. This model is based on the original SORT algorithm. While SORT relies only on a motion model using a Kalman filter and the Hungarian algorithm for data association, DeepSORT introduces a deep learning-based appearance model. Precisely, DeepSORT uses a CNN to extract discriminative appearance features for each target object. This feature space is then used in addition to the motion model to predict the position of each object in subsequent frames, reducing the incidence of identity switches and improving robustness in crowded or occluded scenes.

Compared to SORT and MDNet, DeepSORT offers a balanced trade-off between robustness and computational cost. By adding the Kalman filter and Hungarian data association used in SORT with a deep appearance model, DeepSORT significantly improves robustness under occlusions and in scenes with multiple targets, where motion information alone is often ambiguous. At the same time, it remains far more efficient than trackers such as MDNet, since the appearance network directly performs the re-identification of the target object, without the need for full online model adaptation for each target. However, this design also introduces limitations. DeepSORT still depends on an external detector; its performances degrade when detections are noisy or missing. Its appearance model is less adaptive than the features of MDNet, which can handle more severe, long-term appearance changes thanks to their online adaptation, but at the cost of a much higher computational cost.

3.5.3 LSTM and CNN

Deep learning-based trackers encompass a wide range of architectures, including models where Recurrent Neural Networks (RNN) [91] act as learned motion models. One of the largest sets of these models is the LSTM + CNN tracker. Among these

trackers, a representative example is the Recurrent you Only Look Once (ROLO) model [86], which has been one of the first models to combine deep motion and appearance models together in VOT. In ROLO, a CNN, similar to the one of You Only Look Once (YOLO) detector [92], processes each frame from a sequence to extract spatial features, and generates bounding boxes for all detected objects, which are then concatenated and passed as input to a LSTM [84] network, which regresses the final bounding box and implicitly models the trajectory of the target over time.

ROLO showed that using CNNs and LSTMs can be effective for online, real-time tracking tasks, although its speed (10 frames per second) remains low [86]. The CNN provides rich spatial representations, while the LSTM extracts temporal context to smooth trajectories and recover from missed detections. This combination allows LSTM + CNN trackers to outperform previous deep learning-based trackers like MDNet and DeepSORT. Their modular design also allows for flexibility, as different CNN backbones or LSTM configurations can be adapted to specific application requirements or computational constraints. Despite these advantages, CNN + LSTM trackers, such as ROLO, also have limitations. Their recurrent structure increases their computational cost, as compared to purely feed-forward architectures. Moreover, modelling long-term temporal dependencies with LSTMs can be difficult in the presence of dramatic appearance changes, scale variations, or fast motion, which may lead to loss of the target object.

A variant of ROLO is Tiny-ROLO [93], which replaces the original YOLO detector with Tiny-YOLO [94] in order to reduce the computational cost. In this configuration, the LSTM component of the tracker is unchanged, but the detection is faster. This variation increases the overall processed frames per second, but at the cost of reduced localisation accuracy. Evaluations [93] show that Tiny-ROLO generally tracks less accurately than ROLO, which highlights how the choice of detector strongly governs the final trade-off between speed and precision in CNN +

LSTM trackers.

3.5.4 SiamFC and Siamese-based Trackers

SiamFC [4] is a Siamese network, that is, a network composed of two branches that share the same backbone (i.e, shared weights). The particularity of SiamFC is the replacement of the fully connected layers with convolutional layers in its AlexNet [9] backbone. This replacement enables the use of images of different sizes as input for each branch of the Siamese network. This method is fast and efficient, but is only suited to SOT.

Figure 3.4 shows the global architecture of SiamFC. This architecture learns similarities between the object to track (represented by an exemplar image, that is, a patch of the first frame cropped around the target object) and a search region (called the candidate image), defined as a region of the next frame cropped around the previous location of the target. To this end, a similarity function f is trained offline. This function computes the cross-correlation between the output of the exemplar branch and the output of the candidate image branch. This cross-correlation f outputs a map of intensities. The position of the target is at the maximum intensity of this map. The map is then interpolated with bilinear interpolation to match the coordinate system of the candidate image. Equation 3.1 provides the definition of the similarity function:

$$f(z, x) = \varphi(z) * \varphi(x) + b \tag{3.1}$$

where $*$ is the cross-correlation operator, $z \in \mathbb{R}^{H_z \times W_z \times C}$ is the exemplar image fed to the examplar branch (with H_z, W_z, and C being the height, width, and number of channels of the examplar image, respectively), $x \in \mathbb{R}^{H_x \times W_x \times C}$ is the instance image fed to the instance branch (with H_x and W_x being the height and

width of the instance image, respectively), $b \in \mathbb{R}$ is a similarity margin parameter which shifts all values in the output similarity map to center the distribution of the output similarity map, so that it is compatible with the logistic loss [95], and φ is the feature extractor model (typically, an AlexNet backbone).

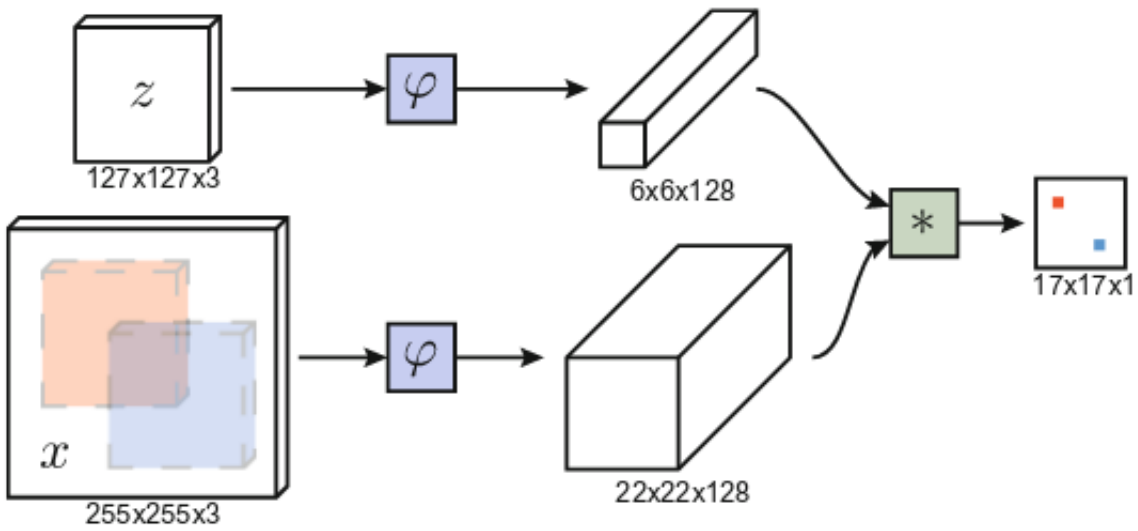


Figure 3.4: Workflow of SiamFC architecture [4]. This figures has been used from the paper of Bertinetto et al. [4].

Compared to the previous deep learning-based trackers, SiamFC offers a trade-off between accuracy and computational cost. As a fully convolutional, purely feed-forward Siamese network trained entirely offline, SiamFC avoids per-sequence online adaptation as in MDNet, and recurrent processing as in ROLO. This enables fast object tracking beyond real time (i.e., between 50 and 100 frames per second, depending on the actual architecture) while still learning a discriminative appearance model from large-scale data. However, this design also has limitations. In its original paper, the target object is extracted from the first frame and kept fixed, and the network parameters are not adapted online. As a result, SiamFC can struggle with large, long-term appearance changes of the target. Furthermore, unlike DeepSORT, it does not explicitly handle MOT.

Several authors have tried to improve the original SiamFC, such as in SiamMask [96], the Siamese Region Proposal Network (SiamRPN) [88], Siamese Multi-Task (SiamMT) [36],

and SiamFC++ [87]. Each of these architectures has brought novelty and increased performance, but also increased the computational cost of SiamFC.

SiamMask is a subtype of Siamese-based model derived from SiamFC that simultaneously performs visual object tracking and semi-supervised object segmentation. Initialized with a single bounding box in the first frame, SiamMask predicts both the bounding box and a binary segmentation mask for the target in each subsequent frame. This ability allows SiamMask to track object location and provide pixel-level segmentation, making it effective when precise object boundaries are important.

SiamRPN is also a Siamese tracking model that integrates a region proposal network, inspired by advances in object detection such as Faster Region-based Convolutional Neural Network (Faster R-CNN)[97]. This allows SiamRPN to generate multiple candidate regions and select the most likely location for the targeted object, improving tracking accuracy and robustness, especially in the presence of distractors or significant appearance changes. The region proposal mechanism enables more precise localisation than earlier Siamese trackers.

SiamFC++ is based on the original SiamFC from Bertinetto et al. [4]. It retains its fully convolutional Siamese design, with new branches and losses. In contrast to SiamFC, which only performs a binary classification of the target object against the background by learning a cross-correlation operator, SiamFC++ also predicts the dimensions of the bounding box (from a regression branch) and a quality score for each location on the map. So, SiamFC++ makes no assumption on the dimensions of the bounding box of the target object. This design removes the need for manually defined fixed scale priors and makes the tracker more flexible when the target object undergoes large changes in size and aspect ratio. This multi-objective training encourages the network to produce classification scores that are less ambiguous, at the cost of a small computational overhead. During inference, the final tracking decision combines classification and quality scores, enabling SiamFC++ to suppress these distractors that may have high raw similarity, but do not correspond to the size

of the original target object, and thus improves robustness and accuracy compared to SiamFC.

SiamMT extends the siamese-based trackers by adding multiple tasks, such as classification, localization, and sometimes segmentation, within one architecture. By simultaneously optimizing these tasks, SiamMT improved the discriminative ability of the learned features and enhanced tracking performance.

Finally, SiamFC and its derived models are trained entirely offline, enabling high-speed, near real-time tracking without the need for online adaptation during inference.

3.5.5 Transformer-based Trackers

Transformer-based trackers were introduced more recently in VOT. This deep learning-based model relies on the success of transformers in image classification [98] and object detection [99]. These models use self-attention mechanisms to identify and re-identify the targeted object. In the literature [100, 101], authors have explored multiple ways of integrating transformers. One common approach is the hybrid Siamese-transformer trackers, extending the classical SiamFC architectures from Bertinetto et al. [4]. These models replace the convolutional backbone with transformer blocks. More recent approaches propose a purely transformer-based and single-branch architecture. These models have rapidly achieved state-of-the-art performance on challenging benchmarks such as LaSOT and GOT-10k [102], particularly in scenarios involving cluttered backgrounds, large appearance changes and long-term disappearance of the target.

A representative hybrid approach is the TransT tracker [100]. TransT uses a Siamese-like tracker architecture with a feature extraction backbone, but replaces the classical correlation-based similarity estimation with an attention-based feature fusion network implemented with transformers. In TransT, the exemplar and search images are first processed by a shared CNN to produce feature maps, which are then

fed to two transformer modules. The first one enhances each feature map with its own global context, and includes a cross-feature module based on cross-attention that explicitly models the interactions between exemplar and search features. This attention-based fusion operation enables the tracker to adaptively focus on informative regions and suppress distractors, effectively replacing the fixed-correlation operation of SiamFC [4]. As a result, TransT achieves higher accuracy and robustness than classical Siamese trackers [4, 87, 88] on multiple benchmarks while still operating at near real-time speed.

Beyond hybrid architectures like TransT, a second family of methods in the literature uses transformer models. This family of models rely on Single Branch Tracking (SBT) architectures [101]. Instead of using a Siamese branch for the exemplar and search images, these models use a common branch that jointly encodes the exemplar image and the search image via a sequence of self-attention and cross-attention layers. The target is represented either by dedicated tokens or by embedding the initial bounding box into the transformer input. The model predicts the target location at each frame by decoding these tokens. This single-branch formulation can simplify the overall pipeline and add the global contextual modelling capacity of transformers. Overall, single-branch transformer trackers achieve competitive or superior accuracy compared to hybrid Siamese-transformer-based models [100], but often at the cost of higher memory usage and increased computational complexity.

3.6 Conclusion

VOT has gradually evolved from early handcrafted and analytic approaches to deep learning-based models that rely on learned feature representations and powerful similarity or classification functions. Along this evolution, the field has converged toward a common problem formulation (SOT vs. MOT, short-term vs. long-term tracking) and a shared set of benchmarks such as OTB, VOT, LaSOT,

GOT-10k, MVTrack, or QuadTrack, which provide standardized evaluation protocols and cover increasingly complex scenarios.

Classical tracking methods can be grouped according to their appearance and motion models. On the appearance side, generative trackers build an explicit model of the target and search for regions that best match this model, whereas discriminative trackers learn a decision function that separates the target object from the background. In practice, discriminative methods based on correlation filters or classifiers tend to outperform generative ones on the standard benchmarks presented here. This is especially the case in cluttered scenes and in the presence of similar distractors, because they explicitly model the difference between target and background rather than only the target itself. On the motion side, Kalman and particle filters provide ways to predict the position of the target object over time by estimating the trajectory of this object. These models improve robustness under partial occlusions or short-term disappearance compared to appearance-only trackers. Despite these advances, combinations of handcrafted appearance models and classical motion models still face a trade-off: simple models are fast but not robust enough to strong appearance changes, long-term occlusions, or complex backgrounds, while more elaborate formulations increase the computational cost and remain difficult to scale to long sequences or large datasets. These limitations on both robustness and efficiency motivate the transition to deep learning-based architectures, which can learn richer appearance representations and exploit temporal information.

Deep learning-based methods, such as MDNet, DeepSORT, CNN–LSTM trackers, Siamese architectures (i.e., SiamFC and its derivatives) and transformer-based models, have significantly improved tracking performance on standard benchmarks by combining more expressive appearance models with more flexible motion or association strategies. At the same time, these models introduce new constraints, in particular in terms of training data requirements, model size, and inference speed, which are critical when tracking has to be performed in real time or on limited

hardware resources.

Table 3.3 summarises all the trackers described in this background chapter, along with their core concepts and main characteristics to facilitate comparison. In summary, SiamFC offers a favourable trade-off between localisation accuracy and computational cost compared to the other deep learning-based trackers reviewed in this chapter. While MDNet and ROLO achieve strong robustness through online adaptation or recurrent modelling, their inference time remains significantly below real-time, which is incompatible with applications that require to process long sequences. In contrast, SiamFC and its derivatives maintain real-time or even beyond real-time performance (around 50–100 frames per second depending on the backbone and hardware) while providing a competitive appearance model on standard VOT benchmarks such as OTB and GOT-10k. SiamFC variants like SiamRPN, SiamMask or SiamFC++ improve the accuracy and robustness of SiamFC, but at the cost of increased architectural complexity, memory consumption, and training and inference cost. Transformer-based model improve accuracy further, but a cost of additional complexity. It should be noted that transformer-based models were released after the beginning of this work, and most of them are build on Siamese architectures.

For the rest of this thesis, we focus on adapting SiamFC from Bertinetto et al. [4] to SNNs. Since more performant trackers (such as SiamFC variants and transformer-based architectures) use the same core architecture as SiamFC, it constitutes the best starting point to develop the first, directly trained SNN trackers.

Table 3.3: Representative VOT trackers reviewed in this thesis, from classical correlation filters to Siamese CNNs and transformer-based architectures.

Tracker	Family	Core idea	Main characteristics
MOSSE [17]	Correlation filter	Linear correlation filter with grayscale	Simple and fast, but limited robustness to appearance changes.
KCF [18]	Correlation filter	Kernelized correlation filter with colour images	Stronger than MOSSE, relies on handcrafted features.
SiamFC [4]	Siamese CNN	AlexNet-like backbone, cross-correlation head	Simple fully-convolutional Siamese baseline, offline training.
SiamRPN [88]	Siamese CNN	Deeper backbone + region proposal head	Higher accuracy than SiamFC; more complexity.
SiamFC++ [87]	Siamese CNN	Anchor-free classification + regression heads	Improves box estimation and accuracy; increased architectural complexity.
TransT [100]	Siamese + Transformer	CNN backbone + transformer feature fusion	Hybrid Siamese–transformer; state of the art accuracy; heavier attention modules.
SBT [101]	Transformer-based	Pure transformer tracking framework	state of the art accuracy; high model and computational complexity.

Chapter 4

Spiking Neural Networks

SNNs are a family of neural models in which information is carried by discrete spikes evolving over time, rather than by continuous activations as in ANNs [21]. Instead of updating all neurons at every layer synchronously, SNNs update membrane potentials only when spikes occur, leading to sparse, event-driven computation well suited to low-power neuromorphic hardware and to sensors that produce spike trains. In this chapter, we introduce the main requirements for building and training SNNs, then focus on how these models can be used for VOT, with an emphasis on spiking Siamese architectures.

4.1 Core Concepts of Spiking Neurons

Spiking neurons provide the basic computational units of SNNs by describing how inputs are integrated and when spikes are emitted. Different neuron models have been proposed, depending on the trade-off between biological realism and computational efficiency, from simple spike integrator models to more detailed ion channel models. This section introduces the main spiking neuron models and neural coding schemes that will be used in the remainder of the thesis, and provides the necessary background to understand the SNN-based tracking methods presented later.

4.1.1 Spiking Neuron Models

Spiking neurons are a large set of neuron models. Each of them can be used for specific problems. Among the most influential models are the Integrate-and-Fire (IF) model [103], the Leaky Integrate-and-Fire (LIF) model [22], Hodgkin-Huxley (HH) model [104], and Spike Response Model (SRM) model [22], each offering distinct properties in computational complexity and representation capacity [105].

Before detailing these models, we briefly recall a few basic concepts common to most spiking neurons. The internal state of a spiking neuron is represented by its membrane potential. Its value increases with Excitatory Postsynaptic Potentials (EPSPs) and decreases with Inhibitory Postsynaptic Potentials (IPSPs), which are generated by presynaptic spikes at excitatory and inhibitory synapses, respectively. When the membrane potential crosses a given firing threshold, the neuron emits an output spike and its internal state is reset. After each spike, the neuron may enter a refractory period, during which the neuron cannot, or is less likely to, emit another spike. In basic IF models, this is always implemented explicitly, whereas in models like HH [104] or Izhikevich neurons [105], this feature is inherent to the mathematical framework of the models. The different spiking neuron models mainly differ in how they describe the evolution of the membrane potential, the form of the synaptic input, and the implementation of the refractory mechanism [22].

The IF model [103] is one of the simplest models of spiking neurons. In this model, the membrane potential of a neuron accumulates incoming synaptic inputs over time. When the potential reaches a fixed threshold, the neuron emits a spike and the potential is reset, often followed by a refractory period. The IF model provides a computationally lightweight model for simulating deep neural networks. Its simplicity, however, comes at the cost of a more restricted representation capability of complex problems. The behaviour of this neuron in discrete time is modelled by Equation 4.1:

$$V_i(t) = V_i(t-1) + \sum_{j=1}^{J} W_{ij} S_j(t-1) \tag{4.1}$$

where the output spike of neuron i at time t is defined as:

$$S_i(t) = \begin{cases} 1, & \text{if } V_i(t) \geq V_{\text{th}}, \\ 0, & \text{otherwise}, \end{cases}$$

and the membrane potential is soft reset after a spike as:

$$V_i(t) = V_i(t) - V_{\text{rest}} \quad \text{if} \quad V_i(t) > V_{\text{th}}$$

where $V_i(t)$ is the membrane potential of the neuron i at time t, $S_j(t)$ represents the input presynaptic spike that arrives at time t for neuron j, W_{ij} is the synaptic weight from neuron j to neuron i, and J is the number of presynaptic neurons. When the membrane potential $V_i(t)$ crosses the threshold V_{th}, the neuron emits a spike and its membrane potential is reset by subtracting the threshold, $V_i(t) = V_i(t) - V_{\text{th}}$. This behavior is called soft reset.

The LIF model [22] introduces a leak term, representing the passive decay of the membrane potential toward a resting value in the absence of input. This is typically modelled as a resistor-capacitor circuit, where the resistor accounts for the leakiness of the membrane. Variants of the LIF model, such as adaptive [22] and generalized forms [106], have been developed to better reproduce the diverse firing patterns observed in the hope of improving the representation capability of the neurons. The LIF model is defined in Equation 4.2:

$$V_i(t) = \lambda V_i(t-1) + \sum_{j} W_{ij} S_j(t-1) \tag{4.2}$$

with

$$\lambda = -\frac{1}{\tau_m}$$

where $V_i(t)$, $S_j(t)$, W_{ij} are the same as Equation 4.1, λ is the leak factor where $0 \leq \lambda \leq 1$, and τ_m is the membrane time constant which regulate the decrease of the membrane potential. Compared to the IF neuron, the LIF model provides a more realistic temporal integration of presynaptic spikes, while remaining simple enough to be efficiently used in deep SNN architectures.

The HH model [104] is used mostly for biophysical realism in neuron modeling. It describes the generation and propagation of action potentials through a set of nonlinear differential equations that model the dynamics of voltage-gated ion channels. The HH model accurately reproduces the shape, timing, and refractory behavior of real neuronal spikes, making it a good candidate for studying the detailed mechanisms of neural excitability. However, its high computational cost limits its use in large-scale simulations and real-time neuromorphic systems, where simpler models like IF or LIF are preferred for efficiency.

The SRM [22] is a kernel-based approach to spiking neuron modelling. Instead of directly simulating membrane dynamics, the SRM describes the membrane potential as a sum of response kernels triggered by incoming spikes and the firing history of the neuron. This allows the SRM to efficiently capture both the effects of synaptic inputs and the aftereffects of spiking, such as refractory periods. The flexibility of the model makes it suitable for a wide range of theoretical and computational studies, particularly where the precise timing of spikes and their influence on network behavior are of primary interest.

A Convolutional Spiking Neural Network (CSNN) [107] is a spiking neural network in which neurons are connected via convolutional filters, so that spikes are processed through spatial convolutions.

4.1.2 Neural Coding

Before using SNNs, real value data needs to be converted into spikes during a preprocessing phase. Various methods of neural coding exist (see [108] for a review).

Among these, rate coding [41], latency coding [109], and direct coding [41] have emerged as the most popular models to convert real-valued data into spikes with a limited loss of information. Rate coding [41] is the most used and widely studied neural coding mechanism. In this scheme, data is expressed as the frequency of the spike train over a specific time window. In practice, the intensity of an input value is represented by the number of spikes transmitted: a higher value leads to a higher firing rate. Rate coding is valued for its robustness and simplicity [41], making it a highly used encoding scheme in SNNs. However, this encoding scheme is quite inefficient for rapid information transmission, as it requires several spikes to encode a single value, and it ignores the precise timing of individual spikes. A common way to implement rate coding is to generate spikes according to a Poisson process [109] (i.e., a stochastic process in which spikes occur independently with a constant average rate over time). In practice, the input value controls the rate of the Poisson process, so that the expected number of spikes in a given time span is proportional to the encoded value. In discrete time, rate coding can be implemented by sampling a spike independently at each time step with probability proportional to the input value. This probability is defined in Equation 4.3:

$$s(t) \sim \text{Bernoulli}\big(p(x)\big), \qquad p(x) = r_{\max} x \tag{4.3}$$

where x is the input value (e.g., a pixel value), $s(t) \in \{0, 1\}$ is the spike at time t, $r_{\max}$ is a maximum firing rate, and $p(x)$ is the resulting spike probability per time step. Over a discrete time window of T time steps, the firing rate has expectation $\mathbb{E}[\bar{s}] \approx r_{\max} x$, so the average number of spikes is proportional to the encoded input value. $\bar{s}$ is defined in Equation 4.4:

$$\bar{s} = \frac{1}{T} \sum_{t=1}^{T} s(t) \tag{4.4}$$

Latency coding [110], a type of temporal coding, encodes data as the precise

timing of a spike. In this model, the latency of a spike represents the strength of the input: a lower latency indicates a larger value. This approach allows for rapid and energy-efficient information processing (i.e., only a single spike is needed to convey meaningful information) [109]. In discrete time, it is defined in Equation 4.6:

$$t_{\text{spike}} = \left\lfloor (1-\epsilon)\, T \right\rfloor + 1, \tag{4.5}$$

and

$$S(t) = \begin{cases} 1, & t = t_{\text{spike}} \\ 0, & \text{otherwise.} \end{cases}, t = 1, \ldots, T \tag{4.6}$$

where $\epsilon \in [0, 1]$ is the input value, t_{spike} is the latency, $t_{\min}$ is the first time step and $t_{\max}$ is the last time step, T the number of time steps, and $S(t)$ indicates the presence of a spike at time t.

Direct coding [41] consists of learning the conversion from continuous value inputs to spikes directly within the first spiking layer, instead of using a hand-crafted encoding such as rate or latency coding. In practice, the input (e.g., image pixels) is fed to a learnable spiking layer trained with the rest of the network. In this encoding, the spike trains produced by this layer are adapted to the targeted task. This makes direct coding a strong candidate for building complex spiking representations of data. It has shown competitive performance against rate coding on large-scale image classification benchmarks [41].

4.2 Training Spiking Neural Networks

SNNs are challenging to train. On the one hand, directly training deep SNNs is difficult due to the non-differentiable thresholding function and the temporal nature of spiking neurons. These problems limit the applicability of standard back-propagation [111] and make local rules such as Spike Timing-Dependent Plasticity (STDP) [112] hard to use in deep models. On the other hand, indirect approaches,

which transfer pretrained ANN parameters to SNNs, often sacrifice the potential energy savings of the training phase. In this section, we review the main methods for training SNNs. First, we present ANN-to-SNN conversion methods [113], then Hebbian learning methods [114], an alternative to gradient propagation: feedback alignment and finally, surrogate gradient.

4.2.1 ANN to SNN Conversion

The standard ANN-to-SNN conversion process involves training an ANN using established methods (such as backpropagation) and then mapping its parameters (weights and biases) onto an equivalent SNN architecture. In this process, artificial neurons are replaced by spiking neurons, often of the IF type. The average firing rate of each SNN neuron is designed to approximate the activation of its analog counterpart over time (similar to rate coding), enabling the SNN to mimic the behaviour of the trained ANN for tasks such as classification or regression.

A key challenge in conversion is the activation mismatch: the information capacity of spike trains in SNNs is inherently lower than the continuous activations in ANNs, leading to potential accuracy loss and requiring longer simulation times for the SNN to reach ANN-level performance [113]. To address this, researchers have introduced techniques such as threshold balancing [115], weight normalization [115], soft-reset mechanisms [113], and layer-wise parameter calibration [113, 115] to minimize conversion errors and preserve information flow through the network.

Recent frameworks, such as SpikeConverter [116] and other optimal conversion methods [117], have demonstrated nearly lossless conversion for deep architectures like Visual Geometry Group (VGG) [118], Residual Neural Network (ResNet) [119], and MobileNet [120] on large-scale datasets, while drastically reducing the number of time steps required for inference. These advances rely on improved theoretical understanding of conversion errors, novel neuron models, and temporal separation schemes that better align spike generation with ANN activations. Moreover,

hybrid coding strategies (e.g., temporal and phase coding [29]) and burst-spiking models [29] have been explored to further compress information into fewer spikes, reducing inference latency and energy consumption.

ANN-to-SNN conversion has enabled SNNs to match or closely approach the accuracy of their ANN counterparts on challenging tasks, while offering substantial gains in energy efficiency at inference, when deployed on neuromorphic hardware. For example, Lemaire et al. [10] report SNN models that are approximately 6 to 8 times more energy efficient than their ANN counterparts, provided that the spike rate is sufficiently low (for further details see Section 4.2.5, Page 83). At the same time, converted models often suffer from performance loss and increased inference latency due to activation mismatch [117, 121, 115], and the need for a large amount of time steps to accurately approximate ANN activations, which recent ANN-to-SNN methods [122, 123] try to mitigate.

4.2.2 Hebbian Learning

Hebbian learning [114] is based on the idea that the strength of the synaptic connection between two neurons should increase when they are activated together—often summarized as "cells that fire together, wire together." In the context of spiking neural networks, Hebbian learning provides a biologically inspired and local mechanism for synaptic plasticity, enabling networks to adapt and learn from temporal patterns of activity without the need for global error signals or backpropagation.

In SNNs, Hebbian learning can be implemented at both the rate and spike levels. Traditional rate-based Hebbian rules [124] adjust synaptic weights based on the correlation between the average firing rates of the pre- and postsynaptic neurons. However, more recent research emphasises spike-based Hebbian learning [125] (e.g., STDP), where the precise timing of individual spikes plays a critical role. For example, synaptic weights are increased if a presynaptic spike precedes a postsynaptic spike within a narrow time window, and decreased if the order is reversed,

Modern SNN models mostly use Hebbian learning for unsupervised representation learning [126, 127, 128] or continual learning [129]. For instance, recent work [126] has demonstrated that Hebbian learning can be used to develop SNNs capable of learning internal representations from data in an online, unsupervised manner, closely mimicking the sparse and irregular firing patterns of cortical neurons. Furthermore, Hebbian learning principles have been extended to address challenges such as catastrophic forgetting in continual learning scenarios [129].

While STDP and Hebbian plasticity are powerful and biologically plausible rules, they are mostly applied in unsupervised learning settings. These methods cannot handle deep supervised tasks. Furthermore, direct training of deep SNNs often suffers from the dead neuron problem [130, 131], where many neurons stop spiking and thus receive no effective learning signal.

4.2.3 Feedback Alignment

Feedback alignment [132] was originally introduced as a biologically motivated alternative to standard backpropagation, in which error signals are propagated through fixed, random feedback weights instead of the exact transpose of the forward weights, which is difficult to implement in neuromorphic hardware. This relaxation of weight symmetry from backpropagation makes it a good candidate for the hardware implementations of SNNs.

In the context of spiking neural networks, feedback alignment and its variants (i.e., direct feedback alignment [133] and broadcast feedback alignment [134]) are hardware-friendly solutions for supervised learning. These methods enable multilayer SNNs to learn by transmitting error signals through random projections or fixed feedback pathways, rather than relying on the exact gradients computed by backpropagation.

Recent research [135, 136] has demonstrated the feasibility and effectiveness of feedback alignment in SNNs. For example, spike-train level direct feedback align-

ment has been shown to support efficient on-chip learning, reducing hardware resource consumption and training latency while maintaining competitive accuracy on real-world tasks such as image and speech classification [136].

Broadcast feedback alignment [134] extends the concept by sending error signals directly from the output layer to all hidden layers via random feedback connections, further simplifying the training process of multiple-layer SNN architectures. This progress position feedback alignment as a promising direction for training deep SNNs in a way that is efficient for hardware deployment.

While feedback alignment is more biologically plausible and could be easier to implement on neuromorphic hardware than backpropagation, empirical studies [137, 138] show that feedback alignment and direct feedback alignment struggle to perform as effectively as backpropagation on deep convolutional architectures and complex datasets.

4.2.4 Backpropagation with Surrogate Gradient

Backpropagation is the standard learning algorithm for deep ANN models. In these models, gradients of a global loss function are propagated backwards through all layers to update the weights of the neural network. The application of backpropagation to SNNs is challenging because spikes are generated by a non-differentiable Heaviside step function [40] and spiking neurons are inherently temporal. The temporal nature of spiking neurons requires the use of Backpropagation Through Time (BPTT) [139]. BPTT is an extension of backpropagation to RNNs, in which the network is unrolled over discrete time steps so that gradients can be propagated through the temporal dimension and used to update synaptic weights. Despite these constraints, gradient-based training through backpropagation remains attractive for supervised deep SNNs. In this section, we introduce the two major methods used for training SNNs with backpropagation, feedback alignment [132] and surrogate gradient [40].

Surrogate gradient methods [40] have become the dominant method for training spiking neural networks using gradient-based optimization, effectively overcoming the challenge posed by the non-differentiable nature of spike generation. In SNNs, a spike is emitted when membrane potentials reach a threshold, which prevents the direct application of backpropagation, as its derivative is zero almost everywhere and a Dirac at the threshold. Surrogate gradient learning addresses this by replacing the non-differentiable spike function with a smooth, differentiable surrogate activation function during the backward pass, while retaining the original spiking dynamics in the forward pass.

A wide variety of surrogate functions have been proposed, including the sigmoid [39], Rectified Linear Unit (ReLU) [39], exponential [39], Gaussian [39], and piecewise linear functions [39]. These differentiable surrogate functions approximate the gradient of the Heaviside step function, and as previously mentioned, enabling the use of BPTT. The choice of a surrogate function and its hyperparameters (e.g, the steepness of the sigmoid), can significantly influence learning performance.

Surrogate gradient methods have demonstrated strong robustness, allowing SNNs to get closer to state-of-the-art results on complex fields such as computer vision, and bridge the performance gap with ANNs. These methods are not limited to specific neuron models and can be applied across a range of spiking architectures, supporting the direct training of deep SNNs for real-world applications. As a result, surrogate gradient learning is widely regarded as a practical and effective solution for functional SNN training, and ongoing research continues to refine these techniques for even greater efficiency.

4.2.5 Energy Consumption of SNNs

Several studies [140, 10] report significant energy savings when SNNs are on neuromorphic hardware with sparse spiking activity. For instance, Chandarana et al. [140] show that SNNs can be between 5 and 27 times more energy-efficient than equiv-

alent feedforward ANNs on tasks like image classification. Conversion-based deep SNNs have also demonstrated substantial reductions in energy consumption [141], even though energy savings are often lower than those of directly trained SNN models because of a higher spike rate. Furthermore, optimisation strategies based on activity regularisation [142] were shown to further reduce SNN energy consumption while maintaining comparable accuracy by controlling spike rates during training.

In recent years, the reduction of the energy consumption of SNNs compared to ANNs has been debated. Research by Davidson and Furber [143] and Lemaire et al. [10] shows that the energy benefit of SNNs is conditioned by the sparsity of the spikes of the model. Thus, the energy efficiency of SNNs depends on the choice of the encoding scheme (i.e., rate coding versus latency coding). Lemaire et al. [10] proposed an analytical comparison of the computational cost between SNN models and their ANN counterparts. To this end, they evaluate the energy consumption of memory accesses and storage (i.e., input and output buffers) for the membrane potentials, weights, and biases on CIFAR-10 [144], Google Speech Commands V2 (GSC) [145], and the Prophesee Neuromorphic CARS (N-CARS) dataset [146]. In addition, they also count the number of synaptic operations. The results from Lemaire et al. are shown in Table 4.1. These results show that their SNN models consume approximately 8.19, 6.22, and 8.17 times less energy than their ANN counterparts on CIFAR-10, GSC, and N-CARS, respectively. These results were obtained with a spike rate of approximately 10%, highlighting the importance of SNN sparsity for energy savings. Nevertheless, Lemaire et al. [10] show that SNNs can still save a significant amount of energy even when using a rate-based encoding scheme.

Table 4.1: Energy consumption estimations of SNNs and ANNs counterparts on CIFAR10, GSC, and N-CARS from Lemaire et al. [10].

	CIFAR10		GSC		NCARS	
	SNN	ANN	SNN	ANN	SNN	ANN
Memory Accesses (nJ)						
Potentials	1.12E+6	–	3.50E+3	–	2.69E+5	–
Weights	4.60E+5	5.59E+6	1.73E+3	1.46E+4	8.10E+4	1.11E+6
Bias	2.79E+2	6.97E+1	6.00E+0	3.00E+0	3.71E+1	7.43E+0
Buffers	7.65E+2	6.09E+6	3.90E+1	1.50E+4	2.88E+2	1.54E+6
Total	1.58E+6	1.17E+7	5.27E+3	2.96E+4	3.50E+5	2.64E+6
Synaptic Op. (nJ)	2.41E+3	1.29E+6	4.08E+1	3.53E+3	6.15E+3	2.67E+5
Addressing (nJ)	1.97E+2	4.94E+1	9.28E+0	1.93E+0	5.05E+1	7.65E+1
Total (nJ)	1.58E+6	1.24E+7	5.32E+3	3.32E+4	3.57E+5	2.86E+6
$E^{\text{ANN}}/E^{\text{SNN}}$	8.19		6.22		8.17	

4.3 SNNs for VOT

SNNs have only recently started to be explored for VOT, and the existing literature remains very sparse compared to ANN-based trackers. Most existing work either focuses on inputs provided by event-based devices or employ SNNs only for specific modules, like object detection, instead of building end-to-end spiking trackers. In the following, we first review previous non-Siamese SNN trackers for SOT, and then focus on spiking Siamese networks, which are the closest to the spiking SiamFC models developed in this thesis.

4.3.1 Previous SNN Trackers for SOT

SNN-based SOT [147] has recently emerged as an alternative to conventional ANN trackers, mainly in event-based vision, where neuromorphic sensors naturally produce sparse spike streams. In this context, the Spiking Convolutional Tracking Network (SCTN) [147] has been proposed for single object tracking on event sequences acquired with Dynamic Vision Sensors (DVS) [148], which convert light directly into spikes by difference of intensity. SCTN takes short segments of events as input and processes them with a deep spiking convolutional architecture trained

directly as an SNN, instead of using ANN-to-SNN conversion. This model uses a temporal integration of events in successive segments to build feature maps, and predicts the target bounding box at each timestep using a tracking loss that combines localisation and confidence terms, achieving competitive accuracy compare to ANN tracker such as SiamRPN [147] with a substantially reduced computational cost and energy consumption on its dedicated benchmark as compared to ANN models.

In parallel, several authors have focused on spiking architectures for object detection [149, 150, 151], which is a core component of many tracking-by-detection pipelines. Among them, the Efficient Multiscale Sampling for Real-Time Object Detection (EMS-YOLO) [151] proposes a fully deep, directly trained SNN detector that uses surrogate gradients learning and a spiking residual backbone to perform object localization and classification, avoiding the usual rate-based ANN-to-SNN conversion and reducing the number of timesteps at inference time. EMS-YOLO demonstrates that directly trained SNNs can reach performance similar to ANN detectors on standard datasets while significantly decreasing the estimated energy usage, which makes these models promising building blocks for low-power tracking systems.

However, SNNs for single object tracking on conventional RGB video remain underexplored, and most existing approaches are limited to event-based data [147] or to using SNNs as isolated components rather than fully integrated trackers [147]. The combination of SNN backbones with ANN-based motion model [152] or association modules [152] raises additional questions regarding gradient backpropagation, training stability, and the design of loss functions that can exploit both spike timing and appearance information. As a result, there is currently no widely adopted spike-based SOT framework evaluated on benchmarks such as OTB, LaSOT, or GOT-10k, and hybrid ANN–SNN designs remain an open research direction to

leverage both the robustness of conventional visual features and the energy efficiency of neuromorphic computation.

4.3.2 Spiking Siamese Networks for Tracking

Spiking Siamese networks extend the SiamFC family to the neuromorphic domain by replacing standard artificial neurons with spiking units while preserving the Siamese correlation paradigm for similarity matching. In this context, only a few papers [29, 30] have explicitly explored the conversion or training of Siamese-based trackers as SNNs, and most of them target energy efficiency rather than pushing absolute tracking accuracy to the current state of the art. These models aim to maintain the fully convolutional cross-correlation between an exemplar and a search region while exploiting sparse spike-based computation to reduce the number of operations during inference.

The first work in this direction is SiamSNN, introduced by Luo et al. [29], which performs ANN-to-SNN conversion of a pretrained SiamFC model. In SiamSNN, the convolutional backbone and correlation head are kept structurally identical to the original SiamFC, but activations are converted into spike trains using rate coding over multiple timesteps, allowing the authors to estimate energy savings on neuromorphic hardware without retraining from scratch. This study shows that a converted spiking Siamese tracker can reach performance close to its ANN counterpart while significantly reducing the number of multiply–accumulate operations, but the method remains constrained by the original ANN design and by the conversion procedure.

More recently, Xiang et al. [30] proposed a spiking version of SiamFC++ [87], which directly integrates spiking neurons into a more advanced Siamese tracking framework. Their architecture replaces ReLU units with LIF neurons and the network is trained using surrogate gradient, enabling end-to-end optimization in the

spiking domain instead of relying on post-hoc conversion. This work demonstrates that training a deeper spiking Siamese tracker is feasible and that such a model can achieve competitive accuracy on OTB-2015 with its ANN counterpart.

This paper on SiamFC++ was published after we started our research on a spiking version of SiamFC. Although this work constitutes an important first step toward spiking versions of modern Siamese trackers, its experimental validation remains limited. The evaluation is restricted to OTB-2015, without results on larger and more diverse benchmarks such as GOT-10k, which makes it difficult to fully assess the generalization capabilities of the proposed spiking SiamFC++ model beyond relatively short and constrained sequences. In addition, the paper provides few ablation studies and limited implementation details, in particular regarding the training schedule (e.g., number of epochs and convergence behaviour), which hinders the reproducibility of the reported results and makes it hard to disentangle the contributions of the spiking formulation from other design choices.

In the context of this thesis, a spiking Siamese tracker is evaluated on both OTB-2015 and GOT-10k using a clearly documented training protocol, with an emphasis on reproducibility and on analysing the impact of architectural and optimization choices. This contemporaneous work complements the existing literature by providing a more systematic study of spiking Siamese networks for single object tracking and by positioning their performance with respect to both classical SiamFC models and more recent deep trackers on standard benchmarks.

4.4 Conclusion

Unlike ANNs, SNNs represent and process data through discrete, sparse spike trains, rather than continuous activations. This property makes them implementable on dedicated, low-energy neuromorphic hardware. However, SNNs also introduce specific modelling and training challenges that differ from those of ANNs.

In this chapter, we introduced the core concepts of spiking neurons, from simple IF/LIF models to more biologically relevant HH and SRM models. We also presented the most popular neural coding schemes, rate, latency, and direct coding, which define how continuous inputs can be transformed into spike trains. We then reviewed the main training strategies for SNNs, comparing the advantages and drawbacks of ANN-to-SNN conversion, Hebbian learning, and backpropagation-based methods, including feedback alignment and surrogate gradients. Finally, we examined existing SNN approaches for VOT, focusing on spiking Siamese architectures. While models such as SiamSNN and spiking SiamFC++ clearly demonstrate the feasibility and potential energy benefits of spike-based Siamese trackers, they provide a limited analysis of the specific challenges that arise when implementing Siamese trackers with SNNs.

Based on this review, we determined that IF/LIF neurons combined with direct input coding and surrogate gradient backpropagation would be highly adapted to the problems developed in this thesis. Direct coding enables the first spiking layer to learn specialised, rich spike representations useful for VOT, and has been shown to achieve higher accuracy than rate coding. Surrogate-gradient learning provides a flexible and robust framework for training deep SNNs end-to-end despite the non-differentiability of spikes and has repeatedly demonstrated competitive performance on challenging vision tasks.

Overall, this analysis provides the foundation for designing a spiking tracker. It gathers the key notions and models in terms of neuron model, neural coding scheme and training strategy. In the next chapters, we will exploit this groundwork to construct and analyse the spiking SiamFC tracker proposed in this thesis.

Part II

Contributions

Chapter 5

Dataset Annotation for Atmospheric Object Tracking Using CAMS Reanalysis

VOT of aerosol plumes could provide re-identification and dynamic analysis of aerosols by enabling researchers in atmospheric sciences to study small events automatically. VOT could also allow an automation of the orientation of instruments of satellites towards an event of interest, if embedded onboard (see Section 1.2, Page 26). Aerosol datasets, and in particular AOD [153] datasets, are well-developed in atmospheric sciences. We choose AOD at wavelength 550nm because it is the most representative of aerosol impact for the general use case [154], and is largely used in the atmospheric science literature [7]. For the rest of this thesis, we will write AOD for AOD at wavelength 550nm. AOD data, such as those provided by CAMS [38], come from satellite observations and atmospheric reanalysis products (e.g, dust, sea salt or total AOD at 550nm). These products provide full spatial and high temporal coverage (i.e., every 3 hours) of Earth. However, they are not annotated with the location of atmospheric plumes at each acquisition (i.e., frame), which is needed for training and evaluating automatic trackers, as discussed in Chapter 2

(Page 35). Consequently, while CAMS provides valuable estimates of aerosol quantities, it lacks some information, such as the centres and sizes of the bounding boxes around the empirical limits of the aerosol plumes. Despite the importance of plume tracking, there is not any publicly available annotated VOT dataset of aerosol data in the literature, to the best of our knowledge.

Motivated by this gap, we present in this chapter the first manually annotated VOT dataset of aerosol plumes, derived from CAMS reanalysis data and containing AOD sequences of dust and sea salt plumes. This dataset aims to provide plume annotations to train SOT trackers on total and by-species AOD provided by CAMS Atmospheric Composition Reanalysis 4 (EAC4), where plumes are not merging nor splitting.

We start this chapter by explaining aerosol data acquisition. In Section 5.1, we describe the CAMS ECMWF EAC4 products and our species of interest. We also detail the acquisition period, spatio-temporal resolution, and preprocessing steps used to facilitate the manual identification and re-identification of aerosol plumes. In Section 5.2, we explain our annotation process and the criteria used to determine aerosol plume boundaries. In Section 5.3, we analyse the quality of the results by comparing the size and intensity distributions of the annotated plumes with atmospheric statistics on CAMS EAC4 from the literature [5, 6]. Finally, we conclude and discuss challenges and opportunities for the improvement of this dataset. In Appendix A.1, we detail the dataset metadata, its format and its structure.

5.1 Aerosol Data Acquisition

CAMS provides atmospheric composition forecasting on a global scale. Among the multiple datasets produced by CAMS from ECMWF, the EAC4 dataset provides a map of atmospheric components, such as aerosols or gases, over multiple years.

CAMS EAC4 is built on an assimilation framework (as discussed in Section 2.2.3, Page 39) that integrates satellite and ground-based observations with prediction models. CAMS EAC4 delivers a temporal and spatial characterisation of atmospheric conditions from 2003 to the present.

The data assimilation process of EAC4 takes as input observational data (such as satellite radiances, ground-based sun photometers and in-situ measurements), model data from previous predictions, and meteorological parameters (e.g., temperature, wind, humidity), as well as previously observed AOD data from sensors, in order to provide a robust representation of future AOD. The assimilation process uses the 4D-Var methodology [155]. It provides hourly forecasts of the atmospheric composition at a global spatial resolution (0.75° by 0.75° of longitude and latitude).

For aerosols, CAMS EAC4 provides both total AOD and by-species AOD (such as dust, sea salt, black carbon, sulfate, and organic matter). This granularity allows us to study aerosol sources and trajectories, which is why we chose this dataset over the others provided by CAMS. EAC4 has been created to be a stable, homogeneous, and long-term reanalysis dataset, compared to near-real-time or intermediate reanalysis datasets, which contain many data gaps. EAC4 has been extensively validated against AERONET sun-photometer observations in the literature [45]. These studies generally report reduced AOD biases and improved temporal consistency compared to earlier atmospheric composition reanalyses produced with the same modelling and assimilation system, such as the Monitoring Atmospheric Composition and Climate (MACC) [156].

Thanks to their reliability and continuity, the CAMS EAC4 reanalysis data have been highly adopted in the atmospheric research community [45] and are used for numerous applications, including climate trend analyses [7], air quality forecasting [45], health impact studies [5], and satellite validation [45]. The large period covered, the multi-component dimension of aerosol characterisation, and assimilation of remote observations make EAC4 a fundamental dataset for environmental

monitoring and for targeted analyses, such as the VOT of aerosol plumes presented in this thesis.

5.1.1 ECMWF CAMS EAC4 Data

The ECMWF data center provides AOD data through the CAMS system and the EAC4 dataset. AOD is a value without dimension that describes the extinction (i.e, the absorption) of solar beams by aerosol particles in a vertical column of the atmosphere. Physically, it measures the integrated attenuation of light due to aerosol scattering and absorption in the air column (i.e., between the surface and the top of the atmosphere). Thus, AOD represents the behaviour of aerosols in the atmosphere. It is essential to understand aerosols and monitor air quality and climate in general [153]. Based on Guilbert et al.'s work [157], we assume that these variations in AOD create spatial patterns that are unique for each aerosol plume, which makes AOD a good candidate to track plumes.

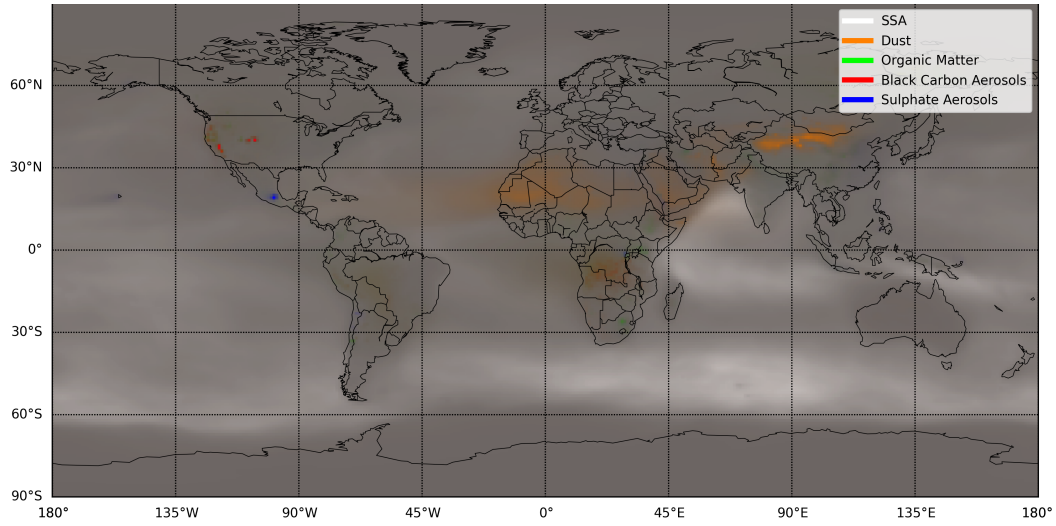


Figure 5.1: Illustration of the AOD for each species of CAMS EAC4. CAMS provides five categories of aerosols: dust, sea salt, organic matter, black carbon, and sulphate.

AOD is illustrated in Figure 5.1. This figure displays the AOD for each species provided by CAMS EAC4. We observe that dust is present for the most part

in the Saharian and Gobi deserts (acting as the main sources of dust on Earth), depending on meteorological events. Dust plumes are well-defined and localised. On the contrary, sea salt AOD is diffused all around Earth, making it more challenging to isolate and define plumes.

Mathematically, the total AOD $\tau(\lambda)$ at wavelength λ is formulated as the integral over the air column of the aerosol extinction coefficient $\sigma_e(z, \lambda)$. AOD has technically no numerical limits; however, in general, it takes values from almost 0 in clean atmospheres to 1 in highly polluted events. It is defined in Equation 5.1:

$$\tau(\lambda) = \int_0^\infty \sigma_e(z, \lambda)\, dz \tag{5.1}$$

where z is the altitude. This integral takes into account both scattering and absorption. The extinction coefficient $\sigma_e(z, \lambda)$ at each altitude is defined as a sum over all aerosol types and their respective size distributions, compositions, and concentrations inside the air column.

In practice, AOD can be obtained either from direct measurements, such as AERONET photometers, or from atmospheric models (e.g., CAMS EAC4). From measurements, we first obtain the attenuation of direct solar radiation with sun photometers at several wavelengths, and derive total AOD from Beer-Lambert's law [153]. In contrast, in the CAMS EAC4 reanalysis dataset, total AOD is computed with IFS models (see Section 2.2.3, Page 39).

From photometers, by-species AOD (e.g. for dust or sea salt) can only be obtained indirectly. It requires the combination of AOD measures at multiple wavelengths and measurements of the radiance of the sky, and the use of inversion algorithms [158]. It also requires assumptions on aerosol size distributions, shapes and refractive indices to retrieve an aerosol size distribution. This distribution gives the contributions of an aerosol species to the total AOD. In CAMS EAC4, by-species AOD are obtained directly from the IFS model.

5.1.2 Preprocessing Steps

The manual annotation of aerosol plumes in CAMS EAC4 is challenging because of the variance of raw AOD values across time steps. This variance is due to background AOD variability, and variations from seasonal cycles or extreme atmospheric phenomena such as volcanic eruptions or forest fires. Therefore, plume patterns often have low contrast, making it challenging to distinguish their visual delimitations, which results in potential inconsistencies across annotations. Moreover, the diffuse nature of plume boundaries further increases this difficulty, as annotators must have the ability to see small changes in pixel intensities and must not include plumes of lower intensity detached from the original one, to avoid scenarios of plume splitting in the training data for the SOT tracker.

To address these problems, we perform two preprocessing steps before manual annotation. First, for each pixel, we compute the AOD fraction of the specific species to annotate. The AOD fraction is the ratio between the AOD of a species and the total AOD. The AOD fraction is defined in Equation 5.2:

$$\mathrm{AOD}_{\mathrm{f,s}}(x, y) = \frac{\mathrm{AOD}_{\mathrm{s}}(x, y)}{\mathrm{AOD}_{\mathrm{t}}(x, y)} \tag{5.2}$$

where $\mathrm{AOD}_{\mathrm{f,s}}(x, y)$ is the fraction of AOD for a specific species with respect to total AOD, $\mathrm{AOD}_{\mathrm{s}}(x, y)$ is the by-species AOD, and $\mathrm{AOD}_{\mathrm{t}}(x, y)$ is the total AOD.

To reduce the visual importance of diluted or old plume remnants mixed with the background around the main event (see Section 1.3, Page 28), we use the AOD fraction map. It highlights areas where a specific aerosol species contributes a large share of the total AOD, so that the most significant plumes of that species stand out more clearly than in the raw AOD maps (see Figure 5.2). Because the fraction normalises by-species AOD by the total AOD, it is less sensitive to broad background variations and produces sharper gradients at plume boundaries, without requiring arbitrary absolute thresholds. This, in the end, helps homogenise

annotation protocols, as annotators can rely on similar AOD fraction values for plume identification and re-identification across a sequence.

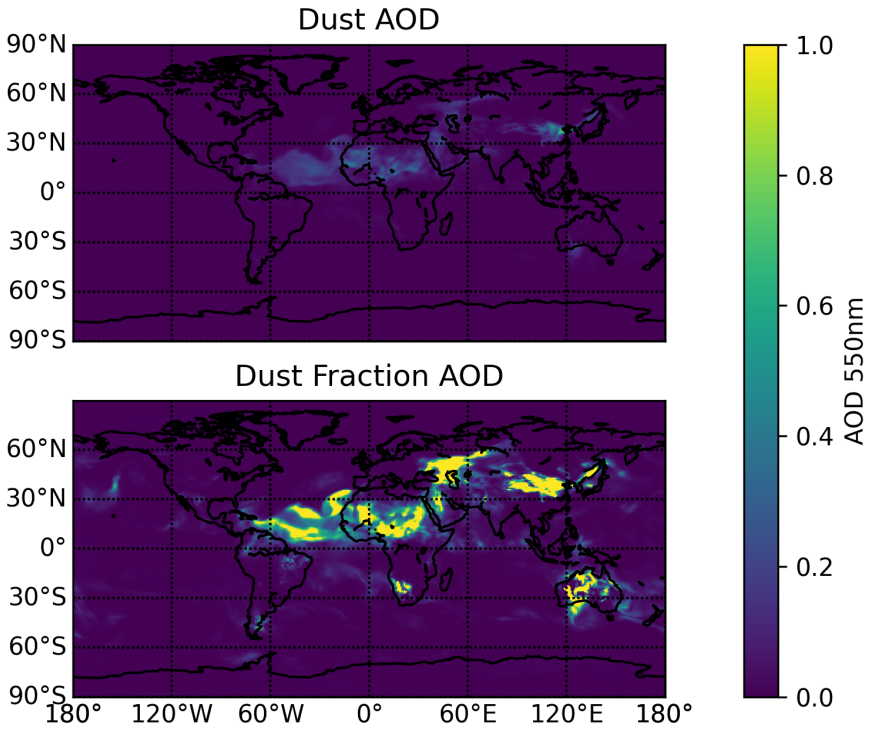


Figure 5.2: CAMS reanalysis maps for 1st September, 2020, showing the dust AOD at 550nm (top), and the corresponding dust aerosol fraction (bottom). The dust fraction map highlights the relative contribution of dust to the overall aerosol content.

However, for a fair annotation, we need a fixed intensity distribution from frame to frame. Despite the advantages of AOD fraction, it can appear saturated on maps when dust aerosols are the only species in the air column. Therefore, min-max normalisation is applied to the AOD fraction of each species at each frame in the sequence to normalize AOD intensities over the sequence. Figure 5.3 shows an example of an AOD frame before and after normalisation. At each pixel location (x, y) of the AOD map, the normalised value is computed as in Equation 5.3:

$$\mathrm{AOD}_{\mathrm{norm}}(x, y) = \frac{\mathrm{AOD}(x, y) - \min(\mathrm{AOD})}{\max(\mathrm{AOD}) - \min(\mathrm{AOD})} \tag{5.3}$$

where max(AOD) and min(AOD) denote, respectively, the minimum and maximum of AOD values within the sequence of AOD frames.

Through this normalisation process, we can annotate frames using the same referential value across the sequence, by taking only AOD values above 90%. The species fraction shows the more salient plumes, but normalising it makes the separation between plumes clearer (see Figure 5.3). Thus, the min-max operation also homogenises annotation protocols across the sequence by retaining only the purest aerosol plumes across the sequence.

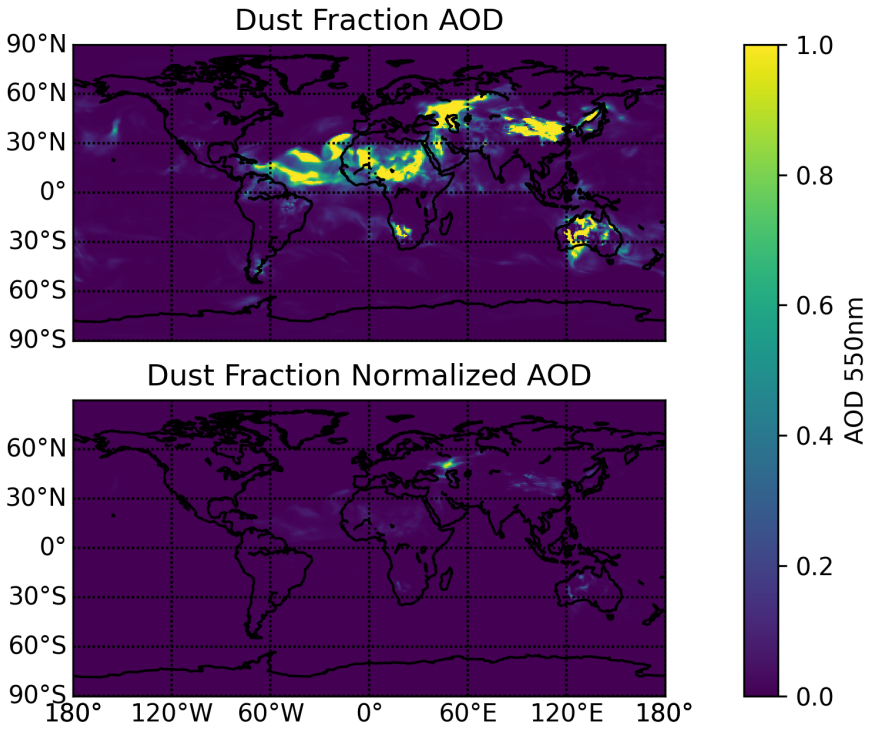


Figure 5.3: CAMS AOD maps of dust aerosols for 1st September, 2020, showing the dust AOD fraction map (top), and the same map after min-max normalisation (bottom). The normalised map enhances local contrasts and the sharpness of dust plume boundaries.

Following the conversion to AOD fractions and min-max normalisation, the resulting AOD map is rendered as a colour image using a colour map, in order to ease the manual annotation process. Indeed, even though the normalised grayscale

images already enhance local contrast, human visual perception is more sensitive to variations in colour than slight differences in brightness [159].

5.2 Annotation Process

The annotation protocol focuses on labelling successive aerosol plumes frame-by-frame, maintaining object identities until the object ceases to exist. The decision not to include the plume of the next frame in an annotated sequence of plumes is based on both physical and practical considerations. The sequence of an aerosol plume ends when the plume fades away, splits, or merges. A plume fades away when it becomes too diluted against the background. Splitting and merging represent fundamental changes in the re-identification and appearance of the observed objects. Moreover, for splitting, this event creates inconsistencies in the object definition, since we need to decide which sub-plumes to track. By ending plume trajectories when these events occur, the dataset is composed of more consistent trajectories, but at the cost of not representing these complex plumes in the final dataset. This results in sequences of variable length.

To summarise, we propose the following protocol to annotate AOD at 550nm of CAMS EAC4 for VOT.

1. AOD (at 550nm) maps for dust and sea salt are extracted from the EAC4 reanalysis dataset from the CAMS service from ECMWF.

2. Maps are prepared by computing the fraction of AOD by-species (dust or sea salt), normalising them to min-max, and colourizing them to highlight aerosol plumes for the annotator.

3. Maps are annotated. We define a bounding box around every visually consistent aerosol plume with AOD values above 0.9 using the previously prepared maps as the guide. Each plume receives a unique identification number that

is used to label it across frames. The sequence ends when the plume presents merging or splitting events, or when the intensity falls below 0.9. The choice of this threshold has been decided empirically to track the most concentrated plumes and to discard diffuse background artifacts and plume remnants.

4. A second annotator controls each annotation and corrects it if needed. And annotations are finally saved in a single file per map, containing the bounding boxes of each plume and its associated identifier.

5. For SOT trackers, we crop the plumes around the largest area which does not overlap with any other plume, to make produce sequences containing a single plume.

To ensure consistency and reproducibility in our annotation process, we use the LabelMe platform, a widely recognised open source tool [160] providing a graphical interface for diverse types of object segmentation and tracking (such as masks, bounding boxes or polygons). The user interface enables the precise spatial delimitation of aerosols and the temporal association of plumes across time. For reusability, we give our annotations the same format as ImageNet VID 2015 [64]. For further details on the format, see Appendix A.1 (Page 173). The final dataset provides bounding boxes for each aerosol species on each sequence. These bounding boxes can be used either to train a model to track over maps of total AOD or maps of by-species AOD, since all maps share the same referential.

5.3 Dataset Analysis

This section characterises our manually annotated AOD dataset of plumes coming from CAMS EAC4. We evaluate the physical consistency through quality control statistics. These analyses summarise the temporal coverage, class distribution, and AOD statistics within bounding boxes. In this section, we verify that annotations

align with the atmospheric science literature on dust and sea salt aerosols to ensure their suitability for future training and benchmarking of tracking models.

5.3.1 Dataset Properties

The raw data was acquired over a period of two years and four months, encompassing a total of 28 months of observations, from the 1st of September 2020 to 31st of December 2022. The dataset focuses on two primary aerosol types: dust and sea salt, with differing temporal coverage and annotation volume for each. For the dust aerosol, data were collected over 28 months, with an average acquisition rate of approximately 247 images per month (i.e, one image every 3 hours). This dataset is composed of 182 annotated SOT sequences, capturing the spatial-temporal dynamics of dust plumes. Within these sequences, we annotated 20,837 bounding boxes, providing detailed spatial coverage for aerosol plume tracking. For the sea salt aerosol, we cover a shorter period of 5 months, with the same average frame rate of 247 images per month. This results in 85 SOT sequences annotated for sea salt aerosol plumes, comprising a total of 7,924 bounding boxes.

Overall, the dataset consists of 267 sequences (182 dust AOD sequences and 85 sea salt AOD sequences) and 28,761 annotated bounding boxes.

5.3.2 Quality Control

In this section, we conduct a quality control of manually annotated bounding boxes, which is essential to ensure that the dataset can be reliable enough for training and evaluating object tracking models in realistic settings. We perform several quality control analyses to check that the annotations of the AOD maps (total or by-species AOD) from CAMS EAC4 are accurate for size, intensity and evolution. These controls aim at ensuring that the annotations have physical meaning.

Figure 5.4 shows the distribution of the sizes of bounding boxes for dust plumes. It shows a peak at 3,000km by 1,500km, with a non-negligible amount of outliers

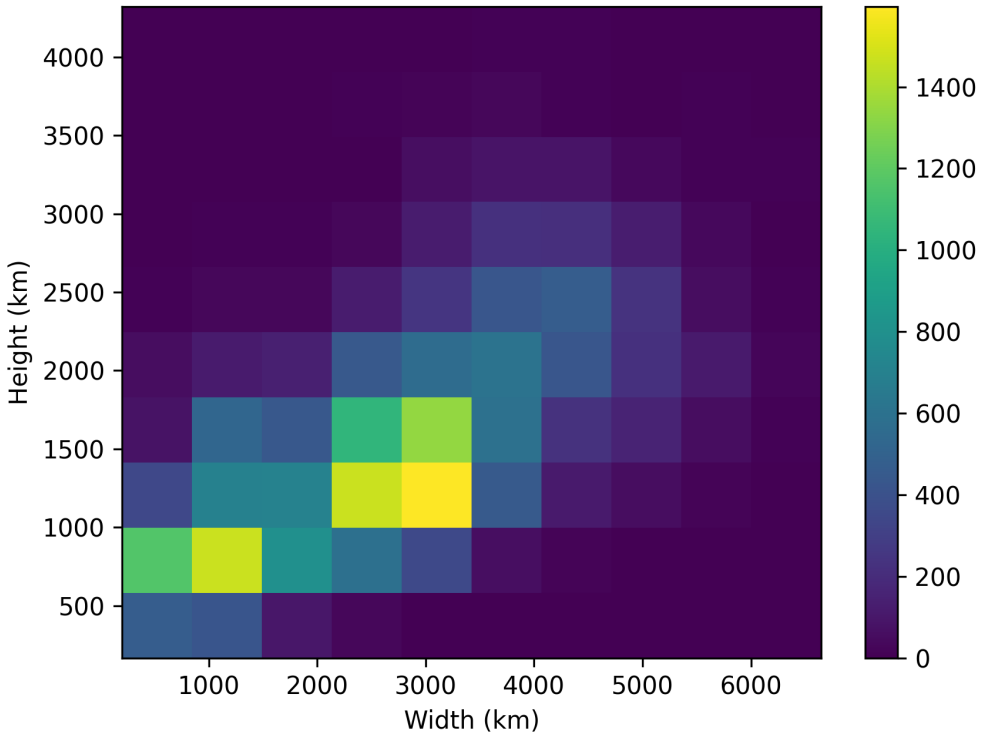


Figure 5.4: Heatmap of the distribution of the sizes of bounding boxes annotated on the dust AOD maps over the Gobi and Saharan desert regions. The distribution peaks correspond to the typical spatial scales of dust plumes observed in CAMS EAC4 dust aerosol data for these areas [5]. The annotations were realised from the 1st of September, 2020 to 31st of December, 2022, reflecting realistic plume size variability over multiple seasons for dust aerosols.

around it. This distribution of dust plume sizes is consistent with observations [5]. This shows that the annotation process matches the natural variability in plume extent through dust-rich locations such as the Sahara and Gobi deserts.

Similarly, Figure 5.5 shows a peak at $1,500$km by $1,000$km in height and width, respectively, for sea salt plumes. Compared to dust, the annotated sea salt plumes tend to have larger bounding boxes, reflecting their more diffuse origin, and shorter atmospheric lifetimes. While the literature does not provide explicit statistics for individual sea salt plume areas, our annotations are consistent with the behaviour of sea salt aerosols, being concentrated near oceanic source regions and fading quickly, as described by Bian et al [6].

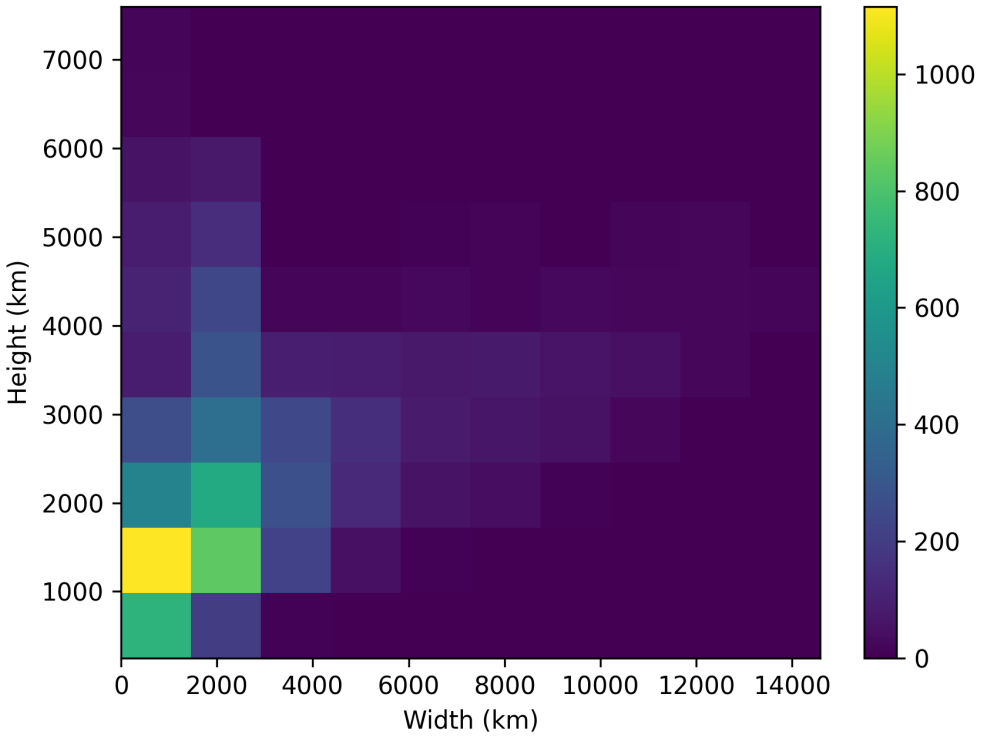


Figure 5.5: Heatmap of the size distribution of bounding boxes annotated on AOD maps of sea salt. Most annotated plumes have moderate horizontal extents, consistent with the short lifetime and marine origin of sea salt aerosol maps discussed by Bian et al. [6].

Figure 5.6 shows the statistics of the distribution of plume areas by species. It shows that half of the bounding boxes have areas below 3.5 million of km^2 for dust and sea salt. Dust and sea salt species have an area of 4 million km^2 and 7 million km^2 on average, respectively, which follows the statistics of Silva et al. [5] for dust. Furthermore, both species display many outliers coming from events such as storms, which correspond to larger plumes (i.e., larger bounding boxes). It is consistent with the previous observations from Figure 5.4 and Figure 5.5.

The class distribution of our dataset is shown in Figure 5.7. Dust represents the majority of annotated sequences and frames (i.e., 68.2% and 72.6%, respectively), while sea salt is less represented (i.e., 31.8% and 27.4%, respectively). This imbalance results from practical time constraints during manual annotation: since dust

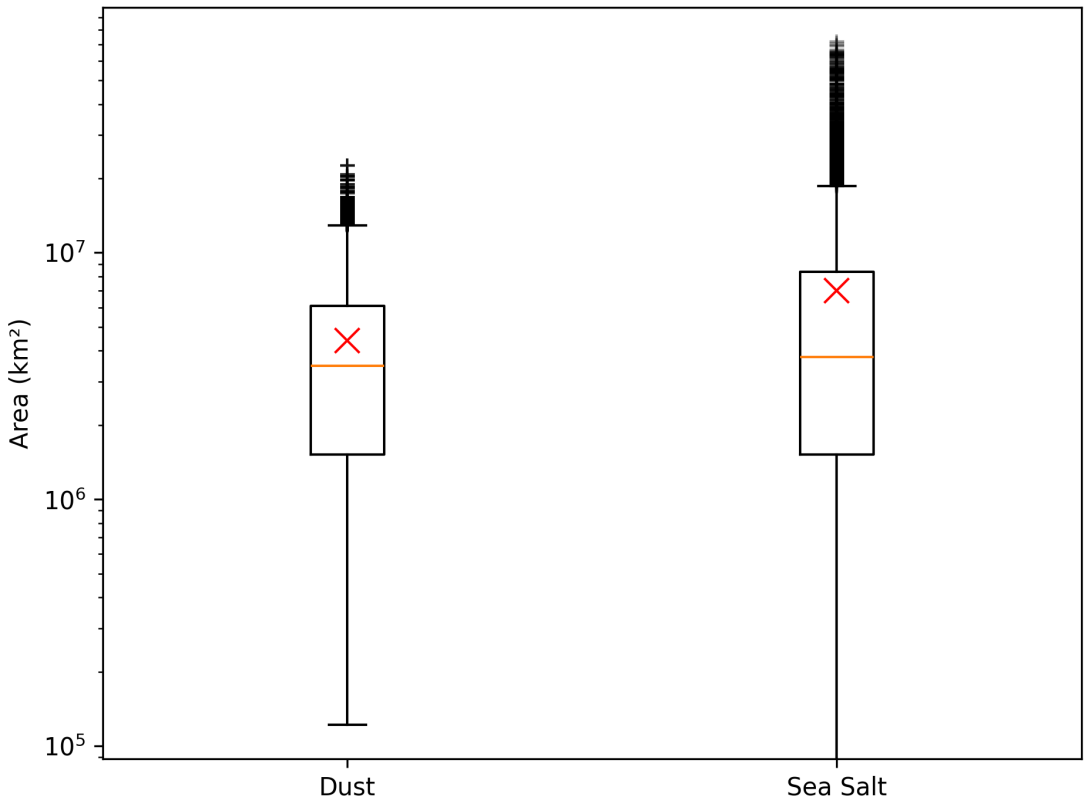


Figure 5.6: By-species statistics of the areas of bounding boxes of groundtruth plumes. Red crosses represent the mean area, and black crosses represent the outlier bounding boxes. The statistics for dust agree with the tables provided by Silva et al. [5].

.

aerosol plumes are more dense and well-defined [6, 5], we decided to prioritise dust annotations over a more extended time period (28 months, compared to 5 months for sea salt) to annotate more sequences in a shorter time. So, the dataset contains proportionally more dust sequences and frames. Future extensions of the dataset should include additional annotations of sea salt plumes to improve class balance.

In addition to size distribution, we analyse the global AOD intensities to support the physical consistency of the annotated plumes further. Figure 5.8 shows the intensity distribution of the by-species AOD. These statistics have been computed globally from the values within bounding boxes annotations for dust and sea salt aerosol plumes. In Figure 5.8, the probability density functions of AOD values have an inverse exponential shape. This reflects the skewness of atmospheric data toward

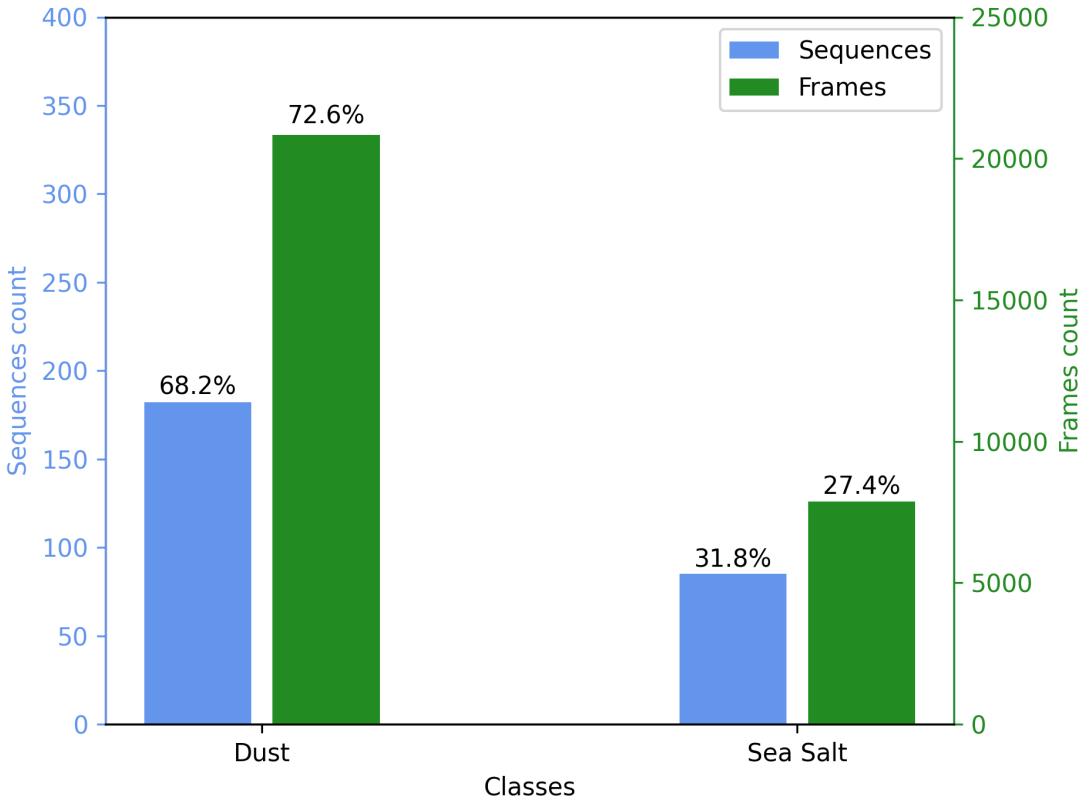


Figure 5.7: Class distribution of the annotated tracking aerosol plume dataset. The number of sequences (blue bars, left axis) and frames (green bars, right axis) are shown for dust plumes (28 months) and sea salt plumes (5 months). Percentages indicate the relative contribution of each class for frames and sequences.

low values. This figure shows that the distribution of aerosols is quite sparse, both for dust and sea salt. The absence of irregularities in the dust density curve reflects more episodic events, supported by higher but less frequent values, which agrees with the literature [5]. Figure 5.9 shows the box plot of by-species AOD intensities (mean, median, and quartiles) over the whole dataset. Figure 5.9 agrees with the previous observations: it shows that half of the AOD values are below 0.1 and 0.04 for dust and sea salt, respectively. Such distributions align with the expected distributions of these aerosol types [6, 5] and confirm that the dataset contains both average and extreme aerosol plume conditions.

Figure 5.10 and Figure 5.11 show the seasonal trends (mean and standard deviation) of AOD for dust and sea salt, averaged by month. Dust seasonal trends on

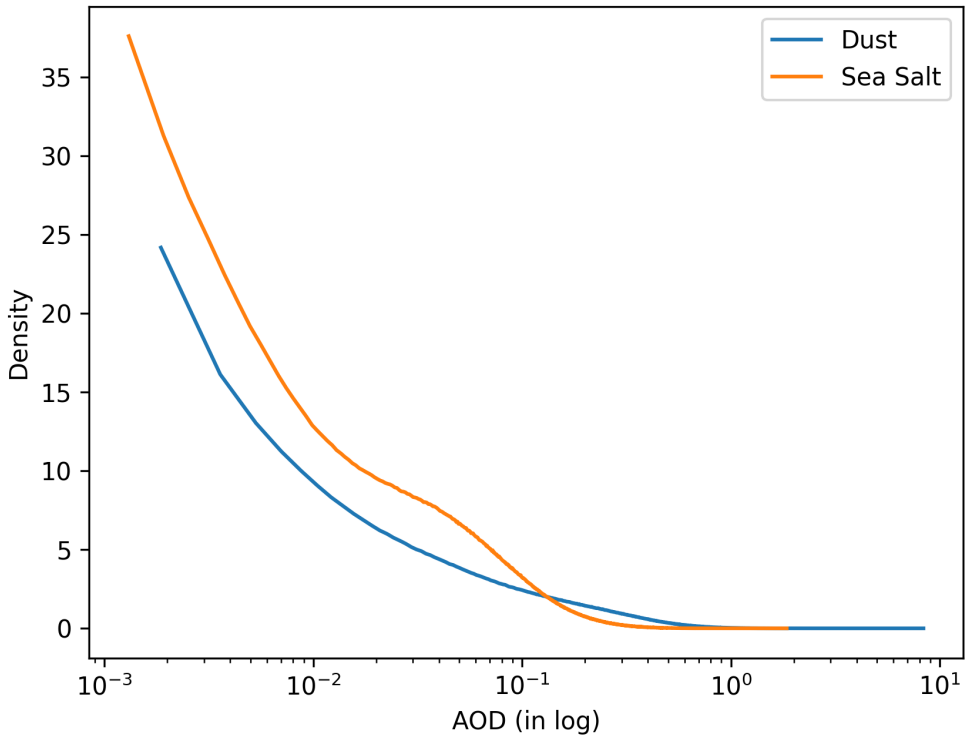


Figure 5.8: Intensity distribution of the by-species AOD within bounding box annotations for dust (blue) and sea salt (orange). The distribution follows an inverse exponential law, expected from aerosols behavior [6, 5]. Most of the values have low intensities and come from the background. Higher intensities represent events that are less frequent. The sea salt density curve shows a slight perturbation compared to dust and gets higher intensity values. This reflects more episodic events and fewer cyclic and continuous emissions of aerosols.

Figure 5.10 confirm the presence of episodic events inferred from Figures 5.8 and 5.9. Sea salt trends in Figure 5.11 show a lower inter-month variation. Both agree with the seasonal trends from the atmospheric science literature [5], supporting the physical relevance of this tracking dataset.

Overall, these quality control analyses confirm that the annotations are physically consistent with CAMS EAC4 data and ready to be used to be validated as a realistic training and evaluation dataset for deep learning–based object trackers.

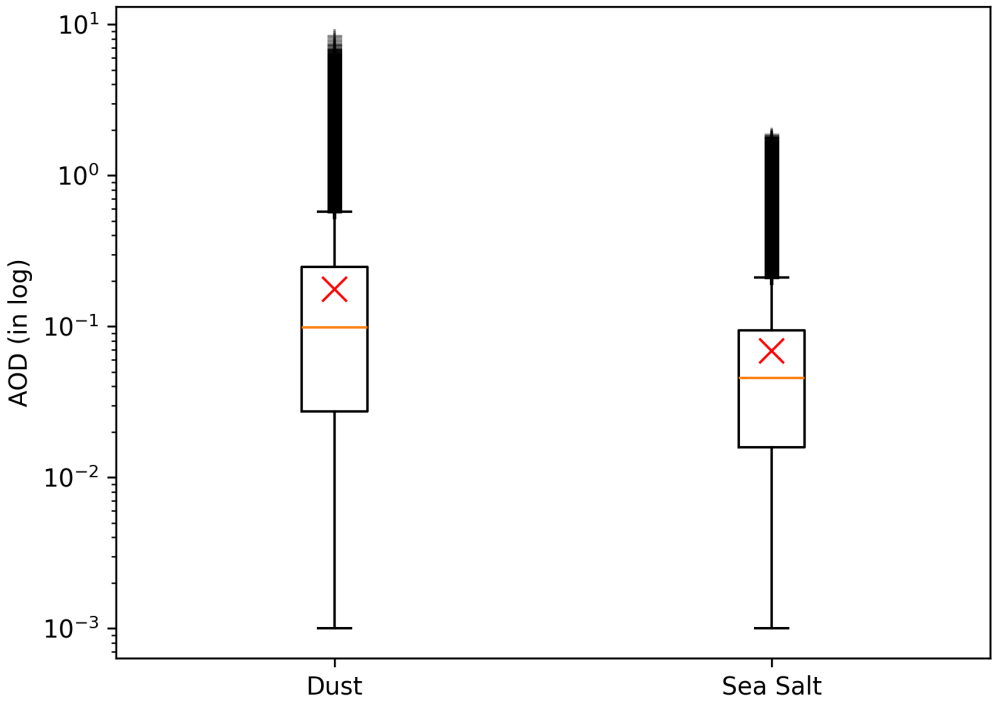


Figure 5.9: Box plot of the intensity distribution of dust and sea salt by-species AOD. This figure shows a median of 0.1 for dust AOD and 0.04 for sea salt AOD. Red crosses represent the mean AOD value, and black crosses represent outliers.

5.4 Limitations and Future Work

While providing a valuable resource for training trackers to track aerosol plumes of multiple species, this dataset shows some limitations.

- The diversity of the aerosol species in the dataset is limited (only dust and sea salt plumes have been annotated, out of the five species contained in CAMS EAC4). Furthermore, the annotated sequences capture plumes that are characteristic of specific regions, such as the Gobi or the Sahara Desert, for dust aerosols. This could limit the number of cases seen by tracking models, which could, in the end, impede the accuracy of the tracker in real use cases. In order to solve this limitation, additional annotation efforts are

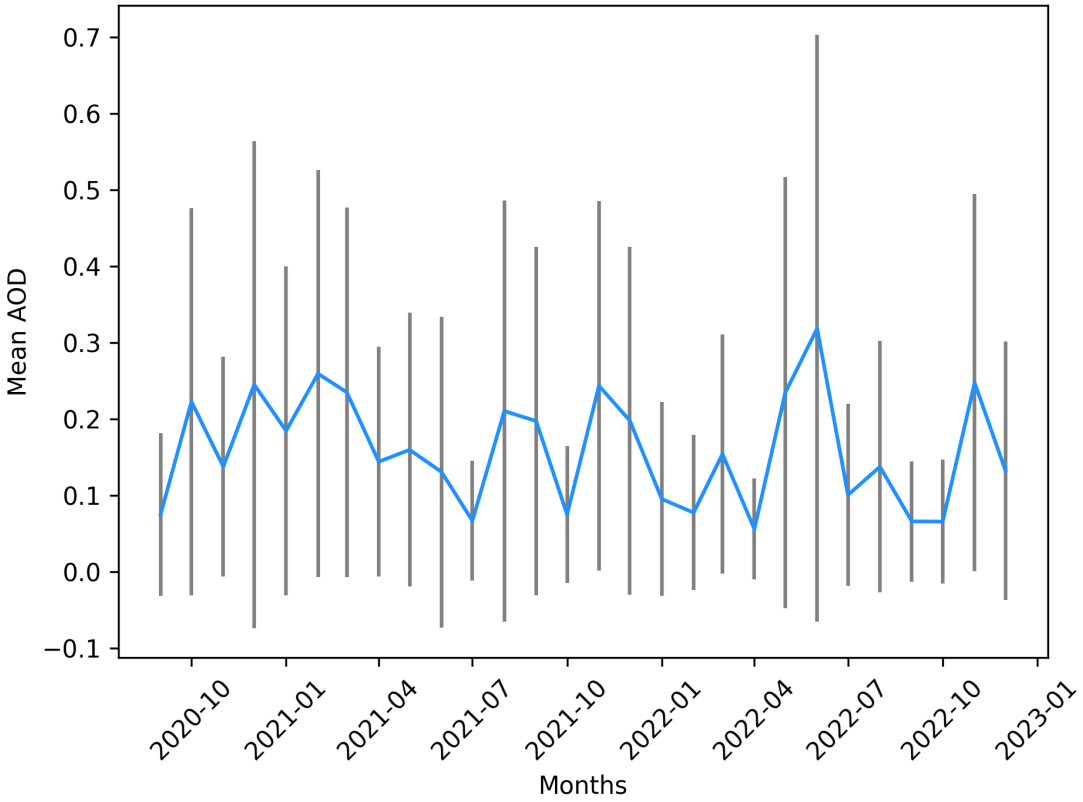


Figure 5.10: Seasonal trends of the mean by-species AOD for dust plumes from the 1st of September, 2020 to 31st of December, 2022. The graph shows the mean (curve) and standard deviation (bars) for each month. These trends agree with the climate statistics provided by Kunchala et al. [7].

required. Future annotation work on sea salt, dust or other species can follow our annotation protocol.

- The resolution of CAMS reanalysis data is relatively low, which can degrade the texture of aerosol plume observations. Moreover, this coarse resolution could lead to the missed detection of more complex plume features or potentially reduce the precision of the annotations of plume boundaries, affecting the overall precision of the final dataset. For example, plumes of size less than 0.75° by 0.75° in latitude and longitude do not appear on maps. To address this limitation, future work should include some efforts in the annotation of datasets of higher resolution, such as the MODerate resolution Imaging Spectroradiometer (MODIS) [154] for total AOD colocated with AERONET [153].

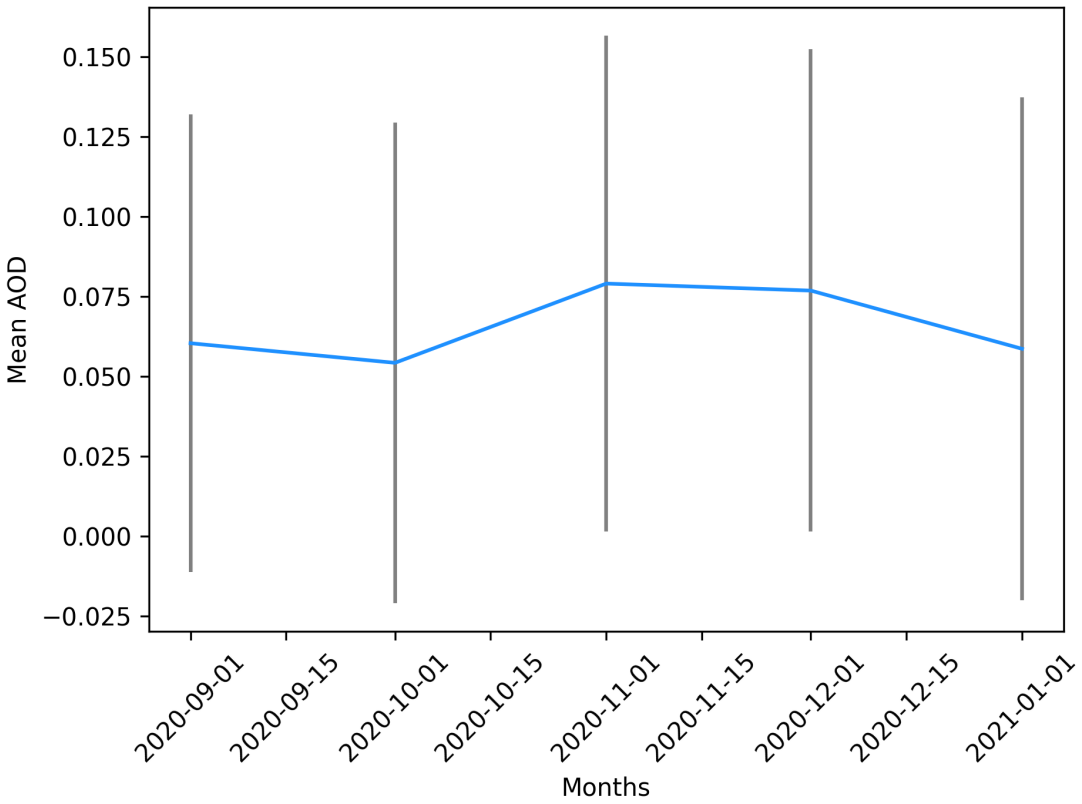


Figure 5.11: Seasonal trends of the mean by-species AOD for sea salt plumes from the 1st of September, 2020 to 31st of January, 2021. The graph shows the mean (curve) and standard deviation (bars) by month. These trends align with those of Aldhaif et al. [8].

This combination of datasets offers a spatial resolution of 0.1° by 0.1° but has a lower temporal resolution (two acquisitions per day), which is why we chose to work with CAMS EAC4 for tracking.

- Atmospheric and meteorological variability influences the visual properties of aerosol plumes. Variations in humidity, cloud cover, and boundary layer height (i.e, the height of the lowest air layer directly influenced by the surface, where most aerosols are concentrated) for different aerosol species or gases can affect the properties of AOD and its acquisition. While we selected a long observation period from September 2020 to December 2022 to capture seasonal and interannual variability, this dataset remains limited in capturing extreme aerosol plume events. To capture such events, future work should

be focused on locating such events specifically, to integrate them into our dataset.

These limitations pave the way to further improvements of our dataset.

5.5 Conclusion

In this chapter, we presented an annotated dataset for VOT of aerosol plumes based on the ECMWF CAMS EAC4 reanalysis AOD dataset. To the best of our knowledge, this is the first VOT dataset for aerosols. While CAMS provides reliable information on AOD at a global scale, CAMS EAC4 does not include plume tracking metadata such as object boundaries or trajectories. This contribution fills this gap by introducing bounding box annotations that identify aerosol plumes through sequences for two aerosol species, dust and sea salt.

To ensure the accuracy of the dataset, we used a preprocessing pipeline based on aerosol fraction maps and min–max normalisation to enhance plume contrast before annotation. We designed an annotation protocol to stop the sequence when a plume is too diffuse (i.e, confused with background noise) or when merging or splitting events occur. Several quality control analyses were presented, including size statistics, class distributions and intensity statistics such as AOD intensity histograms, box plots and monthly trends. All of these statistics of our manually annotated dataset align with the physical properties of aerosol events in CAMS EAC4 [6, 5]. Importantly, the imbalance between the number of annotated frames of dust and sea salt classes comes from annotation time constraints rather than misrepresented atmospheric properties.

This dataset gathers 267 sequences, which represent more than $28,761$ annotated aerosol plumes bounding boxes, organised in a structure compatible with standard VOT benchmarks. It provides a valuable resource to train, evaluate, and benchmark object tracking algorithms in the context of atmospheric science, en-

abling the machine learning and computer vision community to address challenges associated with diffuse, deformable, and evolving objects such as aerosol plumes.

Despite the advantages offered by this dataset, it remains limited in terms of representation of aerosol species, the amount of annotated sequences for sea salt, and it is based only on the CAMS EAC4 dataset, which has a lower resolution than MODIS coupled with AERONET. To address these limitations, future work will have to expand annotations to remaining species provided by CAMS EAC4 (i.e., black carbon, organic and sulphate) and improve the variability of annotated regions to capture extreme events. Nevertheless, the dataset constitutes a first step toward the development of an automated tracking of aerosol plumes using reanalysis and satellite data. It opens new opportunities for both atmospheric science and computer vision research, linking environmental monitoring needs with modern deep learning-based tracking methods.

Chapter 6

Spiking Siamese Tracker for VOT

VOT is a task in computer vision that aims to re-locate a targeted object through a sequence of images. VOT has seen significant advancements in recent years due to its applications in surveillance, autonomous driving, and human-computer interaction (see Chapter 3, Page 45). These trackers are now able to achieve high performance under various challenges, such as occlusions, background clutter, or fast motion.

As deep learning gained prominence in computer vision, researchers developed a new deep learning architecture, called Siamese networks [161]. This architecture has been adapted for VOT and improved trackers significantly [87]. The first Siamese-based network tracker, called the SiamFC [4], learns a similarity function between the target object and candidate regions in subsequent frames (see Section 3.5.4 Page 64).

Despite their success in VOT, SiamFC ANN [4] consumes a significant amount of energy both during training and inference because of its ANN backbone [162], making it unsuitable for embedding in external devices. Described as the third generation of neural networks, SNNs [22] process information through trains of spikes, offering efficient computation and energy savings when deployed on dedicated hardware [162] (see Chapter 4, Page 73).

However, using SNNs introduces new challenges to Siamese trackers. First, it requires handling a supplementary temporal dimension, in addition to the temporal dimension of the sequence of frames, due to the encoding of data as spikes. This additional temporal spiking dimension presents a new challenge, as Siamese-based trackers require perfect synchronisation between the spike train features extracted from the inputs [163]. Another challenge is the adaptation of pooling layers to SNNs, which is a highly discussed topic [164], and, to the best of our knowledge, has not found a definitive answer. Lastly, a representation in spikes needs to be defined, as no common ground has been established yet for spiking Siamese-based trackers (see Section 1.3, Page 28).

Thus, we propose a novel approach to spiking Siamese-based trackers, which uses a mechanism to synchronise thresholds between the two branches of a Siamese network. This threshold synchronisation increases training stability through threshold adaptation while maintaining the synchronisation of spike trains required between the branches of the Siamese tracker. Then, we choose to compute the similarity measure with an asymmetric accumulation of spikes between the two branches, accumulating spikes in the instance branch where we search for the object. To the best of our knowledge, this method has not been reported in the literature for spiking Siamese-based trackers. It can save energy by reducing the number of spikes transmitted to the similarity operator, while performing slightly better than accumulating in the branch of the object being searched. We refer to this method as Temporally Weighted Spike Count (TWSC) correlation and demonstrate its effectiveness in terms of stability and convergence.

In addition, we assess different spike encoding mechanisms, latency coding [165] and direct coding [41], and show that direct coding appears to be the most successful spike encoding strategy. The coupling of direct coding with the synchronisation of thresholds makes the model learn a better spiking representation of colour images [166]. Lastly, we use Temporal Efficient Training (TET) [167] on a spiking

Siamese-based tracker to compute the error at each timestep. It provides better feedback to the weights of our spiking SiamFC model.

In this chapter, we first introduce the baseline SiamFC ANN architecture in Section 6.1. We then describe our spiking SiamFC architecture in Section 6.2, including its neural coding mechanism (Section 6.2.1), synchronized threshold adaptation (Section 6.2.2), backbone architecture (Section 6.2.3), spike rate pooling layer (Section 6.2.4), TWSC similarity function (Section 6.2.5), and training procedure (Section 6.2.6). In Section 6.3, we present our experiments and analyze our experimental results. Lastly, we discuss our results in Section 6.4.

6.1 Baseline: SiamFC ANN

As mentioned in Section 3.5.4 (Page 64), SiamFC ANN aims to create a universal correlator by learning a similarity function and sharing weights across branches for each layer. Equation 6.1 provides the definition of the similarity function of SiamFC from Bertinetto et al [4]:

$$y = f(z, x) = \varphi(z) * \varphi(x) + b \tag{6.1}$$

where $*$ is the cross-correlation operator, y is the output similarity map, $z \in \mathbb{R}^{H_z \times W_z \times C}$ is the exemplar image fed to the examplar branch (with H_z, W_z, and C being the height, width, and number of channels of the examplar image, respectively), $x \in \mathbb{R}^{H_x \times W_x \times C}$ is the instance image fed to the instance branch (with H_x and W_x being the height and width of the instance image, respectively), $b \in \mathbb{R}$ is a similarity margin parameter which shifts all values in the output similarity map to center the distribution of the output similarity map, so that it is compatible with the logistic loss [95], and φ is the feature extractor model (usually, an AlexNet backbone).

SiamFC is trained by computing a logistic loss between the output similarity

scores map y and a generated groundtruth similarity score map $\hat{y}$ with a fixed-sized area representing the center of the object in the instance image. $\hat{y}$ is defined in Equation 6.2:

$$\hat{y}_{ij} = \begin{cases} +1, & \sqrt{(i_c - i)^2 + (j_c - j)^2} \leq \mathrm{R} \\ -1, & \text{otherwise} \end{cases} \tag{6.2}$$

with (i_c, j_c) the center of the bounding box associated with the object to track, (i, j) a location in the groundtruth similarity map $\hat{y}$, $\hat{y}_{ij}$ the value of $\hat{y}$ at location (i, j), and R a radius parameter controlling how much spatial context around the centre of the object is treated as a correct position.

From this, we can compute the logistic loss defined in Equation 6.3:

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{H_o} \sum_{j=1}^{W_o} \beta_{ij} \log\left(1 + e^{-y_{ij}\hat{y}_{ij}}\right) \tag{6.3}$$

and

$$\beta_{ij} = \begin{cases} \frac{1}{2M}, & \hat{y}_{ij} = 1 \\ \frac{1}{2N}, & \hat{y}_{ij} = -1 \end{cases}$$

where $\mathcal{L}$ is the logistic loss, y_{ij} is the predicted similarity score at position (i, j), $\hat{y}_{ij}$ is the value at position (i, j) of the similarity groundtruth map $\hat{y}$, and β_{ij} are balancing weights to equilibrate negative and positive similarity scores (i.e., equilibrate the scores from the background and the object location). H_o and W_o are the height and width of the output map, respectively. M and N are the total number of positive and negative values in the groundtruth map, respectively. Since background is often overly represented, this loss makes the network assign more weight to positive similarity scores (where the presence of an object is detected) and less weight to negative similarity scores (where the object is absent).

The authors use ImageNet Video 2015 [64] for training. They select pairs of exemplar and instance images for each sequence, following a uniform distribution

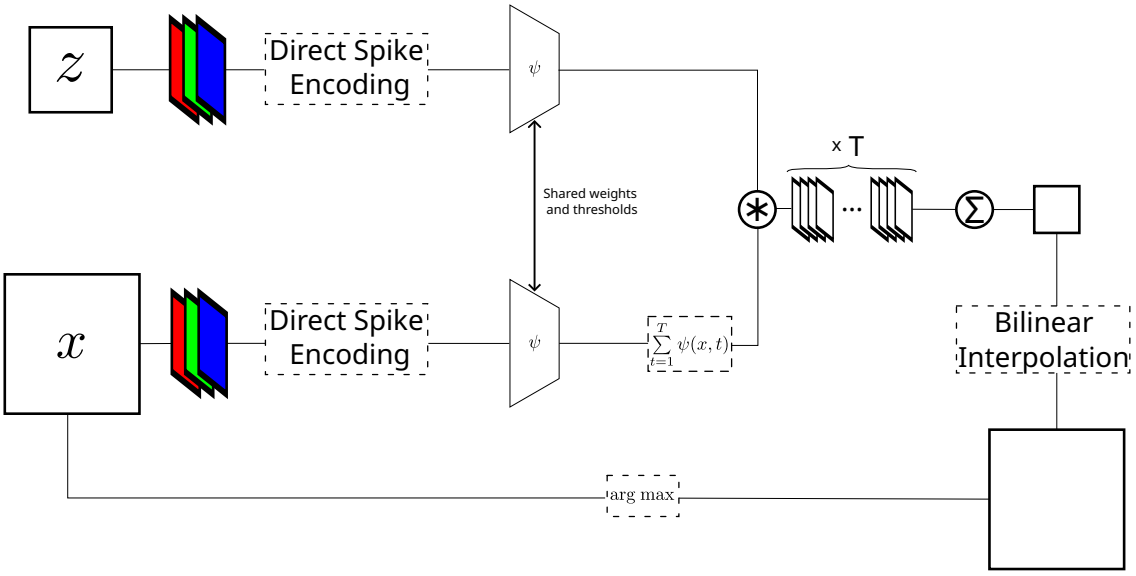


Figure 6.1: Overview of our spiking SiamFC model. x is the instance image, z is the examplar image, and ψ is the backbone.

for the selection of each frame within a sequence. These images are then cropped, around the bounding box for the exemplar image, and to twice the size of the bounding box of the object for the instance image. After cropping, images are padded to a square shape and resized to 127x127 and 255x255 pixels, respectively. The padding step prevents distortions during resizing.

6.2 Proposed Spiking SiamFC

The overall architecture of the proposed spiking SiamFC tracker is provided in Figure 6.1. First, we describe the neural coding strategy used to generate spiking inputs (Section 6.2.1). We then introduce the synchronised threshold adaptation mechanism between the Siamese branches (Section 6.2.2) and the backbone architecture used as spiking feature extractor (Section 6.2.3). Next, we present the spike rate pooling layer, which performs spatial downsampling in the spiking domain (Section 6.2.4) and our TWSC (Section 6.2.5). Finally, we detail the training procedure based on surrogate gradient backpropagation (Section 6.2.6).

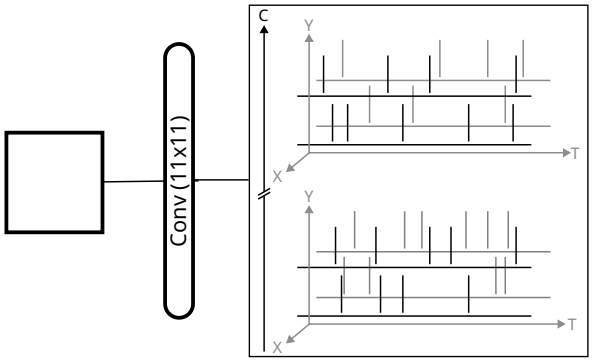


Figure 6.2: Direct coding of an input image.

6.2.1 Neural Coding

Direct coding [41] consists of directly providing the RGB image to the first spiking layer (see Figure 6.2). In our model, we feed input images directly into the first Convolutional Spiking layer (SConv) of each branch. This layer is composed of convolutional IF neurons with soft reset, see Equation 4.1 (Page 75), and their weight are shared between the two branches. In this configuration, the network learns the spike encoding of the pixels.

However, direct coding has some drawbacks. Since the same image is presented at each timestep, neurons in the first layer receive a constant input and tend to generate regular, periodic spike trains. In practice, the resulting spike trains contain redundant periodic spikes. In a Siamese tracker, this redundancy is problematic since the similarity function uses spiking feature maps from the exemplar and instance branches extracted from the same backbone. When computing the similarity, these periodic spike trains may reinforce the spatial match between the two inputs even when their actual similarity is low. This problem reduces the discriminative power of the spiking Siamese tracker by increasing the correlation values at erroneous locations. To counteract this effect, we introduce a synchronised threshold adaptation mechanism between the two branches (see next section) to allow a dynamic evolution of the threshold across time and, in the case of spike encoding, allow the first encoding layer to provide more temporally diverse and sparser spike

trains, which may lower energy consumption, as Lemaire et al. [10] have shown that decreasing spike frequencies can lower the energy consumption of SNNs.

6.2.2 Synchronised Threshold Adaptation for Siamese Networks

As mentioned in Section 6.2.1, we define a new method for synchronizing the threshold adaptation of membrane potentials within a Siamese network, which allows the branches to share their threshold at each layer. We first set a constant initial threshold for each layer of the backbone. The adaptation of the threshold is done by integrating the post-synaptic spikes to compute the post-synaptic spike rate, then decreasing or increasing the threshold accordingly after every time step, like in [168]. It is defined in Equation 6.4:

$$\theta^l(t) = \frac{1}{\tau_s}(\theta_{\text{base}} - \theta^l(t-1)) + \theta_{\text{base}} \cdot \bar{S}^l(t) \tag{6.4}$$

with

$$\theta_{\text{base}} \leq \theta^l(t) \leq 1$$

where $\theta^l(t)$ is the threshold of layer l at time t, $\theta^l(t) \in [0, 1]$, θ_{base} is the initial threshold value ($\theta_{\text{base}} \in [0, 1]$), τ_s is a positive time constant ($\tau_s > 0$) that controls the threshold adaptation, and $\bar{S}^l(t)$ is the average spike rate at layer l at timestep t. θ^l is defined per layer (i.e., one θ^l for all neurons in layer l).

At the end of each time step, the threshold is updated as the mean of the threshold of the two branches, and reset to θ_{base} at the end of each training iteration. The threshold synchronisation between examplar and instance branches is defined in Equation 6.5:

$$\theta_x^l(t+1) = \theta_z^l(t+1) = \frac{1}{2}\left(\theta_z^l(t) + \theta_x^l(t)\right) \tag{6.5}$$

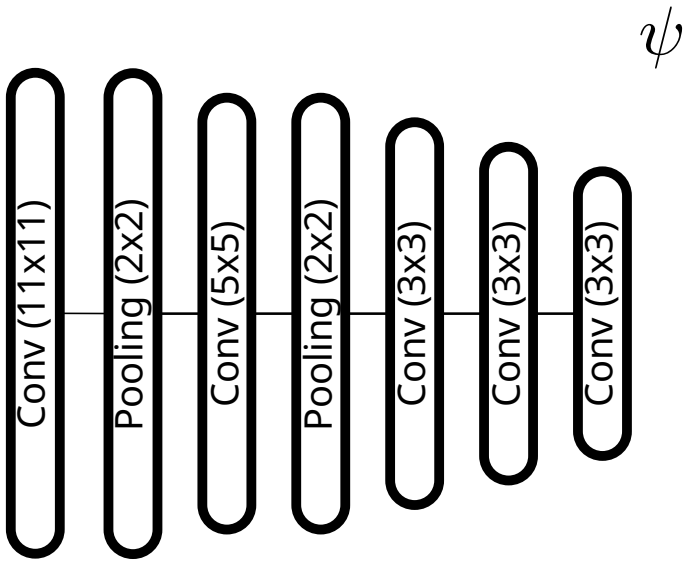


Figure 6.3: Backbone ψ of our model. This backbone is an adaptation of AlexNet [9] used in SiamFC ANN from Bertinetto et al. [4].

where θ_z^l is the threshold from the exemplar branch, and θ_x^l is the one from the instance branch, for a layer l.

6.2.3 Backbone Architecture

Figure 6.3 shows the backbone used in our tracker. For the feature extractor model ψ, we use AlexNet [9] as a basis, like in the SiamFC ANN from Bertinetto et al. [4], and replace the convolutional layers with convolutional spiking layers and the pooling layers with spike rate pooling layers(as defined in Section 6.2.4). Our convolutional spiking layer use the soft reset IF neurons described in Section 4.1.1 (Page 75). Further details on the backbone hyperparameters will be given in Table 6.1 (Page 135).

6.2.4 Spike Rate Pooling

Unlike the original AlexNet backbone used by Bertinetto et al. [4], we need to adapt the pooling operation to process spikes. LIF pooling [169] integrates spikes over time by first applying average pooling, and then feeding the result to an LIF

neuron, which introduces additional weights and a membrane potential that accumulates across timesteps. In our case, we want pooling to remain a simple spatial downsampling step that does not add extra temporal dynamics, to make it a simple spiking equivalent to pooling in ANNs. We therefore define a spiking pooling layer that directly combines average pooling with a thresholding operation. This design makes the pooling layer emit a spike when the average pooled value crosses a threshold θ_p, without learning new weights and without accumulating a membrane potential over time. As a result, pooling uses only presynaptic spikes at the current timestep, which reduces the computational burden, preserves sparsity, and maintains the local spatial spiking patterns extracted by the previous convolutional spiking layers. It is defined by Equation 6.6:

$$\bar{p}_{ij} = \frac{1}{k_h \cdot k_w} \sum_{u=-k_h/2}^{k_h/2} \sum_{v=-k_w/2}^{k_w/2} \eta_{i \cdot s_h + u, j \cdot s_w + v} \tag{6.6}$$

and

$$S_{ij}^{p} = \begin{cases} 0 & \text{if} \quad \bar{p}_{ij} < \theta_p, \\ 1 & \text{if} \quad \bar{p}_{ij} \geq \theta_p \end{cases}$$

where $\bar{p}_{ij}$ is the value at position (i, j) after average pooling over a window of size $k_h \times k_w$, η_{ij} is the input spike map at position (i, j), s_h and s_w are the vertical and horizontal stride, respectively, θ_p is the threshold, and S_{ij}^{p} is the value of the output spiking feature map at position (i, j).

This thresholded spiking average pooling layer allows output spiking feature maps to be reduced spatially while providing binary spikes as output instead of continuous values like standard average pooling, so that its output can be processed directly by the rest of the SNN.

6.2.5 Similarity Function

Like in SiamFC, our model learns a similarity function between the outputs of the two branches. The classic cross-correlation operator defined in Section 6.1 (Equation 6.1, Page 117) assumes feature maps with real values. In our spiking setting, however, each feature map is a sparse binary spike map. A classic cross-correlation operation between two spike maps only produces a significant response when spikes from the examplar and the instance branch occur at exactly the same time step and position. Because spikes are sparse, such exact synchrony is rare, so the resulting correlation output is almost always zero and does not provide a reliable similarity measure. Therefore, we must modify the cross-correlation operator to tolerate temporal misalignment between spikes. Thus, motivated by the work of Luo et al. [29], we define a new similarity function called TWSC. In TWSC, we accumulate spiking instance feature maps over timesteps. We apply this temporal accumulation only on the instance side because the instance image covers a larger search region and therefore contains much more background than the exemplar patch. As a result, the instance feature maps are more prone to spike-time jitter and spike artifacts caused by background structures. Temporal accumulation on the instance branch averages out these unstable spikes and stabilizes the similarity score, while keeping the exemplar feature maps unaccumulated preserves fine spatial spiking representation of the target examplar image, important for comparing the target examplar representation to each instance sub-region. This accumulation at the end of the instance branch does not include a leaky term, to retain the maximum amount of temporal information. Our TWSC similarity function is defined in Equation 6.7:

$$S_{\text{TWSC}}(z, x, t) = \psi(z, t) * \tau(x, t) \tag{6.7}$$

with

$$\tau(x,t) = \sum_{k=1}^{t} \psi(x,k) \tag{6.8}$$

where $S_{\text{TWSC}}(z,x,t)$ is the output similarity score map at time t, $\psi(\cdot,t)$ is the output of the spiking feature extractor at time t, z is the input examplar image, $\tau(x,t)$ is the accumulation of spikes from time steps 1 to t at the output of the instance branch, and x is the input instance image. After computing each similarity map $S_{\text{TWSC}}(z,x,t)$ at each time step until t, we sum these maps over time to generate the final response map and locate the object. This operation is defined in Equation 6.9:

$$y_t = f(x,z,t) = \sum_{k=1}^{t} S_{\text{TWSC}}(z,x,k) \tag{6.9}$$

where $S_{\text{TWSC}}(z,x,k)$ is the temporally weighted spike count at time k, and y_t is the predicted feature map accumulated from 1 to t.

This method allows the model to predict the location from the sum of similarity score maps across timesteps, which allows for reducing spike noise in the final location prediction after the bilinear interpolation step. This interpolation step is identical to the one SiamFC ANN from Bertinetto et al. [4] (for further details, see Section 6.1, Page 117).

6.2.6 Training

To train our spiking SiamFC, we need to define a cost function adapted to the dynamics of the accumulated similarity score maps (see Equation 6.9). Our loss is composed of two terms, a classification term $\mathcal{L}$ and a regularization term $\mathcal{R}$. The total loss is defined in Equation 6.10:

$$\mathcal{L}_{\text{total}}(\boldsymbol{X}, y, \hat{y}) = \mathcal{L}(y, \hat{y}) + \mathcal{R}(\boldsymbol{X}) \tag{6.10}$$

where $\boldsymbol{X}$ is the input batch, y the output similarity map and $\hat{y}$ the groundtruth similarity map defined as in Section 6.1 (Equation 6.2, Page 118).

For the classification loss, we choose to use Binary Cross-Entropy (BCE) [170] at each time step, inspired by the TET [171] framework. However, since the background occupies a larger area compared to the object of interest in the region where we search for the object, we also need to balance this loss function in the same manner as in Equation 6.3 (Section 6.1, Page 118). Thus, we define a balanced temporal BCE, expressed in Equation 6.11:

$$\mathcal{L}(y, \hat{y}) = \sum_{n=1}^{N_b} \sum_{t=1}^{T} -\beta_n [\hat{y}_n \log y_{tn} + (1 - \hat{y}_n) \log (1 - y_{tn})] \tag{6.11}$$

where N_b is the batch size, T is the number of steps, β_n is the balanced weights for the n-th sample of the batch, as defined in Equation 6.3, $\hat{y}_n$ is the groundtruth map for the n-th sample of the batch, y_{tn} is the similarity response map at time t for the n-th sample of the batch, as defined in Equation 6.9 of our model.

For the regularization term, to maintain a sufficient spike rate and avoid the dead neuron problem [39], we add a spike rate regularization term [39] to our total cost function. This spike rate regularization is defined as:

$$\mathcal{R}(\boldsymbol{X}) = \frac{1}{L} \sum_{l=1}^{L} (\rho^l(\boldsymbol{X}) - \hat{\rho})^2 \tag{6.12}$$

$$\rho^l(\boldsymbol{X}) = \frac{1}{N_b Q^l T} \sum_{n=1}^{N_b} \sum_{t=1}^{T} \sum_{m=1}^{Q^l} S^l_{nm}(t)$$

where $\mathcal{R}(\boldsymbol{X})$ is the spike rate regularization term, $\boldsymbol{X}$ is the input batch of data, N_b is the batch size, T is the number of time steps, Q^l is the number of neurons (or spatial positions) in layer l, S^l_{nmt} is the spike of neuron m in layer l at time t for sample n, ρ^l is the mean firing rate of layer l (averaged over N_b, T, and Q^l), L is the number of layers, and $\hat{\rho}$ is the target firing rate, which is a manually set hyperparameter.

Now that we have defined the cost function of our spiking SiamFC, we can apply backpropagation using the surrogate gradient method [39] to the branches of the model. Spikes are not differentiable since the threshold of a spiking neuron acts as a Heaviside step function. Therefore, classic backpropagation cannot be applied. A solution, as described in [40], is to replace the Heaviside function with a continuous one during the backward pass. We decide to use the sigmoid function as a surrogate activation function, defined in Equation 6.13:

$$\sigma(\epsilon) = \frac{1}{1 + e^{-\epsilon}}. \tag{6.13}$$

where ϵ is an input value, $\epsilon \in \mathbb{R}$. Its derivative (i.e., the surrogate gradient function) is given by Equation 6.14:

$$\frac{\partial \sigma(\epsilon)}{\partial \epsilon} = \sigma(\epsilon)\left(1 - \sigma(\epsilon)\right). \tag{6.14}$$

The surrogate function is used in BPTT [172] on the unrolled SNN graph of spiking SiamFC. Figure 6.4 shows a schematic of the unrolled graph of spiking SiamFC. Since the computation of the gradient is similar to that of RNNs, we can use the chain rule to compute the error of each neuron. To apply the gradient descent algorithm, we need to compute the global loss, which, in the case of SNNs, is given by Equation 6.15[1]:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial W}. \tag{6.15}$$

Unrolling the spiking SiamFC architecture (see Figure 6.4), we obtain Equation 6.16:

$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}_t}{\partial \tau(x,t)} \frac{\partial \tau(x,t)}{\partial W} + \frac{\partial \mathcal{L}_t}{\partial \psi(z,t)} \frac{\partial \psi(z,t)}{\partial W} \tag{6.16}$$

where $\mathcal{L}$ is the loss function, W are the weights of the model, $\frac{\partial \mathcal{L}_t}{\partial \tau(x,t)} = \text{Corr}\left(\frac{\partial \mathcal{L}_t}{\partial y_t}, \psi(z,t)\right)$

[1]For clarity of notation, we omit the parameters of $\mathcal{L}_t$, and write only its time index t in the derivations.

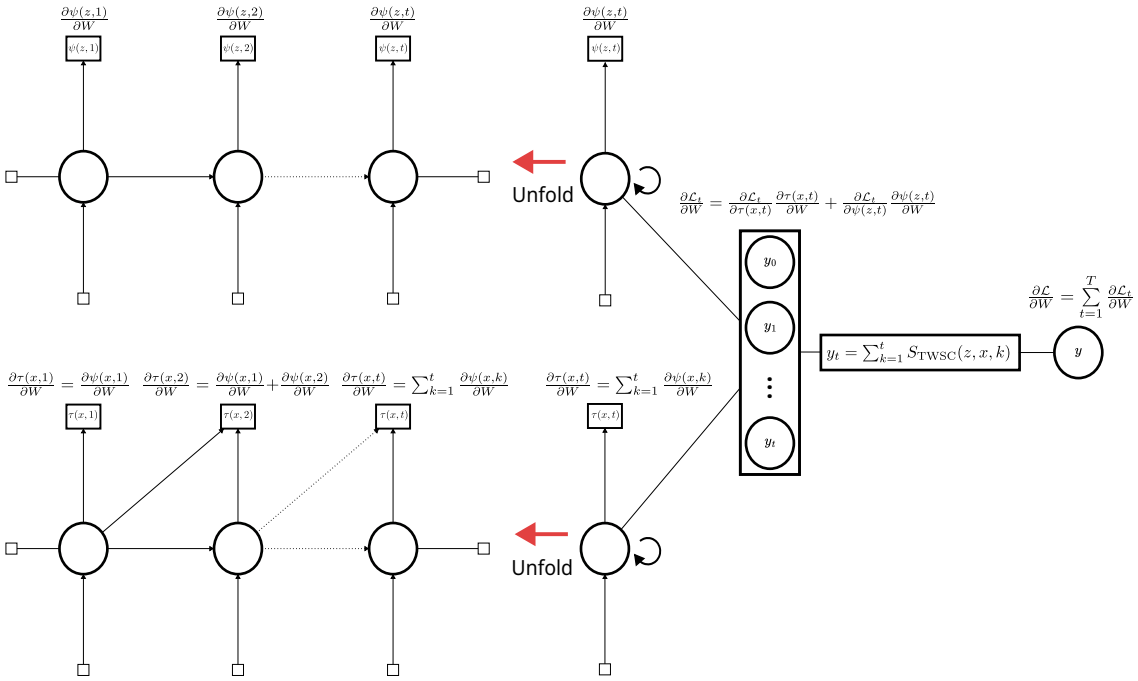


Figure 6.4: Unfolding and chain rule computation of our model for surrogate gradient backpropagation, showing the error gradients across multiple time steps. The left-hand part represents the unfolded computational graph with the output gradients. The right-hand part shows the computation of the similarity with its gradient.

is the adjoint operator [173] (i.e., the linear operator that maps gradients from the output of a layer back to its inputs) of the instance branch, Corr is the correlation operator, and $\frac{\partial \mathcal{L}_t}{\partial \psi(z,t)} = \text{Corr}\left(\frac{\partial \mathcal{L}_t}{\partial y_t}, \tau(x,t)\right)$ is the adjoint operator of the exemplar branch. We denote by $\text{Corr}(a,b)$ the cross-correlation between two feature maps a and b (i.e., the operation that slides a over b and computes a sum of element-wise products at each spatial position). Corr is defined in Equation 6.17:

$$\text{Corr}_{ij}(a,b) = \sum_{u=-H_r/2}^{H_r/2} \sum_{v=-W_r/2}^{W_r/2} a_{u+H_r/2,v+W_r/2}\, b_{i+u,j+v} \tag{6.17}$$

where Corr_{ij} is the value of the cross-correlation at position (i,j), H_r and W_r are the height and the width of the correlation window, respectively. The cross-correlation is a linear operator [173] whose gradient with respect to each input is given by the cross-correlation between the gradient of the loss and the feature map from the opposite branch. Lastly y_t is the output map at time t, as defined on Equation 6.9

on Page 125.

Since $\tau(x,t)$ is the accumulated spiking feature maps without decay, we can define its derivative with respect to the weights as:

$$\frac{\partial \tau(x,t)}{\partial W} = \sum_{k=1}^{t} \frac{\partial \psi(x,k)}{\partial W} \tag{6.18}$$

If we substitute Equation 6.18 into Equation 6.16, we obtain Equation 6.19:

$$\begin{aligned} \frac{\partial \mathcal{L}_t}{\partial W} = &\operatorname{Corr}\left(\frac{\partial \mathcal{L}_t}{\partial y_t}, \psi(z,t)\right) \cdot \left(\sum_{k=1}^{t} \frac{\partial \psi(x,k)}{\partial W}\right) \\ &+ \operatorname{Corr}\left(\frac{\partial \mathcal{L}_t}{\partial y_t}, \tau(x,t)\right) \cdot \frac{\partial \psi(z,t)}{\partial W} \end{aligned} \tag{6.19}$$

Thus, by combining Equations 6.15 and 6.19, we get the complete gradient formula (Equation 6.20):

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \Bigg[&\operatorname{Corr}\left(\frac{\partial \mathcal{L}_t}{\partial y_t}, \psi(z,t)\right) \cdot \left(\sum_{k=1}^{t} \frac{\partial \psi(x,k)}{\partial W}\right) \\ &+ \operatorname{Corr}\left(\frac{\partial \mathcal{L}_t}{\partial y_t}, \tau(x,t)\right) \cdot \frac{\partial \psi(z,t)}{\partial W} \Bigg] \end{aligned} \tag{6.20}$$

Equation 6.20 makes explicit the temporal credit assignment performed by back-propagation through time in our spiking SiamFC. Because we supervise only the final output of the unfolded network (see Figure 6.4), the gradient $\frac{\partial \mathcal{L}_t}{\partial y_t}$ is non-zero only at the last timestep. Nevertheless, this signal is propagated backward through all preceding timesteps via the accumulated representation $\tau(x,t)$ and the integration over m in Equation 6.20. Finally, since the weights are shared between the exemplar and instance branches, the total gradient with respect to W is obtained by accumulating the error contributions from both branches. The full training pipeline is outlined in Algorithm 1.

Algorithm 1 Training algorithm for spiking SiamFC.

Initialize model parameters W
Set number of epochs E
Set number of time steps T
for each epoch $e = 1, 2, \ldots, E$ **do**
    for each batch of training data N_b **do**
        Draw a pair of exemplar and instance images (respectively z and x) from a sequence.
        for each time step $t = 1, 2, \ldots, T$ **do**
            Compute $\psi(z, t)$ (Eq. 6.7)
            Compute $\psi(x, t)$ (Eq. 6.7)
            Add $\psi(x, t)$ to $\tau(x, t)$ (Eq. 6.8)
            Compute $S_{\mathrm{TWSC}(z,x,t)}$ (Eq. 6.7)
            Compute $\theta_z(t)$ (Eq. 6.4)
            Compute $\theta_x(t)$ (Eq. 6.4)
            Compute the mean between $\theta_z(t)$ and $\theta_x(t)$ and update the threshold (Eq. 6.5).
        end for
    end for
    Compute the balanced binary cross-entropy loss $\mathcal{L}$ over all time steps t (Eq. 6.11)
    Compute $\frac{\partial \mathcal{L}}{\partial W}$ (Eq. 6.20)
    Update parameters
    Evaluate the IoU of the model on the validation set
    if The validation IoU does not improve for P consecutive epochs **then**
        Break
    end if
end for

6.3 Experiments

In this section, we present an evaluation of our spiking SiamFC tracker. We start by describing the datasets used for training and benchmarking in Section 6.3.1, followed by the evaluation metrics in Section 6.3.2. Then, we introduce the evaluation protocol in Section 6.3.3, and the implementation details in Section 6.3.4. We present and analyze the experimental results in Section 6.3.5. Finally, we perform an ablation study (Section 6.3.6) to analyse the impact of the following contributions: synchronised threshold adaptation, asymmetric spike accumulation, and direct coding.

6.3.1 Datasets

In order to make a fair comparison of our method to the original SiamFC from Bertinetto et al. [4], we evaluate our Spiking SiamFC model on the OTB-2015 [42] datasets (which includes OTB 2013 [78]), and on the GOT-10k [43] dataset.

OTB is a widely used dataset and evaluation protocol for visual object tracking. This dataset contains 100 sequences of single objects to track. Each sequence in OTB consists of a single object annotated by a bounding box at every frame. Sequence lengths vary across sequences, ranging from 71 to 5,288 frames, with an average length of around 600 frames. The frames have a variable resolution between 240 by 320 and 1080 by 1920 pixels. The objects tracked in OTB are not classified into categories. Each sequence is annotated with one bounding box and features some classical challenges of visual object tracking, such as occlusions, illumination changes, background clutter, or scale changes.

GOT-10k is a large-scale and diverse benchmark dataset designed for SOT. It contains $10,535$ video sequences with 563 classes, making it one of the most extensive VOT benchmarks available. Sequences are annotated with one bounding box at every frame. Sequences vary in length, with an average sequence length of

300 frames, ranging from a few dozen to a few hundred frames, with an average length of around 150 frames. Frames have multiple resolutions, ranging from 480 by 270 to 1920 by 1080 pixels, reflecting real-world conditions. The dataset includes a wide range of object categories, such as animals, vehicles, humans, and multiple everyday objects.

6.3.2 Evaluation Metrics

As introduced in Section 3.2 (Page 47), the primary metrics used to evaluate tracking performance in VOT are the success rate and the precision rate. The success rate is a metric derived from the IoU, and measures the ratio of successfully tracked frames given an overlap threshold (Equation 6.21):

$$\mathrm{SR}(\theta_{\mathrm{IoU}}) = \frac{1}{F}\sum_{k=1}^{F} \mathbf{1}\big[\mathrm{IoU}_k \geq \theta_{\mathrm{IoU}}\big], \tag{6.21}$$

with:

$$\mathrm{IoU}_k = \frac{\mid B_k \cap \hat{B}_k \mid}{\mid B_k \cup \hat{B}_k \mid}$$

where SR is the success rate, and θ_{IoU} is the threshold on the overlap between the two areas. B_k and $\hat{B}_k$ are the predicted and ground truth bounding boxes at frame k, respectively; F is the sequence length, $\mathbf{1}[\cdot]$ is the indicator function, which equals 1 if the condition is true and 0 otherwise, $\mid \cdot \cap \cdot \mid$ is the intersection operator, and $\mid \cdot \cup \cdot \mid$ is the union operator.

The success plot is obtained by computing the average success rate over all sequences for thresholds ranging from 0% to 100%. From this curve, we deduce the AUC, which is used as a single score to rank trackers.

Precision is defined as the percentage of frames where the Euclidean distance between the center of the predicted bounding box and the center of the groundtruth bounding box is within a specified threshold. Precision is defined in Equation 6.22:

$$\mathrm{P}(\theta_p) = \frac{1}{T}\sum_{k=1}^{F} \mathbf{1}\big[\|c_k - \hat{c}_k\|_2 \leq \theta_p\big], \tag{6.22}$$

where $\mathrm{P}(\theta_p)$ is the precision score at the threshold θ_p, c_k and $\hat{c}_k$ are the centers of the predicted and groundtruth bounding boxes, respectively, at frame k; θ_p is the threshold on the distance, F is the sequence length, and $\mathbf{1}[\cdot]$ is the indicator function.

6.3.3 Evaluation Protocol

Our tracker is trained on ImageNet VID 2015 [64] following the same training protocol as the original SiamFC paper (see Section 6.1, Page 117), using the 4,417 videos that we decompose into 10,696 sequences of 152 frames on average. We follow the standard splits of ImageNet VID 2015 into training, validation, and test sets. We use the training and validation sets for our training and validation phase, respectively. The training set contains $3,862$ videos, while the validation set contains 555 videos. We filtered out sequences containing fewer than 10 frames to ensure that the model always learns from objects with sufficient motion speed. For each sequence, two frames distant in time by at most 50 frames are sampled following a uniform distribution. The earliest frame is cropped around the bounding box of the target object to produce the examplar image. The other frame is cropped around the bounding box of the target object, but with dimensions twice the ones of the bounding box, to produce the instance image. These cropped images are then padded to a square shape (using their respective mean colour) and resized to 127×127 and 255×255 for the examplar and instance images, respectively. After the training phase, we evaluate our spiking SiamFC tracker using the standard One-Pass Evaluation (OPE) protocol [42], which is widely adopted in the VOT literature. In OPE, the tracker is initialized with the ground-truth bounding box on the first frame of each sequence and then runs forward until the last frame

without any re-initialization or correction after failures. This setting measures how well the tracker can continuously follow the target over the entire sequence, and all performance metrics (e.g., success rate and precision) are computed over this single pass for each sequence. Due to the prohibitive training time of our spiking SiamFC, we report results from a single training run for this comparison.

6.3.4 Implementation Details

During training, we minimize the loss defined by Equation 6.11. The parameters are initialized with a normal distribution $\mathcal{N}(0, 1)$ following the Xavier scheme [174]. We train the network for a maximum of 1000 epochs and monitor the validation loss at each epoch. In practice, in order to observe the learning stability, training is not stopped early, but we select our final model as in early stopping: we keep the model at the epoch at which the validation IoU first reaches a plateau. This plateau is found when no further improvement on IoU occurs over the next 50 consecutive epochs. In practice, this plateau typically occurs after roughly 120 training epochs. We use the Adaptive Moment Estimation (Adam) optimizer [175] with an exponential scheduler [176] decreasing the learning rate by a factor γ after each epoch. γ is defined in Equation 6.23:

$$\gamma = \left(\frac{\alpha_{\text{final}}}{\alpha_{\text{init}}} \right)^{\frac{1}{E}} \tag{6.23}$$

where α_{final} is the minimum learning rate, α_{init} is the initial learning rate, and E is the number of epochs. After an extensive search, we found that our model starts to converge at an initial learning rate of learning rate of $\alpha_{\text{init}} = 0.0005$. We set the final learning rate α_{final} to 0.00001. The time constant τ_s is set to 5. We define the mini-batch size as $N_b = 64$ samples. The kernel sizes, numbers of filters, and the initial thresholds of each layer are provided in Table 6.1. We set the number of time steps to $T = 5$ matching the period used in SiamSNN [29]. This number

of timesteps offers a good trade-off between tracking accuracy and computational cost. Finally, the threshold of Table 6.1 for SCONV and spike rate pooling layers has been selected to match at least a target spike rate $\hat{\rho} = 10\%$ of the spike rate in the first five epochs in each layer, in order to prevent the dead neuron problem.

In this chapter, we reproduce the experiments from Bertinetto et al. [4] on OTB-2015 to get the evaluation metrics of SiamFC ANN on this dataset, but without introducing 25% of grayscale images in the training set, unlike Bertinetto et al. [4]. Experiments in this chapter have been done on a single run, which is a common practice in the tracking community [3, 4, 177]. For the spiking SiamFC model equipped with all of our contributions, we trained 10 models and report the results of the one with the highest validation IoU.

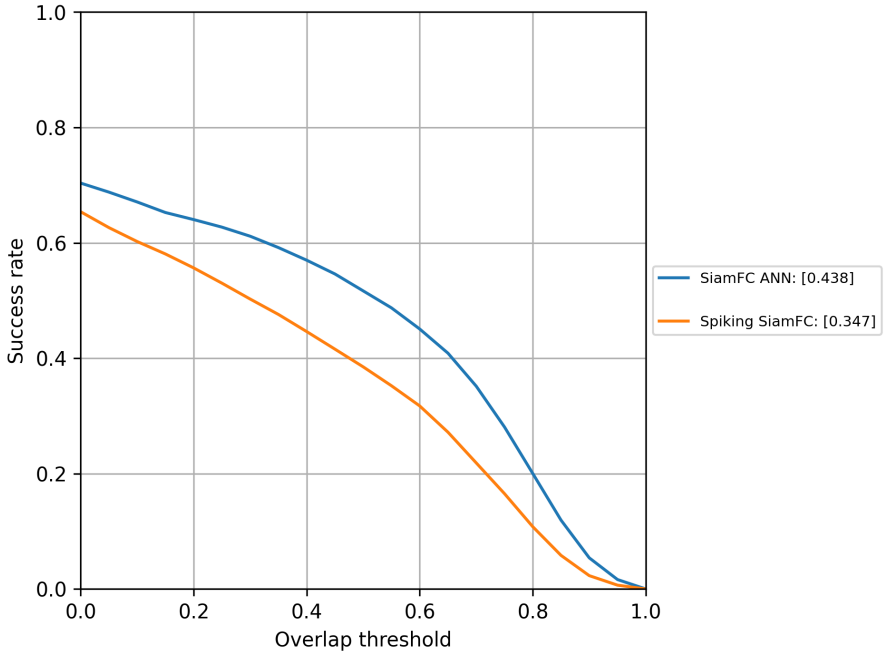
Layer type	**Kernel size**	**Number of filters**	**Stride***	**Threshold****
SCONV	11x11	96	2	0.03
Pooling	3x3	96	2	0.2
SCONV	5x5	256	1	0.05
Pooling	3x3	256	2	0.2
SCONV	3x3	384	1	0.05
SCONV	3x3	384	1	0.05
SCONV	3x3	256	1	0.05

Table 6.1: Details of the backbone ψ. Convolution layers are two-dimensional spiking convolutional layers equiped with our synchronized threshold adaptation (see Section 6.2.2, Page 121), and pooling layers use our spike rate pooling layer (see Section 6.2.4, Page 122). * same in both dimensions. ** threshold is $\theta_{\mathbf{base}}$ for convolutional layers and θ_p for pooling layers.

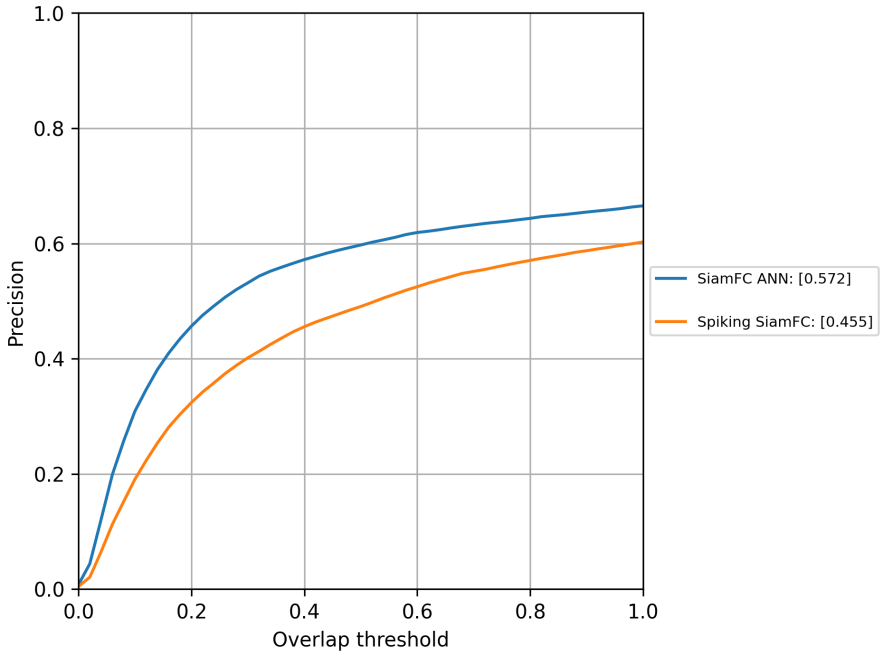
6.3.5 Benchmark Evaluation

In this section, we compare the tracking performance of our spiking SiamFC to the original SiamFC ANN baseline [4, 43] on the OTB-2015 and GOT-10k benchmarks. We report the success plots, the precision plots, and their AUCs under the standard OPE protocol.

Figure 6.5a shows the success rate plots and AUCs of our spiking SiamFC and



(a)



(b)

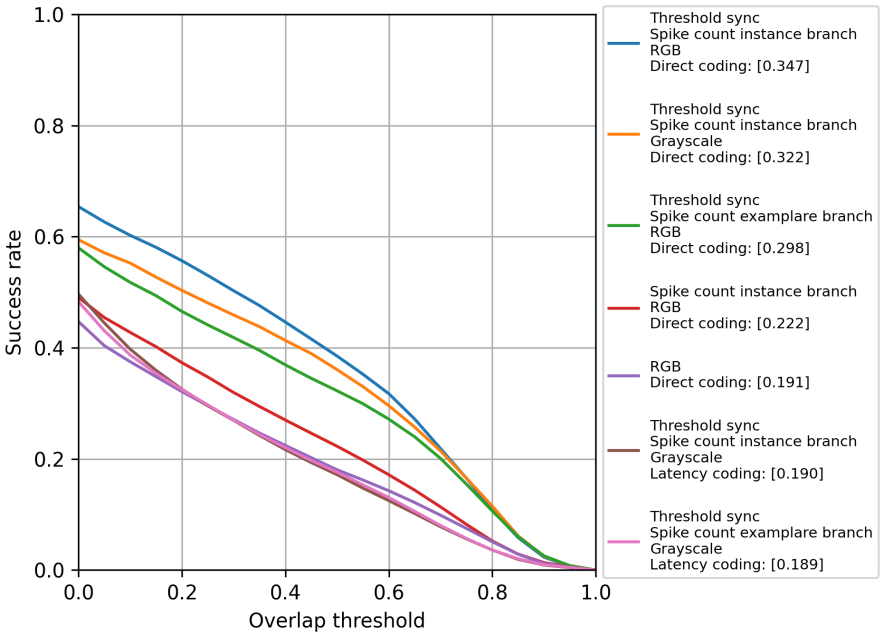
Figure 6.5: Success plot and precision plot for OPE of spiking SiamFC on the OTB-2015 benchmark (values in legends are AUC scores). Values of the SiamFC ANN have been obtained by training our own SiamFC ANN, following the protocol provided by Bertinetto et al. [4] (except for the conversion of 25% of the training set to grayscale). (a): Comparison of the success rate across the whole overlap threshold range between our spiking model and the original SiamFC from Bertinetto et al. [4]. (b): Comparison of the precision rate across the whole overlap threshold range between our spiking model and the original SiamFC from Bertinetto et al. [4].

the original SiamFC on OTB-2015. Results show that our model has a lower performance than its ANN counterpart (an AUC of 0.347 for spiking SiamFC against 0.438 for SiamFC ANN); this behavior was expected since SNNs are typically more difficult to train than ANNs because of the loss of information due to the quantization of the original input and activations into spikes. However, we can see that the curves of our model and of SiamFC (orange and blue curves, respectively) have similar shapes, with a gap of 9.1% in AUC. Figure 6.5b shows the precision plots and AUCs of our spiking SiamFC and the original SiamFC on OTB-2015. Results confirm that our model has a lower performance than its ANN counterpart (an AUC of 0.455 for spiking SiamFC against 0.572 for SiamFC ANN). For GOT-10k, we get an AUC score of 0.397 for spiking SiamFC, while Huang et al. [43] report 0.392 for SiamFC ANN [43]. This slight difference suggests that spiking SiamFC can be as good as SiamFC ANN on this benchmark. These results indicate that, on OTB-2015, the ANN version of SiamFC provides more accurate localization of the target center than its spiking counterpart.

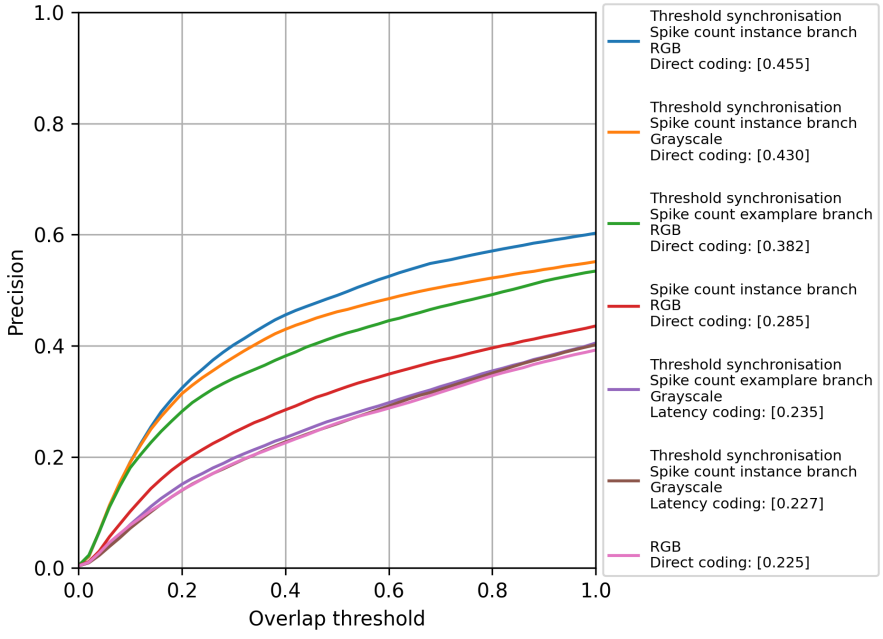
Table 6.2: Success rates at 50% of SiamSNN and spiking SiamFC on OTB-2015 and GOT-10k.

Tracker	OTB-2015	GOT-10k
SiamSNN (reported from [29])	0.443	0.327
Spiking SiamFC (ours)	0.385	0.411

Lastly, we compare the success rate at 50% IoU reported by Luo et al. [29] for SiamSNN to that of our spiking SiamFC. These results are displayed in Table 6.2. SiamSNN achieves success rates of 0.443 and 0.327 on OTB-2015 and GOT-10k, respectively, while our model obtains 0.385 and 0.411 on the same benchmarks. These results suggest that our model performs slightly worse on OTB-2015 but clearly better on GOT-10k. However, although it relies on a single run (both for SiamSNN and Spiking SiamFC), this result clearly shows a trend in the performance of these trackers.



(a)



(b)

Figure 6.6: Success plot and precision plot for OPE of spiking SiamFC on the OTB-2015 benchmark. (a): Success curve of each specific ablation setting in Table 6.3. Values in the legend represent the AUCs. (b): Precision curve of each specific ablation setting in Table 6.3.

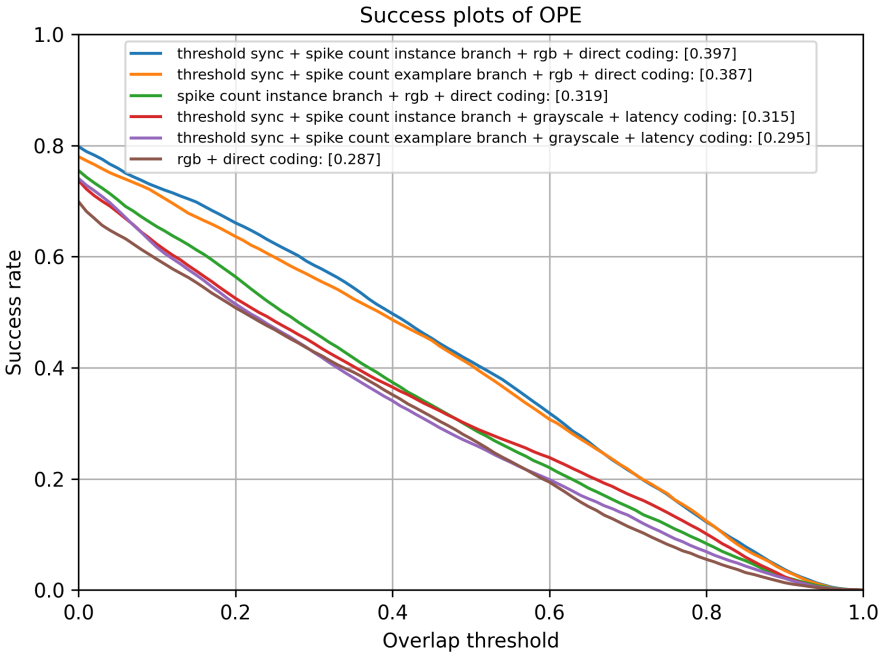


Figure 6.7: Success plot for OPE of spiking SiamFC on the GOT-10k benchmark (values in legends are AUC scores). Each curve represents the success rate of a specific ablation setting with respect to the overlap threshold. The orange curve shows our model without ablation.

6.3.6 Ablation Study

In order to analyse the influence of each of its components on the generalization ability of our spiking Siamese tracker, we consider the following configurations:

- synchronized threshold adaptation: with and without synchronization between the exemplar and instance branches;
- head symmetry and accumulation branch: symmetric head (i.e., using spike accumulation in both branches) against asymmetric head (i.e., using spike accumulation only in one branch, either the examplar branch or the instance branch);
- neural coding: direct input coding (on RGB or grayscale images) against latency coding (on grayscale images).

The result of success rates at 50% of our ablation study are summarised in

Model	Threshold synchronisation	Asymmetry	Accumulation branch	Encoding	Success rate at 50% OTB-2015	Success rate at 50% GOT-10k
Spiking SiamFC	✓	✓	Instance	Direct (RGB)	0.385	0.411
Spiking SiamFC	✓	✓	Instance	Direct (Gray)	0.361	0.421
Spiking SiamFC	✓	✓	Exemplar	Direct (RGB)	0.322	0.405
Spiking SiamFC	✗	✓	Instance	Direct (RGB)	0.223	0.291
Spiking SiamFC	✗	✗	N/A	Direct (RGB)	0.180	0.272
Spiking SiamFC	✓	✓	Exemplar	Latency (Gray)	0.176	0.264
Spiking SiamFC	✓	✓	Instance	Latency (Gray)	0.172	0.295

Table 6.3: Results of the ablation study on OTB-2015 and GOT-10k.

Table 6.3. First, the results in Table 6.3 show that our synchronised threshold adaptation mechanism is necessary to improve model accuracy (0.385 and 0.411 against 0.223 and 0.291, for OTB-2015 and GOT-10k, respectively). In addition to this, Figure 6.8 shows the evolution of the validation IoU over training epochs, comparing runs with and without synchronised threshold adaptation. It shows a stabilisation of the validation IoU with synchronised threshold adaptation during training. Without this mechanism, we can see a sudden drop in IoU during training (from 0.45 to almost 0 over two epochs after epoch 110). This result tends to confirm that synchronised threshold adaptation enables the model to have more robust training.

Next, we analyse the effect of using a symmetric head compared to spike accumulation only in the instance branch. The results in Table 6.3 show that a symmetric head (i.e., accumulation in both branches) decreases the performance of the tracker: the success rate at 50% drops from 0.0.291 to 0.272 on GOT-10k, and from 0.223 to 0.180 on OTB-2015. This superiority of asymmetric heads may be due

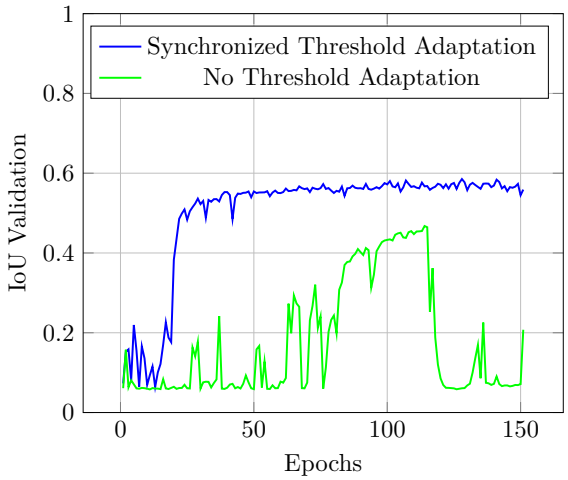


Figure 6.8: IoU on the validation set with and without synchronized threshold adaptation (blue curve and green curve, respectively).

to a loss of temporal information when spikes are accumulated in both branches, leading the model to learn poorer representations of the target object. We also observe that accumulating spikes in the instance branch yields better results than accumulating them in the exemplar branch (from 0.411 to 0.405 on GOT-10k, and from 0.385 to 0.322 on OTB-2015, respectively), which suggests that preserving a spiking (non-accumulated) representation for the exemplar feature maps is particularly important. In addition to providing better accuracy, accumulation only in the instance branch is expected to lower computational cost, since it reduces the number of spikes involved in the similarity operation (see Equation 6.8).

Now, we analyse the impact of using colour versus grayscale inputs. The results in Table 6.3 show that encoding RGB images with direct rate coding yields similar success rates at 50% IoU than encoding grayscale images: for the asymmetric head with accumulation in the instance branch, performance decreases from 0.385 to 0.361 on OTB-2015, but increase from 0.411 to 0.421 on GOT-10k, when switching from RGB to grayscale. This mixed result shows that colour information is not really exploited by the model and suggests that the model could be improved in future work on this aspect.

Finally, regarding the neural coding method, experimental results in Table 6.3

show that the direct coding approach outperforms latency coding in terms of success rate (0.361 and 0.421 against 0.172 and 0.295 on OTB-2015 and GOT-10k, respectively). Figure 6.6b shows the same results with precision metrics, with a precision score of 0.227 for latency coding and 0.430 for direct coding. So, direct coding spiking representation of input appears to provide better features for differentiating the target object from the background. For each configuration, Figure 6.6a and Figure 6.7 show the success plots on OTB-2015 and GOT-10k, respectively. The shape of the plots of precision and success rate is consistent with the previous observations.

The results of our ablation study indicate that each of the proposed components contributes significantly to closing the gap between the ANN and SNN versions of SiamFC, showing their necessity for designing an accurate spiking Siamese tracker.

6.4 Discussion

In this chapter, we compared our model to the original SiamFC model [4], and the ANN-to-SNN model SiamSNN [29]. During our experiments, we noticed a difference in performance between our model and SiamFC on OTB-2015, but not on GOT-10k. This difference may be due to the number of classes in the two datasets, suggesting that our model is better at handling large amounts of classes, such as in GOT-10k, than at handling fewer classes, as in OTB-2015. Furthermore, OTB-2015 contains a non-negligible amount of grayscale data. As a consequence, the lower performance we observe on OTB-2015 may be due to this difference, since we trained spiking SiamFC exclusively on colour images, contrary to Bertinetto et al. [4], who included 25% of grayscale images in their training set. Compared to SiamSNN, our model shows better performance on GOT-10k while performing worse on OTB-2015. This difference on OTB-2015 was expected, since SNN models trained directly with surrogate gradient rarely outperform SNNs converted from ANNs. However, this

result suggests that SiamSNN is slightly better adapted to the specific classes of objects of OTB-2015, while our spiking SiamFC provides stronger generalization on large-scale benchmarks with a larger number of more diverse classes, such as in GOT-10k.

We also noticed, during our experiments, a strong dependence of the performances of the model to the initial selection of training sequences (for instance, whether objects with a low resolution where filtered out or not). This insight is supported by the reports of Xu et al. [87] about SiamFC++ ANN. However, the original SiamFC paper [4] provides only limited details on the pre-processing applied to the ImageNet VID 2015 dataset (e.g., filtering out of sequences, minimal sizes of target objects, or maximum time gap used when sampling frames for training). Therefore, a study on the impact of the training data on SiamFC ANN could also allow improvements of our spiking SiamFC.

Additionally, we show in our model that spike accumulation in the instance branch provides better performance than accumulation in the exemplar branch. This result shows that we could further reduce spike counts of the Siamese-based trackers since the spatial accumulation of spikes in the instance branch after backbone processing covers twice the spatial area of the spiking feature maps from the examplar branch. This result may lead to reduced energy consumption on suitable hardware.

Lastly, since spiking SiamFC++ [30] was not evaluated using the same standard protocol as SiamFC, it would have been difficult to compare our model to theirs. Furthermore, the experiments were not sufficiently detailed in the paper to reproduce the reported results. However, some of our results appear to contradict those reported by Xiang et al. [30], who did not observe an early non-learning phase in their training curves. In our experiments, this plateau of almost-zero improvement at the beginning of training seems to be a common side effect of using surrogate gradient backpropagation in deep SNNs. It is exacerbated in Siamese-based trackers

because both branches share weights, and thus receive highly correlated gradients. An explanation for this discrepancy is that the additional loss term introduced in SiamFC++ (i.e., the quality control loss on predicted bounding boxes) may provide stronger and more informative supervision signals from the beginning of training. This stronger signal could inject more informative gradients and mitigate the initial non-learning phase of SNNs trained with surrogate gradient backpropagation.

6.5 Conclusion

In this chapter, we introduced a novel spiking SiamFC model for VOT. It also provides a comprehensive ablation study analysing individual contributions of each part of the proposed architecture. This work tackles several challenges that have not been widely addressed in the literature. The model was effectively trained using the surrogate gradient method on the ImageNet VID 2015 dataset, and evaluated on OTB-2015 and GOT-10k. The results demonstrated promising tracking performance, although there are still open questions about the management of colour images against grayscale images and the lower performance on OTB-2015 as compared to GOT-10k.

One major contribution introduced in this work is the synchronised threshold adaptation mechanism. This mechanism enhances the regulation of spike trains across the network, which in turn improves the stability of training and leads to better tracking accuracy. Inspired by prior spiking feature extraction models and the work of Luo et al. [178], our model employs an asymmetric correlation operation with spike accumulation performed in the instance branch. This design shows better tracking performance than the accumulation in the exemplar branch while potentially reducing energy consumption by processing fewer spikes.

Additionally, a novel spiking pooling layer was developed. This layer adapts the average pooling operation to output spike trains without information loss, enabling

a fully spiking backbone that is better suited for deployment on neuromorphic hardware.

Together, these contributions advance the goal of building a potentially energy-efficient and robust SNN for VOT, enabling us to track atmospheric data, which we will study in further detail in the next chapter.

Chapter 7

Spiking SiamFC on Atmospheric Products

Aerosol impact the health of populations (see Chapter 1). These plumes are responsible for millions of deaths per year [12] and are mainly due to human activities. Therefore, we need an efficient and robust automatic plume-tracking system that enables faster air quality prediction and supports environmental disaster response (e.g., pollution peaks). To achieve this, satellites should be able to move their instruments directly to areas of interest to obtain the maximum amount of data from specific events. Furthermore, due to the limited energy resources onboard satellites, such tracking models should consume as little energy as possible. However, to the best of our knowledge, no deep learning-based tracker methods for VOT have been applied to atmospheric plumes, let alone low-energy SNN trackers.

In Chapter 6, we introduced a new spiking SiamFC tracker that can track generic objects based on an SNN architecture that has the potential to be embedded on satellites if implemented on suited neuromorphic hardware. We propose in this chapter a case study in which we evaluate the application of our SNN tracker to atmospheric plume data. This case study uses the aerosol VOT dataset derived from CAMS EAC4 reanalysis [38] data that we presented in Chapter 5, focusing on

the sea salt and dust species.

For practical reasons, we choose to track dust and sea salt plumes on AOD maps. This choice is motivated by pushing the tracker to its limits through scenarios that challenge it under different physical and observational conditions. Dust aerosols present well-defined plumes with sharp edges, which are used to validate VOT on the plume. Sea salt aerosols present diffuse plumes with smaller variations in AOD intensity. It allows testing the model on low-contrast, fast-evolving boundary plumes.

Compared to tracking generic objects, tracking plumes poses unique challenges. First, plumes have no definite shape and require trackers to verify the initial hypothesis proposed by Francis et al. [19]. This assumption states that a significantly unique optical signature can be extracted from plume textures. Next, the model should handle aerosols with completely different dynamics. Furthermore, these aerosol plumes can undergo renewal or drop, posing an unprecedented challenge to VOT by altering their presumed optical signature.

These challenges directly affect the spiking SiamFC tracker. Since plumes have no specific shape, the tracker will have to extract an optical signature from only the texture of the plume. Since spiking SiamFC relies only on appearance, the extracted signatures should be sufficiently unique not to confuse plumes. Furthermore, the continuous drop[1] and plume renewal[2] expose the spiking SiamFC tracker to objects that have changed their optical signature from one frame to another.

Therefore, considering the specificities of these atmospheric phenomena, we aim to assess the tracking capabilities of our spiking SiamFC model. First, we evaluate the spiking SiamFC tracker on aerosol plumes (i.e, dust and sea salt) from AOD data.

This chapter presents our tracker applied to the annotated CAMS EAC4 aerosol

[1]By *drop*, we refer to the progressive removal of aerosol from the atmosphere, which reduces the contrast of the plume.

[2]By *renewal*, we refer to the injection of new aerosol particles into an existing plume.

dataset introduced in Chapter 5 (Page 93). We aim to evaluate the ability of the model to track sea salt and dust aerosol plumes across diverse temporal and spatial conditions. We start by detailing the methodology (i.e., the specific dataset preparation, processing, model configuration, and training protocols). Then, we discuss results under multiple experimental conditions.

7.1 Dataset Properties

The CAMS EAC4 aerosol tracking dataset, detailed in Chapter 5 (Page 93), is used as the basis for all experiments in this section. We recall its major properties here. It provides annotated sequences for both sea salt and dust aerosol events for the VOT training and evaluation processes. This dataset uses the AOD at 550nm product. It has been annotated from 1st September 2020 to 31st December 2022 for dust, and from 1st September 2020 to 31st January 2021 for sea salt. The dataset is annotated on a 0.75° by 0.75° grid in latitude and longitude. Overall, the dataset contains 267 sequences with 182 AOD sequences of dust and 85 AOD sequences of sea salt. The total number of annotated plume bounding boxes is $28,761$. Provided bounding boxes can be used either on the total AOD or by-species AOD.

7.2 Preprocessing

Preprocessing for AOD is limited to essential steps. The input data maps already operate in a low dynamic range, which prevents the distortion of physical values and avoids noise amplification. As mentioned in Section 5.1.2 (Page 98), the only preprocessing step is the spatial cropping of regions of interest during data curation, to ensure an SOT setting (see Section 5.2, Page 101). Furthermore, no data augmentation using geometric transformations was applied to the input data. Rotation and flipping of AOD data break the physical meaning of the plumes. For aerosol products, these operations can create distributions and spatial features that

would not be physically possible under real atmospheric transport and emission conditions. This could introduce physically aberrant examples into the training sets.

All sequences in the dataset are randomly split into non-overlapping training, validation, and evaluation sets (with ratios of 60%, 20%, and 20%, respectively). By ensuring diverse data exposure, this strategy prevents overfitting to specific seasonal patterns and enables better generalisation of trackers through varied training scenarios.

7.3 Experimental Protocol

In this section, we propose a protocol to evaluate the generalisation abilities of the tracking model across realistic (total AOD) and simulated (by-species AOD) settings. Four main experimental settings are defined to cover all possible application settings:

- species-specific training on by-species AOD: this setting is composed of two trackers, each trained either on dust AOD sequences (setting code: E5) or on sea salt AOD sequences (E6), with by-species AOD maps as input. This setting serves as a baseline for our comparison since it represents the ideal tracking conditions (i.e. clear separation between species in inputs, and one tracker per species);

- mixed-species training on by-species AOD: one tracker is trained on a dataset containing both dust AOD sequences and sea salt AOD sequences, with by-species AOD maps as input (E4). This setting tests the ability of the tracker to track multiple species at once;

- species-specific training on total AOD: in this setting, one tracker is trained on dust AOD sequences (E2), and another one is trained on sea salt AOD

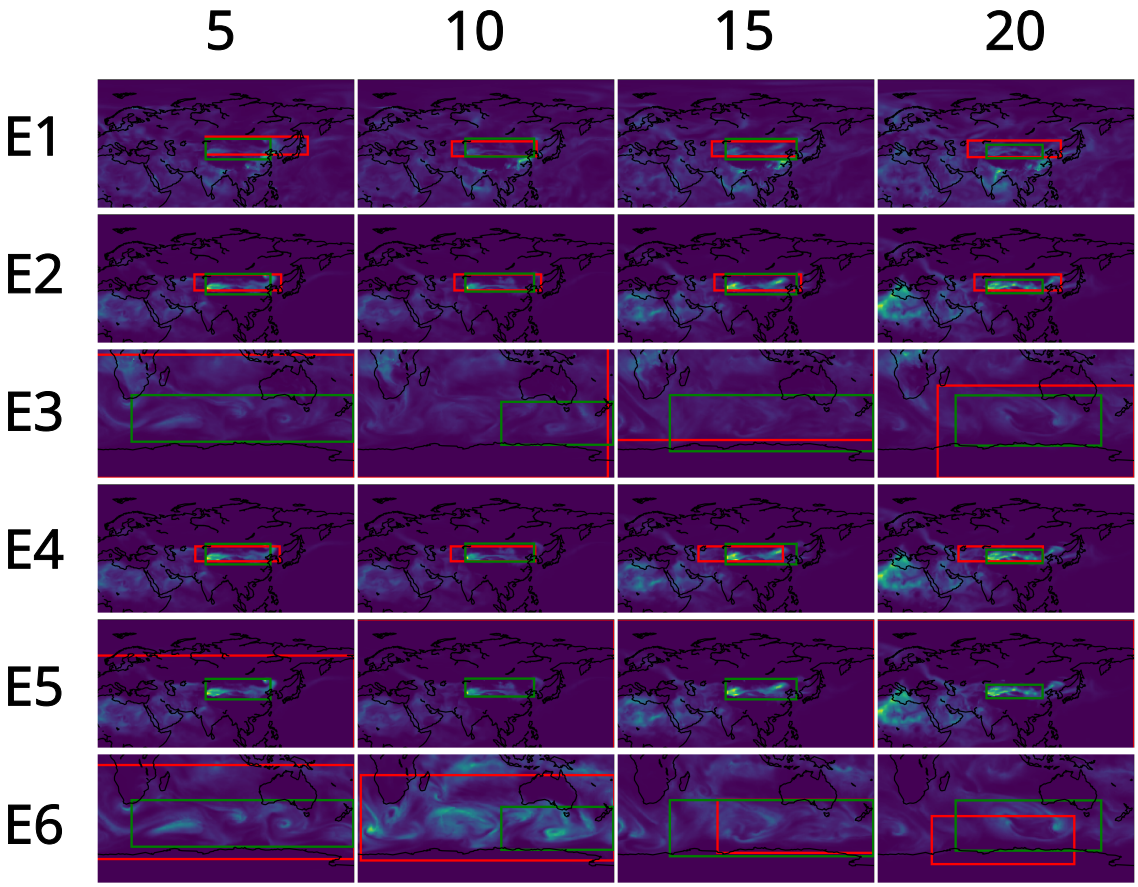


Figure 7.1: Illustration of the spiking SiamFC for each setting from Table 7.1. For each setting, we display the frames at time steps 5, 10, 15 and 20. Each time step is equivalent to 3 hours. Green rectangles are the ground truth bounding boxes, while red rectangles represent the predicted bounding boxes.

sequences (E3), with total AOD maps as input. This setting evaluates the ability of the tracker to track aerosols in real conditions (as total AOD can be obtained directly from measurements [153]);

- mixed training on total AOD: one tracker is trained on a dataset containing both dust AOD sequences and sea salt AOD sequences, with total AOD maps as input (E1). This setting shows the ability of the tracker to process data that can be obtained directly from sensors and that includes multiple species at once.

Table 7.1 sums up the trackers trained in different settings. We illustrate each setting in Figure 7.1.

Setting Code	Aerosol(s) Tracked	Input product
E1	Dust and sea salt	Total AOD
E2	Dust	Total AOD
E3	Sea salt	Total AOD
E4	Dust and sea salt	By-species
E5	Dust	By-species
E6	Sea salt	By-species

Table 7.1: Experimental settings for the tracking of dust and sea salt aerosols. Each setting corresponds to the type of aerosols we want to track and the product on which we track them.

Following the protocol of Chapter 6 (Page 133), on each epoch during training, sequences are uniformly sampled, and within each sequence, a random pair of total AOD (or by-species AOD, depending on the experiment) maps is sampled as the exemplar and instance inputs for our spiking SiamFC tracker. During evaluation, for each tracking experiment, we started with the ground-truth bounding box from the first frame and performed tracking frame-by-frame, without resetting the target object in case of failure, following the OPE protocol for tracker evaluation.

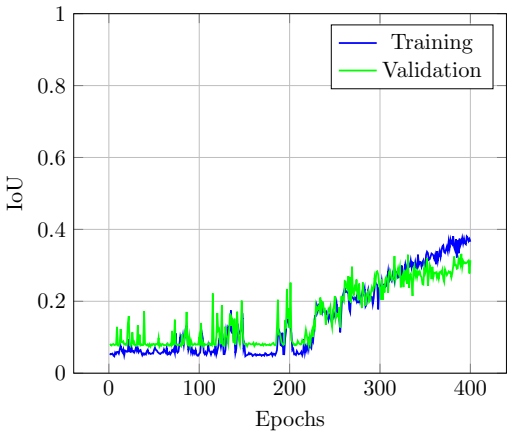
7.4 Implementation Details

Unless stated otherwise in this chapter, all spiking SiamFC hyperparameters are identical to those used in Chapter 6 (Section 6.3.4, Page 134), including the backbone architecture, number of time steps, and spike-rate regularisation settings. We train separate trackers for each experimental setting in Table 7.1, using the same Adam optimiser, initial and final learning rates, batch size and number of epochs as in Chapter 6. Unlike the experiments in Chapter 6, we train models over 400 epochs.

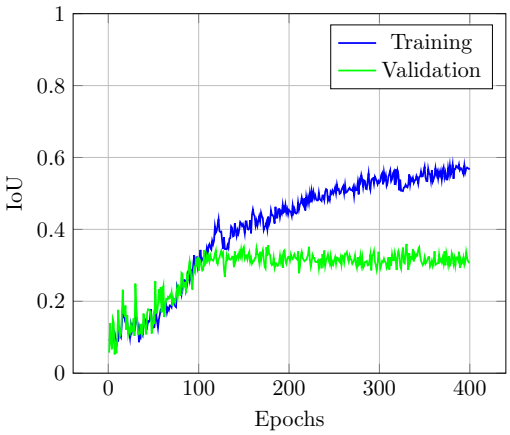
7.5 Results

Figures 7.2a, 7.2b, and 7.2c show the evolution of the training and validation IoU during training for, respectively, settings E3, E2, and E1 (see Table 7.1, Page 152), using total AOD maps as input to the tracker. The learning curves in Figures 7.2a, 7.2b, and 7.2c show that the validation IoU stays low most of the time. For sea salt tracking (see Figure 7.2a), the validation IoU goes from 0.27 to 0.32. At the same time, the training IoU shows the same behaviour. This indicates some capacity of the tracker to extract features and learn from total AOD data. For dust aerosols (see Figure 7.2b), the validation IoU stays close to 0.3, with the training IoU continuing to increase. This behaviour suggests that the model has reached its generalisation capacity. Mixing dust AOD maps and sea salt AOD maps for training (see Figure 7.2c) achieves a validation IoU of 0.3, with a lower difference between training and validation IoU. This behaviour suggests a slightly more stable generalisation across species when training on mixed dust and sea salt plumes than when training on dust alone. This behaviour can also be due to the imbalance between classes in terms of sequences and frames, which biases the results towards the tracking of dust AOD, instead of improving sea salt AOD tracking performances.

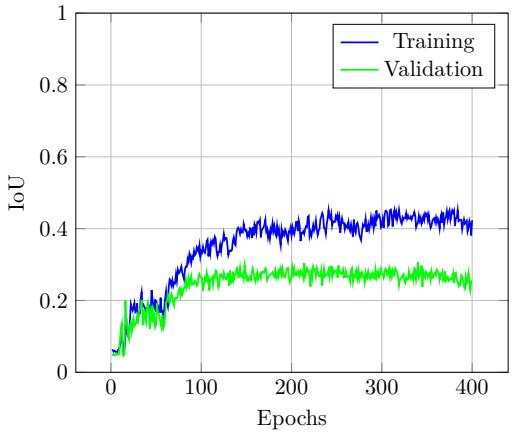
Figures 7.3a, 7.3b, and 7.3c show the evolution of the training and validation IoU during training for, respectively, settings E6, E5, and E4 (see Table 7.1, Page 152), using by-species AOD as input to the tracker. By using species-specific AOD data for tracking, we have observed better generalisation capacity for sea salt (with almost 0.4 for validation IoU against 0.3), while dust (Figure 7.3b) and combined case AOD give almost the same validation IoU as tracking on total AOD. This suggests that the model can extract from total AOD the same information for dust than from by-species AOD, but not for sea salt AOD. The large gap between the IoU of training and validation suggests that the model has also reached its generalisation capacity. When the model is trained on data from both dust and



(a) Sea salt only (total AOD)



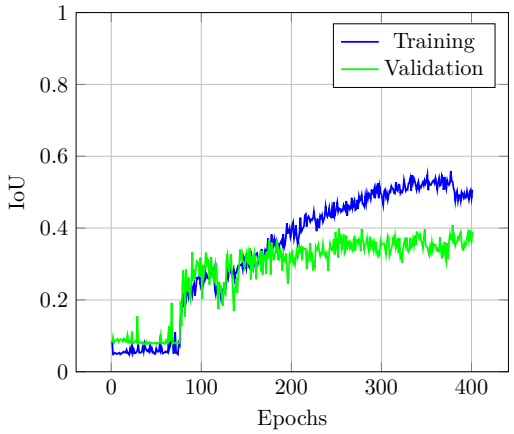
(b) Dust only (total AOD)



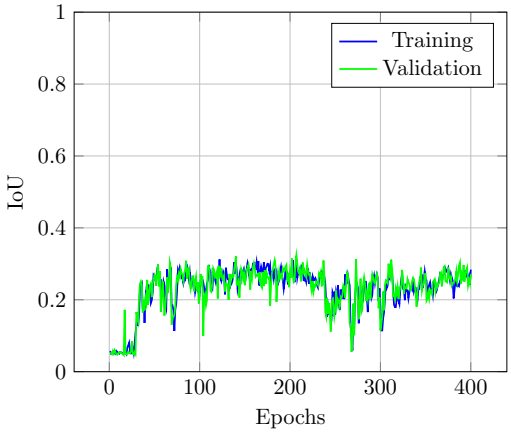
(c) Combined dust and sea salt (total AOD)

Figure 7.2: IoU learning curves for aerosol tracking with total AOD input. Top panels compare the tracking of sea salt only (a) and dust only (b), while the bottom panel presents combined the tracking of dust and sea salt (c).

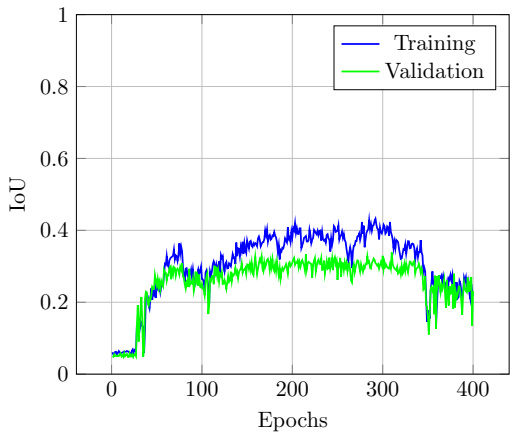
sea salt (Figure 7.3c), the results improve further, achieving a higher validation IoU than when trained on dust plumes only; we will discuss this result later. This approach also results in more stable differences between training and validation performance.



(a) Sea salt only (by-species)



(b) Dust only (by-species)



(c) Combined dust and sea salt (by-species)

Figure 7.3: IoU learning curves for aerosol tracking with by-species AOD. Top panels compare the tracking of sea salt only (a) and dust only (b), while the bottom panel presents the tracking of combined dust and sea salt (c). The results show the improved convergence with by-species for sea salt, but not for dust. The combined species case shows similar results for validation as total AOD.

The success plots in Figures 7.4 and 7.5 offer a detailed view of the performance of the tracker for total and by-species AOD. AUC scores are shown in the legends of Figures 7.4 and 7.5 for each setting. For total AOD, the scenarios involving combined dust and sea salt, as well as dust only, yielded the highest AUC scores

(i.e., 0.128 and 0.127, respectively). Furthermore, by computing individually the AUC for the sequences of each species in setting E1, we see that the tracking of dust is improved with an AUC of 0.14. However, sea salt AUC decreases slightly to 0.10. This increase in AUC for dust in E1 indicates that tracking of aerosols can profit from adding other species to the dataset. For sea salt tracking from total AOD (i.e., setting E3), the tracker shows an AUC score of 0.11, while dust (i.e., setting E2) gets an AUC of 0.127. These results show that dust is better tracked than sea salt in general, which could be explained by the fact that dust plumes have more compact and clearer boundaries than sea salt ones, or by physical changes affecting the appearance of plumes of sea salt, but not of dust plumes. For the combined case on by-species AOD (i.e., setting E4), we get an AUC score of 0.15, showing an improvement compared to case E1. Furthermore, when computing the AUC scores for each species in setting E4, we also notice an improvement, with an AUC of 0.18 for dust, and an AUC of 0.09 for sea salt. This result confirms our previous observations from setting E1 for dust. In E5 and E6, we get an AUC of 0.116 and 0.11 for dust and sea salt, respectively. These AUCs show that dust and sea salt AOD tracking are slightly worse on by-species AOD than on total AOD. This difference needs to be confirmed on additional runs.

We observe from Figures 7.2 and 7.3 that, in all settings, the SNN learns, to some extent, from total and by-species AOD, and does not undergo overfitting. However, results in Figures 7.4 and 7.5 show only modest results. To further study this poor performance of our tracker, we decided to analyse the correlation between the variation in the target AOD plume and the predicted IoU across sequences. Figure 7.6 shows the cross-correlation of the mean AOD of the groundtruth bounding boxes, by the IoU of the tracker for each experimental setting. Results are averaged across sequences to provide general trends for each setting. By computing this cross-correlation operator measures how similar one signal is to a time-shifted (i.e., under

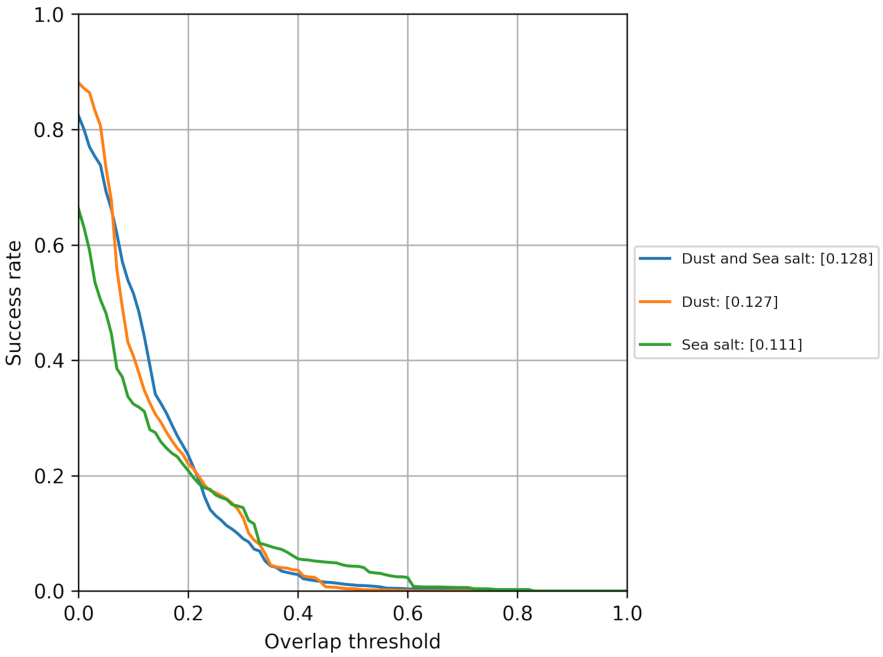


Figure 7.4: Success plot following the OPE evaluation protocol for tracking plumes using CAMS EAC4 total AOD. Each line represents a different aerosol class: dust and sea salt combined (blue), dust (orange), and sea salt (green). The success rate at 50% for each species is given in the legend. The dust and sea salt class and the dust class show similar tracking performance, while sea salt plumes are notably more difficult to track, reflected by the lower success rates across thresholds. The results highlight the challenge of accurately following diffuse and low-contrast aerosol plumes from reanalysis data.

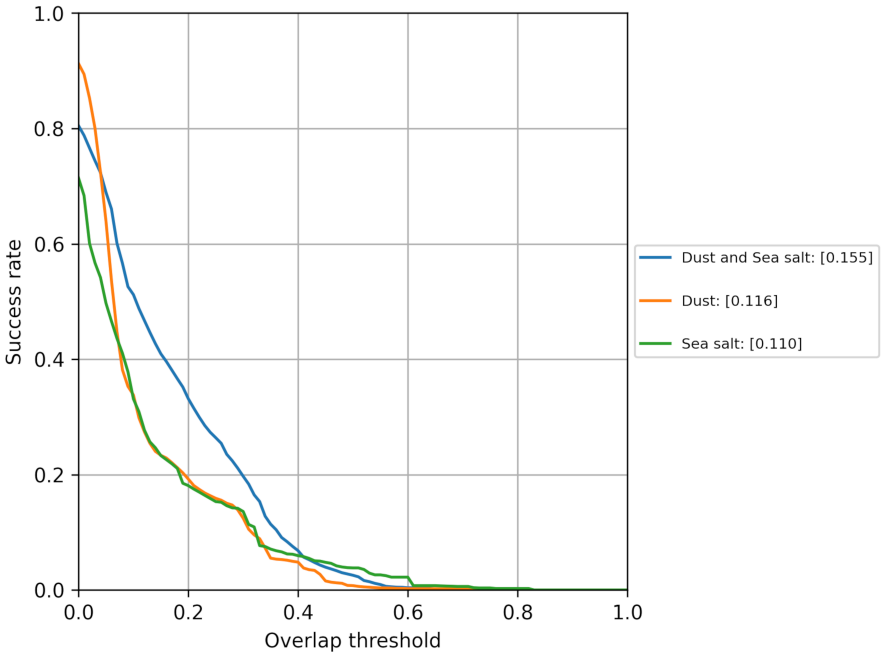


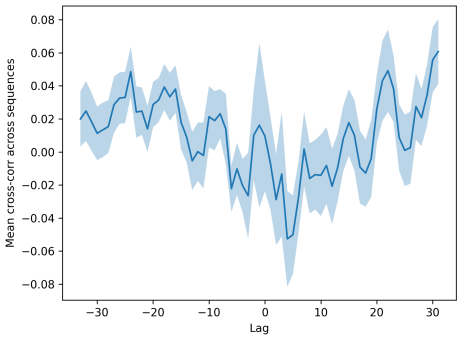
Figure 7.5: Success plot following the OPE evaluation protocol for plume tracking using CAMS EAC4 by-species AOD. The blue curve shows combined dust and sea salt performance, the orange one shows dust only, and the green one shows sea salt only. The success rate at 50% indicates that tracking is most successful for combined plumes, with lower performance for individual dust and especially sea salt plumes. This highlights the increased difficulty of localising and following diffuse or low-contrast events.

lag[3]) version of the other. Using this cross-correlation function, we aim to determine whether drops or renewals of AOD within the target plumes lead to decreases in IoU. If the minimum of the function is in the positive lag, it shows that renewals of the AOD plumes leads to drops of predicted IoU, while if the maximum is in the positive lag, drops of AOD plumes leads to a drop of IoU. In Figure 7.6a corresponding to case E1, we observe the minimum value on a positive lag, which suggest a correlation between increases in AOD (i.e., renewal of the plumes) and decreases of IoUs. This result can explain the low performance discussed earlier. We observe the same behaviour in Figure 7.6b representing case E4. On Figure 7.6a and 7.6b, we also observe a significant increase just after this minimum, suggesting that the tracker tries to relocate the renewed plume. Figure 7.6c and 7.6d, corresponding to settings E2 and E5, show a minimum in the negative lag, which could be either due to low performance of the tracker or a change on the appearance of the plume due to the drop of aerosols. The cross-correlation of sea salt sequences (settings E3 and E6) shows more mixed results with a minimum closer to a 0 lag, indicating a consistent anti-correlation between the variations of AOD and IoU, but we cannot conclude on the relationship between the two, indicating that AOD is not responsible for the tracker failure in this case.

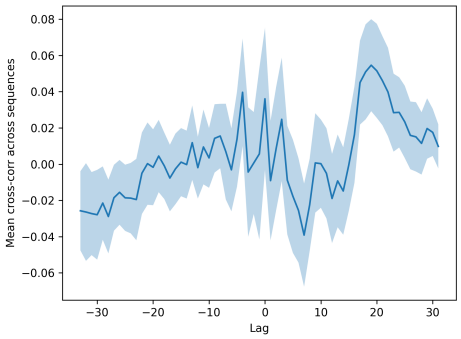
7.6 Discussion

Although achieving high tracking accuracy remains challenging due to the complex nature of aerosol plumes, which include noise and overlapping signals [179], using a dataset that has dust and sea salt sequences improves the tracking of dust, suggesting that adding more species could improve the tracking further. Despite modest results, the statistics we conducted could indicate that renewals or drops of plumes may be one cause of the failure of the tracker. These results will need

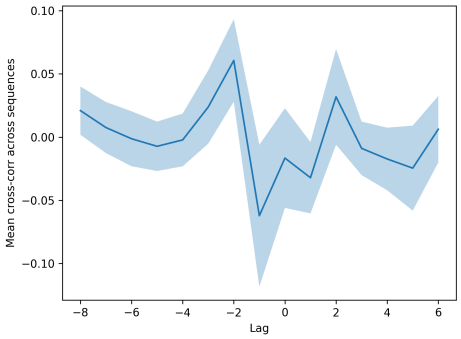
[3]The *lag* corresponds to the temporal shift between the two signals when computing their cross-correlation (i.e., how far one signal is shifted in time relative to the other).



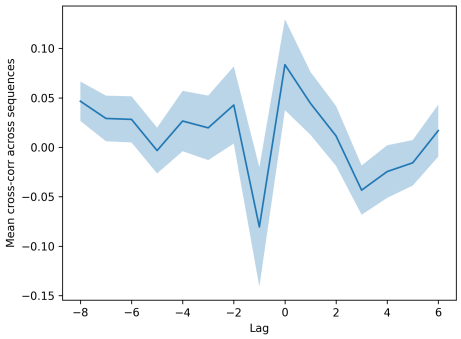
(a) Dust and Sea salt, total AOD



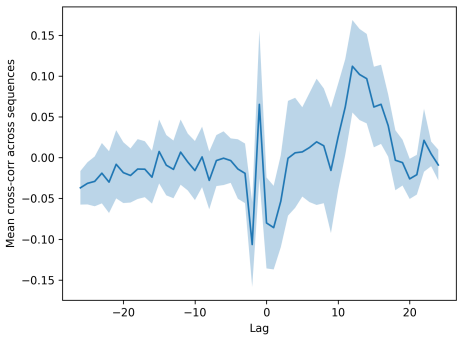
(b) Dust and Sea salt, species AOD



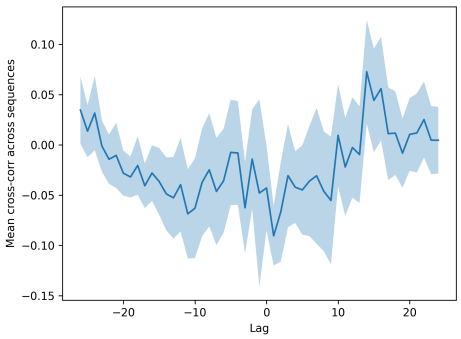
(c) Dust, total AOD



(d) Dust, species AOD



(e) Sea salt, total AOD



(f) Sea salt, species AOD

Figure 7.6: Cross-correlation of the mean AOD over ground truth regions by the IoU through time. Figures (a) and (b) show this cross-correlation for the E1 and E4 settings, respectively (see Table 7.1). (c) and (d) show the ones for settings E2 and E5, respectively. (e) and (f) show the cross-correlation from settings E3 and E6, respectively. Each cross-correlation has been computed over at least 25 samples to ensure statistical robustness.

future investigation to be confirmed. We notice in this chapter that dust plumes are generally easier to track due to their more distinctive spatial structures [6, 5]. In contrast, sea salt plumes remain more challenging for trackers, potentially due to their diffuse, low-contrast nature, as confirmed by physical observation. To the best of our knowledge, there is no prior work on tracking aerosol plumes with SNNs, making our spiking SiamFC a baseline for VOT-based tracking.

Although the quantitative results remain limited, this work is still valuable. To the best of our knowledge, it is the first attempt to track aerosol plumes using a low-energy deep learning SNN model, paving the way for future low-energy approaches to aerosol plume tracking. In addition, our experiments highlight important physical factors that must be considered when tracking aerosol plumes, such as changes in the visual appearance of the target caused by the decay or renewal of aerosols, which in turn modify the optical signature of targeted plumes. It demonstrates that standard approaches to tracking are not enough to tackle aerosol plume tracking, and calls for additional efforts in this field.

Chapter 8

Conclusion and Future Work

8.1 Summary of Contributions

Understanding and monitoring atmospheric plumes is crucial for studying the impact of aerosol and gas plumes on the climate. However, doing so requires models to be embedded on instruments (e.g., LIDAR embedded on a satellite) in order to get faster response in case of pollution peaks or low latency to follow events of interest. Moreover, VOT for these plumes poses significant challenges due to their complex spatio-temporal variability. At the intersection of atmospheric science, VOT, and energy-efficient deep learning, this thesis proposed methods to transfer VOT methods to SNNs and to the tracking atmospheric plumes. To the best of our knowledge, this thesis is the first research to use SNN on atmospheric plume tracking.

The following sections detail the main contributions of this thesis. First, a primary focus has been made on the creation of an atmospheric plume tracking dataset from the CAMS EAC4 dataset. Then, we have adapted to directly-trained SNNs to the SiamFC ANN from Bertinetto et al [4], an efficient tracking-by-detection model. Lastly, a comprehensive evaluation of our spiking SiamFC on aerosol plume data has been realised to evaluate the efficiency and the limits of the proposed method.

8.1.1 Construction of a Dataset for the VOT of Atmospheric Plumes

We introduced a new VOT dataset designed for atmospheric aerosol plume tracking. This dataset uses ECMWF CAMS EAC4 reanalysis products. The dataset focuses on sea salt and dust aerosol plumes. These two aerosol species were chosen for their visual properties in the atmosphere. Dust often presents well-defined atmospheric plumes, while sea salt shows a large and quite diluted aerosol plume.

We also provide a detailed annotation protocol and bounding boxes, along with statistics on the bounding boxes of the dataset. The bounding boxes capture the spatial extents of plumes across frames every 3 hours. The annotated data was acquired from 1st September 2020 to 31st December 2022. This time span allows for the integration of diverse atmospheric plume shapes and sizes, enabling training or at least benchmarking of visual trackers on atmospheric plumes.

We have also validated the physical relevance of bounding boxes in terms of areas and AOD intensities [5, 6, 7]. The dataset emphasises realistic atmospheric conditions by incorporating spatio-temporally diverse plume events extracted from a high-quality reanalysis model from ECMWF. This enables the study of VOT under challenging scenarios representative of typical atmospheric variability. It settles a new standard for aerosol and plume tracking research.

8.1.2 Spiking SiamFC Tracker

The core contribution of this thesis is the development of the a directly trained SNN adaptation of the SiamFC from Bertinetto et al. [4]. This work introduces several novelties in spiking Siamese trackers to enable effective learning with such architectures.

First, this model provides a synchronised threshold adaptation method to stabilise learning with surrogate gradient-based methods. We also proposed a compar-

ison between two spike encoding strategies: direct and temporal coding. Furthermore, our model includes is the first implementation of spike rate pooling. We also show that asymmetrical head accumulation on the instance branch gives slightly better results than accumulation on the exemplar branch, while decreasing the number of spikes to be processed. Lastly, an ablation study has been conducted to validate the benefits of each contribution in achieving stable and accurate tracking, demonstrating clear performance improvements.

Overall, this work advances SNNs in VOT by providing an energy-efficient alternative to traditional ANN architectures. By doing so, it provides an SNN tracker applicable to remote sensing and atmospheric plume applications.

8.1.3 Evaluation of Spiking SiamFC on Atmospheric Plumes

Finally, our spiking SiamFC model was evaluated on the annotated aerosol plume tracking dataset that we developed. In these experiments, we have shown that our SNN tracker is able to learn features from aerosol plumes, even though tracking performance remains low. One explanation for these results is that, although we developed a large dataset for aerosol plume tracking, it remains insufficient in terms of the number of sequences to train an effective tracker. However, we have observed an improvement in tracking for dust plumes when combined with sea salt plume sequences; this observation needs to be confirmed by future research on plume tracking. Evaluation on this dataset has highlighted major challenges unique to plume tracking, such as drop or renewal, causing a fast change of appearance of these plumes.

This work represents the first application VOT to atmospheric plumes, establishing foundational benchmarks and methodology for the intersection of SNNs and atmospheric science.

8.2 Dissemination

The work in this thesis has not been published yet because of the time required to develop an effective SNN Siamese tracker from scratch. As the tracker needed to be developed as a whole, no incremental or partial contribution could be published during research. Furthermore, all contributions in this thesis are closely related: the evaluation of the tracker on atmospheric data could only be done after the tracker was completed.

The contributions of this thesis will be disseminated into two main communities: neuromorphic computing, and atmospheric science and remote sensing.

The contribution proposed in Chapter 6, i.e. the SNN Siamese tracker with the synchronised threshold adaptation, comparison of direct and temporal coding, and asymmetric head accumulation, will be submitted to a neuromorphic computing journal. Making the implementation available as an open-source package will support reproducibility and encourage its use as a baseline for future spiking Siamese trackers.

The dataset from Chapter 5, which includes the annotation protocol and quality-control analysis, and the baseline results of our spiking SiamFC (Chapter 7) will be submitted to a journal on remote sensing and atmospheric science. Furthermore, the public diffusion of the data and of the code will facilitate their reuse by other researchers for training and benchmarking.

8.3 Future Work

This section outlines future advancements for VOT on atmospheric plumes using SNN-based trackers. First, we discuss how richer annotations and the inclusion of additional aerosol species could improve VOT performance for aerosol plumes. We then consider integrating motion models and Physics-Informed Neural Networks

(PINNs) to better exploit prior physical knowledge of aerosol transport. Finally, we explore how extending the spiking SiamFC architecture to MOT could address plume splitting and merging events that are common in realistic atmospheric conditions.

8.3.1 Annotation of other AOD Species from CAMS EAC4

The priority should be given to the annotations of a larger number of atmospheric plumes. One promising route to do this is to expand annotations to include more aerosol species (such as sulfate, black carbon, organic carbon, and nitrate). Such an annotation would make the datasets more representative of all products coming from ECMWF, which will allow to evaluate trackers across multiple classes. This would allow researchers to potentially improve VOT by making the model observe species-specific plume behaviour. In the end, this work could help to study interactions between different aerosols and help study the impact of aerosols on air quality and climate change. The main challenges will be to annotate enough sequences for effective training on each species. Another challenge is to accounting for the physical properties of each aerosol during training and evaluation (e.g., velocity and atmospheric lifespan). This could allow us to confirm by more solid evidence the relationship between the changes in AOD values and the failure of the trackers.

8.3.1.1 Training and Evaluation on Data from Real Instruments

Transitioning from model-generated reanalysis to real instrument data (such as data from MODIS coupled with AERONET) would make the tracker more relevant for realistic scenarios and validate the spiking tracking approach under true observational uncertainty, sensor noise, and atmospheric retrieval artefacts. Lastly, data from instruments often present missing values, which require careful selection and annotations.

8.3.2 Motion Model and Physics Informed Neural Networks

Incorporating physically-informed motion models, such as atmospheric transport trajectories, could help the tracker anticipate plume evolution and avoid tracker failure when velocities of plumes are too high or when observations are missing. However, models such as PINN [180] could be difficult to integrate into a Siamese-based tracker because discrete nature of SNNs enter in conflict with the need for continuity from physical constraints. This could allow for a lower need for data from such trackers. One drawback is that it usually comes at the expense of a larger computational cost. Key challenges of such work include the integration of physical models into SNN trackers, handling SNNs training instability, choosing meaningful physical parameters that would guide the tracker correctly, and finding a spiking representation for those. Prior work already used PINN with SNNs [181], which could serve as a basis for this future tracker.

8.3.3 MOT with Spiking SiamFC

All the contributions presented here are based on SOT. Thus, another axis of development for the spiking SiamFC could be to include multi-target tracking capabilities to explore the problem of tracking splitting and merging AOD plumes, which frequently occur during aerosol transport. Contrary to SOT, MOT would be able to adapt to the tracking of a single plume splitting into multiple plumes by creating groups of target objects and, as such, following the original single plume at multiple positions. MOT would also be useful for merging scenarios by detecting the plume of origins from an already merged plume in a reanalysis scenario. Furthermore, MOT would be able to integrate awareness of the target objects in the map, making it more robust to background clutter. This property, in the case of aerosols or gas plumes, is particularly useful when two plumes merge together. This could also be useful for splitting plumes since the detector integrated in MOT could detect

these two plumes as new objects. This allows for more accurate tracking for the analysis of the evolution of plumes.

Appendices

Appendix A

Atmospheric Composition Reanalysis 4 Tracking Annotation

A.1 Data Format and Structure

Tracking annotations of CAMS EAC4 aerosol plumes have a similar file organisation as classic VOT datasets, such as the ImageNet dataset [64], to facilitate efficient access and usage of this dataset for the VOT research community. The top-level directory, named `export_data/annotations`, contains the sequences stored in subdirectories. Each of these subdirectories is formatted as `cams-global-reanalysis-eac4_<YYYY-MM-DD>_<YYYY-MM-DD>`, with `<YYYY-MM-DD>` being the placeholder for the start and end dates of sequences.

Each subdirectory contains the following elements:

- a file containing the list of paths (e.g., `duaod550_list_path.txt` and `ssaod550_list_path.txt`) per sequence for dust and sea salt aerosol data;
- a directory (e.g., `duaod550`, `ssaod550`) containing the annotations inside JSON files. These JSON files contain bounding boxes of AOD plumes for each frame.

The pre-cropped data directories `duaod550_crop` and `ssaod550_crop` contain sequences cropped around the annotation. This is used as preprocessing for trackers searching target objects in sub-regions instead of the whole image for the training and validation phase. These cropped data directories are further divided into sub-folders (e.g., `dust1`, `sea_salt1`, `dust2`, `sea_salt2`) that correspond to individual aerosol plume sequences.

The original raw CAMS EAC4 reanalysis data can be downloaded on the ECMWF atmosphere data store at the following address: https://ads.atmosphere.copernicus.eu/datasets/cams-global-reanalysis-eac4.

Bibliography

[1] ECMWF, "IFS Documentation CY41R2 - Part IV: Physical Processes," in *IFS Documentation CY41R2*. ECMWF, 2016, no. 4.

[2] Weiwei Xing, W. Liu, Jun Wang, S. Zhang, Lihui Wang, Y. Yang, and B. Song, "Visual Object Tracking from Correlation Filter to Deep Learning," 2021.

[3] H. Nam and B. Han, "Learning Multi-Domain Convolutional Neural Networks for Visual Tracking," 2016.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Springer International Publishing, 2016, pp. 850–865.

[5] T. Silva, M. Fragoso, R. Almendra, J. Vasconcelos, A. Lopes, and A. Faleh, "North African dust intrusions and increased risk of respiratory diseases in Southern Portugal," *International Journal of Biometeorology*, vol. 65, no. 10, pp. 1767–1780, 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8437926/

[6] H. Bian, K. Froyd, D. M. Murphy, J. Dibb, A. Darmenov, M. Chin, P. R. Colarco, A. da Silva, T. L. Kucsera, G. Schill, H. Yu, P. Bui, M. Dollner, B. Weinzierl, and A. Smirnov, "Observationally constrained analysis of sea salt aerosol in the marine atmosphere," *Atmospheric Chemistry and Physics*, vol. 19, no. 16, pp. 10 773–10 785, Aug. 2019, publisher: Copernicus GmbH. [Online]. Available: https://acp.copernicus.org/articles/19/10773/2019/

[7] R. K. Kunchala, R. Attada, R. K. Karumuri, V. Seelanki, B. B. Singh, K. Ashok, and I. Hoteit, "Climatology, trends, and future projections of aerosol optical depth over the Middle East and North Africa region in CMIP6 models," *Frontiers in Climate*, vol. 6, Jul. 2024, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/journals/climate/articles/10.3389/fclim.2024.1384202/full

[8] A. M. Aldhaif, D. H. Lopez, H. Dadashazar, D. Painemal, A. J. Peters, and A. Sorooshian, "An Aerosol Climatology and Implications for Clouds at a Remote Marine Site: Case Study Over Bermuda," *Journal of geophysical research. Atmospheres : JGR*, vol. 126, no. 9, p. e2020JD034038, May 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8216143/

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[10] E. Lemaire, L. Cordone, A. Castagnetti, P.-E. Novac, J. Courtois, and B. Miramond, "An analytical estimation of spiking neural networks energy efficiency," in *Neural Information Processing*, M. Tanveer, S. Agarwal, S. Ozawa, A. Ekbal, and A. Jatowt, Eds. Cham: Springer International Publishing, 2023, pp. 574–587.

[11] M. T. Lund, T. K. Berntsen, B. H. Samset, and et al., "Contrasting trends in anthropogenic aerosol emissions and their climate impacts across regions," *Atmospheric Chemistry and Physics*, vol. 23, pp. 6647–6667, 2023. [Online]. Available: https://acp.copernicus.org/articles/23/6647/2023/

[12] J. Lelieveld, A. Pozzer, U. Pöschl, M. Fnais, A. Haines, and T. Münzel, "Effects of fossil fuel and total anthropogenic emission removal on public health and climate," *Proceedings of the National Academy of Sciences*, vol. 116, no. 15, pp. 7192–7197, 2019. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1819989116

[13] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: a comprehensive review and list of resources,"

IEEE Geoscience and Remote Sensing Magazine, vol. 5, no. 4, pp. 8–36, 2017. [Online]. Available: https://arxiv.org/abs/1710.03959

[14] M. Boichu and T. Mathurin. (2022) Volcplume, an interactive web portal for the multiscale analysis of volcanic plume physico-chemical properties. AERIS. Interactive Web based Resource. [Online]. Available: https://volcplume.aeris-data.fr

[15] M. J. Pavolonis, J. Brunner, C. C. Schmidt, G. S. Harkema, and B. Callicutt, "A blended, multi-sensor technique for volcanic ash cloud detection and height estimation using polar-orbiting and geostationary infrared sensors," *Journal of Geophysical Research: Atmospheres*, vol. 120, no. 18, pp. 9736–9758, 2015, vOLCAT: Volcanic Cloud Analysis Toolkit, NOAA. [Online]. Available: https://doi.org/10.1002/2015JD023179

[16] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual Tracking: An Experimental Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[17] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2544–2550.

[18] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[19] A. Francis, P. Sidiropoulos, and J.-P. Muller, "CloudFCN: Accurate and Robust Cloud Detection for Satellite Imagery with Deep Learning," *Remote Sensing*, vol. 11, no. 19, p. 2312, 2019.

[20] L. Bouza Heguerte, A. Bugeau, and L. Lannelongue, "How to estimate carbon footprint when training deep learning models? a guide and review," *Environmental Research Communications*, 2023. [Online]. Available: https://arxiv.org/abs/2306.08323

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

[22] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[23] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks," *Pattern Recognition*, vol. 94, pp. 87–95, 2019.

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

[25] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, N. Imam, Y. Guo, Y. Nakamura, I. Morley, M. Schatz, B. Jackson, W. Bernstein, D. Mountain, M. Sima, S.-C. Fok, S. Choudhary, B. Taba, A. Amir, M. D. Flickner, P. Patil, M. Rizk, B. Blau, S. K. Esser, R. Appuswamy, D. Barch, C. Di Nolfo, R. Preissl, K. Wang, P. Fox, J. Nagle, A. Feldman, P. Raghavan, S. Srivastava, S. Chandra, J. Yang, A. Ndirango, D. Lou, B. Rajendran, Y.-H. Weng, S. Kim, B. Choi, G. Venkataramanan, C.-L. Ye, J. Wu, P. Narayanan, M. V. Debole, S. Esser, R. Appuswamy, D. Barch, C. Di Nolfo, R. Preissl, K. Wang, P. Fox, J. Nagle, A. Feldman, P. Raghavan, S. Srivastava, S. Chandra, J. Yang, A. Ndirango, D. Lou, B. Rajendran, Y.-H. Weng, S. Kim, B. Choi, G. Venkataramanan, C.-L. Ye, J. Wu, P. Narayanan, M. V. Debole, A. Amir, M. D. Flickner, P. Patil, M. Rizk, B. Blau, S. K. Esser, R. Appuswamy, D. Barch, C. Di Nolfo, R. Preissl, K. Wang, P. Fox, J. Nagle, A. Feldman, P. Raghavan, S. Srivastava, S. Chandra, J. Yang, A. Ndirango, D. Lou, B. Rajendran, Y.-H. Weng, S. Kim, B. Choi, G. Venkataramanan, C.-L. Ye, J. Wu, P. Narayanan, M. V. Debole, A. Amir, and M. D. Flickner,

"A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[26] C. Ostrau, C. Klarhorst, M. Thies, and U. Rückert, "Benchmarking neuromorphic hardware and its energy expenditure," *Frontiers in Neuroscience*, vol. 16, p. 873935, 2022. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2022.873935/full

[27] M. Davies *et al.*, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, 2021. [Online]. Available: https://dynamicfieldtheory.org/upload/file/1631291311_c647b66b9e48f0a9baff/DavisEtAl2021.pdf

[28] Y. Xiao *et al.*, "Recent progress in neuromorphic computing from fundamental devices to large-scale systems," *Advanced Devices and Instrumentation*, vol. 4, p. 0044, 2024.

[29] Y. Luo, M. Xu, C. Yuan, X. Cao, L. Zhang, Y. Xu, T. Wang, and Q. Feng, "SiamSNN: Siamese Spiking Neural Networks for Energy-Efficient Object Tracking," 2021.

[30] S. Xiang, T. Zhang, S. Jiang, Y. Han, Y. Zhang, X. Guo, L. Yu, Y. Shi, and Y. Hao, "Spiking SiamFC++: deep spiking neural network for object tracking," *Nonlinear Dynamics*, vol. 112, no. 10, pp. 8417–8429, 2024.

[31] S. Manavi *et al.*, "Atmospheric aerosol spatial variability: Impacts on air quality and climate change," *One Earth*, vol. 7, no. 10, p. 101237, 2025.

[32] A. F. Stein, R. R. Draxler, G. D. Rolph, B. J. B. Stunder, M. D. Cohen, and F. Ngan, "NOAA's HYSPLIT Atmospheric Transport and Dispersion Modeling System," *Bulletin of the American Meteorological Society*, vol. 96, no. 12, pp. 2059–2077, Dec. 2015, publisher: American Meteorological Society Section: Bulletin of the American Meteorological Society. [Online]. Available: https://journals.ametsoc.org/view/journals/bams/96/12/bams-d-14-00110.1.xml

[33] W. Xing, W. Liu, J. Wang, S. Zhang, L. Wang, Y. Yang, and B. Song, "Introduction," in *Visual Object Tracking from Correlation Filter to Deep Learning*, W. Xing, W. Liu, J. Wang, S. Zhang, L. Wang, Y. Yang, and B. Song, Eds. Springer, 2021, pp. 1–14.

[34] D. Weidmann, K. Antonini, D. M. Pino, B. K. Brodersen, G. Patel, M. I. Hegglin, C. Sioris, W. Bell, K. Miyazaki, L. K. Alminde, A. Gabriele, M. Pastena, and A. Hoffmann, "Cubesats for monitoring atmospheric processes (CubeMAP): a constellation mission to study the middle atmosphere," in *Sensors, Systems, and Next-Generation Satellites XXIV*, vol. 11530. SPIE, Sep. 2020, pp. 141–159. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11530/115300U/Cubesats-for-monitoring-atmospheric-processes-CubeMAP--a-constellation-mission/10.1117/12.2573727.full

[35] D. Przewlocka-Rus and T. Kryjak, *Towards Real-Time and Energy Efficient Siamese Tracking – A Hardware-Software Approach.* Springer International Publishing, 2022, p. 162–173. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-12748-9_13

[36] L. Vaquero, M. Mucientes, and V. M. Brea, "SiamMT: Real-Time Arbitrary Multi-Object Tracking," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 707–714.

[37] X. Dong and J. Shen, "Triplet Loss in Siamese Network for Object Tracking," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, vol. 11217, pp. 472–488, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-01261-8_28

[38] "Cams global reanalysis (eac4)," 2019. [Online]. Available: https://ads.atmosphere.copernicus.eu/cdsapp#!/dataset/cams-global-reanalysis-eac4?tab=overview

[39] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks," 2019.

[40] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training Spiking Neural Networks Using Lessons From Deep Learning," *arXiv:2109.12894 [cs]*, 2022.

[41] Y. Kim, H. Park, A. Moitra, A. Bhattacharjee, Y. Venkatesha, and P. Panda, "Rate Coding or Direct Coding: Which One is Better for Accurate, Robust, and Energy-efficient Spiking Neural Networks?" 2022.

[42] Y. Wu, J. Lim, and M.-H. Yang, "Object Tracking Benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[43] L. Huang, X. Zhao, and K. Huang, "GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1562–1577, 2021.

[44] F.-M. Bréon and O. Boucher, Eds., *Aerosols: properties, processes, and impacts.* London: Éditions ISTE, 2012.

[45] A. Inness, M. Ades, A. Agustí-Panareda, J. Barré, A. Benedictow, A.-M. Blechschmidt, J. J. Dominguez, R. J. Engelen, H. Eskes, J. Flemming, V. Huijnen, L. Jones, Z. Kipling, S. Massart, M. Parrington, V.-H. Peuch, M. Razinger, S. Remy, M. Schulz, M. Suttie, J. Thompson, A. Wagner, J. Walters, and M. Ziółkowski, "The cams reanalysis of atmospheric composition," *Atmospheric Chemistry and Physics*, vol. 19, no. 6, pp. 3515–3556, 2019. [Online]. Available: https://acp.copernicus.org/articles/19/3515/2019/

[46] Copernicus Sentinel-5P (processed by ESA), "Sentinel-5p tropomi level 2 sulphur dioxide total column," 2020, accessed via the Copernicus Data Space Ecosystem. [Online]. Available: https://doi.org/10.5270/S5P-74eidii

[47] C. Clerbaux, A. Boynard, L. Clarisse, M. George, J. Hadji-Lazaro, H. Herbin, D. Hurtmans, M. Pommier, A. Razavi, S. Turquety, C. Wespes, and P.-F. Coheur,

"Monitoring of atmospheric composition using the IASI instrument: First results and future prospects," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 110, no. 11, pp. 2359–2373, 2009.

[48] L. E. Flynn, C. J. Seftor, J. C. Larsen, and P. Xu, "The ozone mapping and profiler suite (OMPS). the next generation of u.s. ozone monitoring instruments," *Proc. SPIE*, vol. 6296, p. 62960C, 2006.

[49] P. F. Levelt, G. H. J. van den Oord, M. R. Dobber, A. Malkki, H. Visser, J. de Vries, P. Stammes, J. O. V. Lundell, and H. Saari, "The ozone monitoring instrument," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 5, pp. 1093–1101, 2006.

[50] W. P. Menzel and J. F. W. Purdom, "Introducing GOES-I: The first of a new generation of geostationary operational environmental satellites," *Bulletin of the American Meteorological Society*, vol. 75, no. 5, pp. 757–781, 1994.

[51] J. Schmetz, P. Pili, S. Tjemkes, D. Just, J. Kerkmann, S. Rota, and A. Ratier, "An introduction to Meteosat second generation (MSG)," *Bulletin of the American Meteorological Society*, vol. 83, no. 7, pp. 977–992, 2002.

[52] K. Bessho, K. Date, M. Hayashi, A. Ikeda, T. Imai, H. Inoue, Y. Kumagai, T. Miyakawa, H. Murata, T. Ohno, A. Okuyama, R. Oyama, Y. Sasaki, Y. Shimazu, K. Shimoji, Y. Sumida, M. Suzuki, H. Taniguchi, H. Tsuchiyama, D. Uesawa, H. Yokota, and R. Yoshida, "An introduction to Himawari-8/9: Japan's new-generation geostationary meteorological satellites," *Journal of the Meteorological Society of Japan*, vol. 94, no. 2, pp. 151–183, 2016.

[53] C. Roberts, G. Balsamo, S. Bousquet, R. Cullum, P. Dominguez, B. Fox, S. Good, JeanJorda, and et al., "Climate model configurations of the ecmwf integrated forecasting system (ecmwf-ifs cycle 43r1) for highresmip," *Geoscientific Model Development*, vol. 11, no. 9, pp. 3681–3712, 2018. [Online]. Available: https://gmd.copernicus.org/articles/11/3681/2018/

[54] ECMWF, *IFS Documentation CY41R2*. ECMWF, 2016.

[55] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[56] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2005, pp. 886–893.

[57] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[58] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[59] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[60] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[61] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679, 1981.

[62] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, no. 2, pp. 107–113, 1993.

[63] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008, pp. 234–247.

[64] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2015.

[65] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin Zajc, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, A. Khajenezhad, A. Salahledin, A. Soltani-Farani, A. Zarezade, A. Petrosino, A. Milton, B. Bozorgtabar, B. Li, C. S. Chan, C. Heng, D. Ward, D. Kearney, D. Monekosso, H. C. Karaimer, H. R. Rabiee, J. Zhu, J. Gao, J. Xiao, J. Zhang, J. Xing, K. Huang, K. Lebeda, L. Cao, M. E. Maresca, M. K. Lim, M. E. Helw, M. Felsberg, P. Remagnino, R. Bowden, R. Goecke, R. Stolkin, S. Y. Lim, S. Maher, S. Poullot, S. Wong, S. Satoh, W. Chen, W. Hu, X. Zhang, Y. Li, and Z. Niu, "The visual object tracking vot2013 challenge results," Dec 2013.

[66] H. Fan, H. Bai, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, Harshit, M. Huang, J. Liu, Y. Xu, C. Liao, L. Yuan, and H. Ling, "Lasot: A high-quality large-scale single object tracking benchmark," *CoRR*, vol. abs/2009.03465, 2020. [Online]. Available: https://arxiv.org/abs/2009.03465

[67] K. Li, Y. Wang, Y. He, Y. Li, Y. Wang, Y. Liu, Z. Wang, J. Xu, G. Chen, P. Luo, L. Wang, and Y. Qiao, "Mvbench: A comprehensive multi-modal video understanding benchmark," 2024. [Online]. Available: https://arxiv.org/abs/2311.17005

[68] K. Luo, H. Shi, S. Wu, F. Teng, M. Duan, C. Huang, Y. Wang, K. Wang, and K. Yang, "Omnidirectional multi-object tracking," 2025. [Online]. Available: https://arxiv.org/abs/2503.04565

[69] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 415–422.

[70] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[71] Z. Soleimanitaleb and M. A. Keyvanrad, "Single object tracking: A survey of methods, datasets, and evaluation metrics," 2022. [Online]. Available: https://arxiv.org/abs/2201.13066

[72] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[73] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1822–1829.

[74] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, pp. 58:1–58:48, 2013.

[75] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[76] M. Mueller, N. Smith, and B. Ghanem, "Context-aware correlation filter tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1396–1404.

[77] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2015, pp. 58–66.

[78] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark," 2013, pp. 2411–2418.

[79] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei, "Deep learning for visual tracking: A comprehensive survey," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–26, 2021.

[80] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[81] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

[82] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[83] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," *arXiv:1703.07402 [cs]*, 2017.

[84] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[85] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[86] G. Ning, Z. Zhang, C. Huang, Z. He, Z. Ren, and H. Wang, "Rolo: Spatial-temporal correlation and deep learning for multiple object tracking," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 1540–1544.

[87] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines," 2020.

[88] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4277–4286.

[89] I. Jung, J. Son, M. Baek, and B. Han, "Real-time mdnet," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 86–101.

[90] Y. Qin, S. He, Y. Zhao, and Y. Gong, "Roi pooling based fast multi-domain convolutional neural networks for visual tracking," in *Proceedings of the 2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE)*, 2016, pp. 195–201.

[91] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[92] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[93] S. Larspers Qvist, "A performance comparison of the single object tracking algorithms rolo and tiny-rolo," Master's thesis, Mid Sweden University, 2022, tiny-ROLO uses Tiny-YOLO as the detection backbone in the ROLO framework.

[94] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[95] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer, 1995.

[96] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast Online Object Tracking and Segmentation: A Unifying Approach," 2019.

[97] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[98] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[99] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV*

2020, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 213–229.

[100] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," *CoRR*, vol. abs/2103.15436, 2021. [Online]. Available: https://arxiv.org/abs/2103.15436

[101] F. Xie, W. Yang, C. Wang, L. Chu, Y. Cao, C. Ma, and W. Zeng, "Correlation-embedded transformer tracking: A single-branch framework," 2024. [Online]. Available: https://arxiv.org/abs/2401.12743

[102] J. Kugarajeevan, T. Kokul, A. Ramanan, and S. Fernando, "Transformers in single object tracking: An experimental survey," *IEEE Access*, vol. 11, pp. 80 297–80 326, 2023.

[103] L. Lapicque, "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation," *Journal de Physiologie et de Pathologie Générale*, vol. 9, pp. 620–635, 1907.

[104] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[105] E. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[106] R. Jolivet, T. J. Lewis, and W. Gerstner, "Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy," *Journal of Neurophysiology*, vol. 92, no. 2, pp. 959–976, 2004.

[107] M. Matsugu, K. Mori, M. Ishii, and Y. Mitarai, "Convolutional spiking neural network model for robust face detection," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, vol. 2, 2002, pp. 660–664 vol.2.

[108] D. Auge, J. Hille, E. Mueller, and A. Knoll, "A survey of encoding techniques for signal processing in spiking neural networks," *Neural Processing Letters*, vol. 53, no. 6, pp. 4693–4710, Dec 2021. [Online]. Available: https://doi.org/10.1007/s11063-021-10562-2

[109] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems," *Frontiers in Neuroscience*, vol. 15, 2021.

[110] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," 2017.

[111] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[112] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological Cybernetics*, vol. 98, no. 6, pp. 459–478, 2008.

[113] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification," *Frontiers in Neuroscience*, vol. 11, 2017.

[114] D. O. Hebb, *The organization of behavior; a neuropsychological theory.* Wiley, 1949.

[115] Y. Li, S. Deng, X. Dong, and S. Gu, "Converting artificial neural networks to spiking neural networks via parameter calibration," *arXiv preprint arXiv:2205.10121*, 2022. [Online]. Available: https://arxiv.org/abs/2205.10121

[116] Y. Kim, J. Park, and S. Lee, "Spikeconverter: An automated tool for converting artificial neural networks to spiking neural networks," *Neurocomputing*, vol. 400, pp. 297–305, 2020.

[117] J. Ding, Z. Yu, Y. Tian, S. Deng, H. Tang *et al.*, "Optimal ann-snn conversion for fast and accurate deep spiking neural networks," in *Proceedings of the*

30th International Joint Conference on Artificial Intelligence (IJCAI), 2021, pp. 2328–2334. [Online]. Available: https://www.ijcai.org/proceedings/2021/0321.pdf

[118] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[119] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. [Online]. Available: https://arxiv.org/abs/1512.03385

[120] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[121] H. Jiang *et al.*, "A unified optimization framework of ann-snn conversion," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023, pp. 14 929–14 949. [Online]. Available: https://proceedings.mlr.press/v202/jiang23a.html

[122] T. Bu, W. Fang, J. Ding, P. Dai, Z. Yu, and T. Huang, "Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks," in *International Conference on Learning Representations (ICLR)*, 2022. [Online]. Available: https://arxiv.org/abs/2303.04347

[123] H. Gao, W. Fang, Z. Yu, and et al., "High-accuracy deep ann-to-snn conversion using calcium-gated bipolar lif neurons," *Frontiers in Neuroscience*, vol. 17, p. 1141701, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2023.1141701/full

[124] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, 1982.

[125] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: A hebbian learning rule," *Annual Review of Neuroscience*, vol. 31, pp. 25–46, 2008.

[126] N. Ravichandran, A. Lansner, and P. Herman, "Spiking neural networks with hebbian plasticity for unsupervised representation learning," *arXiv preprint arXiv:2305.03866*, 2023. [Online]. Available: https://arxiv.org/abs/2305.03866

[127] M. El-Assal, P. Tirilly, and I. M. Bilasco, "2D versus 3D Convolutional Spiking Neural Networks Trained with Unsupervised STDP for Human Action Recognition," in *International Joint Conference on Neural Networks (IJCNN) 2022*, Padova, Italy, Jul. 2022. [Online]. Available: https://hal.science/hal-03679597

[128] P. Falez, P. Tirilly, I. Marius Bilasco, P. Devienne, and P. Boulet, "Multi-layered spiking neural network with target timestamp threshold adaptation and stdp," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[129] D. E. A. Safa *et al.*, "Continual learning with hebbian plasticity in sparse and predictive coding networks: A survey and perspective," *arXiv preprint arXiv:2407.17305*, 2024. [Online]. Available: https://arxiv.org/abs/2407.17305

[130] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains," *PLOS ONE*, vol. 3, no. 1, p. e1377, 2008.

[131] S. B. Shrestha and G. Orchard, "SLAYER: Spike Layer Error Reassignment in Time," *arXiv:1810.08646 [cs, stat]*, 2018.

[132] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, vol. 7, p. 13276, 2016. [Online]. Available: https://www.nature.com/articles/ncomms13276

[133] A. Nøkland, "Direct Feedback Alignment Provides Learning in Deep Neural Networks," 2016.

[134] A. Samadi, T. P. Lillicrap, and D. B. Tweed, "Deep learning with dynamic spiking neurons and fixed feedback weights," *Neural Computation*, vol. 29, no. 3, pp. 578–602, 2017.

[135] F. Bacho and D. Chu, "Forward direct feedback alignment for online gradient estimates of spiking neural networks," *arXiv preprint arXiv:2403.08804*, 2024. [Online]. Available: https://arxiv.org/abs/2403.08804

[136] J. Lee, R. Zhang, W. Zhang, and P. Li, "Spike-train level direct feedback alignment: Sidestepping backpropagation for on-chip training of spiking neural nets," *Frontiers in Neuroscience*, vol. 14, p. 143, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00143/full

[137] J. Launay, M. Poli, F. Krzakala, J. Brazdil, J. Fiúza, and S. Spigler, "Direct feedback alignment scales to modern deep learning tasks and architectures," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 9345–9355. [Online]. Available: https://arxiv.org/abs/2006.12878

[138] S. Bartunov, A. Santoro, B. A. Richards, L. Marris, G. E. Hinton, and T. P. Lillicrap, "Assessing the scalability of biologically-motivated deep learning algorithms and architectures," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 9368–9378.

[139] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[140] P. Chandarana, M. Mohammadi, J. Seekings, and R. Zand, "Energy-efficient deployment of machine learning workloads on neuromorphic hardware," in *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*, 2022, pp. 1–7.

[141] G. Datta and P. A. Beerel, "Can deep neural networks be converted to ultra low-latency spiking neural networks?" in *2022 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2022, pp. 718–723.

[142] S. Narduzzi, S. A. Bigdeli, S.-C. Liu, and L. A. Dunbar, "Optimizing the consumption of spiking neural networks with activity regularization," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 61–65.

[143] S. Davidson and S. B. Furber, "Comparison of artificial and spiking neural networks on digital hardware," *Frontiers in Neuroscience*, vol. Volume 15 - 2021, 2021. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2021.651141

[144] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009, technical report.

[145] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *ArXiv e-prints*, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.03209

[146] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: histograms of averaged time surfaces for robust event-based object classification," *CoRR*, vol. abs/1803.07913, 2018. [Online]. Available: http://arxiv.org/abs/1803.07913

[147] M. Ji, Z. Wang, R. Yan, Q. Liu, S. Xu, and H. Tang, "SCTN: Event-based object tracking with energy-efficient deep convolutional spiking neural networks," *Frontiers in Neuroscience*, vol. 17, 2023.

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

[149] X. Liu *et al.*, "Spiking transformer-cnn for event-based object detection," in *International Conference on Learning Representations*, 2025.

[150] D. Yang *et al.*, "Event-based spiking neural networks for object detection," *arXiv preprint arXiv:2411.17006*, 2024.

[151] J. Su, X. Lu, H. Yang, Y. Sun, L. Zhao, and B. Xu, "Deep directly-trained spiking neural networks for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 19 764–19 774. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Su_Deep_Directly-Trained_Spiking_Neural_Networks_for_Object_Detection_ICCV_2023_paper.html

[152] P. Zhong, X. Wang *et al.*, "Smtrack: End-to-end trained spiking neural networks for multi-object tracking in rgb videos," *arXiv preprint arXiv:2508.14607*, 2025.

[153] B. Holben, T. Eck, I. Slutsker, D. Tanré, J. Buis, A. Setzer, E. Vermote, J. Reagan, Y. Kaufman, T. Nakajima, F. Lavenu, I. Jankowiak, and A. Smirnov, "Aeronet—a federated instrument network and data archive for aerosol characterization," *Remote Sensing of Environment*, vol. 66, no. 1, pp. 1–16, 1998.

[154] R. C. Levy, S. Mattoo, L. A. Munchak, L. A. Remer, A. M. Sayer, F. Patadia, and N. C. Hsu, "The collection 6 modis aerosol products over land and ocean," *Atmospheric Measurement Techniques*, vol. 6, pp. 2989–3034, 2013.

[155] Y. T. M. Fisher, D. Salmond, A. Fouilloux, and T. Wilhelmsson, "4D-Var, Scalability and Code Design," *T R*, p. 28.

[156] A. Inness, F. Baier, A. Benedetti, I. Bouarar, S. Chabrillat, H. Clark, C. Clerbaux, P.-F. Coheur, R. Engelen, Q. Errera, J. Flemming, M. George, C. Granier, J. Hadji-Lazaro, V. Huijnen, D. Hurtmans, L. Jones, J. W. Kaiser, J. Kapsomenakis, K. Lefever, J. Leitão, M. Razinger, A. Richter, M. G. Schultz, A. J. Simmons, M. Suttie, O. Stein, J.-N. Thépaut, V. Thouret, M. Vrekoussis, and C. Zerefos, "The macc reanalysis: an 8 yr data set of atmospheric composition," *Atmospheric Chemistry and Physics*, vol. 13, no. 8, pp. 4073–4109, 2013. [Online]. Available: https://acp.copernicus.org/articles/13/4073/2013/

[157] E. Guilbert and H. Lin, "A new model for cloud tracking and analysis on satellite images," *GeoInformatica*, vol. 11, no. 3, pp. 287–309, 2007.

[158] O. Dubovik and M. D. King, "A flexible inversion algorithm for retrieval of aerosol optical properties from sun and sky radiance measurements," *Journal of Geophysical Research: Atmospheres*, vol. 105, no. D16, pp. 20 673–20 696, 2000.

[159] L. M. Hurvich and D. Jameson, "An opponent-process theory of color vision," *Psychological Review*, vol. 64, pp. 384–404, 1957.

[160] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1–3, pp. 157–173, 2008.

[161] J. Bromley, J. W. Bentz, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 669–688, 1993.

[162] Z. Yan, Z. Bai, and W.-F. Wong, "Reconsidering the energy efficiency of spiking neural networks," *arXiv preprint arXiv:2409.08290*, 2024.

[163] C. Houghton and T. Kreuz, "On the efficient calculation of van Rossum distances," *Network: Computation in Neural Systems*, vol. 23, no. 1-2, pp. 48–58, 2012.

[164] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47–63, 2019.

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

[166] W. Wei, M. Zhang, H. Qu, A. Belatreche, J. Zhang, and H. Chen, "Temporal-Coded Spiking Neural Networks with Dynamic Firing Threshold: Learning with Event-Driven Backpropagation," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 10 518–10 528.

[167] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal Efficient Training of Spiking Neural Network via Gradient Re-weighting," 2022.

[168] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018, pp. 787–797.

[169] Y. Li, X. Guan, W. Yue, Y. Huang, B. Zhang, and P. Duan, "A reinforced, event-driven, and attention-based convolution spiking neural network for multivariate time series prediction," *Biomimetics*, vol. 10, no. 4, 2025. [Online]. Available: https://www.mdpi.com/2313-7673/10/4/240

[170] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[171] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient re-weighting," in *International Conference on Learning Representations (ICLR)*, 2022.

[172] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[173] G. Folberth, S. Lefkimmiatis, and M. Unser, "Efficient adjoint computation for wavelet and convolution operators," *Journal of Mathematical Imaging and Vision*, vol. 61, no. 4, pp. 551–571, 2019.

[174] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256.

[175] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015, arXiv:1412.6980.

[176] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *arXiv preprint arXiv:1910.07454*, 2019.

[177] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," 2016. [Online]. Available: https://arxiv.org/abs/1604.01802

[178] Y. Luo, H. Shen, X. Cao, T. Wang, Q. Feng, and Z. Tan, "Conversion of Siamese networks to spiking neural networks for energy-efficient object tracking," *Neural Computing and Applications*, 2022.

[179] R. A. Kahn, G. Chen, D. J. Diner *et al.*, "Wildfire smoke detection and classification using satellite observations: An overview of challenges and perspectives," *Atmospheric Environment*, vol. 224, p. 117288, 2020. [Online]. Available: https://doi.org/10.1016/j.atmosenv.2020.117288

[180] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[181] S. Wang, P. Maris Ferreira, and A. Benlarbi-Delai, "Physics Informed Spiking Neural Networks: Application to Digital Predistortion for Power Amplifier Linearization," *IEEE Access*, vol. 11, pp. 48 441–48 453, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10122961/