Optical flow estimation from event-based data using spiking neural networks

Javier Cuadrado Anibarro

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

préparé à l'Université Toulouse III - Paul Sabatier

Estimation du flux optique à partir des données issues des capteurs événementiels avec des réseaux de neurones impulsionnels

---

Thèse présentée et soutenue, le 24 septembre 2024 par

Javier CUADRADO ANIBARRO

École doctorale
EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité
Informatique et Télécommunications

Unité de recherche
CERCO - Centre de Recherche Cerveau et Cognition

Thèse dirigée par
Timothée MASQUELIER

Composition du jury
M. Benoît MIRAMOND, Rapporteur, LEAT (CNRS UMR 7248) - Université Côte d'Azur
Mme Eduardo ROS, Rapporteure, Universidad de Granada
Mme Manon DAMPFHOFFER, Examinatrice, CEA - List (Univ. Grenoble Alpes)
M. Timothée MASQUELIER, Directeur de thèse, CerCo (CNRS UMR 5549) - Université Toulouse III Paul Sabatier

Membres invités
M. Francisco BARRANCO, Universidad de Granada
M. Benoît COTTEREAU, CerCo (CNRS UMR 5549) - Université Toulouse III Paul Sabatier

Table of Contents

List of Figures

List of Tables

List of Acronyms

AC ..... Accumulate
AAE ..... Average Angular Error
AEE ..... Average Endpoint Error
ANN ..... Analog Neural Network
ARE ..... Average Relative Error
BPTT ..... Back-propagation through time
CNN ..... Convolutional Neural Network
DVS ..... Dynamic Vision Sensor
EBS ..... Event-based Sensor
EC ..... Event Camera
ERF ..... Effective Receptive Field
DNN ..... Deep Neural Network
FBS ..... Frame-based sensor
FLOPS ..... Floating point operations per second
FOV ..... Field of view
FPGA ..... Field-programmable gate array
GAN ..... Generative Adversarial Network
GPU ..... Graphics Processing Unit
GRU ..... Gated Recurrent Unit
HPC ..... High-performance Computing
IF ..... Integrate-and-fire
IMU ..... Inertial Measurement Unit
LiDAR ..... Light Detection and Ranging
LIF ..... Leaky Integrate-and-fire
LSTM ..... Long short-term memory
MAC ..... Multiply-accumulate
MSE ..... Mean Square Error
PLIF ..... Parametric Leaky Integrate-and-fire

RNN ..... Recurrent Neural Network
SGL ..... Surrogate Gradient Learning
SNN ..... Spiking Neural Network
STDP ..... Spike-timing-dependent plasticity
TAGC ..... Time-aware grouped convolutions
TBPTT ..... Truncated back-propagation through time
TCN ..... Temporal Convolutional Network

Résumé [FR]

Ce document présente les travaux de thèse effectués par M. Javier CUADRADO ANÍBARRO lors de son contrat en tant qu'étudiant en thèse au sein du Centre de Recherche Cerveau et Cognition (CerCo - CNRS UMR 5549) à Toulouse entre Septembre 2021 et Août 2024. Ce projet de thèse, rendu possible grâce au financement attribué par l'Agence Nationale de la Recherche (ANR) au projet DeepSee (bourse ANR-20-CE23-0004-04), a eu pour but le développement d'algorithmes de vision artificielle, reposant sur les réseaux de neurones impulsionnels, pour la prédiction du flux optique à partir des données issues des caméras événementielles.

Le premier pilier sur lequel ce projet de thèse trouve son socle est le flux optique, c'est à dire, le champ de vitesse dans la scène visuelle, dû à la fois au mouvement relatif entre l'observateur et la scène et à leur profondeur rélative. Cette grandeur a une importance clé chez les êtres vivants, notamment due à sa contribution pour la détection des obstacles et à leur évitement. Pour ces raisons, elle a été étudiée en profondeur en vue de potentielles applications dans le domaine de la navigation artificielle (e.g. déploiement dans des voitures autonomes). Par ailleurs, le flux optique est une grandeur essentiellement dynamique, qui prend en compte non seulement l'information spatiale (i.e. la profondeur) mais surtout la temporalité des informations (une scène sans mouvement présenterait un champ de flux optique nul), et ce sont en effet les informations liées à la temporalité du mouvement qui joueront le rôle principal dans son estimation.

La deuxième composante de ce projet de thèse repose sur les caméras basées sur des événements (*event cameras*, ECs, ou *event-based sensors*, EBSs). Ce type de capteur se différentie des caméras conventionnelles principalement dans son fonctionnement: au lieu de produire une image

(*frame*) dense avec une cadence fixe, les EBSs sont constitués par des pixels asynchrones, qui déclenchent des événements quand les variationes locales de luminosité dépassent un certain seuil préfixé, soit avec une polarité positive pour des increments de luminosité, soit avec une polarité négative pour des décrements. Ces capteurs, inspirés du fonctionnement des cellules ganglionnaires trouvées dans la rétine des animaux, présentent plusieurs avantages par rapport à leur contrepartie basée sur des *frames*, à savoir:

- Latence inférieure: les pixels des EBS sont capables de déclencher des événements avec un délai dans l'ordre de grandeur des microsecondes, ce qui les rend capables de fonctionner virtuellement en temps réel.

- Gamme dynamique très large: les EC peuvent fonctionner dans une pléthore de conditions lumineuses, avec de très rapides variations d'intensité lumineuse, sans poser des problèmes de saturation. Cette propriété est cruciale pour des scénarios de conduite, car elle permet l'utilisation de ce type de caméras dans des situations très diverses: en journée, le soir, dans des tunnels, etc.

- Efficacité énergétique: contrairement aux caméras traditionnelles, qui produisent une image à chaque pas de temps indépendamment des variations dans la scène visuelle, les EC ne produisent pas d'événements lors que la scène reste statique (en absence de mouvement rélatif). Par conséquent, la production d'information est déclenchée par les variations de luminosité dans la scène, sans redondance d'information.

En revanche, ces avantages sont mitigés par les désavantages de ce type de capteur, consistant surtout en un manque d'information sur la vraie intensité lumineuse (la seule information disponible est liée aux variations de luminosité), à une faible résolution spatiale (normalement de l'ordre des Megapixels) et à leur coût économique élevé. Toutefois, l'information de nature éminemment temporelle qu'elles produisent, intrinsiquement liée au mouvement, se présente comme étant particulièrement adaptée pour la prédiction du flux optique. Dès lors, une absence de mouvement serait répresentée à la fois par une absence de flux optique et d'événements. Ainsi, nous avons décidé de choisir cette famille de capteurs pour le développement de notre modèle.

Enfin, le troisième pilier de ce projet de thèse est les réseaux de neurones impulsionnelles (*spiking neural networks*, SNNs). Ce type de neurones artificiels, inspirés des neurones biologiques trouvés dans le cerveau des animaux, présente deux différences clés par rapport aux neurones artificielles analogiques souvent trouvées dans les modèles d'apprentissage profond:

- Les neurones impulsionnels, tout comme les neurones biologiques, présentent une récurrence (mémoire) intrinsèque, répresentée par le potentiel de membrane du neurone. Ce potentiel est affecté par les connexions présynaptiques des neurones afférents, ainsi que par le phenomène de fuite cherchant à ramener le potentiel de membrane vers sa valeur de repos.

- Quand la valeur du potentiel de membrane dépasse un certain seuil, une impulsion (*spike*) se déclenche. À ce moment-là, une information binaire est trainsmise aux neurones post-synaptiques, et la valeur du potentiel de membrane est remise à sa valeur de *reset*.

Nous pouvons constater que ce type de neurones artificielles semblent être adaptées pour le traitement des informations temporelles, notamment à cause des phenomènes de mémoire/oubli répresentés par le couple potentiel/fuite. Par ailleurs, en absence d'information d'entrée, aucune impulsion ne pourra pas être déclechée (en absence de biais dans le modèle), et aucune information ne sera traitée, ce qui rend les SNNs très efficaces en termes de consommation d'énergie quand déployés sur des puces dédiées. De plus, cette propriété est pertinente vis-à-vis de notre problématique à résoudre, car une absence de flux optique dans la scène visuelle ne déclencherait pas d'événements au niveau du capteur, et donc aucune information n'aura à être traitée par le réseau. En revanche, ce type de neurones artificiels bio-inspirés posent de nouveaux problèmes au niveau de leur entraînement du fait de leur activation binaire (fonction de Heaviside, à derivée nulle partout hormis à zéro où elle passe à l'infini). Afin de parvenir à entraîner des SNNs avec les techniques de différentiation automatique implementées dans les librairies Python les plus connues (e.g. PyTorch), nous employerons la technique du *surrogate gradient*, c'est à dire, une approximation du gradient de la fonction d'activation permettant le flux d'information dans l'ensemble du modèle lors de l'entraînement.

Nos premiers pas vers le développement d'un modèle impulsionnel pour la prédiction du flux optique basée sur des données issues des ECs ont consisté à adapter une architecture du type U-Net pour l'estimation du flux optique en exploitant la mémoire des neurones, avec un entrainement séquentiel sur des histogrammes d'événements (cumul des événements par pixel e polarité pendant une fenêtre de temps fixe) consécutifs. Cependant, nous avons vite constaté les difficultés liées à ce type d'entraînement:

- D'un côté, le coût computationnel associé à dérouler le réseau lors de la rétropropagation des gradients rend l'entraînement impossible pour des très longues séquences.
- D'un autre côté, il est apparu compliqué de trouver le bon équilibre entre mémoire et oubli, ce qui menait à des résultats d'entraînement incohérents et désestructurés.

Nous avons alors tourné notre attention vers les connaissances acquises lors de nos travaux précedents sur l'estimation de la profondeur à partir des EBS avec des SNNs, employant des unités sans mémoire pour sa computation. Néanmoins, cette stratégie n'a pas fonctionné pour l'estimation du flux optique, car elle prend uniquement en compte un seul histogramme d'événements, qui ne permet pas de restituer l'information temporelle. Cette information temporelle joue le rôle principale pour l'estimation du flux optique. Nous avons donc ciblé nos efforts à explicitement l'introduire dans le modèle d'une façon pertinente, tout en respectant les contraintes *bio-inspirées* qui permettraient à notre modèle d'être déployé sur des puces neuromorphiques. La solution que nous avons trouvée consiste à utiliser un tenseur cinq-dimensionnel pour répresenter l'information d'entrée du réseau:

- La première dimension réprésente le *batch size*.
- La deuxième dimension comporte les canaux des histogrammes: un pour le cumul des événements à polarité positive par pixel, et un autre pour les événements à polarité négative.
- La troisième dimension contient l'information temporelle, c'est à dire la suite d'histogrammes fournis au modèle.
- Les deux dernières dimensions réprésentent la résolution spatiale des histogrammes d'événements.

Ainsi, nous avons modifié notre modèle de base pour que l'information temporelle soit gérée par des convolutions 3D le long de la dimension temporelle, analogues à des lignes de délais, de sorte qu'elle collapse au niveau du *bottleneck* du réseau pour augmenter son champ récepteur effectif tout en gardant la temporalité le long de l'encodeur. En imposant à la fois la diminution du modulo du vecteur erreur et de l'écart angulaire entre la prédiction et la vraie valeur du flux optique, nous avons réussi à entraîner notre modèle, qui présente les meilleures performances de l'état de l'art atteintes par un SNN sur cette tâche.

La suite du projet a consisté à essayer d'expoîter des images en niveau de gris, en conjonction avec des données événementielles, pour de meilleures estimations du flux optique. Étant donné que beaucoup de caméras bassées sur des événements sont capables de produire à la fois des événements (dotés d'une incroyable finesse temporelle) et de "vraies" images (avec une meilleure résolution spatiale), nous avons essayé d'exploîter ces deux modalités à la fois pour améliorer nos résultats. L'approche suivie, déjà introduite dans la littérature, a consisté à une architecture à double encodeur (un pour chacune des modalitées d'entrée), qui combine les sorties des encodeurs au niveau du bottleneck et décode l'information bimodale pour l'estimation du flux optique.

Pour récapituler, ces travaux de thèse comportent deux études:

- Premièrement, une méthode d'estimation du flux optique à partir de données issues des EBS en employant des SNNs. Nos travaux ont abouti à une publication qui introduit le SNN le plus performant de l'état de l'art sur cette tâche, à la fois sur des scénarios de vol intérieur (MVSEC Dataset) et des scénarios de conduite à l'extérieur (DSEC Dataset). Son implémentation sur des puces neuromorphiques réprésente un axe d'amélioration potentiel. À cet égard, deux lignes de recherche sont possibles:
  - D'un côté, le réseau pourrait être simplifié en nombre de paramètres et de *throughput* si les convolutions tridimensionnelles étaient remplacées par des convolutions 2d standard. Il y aurait plusieurs façons d'atteindre cet objectif, soit en introduisant la recurrence intrinsèque des SNN dans le modèle (avec un modèle *stateful*), soit en traitant toute l'information temporelle d'un coup

avec une convolution unidimensionnelle qui apprend le long de la dimension temporelle.

  - D'un autre côté, les améliorations pourraient viser l'amélioration de la consommation énergétique du modèle en encourageant explicitement l'encodage d'information dans des tenseurs creux (*sparse*), ainsi que la réduction du coût de mémoire associé aux paramètres à travers de leur quantization.

- Deuxièmement, nous avons ciblés nos efforts de recherche sur la fusion bi-modale des données issues des EBS et des FBS pour une amélioration des performances. Malgré nos meilleurs efforts, nous n'avons pas encore réussi à améliorer notre réseau unimodal. Néanmoins, nous avons réussi à developper des modèles exploitant à la fois les images et les événements pour l'estimation du flux optique. Cela conforte la faisabilité de l'estimation du flux optique reposant sur la fusion de données, ce qui devrait constituer la future ligne de recherche du projet.

Pour conclure, cette thèse a été complétée par le suivi de plusieurs formations, détaillées en profondeur dans le Chapitre 1. Finalement, il y a eu plusieurs acteurs qui ont rendu possible cette thèse outre l'Agence Nationale de la Recherche, que nous tenons à remercier: l'Agence National de Recherche espagnole (Spanish National Grant PID2019-109434RA-I00/SRA), le financement FLAG-ERA (projet DOMINO), le programme DesCartes et la région Occitanie (grâce à l'attribution du projet 2022-p22020 pour l'utilisation du supercalculateur Olympe).

Chapter 1

Introduction

Artificial intelligence has always been a pivotal interest to humanity. It probably comes as no surprise, since being able to develop intelligent, self-operating machines could in turn help us understand our very own nature, as well as providing answers to some of life's deep-rooted questions, which most likely belong in a philosophical dissertation, instead of in a PhD manuscript about informatics. It is this interest, though, which has pushed innovation in many fields, and a journey through history can help us understand not only where we come from, but also where we are going.

If we think about artificial intelligence in the sense of autonomous machines, we can go back in time to Jewish mythology, in particular to the figure of golems: autonomous clay-beings obeying the instructions that they were given by their masters. While more in the realm of fantasy than science, it is the will of achieving such a machine that would push scientists in the not so distant past to develop ingenuities such as robots. In fact, many centuries later, the most simple form of automatons were presented to the world, and although earlier versions were purely mechanical devices, they would nonetheless represent the missing piece that would culminate in the development of robots and other machines alike. Not only researchers and engineers would be inspired by these myths: we can also find examples in literature, where the famous Isaac Asimov formulated his *Three Laws of Robotics*.

Science fiction lived a golden age during the 20th century, and the topic of artificial intelligence and human/machine interaction was ever

present. Examples can be found as early as 1925, with the film *Metropolis* presenting robots impersonating real people to the point of tricking most of the characters around them. This recurrent topic would also appear in the 1968 novel *Do androids dream of electric sheep?* by Philip K. Dick, later adapted into the movie *Blade Runner* by Ridley Scott in 1982, and it invites the spectators to reflect on what being human and being alive really means. This topic is closely intertwined with the natural human strive to live forever, and has inspired many to dream about transferring their consciousness into computers and automatons alike (e.g. the 1987 film *Robocop*, which has to be mentioned when talking about this topic). Dreaming of developing artificial brains has undoubtedly been at least partially motivated by this desire, and the development of spiking neural networks (bio-inspired artificial units with brain-like behaviour) is a natural milestone towards the ultimate goal of immortality. Finally, the 1982 film *Koyaanisqatsi* invites its viewers to think about the role of technology in our daily life and our societies, as well as pondering how we interact with it and how it shapes the world around us.

Reality, however, is still far from these science-fiction inventions. While recent developments such as ChatGPT [5] or DALL-E [6] make us dream of (and even fear) the rise of intelligent computers like *HALL 9000* (firstly imagined by Sir Arthur C. Clarke in his novel *2001: A Space Odyssey*, then masterfully adapted to the big screen by Stanley Kubrick on a namesake film), artificial intelligence in our daily life remains limited to more prosaic applications, ranging from intelligent vacuum cleaners to personal voice assistants, tumor detection software or collision avoidance systems in cars. It is in the automotive field that this PhD finds its inspiration, since our goal is to develop optical flow estimation algorithms that may one day be deployed in cars and other similar vehicles. Our motivation, as we will further explain in Section 2.1, comes from the rich information on both depth and motion that optical flow contains. To achieve our goal, we will use data obtained from event-based cameras because of their intrinsic ability to capture temporal information (more on event cameras will be presented in Section 2.2). The final piece of the puzzle, and the glue holding everything together, will be an artificial neural network composed of spiking neurons: bio-inspired units that mimic the brain's behaviour, and which achieve superior energy efficiency when cleverly deployed on dedicated hardware. We will introduce spiking neural networks (SNNs) in Section 2.3, and we

will explain the nuances of our model in Chapter 3. Finally, we will explore the possibility of combining event-based sensors (EBS) and frame-based cameras (FBS) in Chapter 4, seeking to exploit the high spatial resolution provided by FBSs when combined with the fine temporal information that EBSs can provide.

This PhD project, which started as a Master's Degree internship, has been possible thanks to the funding of the Agence Nationale de la Recherche under Grant ANR-20-CE23-0004-04 *DeepSee* (Figure 1.1). This project seeks to develop event-based vision using spiking neural networks and neuromorphic hardware for real-world automotive applications (e.g. autonomous driving). Three laboratories integrate this ANR grant: the Laboratory of Electronics, Antennas and Telecommunications (LEAT - University Côte d'Azur / UMR 7248 CNRS), the Laboratory of Computer Science, Signals and Systems of Sophia Antipolis (I3S - University Côte d'Azur / UMR 7271 CNRS) and the Brain and Cognition Research Center (CerCo - University Toulouse 3 / UMR 5549 CNRS). In addition, the automotive company Renault makes part of the project as an industrial contributor due to their interest in the potential applications of the research carried out during the ANR. Finally, the company Prophesee will act as an industrial consultant due to their expertise in event-cameras.

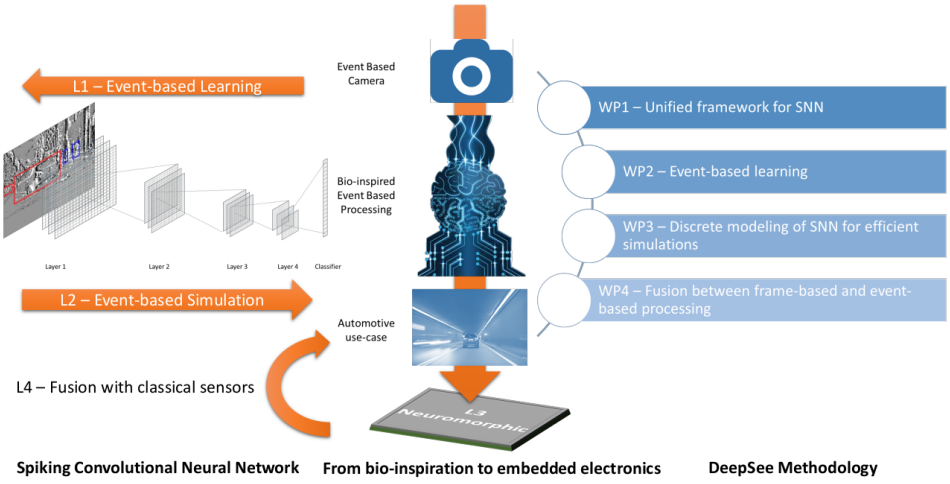


Figure 1.1: Overview of the DeepSee Project

The DeepSee Project has been structured in three work packages (WP), each of them concerning one or several levels of study of event-based processing (Table 1.1). This PhD project will focus on WP2 (even-based learning) and WP4 (fusion between FB and EB processing). The project will therefore cover all of the different levels of study to some extent, i.e. we will develop spiking neural networks (L2) that learn from event-based data (L1) under the constraint of potential neuromorphic hardware implementations (L3), and we will finally seek to combine event-based and frame-based data for optimal performances (L4).

	WP1	**WP2**	**WP3**	**WP4**
L1 – Event-based learning	x	x	x	
L2 – Event-based discrete processing	x		x	x
L3 – Event-based neuromorphic hardware	x	x	x	x
L4 - EBS/FBS fusion	x			x

Table 1.1: Levels of study of event-based processing

This PhD project has managed to publish one contribution in *Frontiers in Neuroscience – Neuromorphic Engineering*, on a special issue on *Spike-based learning applications for neuromorphic engineering*, entitled "Optical flow estimation from event-based cameras and spiking neural networks" [7]. Our model achieved state-of-the-art performance on the MVSEC dataset [8], as well as good performance on the DSEC dataset [4], showing the best performance so far on the latter dataset using spiking neural networks. The publication has received a good number of citations during its first year of life (21 citations in Google Scholar as of July 11th 2024), showing its relevance as well as the interest that this research topic is still generating. In addition, we have also collaborated in a project involving depth estimation using event-based data and spiking neural networks [9], which was published in 2022 in *IEEE Access*. Although this work is indeed heavily linked to this doctoral thesis' subject, we believe that this manuscript's *leitmotiv* is temporal information, present in the three legs of the tripod that supports our research works: optical flow as a metric of relative motion, event sensors as asynchronous devices triggered by motion, and spiking neural networks as dynamic computational units. Therefore, we have decided not to include this project in this thesis' manuscript, although multiple references to it will

be found along this document due to the precious insights we gained while working on this publication. Finally, this PhD project has explored the introduction of frame-based images into a dual spiking neural network for enhanced optical flow estimation (Chapter 4). Although our results so far are rather lackluster in terms of quality, their progression leads us to remain optimistic, and we believe that such a model is indeed possible to be achieved.

The doctorate program has not only been an opportunity to work on the fields of computer vision and neuromorphic computing, but also to publicly share ongoing investigations and results in forums such as the GDR BioComp PhD Forum (held online in November 2021) or the GDR BioComp Colloquium (which took place in Banyuls-sur-Mer in November 2023). Furthermore, the preliminary results on optical flow estimation with event-camera data using spiking neural networks were presented during the poster session of the SNUFA Workshop in 2022.

In addition, several formations and missions have taken place in parallel to the PhD works, the most relevant to the PhD subject being:

- A trip to Grenoble where we could exchange with Prophesee, allowing us to delve into the hardware and simulation of event data.
- A one-week visit to the LEAT laboratory in Sophia-Antipolis (Université Côte d'Azur, Nice), where we could learn more about neuromorphic hardware in general and their custom-made FPGAs in particular.
- A two-day intensive formation on High-Performance Computing (HPC) provided by the CALMIP supercalculator's staff, teaching us about the generalities of HPC and about the usage of the *Olympe* supercalculator in particular.
- A three-day formation on Docker, delving into the basics of the system and the deployment of containers, as well as introducing us to some simple use-cases.
- Two one-day formations on mastering the *Zotero* bibliography handling tool: one introductory session, and an additional one covering more advanced applications.

- A one-day formation on search engines, learning about how to best use them, their working mechanisms and the legislation they are subject to. This formation also targeted how to best exploit research engines to improve the impact factor of research works, as well as optimizing their indexation.

- A one-day formation on ethics and integrity applied to research, where the subject of morality was considered as a whole during the first half, and it was then applied to the research environment.

In addition to the funding provided by Grant ANR-20-CE23-0004-04, this PhD would have not been possible without the support of the Spanish National Grant PID2019-109434RA-I00/ SRA (State Research Agency /10.13039/501100011033), by a FLAG-ERA funding (Joint Transnational Call 2019, project DOMINO), and by the Program DesCartes and the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) Program. We would also like to express our gratitude to the Occitanie's region for allowing us to use the *Olympe* Supercalculator in the CALMIP center thanks to the allocation 2022-p22020, which greatly helped us to speed up calculations and the obtention of results.

Chapter 2

Related Work

All the work performed during this thesis relies on three main pillars: optical flow, event-based cameras and spiking neural networks. A thorough revision of each of these fields literature will be performed, in order to better understand our study case. However, there is substantial overlap between these fields, suggesting an implicit connection of these domains. While we will try our best to classify the different references, it may be argued that some of them belong in a different subsection, so it is advised to read this whole section to better understand each of the different fields, in spite of some parts being less pertinent to some readers.

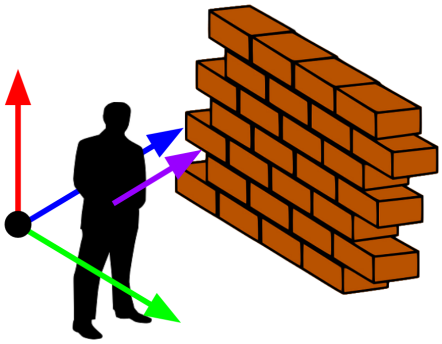
2.1 Optical Flow

The first leg upon which our founding tripod relies on is optical flow. Optical flow, or optic flow, was a magnitude first introduced in the 1940s by the American psychologist James J. Gibson [10]. It consists of the apparent pattern of motion present within a visual scene, due to the relative displacement between the observer and its surroundings. Due to its richness in information, containing displacement and depth information alike, it has remained a relevant research topic ever since its inception, intriguing neuroscientists and engineers alike (e.g. motion parallax and how it is exploited in the animal kingdom for perception and navigation).

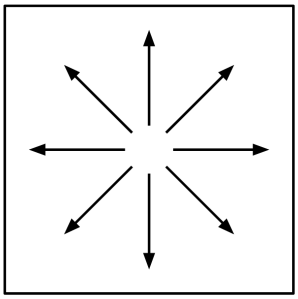
Optical flow has been widely studied in the neuroscience community. A great number of publications have focused on optical flow processing

in macaques, discovering they cortical networks involved in optic flow processing in the monkey's brain through functional magnetic resonance imaging (fMRI) [11], and showing selectivity to optical flow patterns in cortical areas (e.g the medial superior temporal area [12, 13, 14], the ventral intraparietal area [15] or the V6 area [16]). One particularly interesting outcome of these studies is that selectivity to optical flow patterns across different brain areas is a phenomenon akin to the way ANNs learn optical flow, since the kernels usually obtained during training within the CNN naturally end up encoding the combination of the different optical flow patterns. As far as humans are concerned, fMRI studies have shown the influence that optical flow processing has on perception of ego-motion and navigation [17, 18], although the role that each brain region plays in optical flow perception and processing remains unknown.

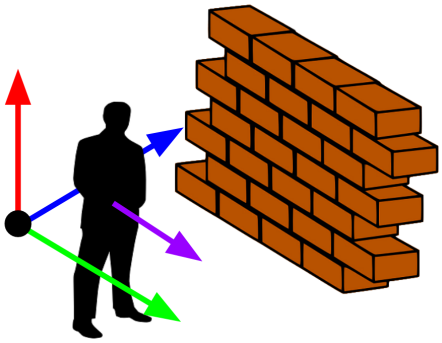
As far as the artificial intelligence community is concerned, optical flow was quickly adopted as one of the major computer vision tasks, due to its wide range of potential applications: video compression, object detection and tracking, obstacle avoidance, visual odometry, etc. However, since optical flow is not a measurable magnitude (due to its intrinsic subjective nature, which is dependent on relative motion), it is hard to find high-quality annotated datasets. Indeed, most datasets rely on a combination of depth measurements obtained with LiDAR and IMUs to infer it, such as MVSEC [8] or DSEC [4] for optical flow estimation from event-based data. Additionally, there are simulators that allow researchers to generate artificial ground-truth following this approach, like the CARLA Simulator [19]. Nevertheless, the simulated nature of the data means that no realistic noise is usually present (realistic noise models providing ecological perturbations are difficult to achieve), therefore limiting their applicability for real-world applications. Other available datasets rely on synthetic data [20] for optical flow estimation, superposing foreground images to random background images and then imposing relative motion between both layers, or measure optical flow from simple artificial scenes [21]. Another approach which has been explored to improve optical flow ground truth consisted of trying to increase size, complexity and variety by simulating optical flow from movies, ideally keeping challenging conditions such as reflections or blur for example [22]. Nevertheless, these methods also suffer from the lack of noisy data associated to simulated optical flow, with the added limitation of not providing ecological images. Even



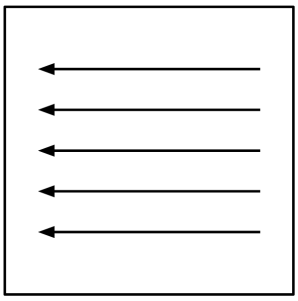
(a) Longitudinal displacement relative to an object.



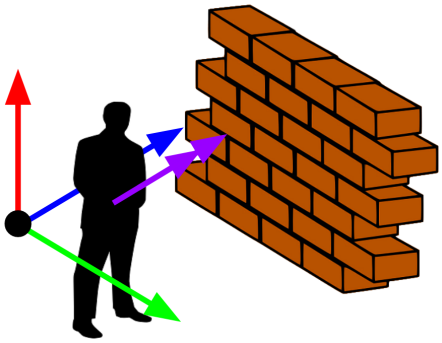
(b) Corresponding radial optical flow pattern.



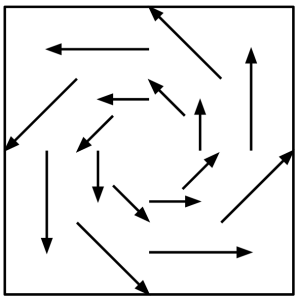
(c) Transversal displacement relative to an object.



(d) Corresponding horizontal optical flow pattern.



(e) Axial rotation relative to an object.



(f) Corresponding twisting optical flow pattern

Figure 2.1: Basic optical flow patterns

publications trying to tackle these issues were eventually bound to admitting the inherent limitations of simulated optical flow [23]. Additional datasets have been proposed to automatically generate optical flow ground-truth from frame sequences captured on real-world scenarios with standard cameras [24], providing real-world optical flow ground-truth with minimal investment in recording gear. Other datasets [25] propose different scene representations for optic flow estimation, taking advantage of the statistical data distribution of outdoor scenes. Finally, the KITTI dataset [26] provides optical flow ground-truth obtained from a combination of depth and ego-motion measurements, thus presenting highly-ecological optical flow labels for driving scenarios.

When considering the estimation of optical flow, the task that is usually tackled is the projection of motion in the 2D focal plane (often neglecting local luminance changes due to brightness variations), measured as the pixel displacement between two consecutive frames/timestamps. The first methods successfully processing optical flow date back to 1981: on the one hand, Lucas and Kanade [27]proposed a method relying on a *data* term (representing the brightness constancy assumption), on the other hand, Horn and Schunk [28] introduced the first method using also a *regularization* term (representing the smoothness constraint) for optical flow estimations. From that point onwards, traditional optical flow estimation approaches have relied on variational methods [29], which consist of minimizing an energy function containing a combination of the *data* term D and the *regulariser* term R. The α factor is a positive coefficient balancing the influence of both terms.

$$E(u) = \int_{\Omega} [D(u) + \alpha \cdot R(u)] d\Omega \tag{2.1}$$

This method assumes that the image brightness of grayscale images is constant within short timespans (a hypothesis know as *the optic flow constraint*), and its applicability has been extensively studied in the literature [30], even demonstrating their viability for real-time optical flow estimation on standard computers [31]. The method has been extensively used since its inception, and has suffered many iterations and modifications. In fact, the problem with local optical flow estimation methods is the *aperture problem* or the *window problem*, i.e. the ambiguity in determining the true velocity of moving objects within the visual scene during a sequence when its trajectory is not perpendicular to the intensity gradient in the

image. This challenge has led more recent solutions to combine local-global optical flow estimations, which provide higher accuracy at the cost of increased computational power requirements. Among the modifications to traditional optical flow estimation methods, we can find models penalizing the Laplacian of the flow components instead of the standard Regularization term R [32] (suitable for ecological driving scenarios due to large optical flow variations). Some other publications propose using both D, R and the Laplacian of R in a combined energy function for better performance on video interpolation [33]. Finally, other publications propose a completely different function yet follow the same philosophy, e.g. using a Laplacian mesh energy term [34]. For more information on traditional optical flow computation methods, a keen reader can consult [35], showing an extensive survey of optical flow algorithms published until 1995, and [36] to find more information on how optical flow estimation methods have evolved across time. Moreover, the *Scholarpedia* article on optical flow [37] provides very clear definitions and explanations of optical flow, in addition to reviewing the more recent literature on the topic.

The advent of deep learning led to convolutional neural networks taking over the computer vision field, and optical flow estimation was also subject to this trend. CNNs have been used in a plethora of tasks ever since their adoption by the community, be it for direct optical flow estimation of for optical-flow-dependent tasks. For example, future frame prediction algorithms exploiting optical flow found in GANs a way to improve the quality of their predictions [38]. Also for future-frame video prediction, [39] achieves multistep flow prediction from still images leveraging spatio-temporal information learnt with 3D convolutions. Additionally, optical flow has been employed in information technology as a technique for video compression [40]. Looking at optical flow estimation itself, some models (e.g. [41]) have been proposed to exploit its vector nature (Figure 2.1) for better per-pixel estimations between two consecutive images. A wide range of different techniques has been used to achieve DNNs with better estimation scores, e.g. adaptive pyramid sampling [42] (for better feature representation), Hampel filters [43] (for better denoising), cross-strip correlations [44] (to better capture long-distance dependencies), or the aforementioned GANs (for more ecological estimations). SAMFlow [45] proposes an implementation of *Segment Anything* [46] for better contour definition, allowing for sharper optical flow maps. Finally, much like

CNNs were predominant in deep architectures, transformers [47] took the field by storm, dominating the computer vision scene until the resurgence of convolutional models in recent years. Many models have appeared inspired by the transformer architecture, either trying to exploit attention to improve performance [48], or using transformers within their architecture (e.g. [49] introducing a transformer-based model for optical flow estimation, or [50] proposing a unified approach for flow, depth and disparity estimation).

2.2 Event-based cameras

The idea of event cameras was introduced as early as 1991 in *The Silicon Retina* by Misha Mahowald [51], during her time as a PhD student at the California Institute of Technology (Caltech). The main idea behind this innovative device was mimicking the way in which a biological retina operates, with ganglion cells asynchronously triggering signals in the form of electrical impulses when subject to luminance variations. In the same vein, an event camera would consist of an array of independent pixels, where signed events would asynchronously be triggered due to local brightness variations (Figure 2.2). Unfortunately, Misha Mahowald would pass away in 1996 at age 33, and the computer vision community still mourns her loss, wondering what could have been achieved were she still actively researching bio-inspired vision systems. Her impact on the event-vision field has been acknowledged by many researchers over the years [52], and we can find solace in her legacy living on thanks to the *Misha Mahowald Prizes for Neuromorphic Engineering*, awarded yearly to the most influential research results in neuromorphic engineering.

When compared to traditional frame-based sensors, event cameras present some qualities that set them aside, the most important ones being:

- Advantages:
  - Low latency, since the pixels fire independently in virtually real-time.
  - Low energy consumption, since still scenes do not trigger events.

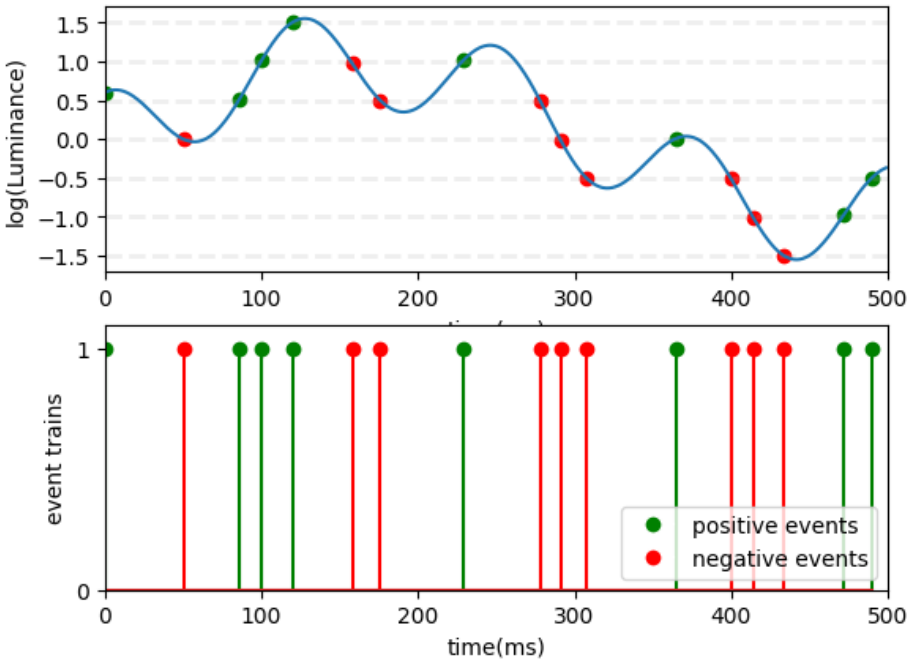


Figure 2.2: Events triggered by an event camera (firing threshold of 0.5 on the log(Luminance))

  - Higher dynamic range, being able to work in a wide range of brightness conditions without saturating.

- Disadvantages:

  - Low spatial resolution, usually in the range of 1Mp, although this feature is constantly improving.
  - Event data does not provide information on intensity within the image, but rather about brightness variations over time.
  - Event-based sensors are usually more expensive than their frame-based counterparts.

Nowadays, a number of commercial solutions exist for event sensors, and a variety of available datasets for event-based vision have arisen. For instance, Innivation has developped the DAVIS 346 camera [53], which was used to record the UZH-FPV Drone Racing dataset [54] and the

MVSEC dataset [8]. Other companies such as Prophesee have specialized on event-based sensors, providing not only event cameras (like the Prophesee Gen3.1 camera [55] used to record the DSEC dataset [4]) but also the Metavision software toolkit for event-based vision [56]. In addition, they also work with industrial partners to develop event sensors according to their needs, like the SilkyEvCam by Century Arks [57], the VisionCam by IMAGO technologies [58], the Triton™ GigE vision camera prototype by LUCID Vision Labs [59] or the Sony IMX636 and IMX637 stacked event-based vision sensors by Sony [60]. There are also event-based solutions developed in academia, like the CeleX-V event sensor [61].

Although event cameras have gained ground in the computer vision scene, their availability remains a concern, for their price has held them back from widespread applications. Several publications have tried to tackle this issue, trying to provide good quality event data at a minimal cost. Such is the case of the aforementioned CARLA simulator [19], which has a built-in DVS simulator. In the same line, [62] and [63] both propose rendered event datasets for optical flow estimation. Other approaches have tried to exploit the plethora of available video datasets by transforming them into event-like data [64], but the intrinsic high latency of frame-based cameras prevents them from achieving competitive temporal resolution, in addition to making them subject to motion blur. Furthermore, these methods lack the real-world noise associated to event cameras, providing unrealistic event data. On the opposite side of the spectrum, others studies have targeted their efforts in denoising event data [65]. Finally, some mobile applications have been developed for real-time optical flow simulation on smartphones [66], providing an inexpensive alternative for real-world event data at the expense of temporal resolution.

The state of the art of event-based vision has been thoroughly reported over the years [67, 68], and the technology has been applied for a wide range of tasks: from image reconstruction [69] to object detection and tracking [70, 71, 72], disparity estimation [73] and even semantic segmentation [74]. Moreover, numerous studies have been carried out concerning data representation for event-based cameras (e.g. [75]) and on the loss functions to use for event-based vision (e.g. [76]).

As far as optical flow estimation is concerned, many different methods

have been proposed over the years. Unsupervised methods have been often used, since they do not require ground-truth data for their training. These methods often rely on warping events to a reference time through the optical flow estimation, and then computing a photometric loss function between the real measured events and the warped events, usually imposing other additional constraints such as image deblurring or contrast maximization [77, 78], but other alternatives exist relying on probability distributions, such as the Fisher-Rao metric [79]. While these methods have historically provided good results, the metrics are constrained to pixels containing events, thus being inadequate for dense optical flow estimation. On the other hand, other architectures have opted for recurrent neural networks trained in a supervised manner, either with an standard MSE loss [80] or combining a simple L1 loss with other constraints, such as image deblurring [81] or motion compensation [82]. Finally, some recent models try to exploit the latest state-of-the-art advancements in the deep learning scene and apply them to event-based vision, e.g. state space models specifically tailored to event data [83].

2.3 Spiking Neural Networks

The final building block of our project consists of spiking neural networks, i.e. artificial neural networks where stateful neurons take the place of standard activation functions (e.g. ReLU, sigmoid, etc.), thus enabling biologically-plausible algorithms. Much like their biological counterparts, artificial spiking neurons emit spikes, i.e. stereotype "all-or-none" electrical impulses emitted when a neuron is sufficiently stimulated (in other words, when its membrane potential reaches the firing threshold).

This section starts by introducing some mathematical models for spiking neurons. Afterwards, we will discuss about information encoding, differentiating between spike- and rate-based approaches. Next, the different learning mechanisms used to train SNNs will be presented. We will then proceed to compare spiking neural networks with their standard analog counterparts, and we will end up by listing some of the many applications of spiking neural networks in the literature.

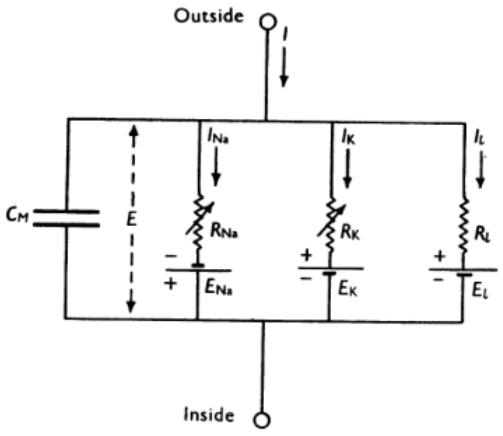


Figure 2.3: Hodgkin-Huxley neuron model (borrowed from [1])

2.3.1 Spiking Neuron Models

Examples of brain-like artificial units can be found as early as 1952, with the introduction of the Hodgkin-Huxley neuron model [1]. This bio-physical model, equivalent to the electrical circuit shown in Figure 2.3, is as close to a biological brain as possible, modelling the different ion currents present during a neuron's charge/discharge cycle (sodium and potassium), was well as the membrane potential and a leakage current. Although this model is able to reproduce the behaviour of a real brain, its high computational cost (four differential equations) has relegated it to toy examples and proofs of concept rather than real-world applications.

Since the publication of the Hodgkin-Huxley model, many more neuron models have arisen, sacrificing biological plausibility for ease of implementation. Such is the case of the Izhikevich neuron model [84], published in 2003. This model opts to model the membrane potential and the membrane recovery value instead of the ion currents, thus requiring only two differential equations. While significantly simple than its predecessors, it manages to capture a wide range of neural dynamics, from regular spiking to bursting or chattering. Nevertheless, it remains too complex for large-scale deep learning tasks, and it has mostly been used to model small populations of neurons.

The simplest artificial neuron model that has prevailed is the Leaky

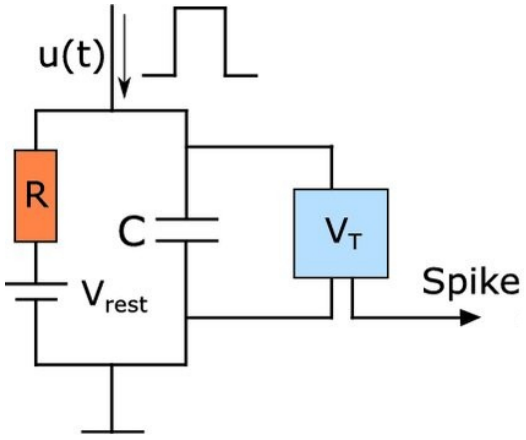


Figure 2.4: LIF neuron model (borrowed from [2])

Integrate-and-Fire neuron, i.e. an exponential decay of the membrane potential (Equation 2.2). Although this model places itself rather far from biology, its simple implementation (see Figure 2.4 for its equivalent electric circuit, very simple to a standard resistor-capacitor architecture) and low associated computational power have propelled it to the go-to spiking neuron model for large-scale deep learning algorithms.

$$\frac{dV}{dt} = -\lambda \cdot V \implies V_t = V_{t-1} \cdot e^{-\lambda \cdot \Delta t} \tag{2.2}$$

The LIF neuron model in itself is very easy to understand, yet it poses some challenges when being deployed in a computer for its training. The main issue is that the evolution we have just described happens in real-time, whereas computers need data to be discretized to perform their computations. The evolution of the neuron's membrane potential can be modelled with three equations:

- Neuronal charge (Equation 2.3): at time t, the membrane potential before firing H_t is a function of its previous value V_{t-1} (membrane potential after firing at time *t-1*), affected by the neuron's time constant τ_m (leak factor), and by all of the input spikes X_t triggered on afferent neurons during the time window $(t-1 \rightarrow t)$. However, these input events are seen as if they had happened simultaneously, potentially hindering the temporal resolution. Since event data is continuous,

though, the time window $(t-1 \rightarrow t)$ can be as short as desired, increasing computational cost but also improving temporal resolution.

$$H_t = f(V_{t-1}, X_t) = V_{t-1} + \frac{1}{\tau_m}[-(V_{t-1} - V_{reset}) + X_t] \tag{2.3}$$

This neuron model combines a good approximation of the membrane potential's temporal evolution and a reasonably low computation cost. Related to this model, it is also worth mentioning integrate-and-fire neurons, a further simplification where the spiking neurons have no leak, therefore acting as perfect integrators until their firing threshold is reached (hence their name). The integrate-and-fire membrane potential evolution can be seen in Equation 2.4.

$$H_t = f(V_{t-1}, X_t) = V_{t-1} + X_t \tag{2.4}$$

- Neuronal fire (Equation 2.5): the membrane potential before firing is compared to the firing threshold $V_{threshold}$ using the Heaviside step function (Θ), and a spike S_t is triggered if the threshold is reached.

$$S_t = g(H_t - V_{threshold}) = \Theta(H_t - V_{threshold}) \tag{2.5}$$

- Neuronal reset (Equation 2.6): the membrane potential after firing V_t is either set to the reset value if a spike has been triggered (hard reset), or it remains unchanged otherwise (its value set to H_t).

$$V_t = H_t \cdot (1 - S_t) + V_{reset} \cdot S_t \tag{2.6}$$

The model itself may be very simple, but the actual implementation took researchers many years. Indeed, the challenge here is the Heaviside step function, which is non-differentiable at zero and constant everywhere else. While it may seem that the problem with this function is the discontinuity at zero, it is actually the other way round. In real-world applications (and specially when making computations), it is virtually impossible to have an exactly-zero value at any point. On the other hand, for every other possible value that the Heaviside function may take as an input, the derivative of the step function would be exactly zero, thus preventing information to flow through the gradients. This impossibility to integrate spiking neurons into standard autodifferentiaton tools (commonly used in deep learning)

relegated spiking neurons to mere coincidence detectors for a very long time, preventing them from achieving competitive results on standard, real-world tasks. It would be in 2019 when the field of SNNs was revolutionized thanks to the introduction of Surrogate Gradient Learning [85], which is explained in Subsection 2.3.3.

2.3.2 Description Levels

Research in spiking neural networks has not only focused on neuron models, though, and information coding has remained a major research topic since their inception. Two possible description levels exist: on the one hand, spike-coding relies on spikes themselves to carry information, on the other hand, rate-coding encodes information in firing rate rather than in spikes alone [86]. Both approaches have extensively studied, and many iterations have emerged for each of them. For instance, some spike-based models have decided to encode to information in the first neuron to spike [87, 88], therefore proposing models following a *winner takes all* philosophy. Other spiking models have relied on allowing at most one spike per neuron, for efficient information large-scale artificial intelligence tasks [89]. Meanwhile, other spike-based models encode information the neurons' time to first spike [90] instead of in the firing rate itself, showing the power of spikes as encoding mechanisms by exploiting a floating-point value (the firing time) all while encoding the information in the spikes themselves. A keen reader will have realized that, in a nutshell, the main difference between both approaches is the nature of the encoded information: spike-based methods use binary information (i.e. spikes), whereas rate-based models work with real numbers (e.g. firing rate or time to spike). This key difference, as naive as it may seem, is actually crucial when training a SNN: rate-based models may be more limited (e.g. they cannot act as coincidence detectors), but they can be trained through regular gradient descent and are therefore much easier to implement. Spike-based models theoretically represent a more powerful information encoding, since firing rate and time to spike are implicit information, but their binary nature has long prevented their application for real-world tasks.

2.3.3 Learning Mechanisms for SNNs

Of course, neuron models and data representation are worthless on their own, since algorithms must be able to learn to perform relevant tasks. As far as spiking neural networks are concerned, the inherent limitation of neuronal activation, i.e. a Heaviside function with infinite derivative at zero and null derivative elsewhere, has prevented them from integrating into standard deep-learning libraries exploiting automatic differentiation (see Equation 2.5, showing the firing of a neuron as a function of the input spike train S_i, the synaptic connections w_i and the neuron's own firing threshold assuming a zero membrane potential before the input spike train's arrival). SNNs have therefore traditionally been trained using Hebbian learning rules [91], succinctly resumed in the now famous expression *those who fire together wire together*. This learning rule, combined with the complementary statement *those who fire out of sync, lose their link* led to Spike-Timing-Dependent Plasticity (STDP) [92], a biologically-plausible learning mechanism consisting of excitation or inhibiting synapses depending on the relative temporal offset of pre- and post-synaptic spikes. This method has been studied in depth over the years, and also some promising and interesting results have been obtained with it (e.g. [93]).Otherwise, spiking neural networks were trained either by adapting a pre-existing ANN [94] or by using a proxy ANN [95] as the learning mechanism, often with lackluster results. It would be in 2019 when the popularity of SNNs would explode thanks to the apparition of surrogate gradient learning (SGL) [85]. In a nutshell, the method consists of using a proxy function when computing the gradients for the backward pass (see Figure 2.5): the forward pass is performed using a Heaviside step activation, but the backward pass pretends that a sigmoid-like function has been used, utilizing its gradient to keep information flowing through the network. This approximation allows spiking neurons to be integrated into standard deep-learning libraries, and has allowed them to reach state-of-the-art results on a variety of tasks. Since then, many libraries have been developed proposing a variety of neuron models for deep learning tasks (e.g. Spikingjelly [96], Norse [97] or snnTorch [98], all of them with direct PyTorch integration), and many results have appeared showing the advantages SNNs may present when compared to their analogical counterparts.

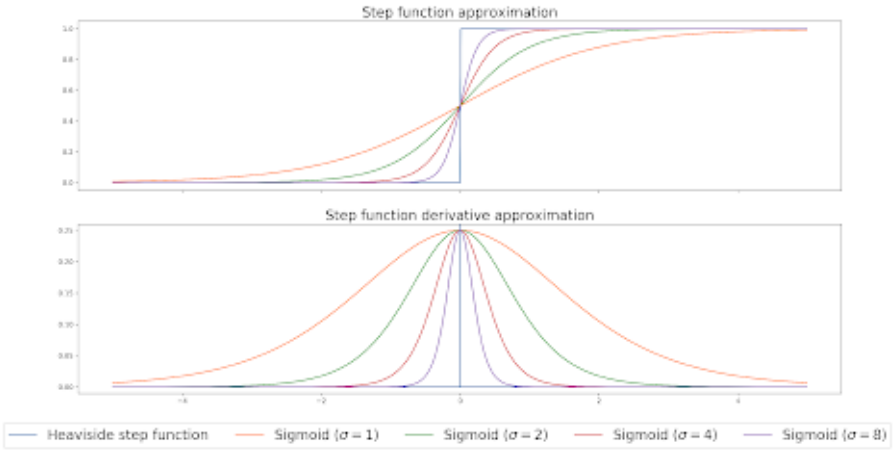


Figure 2.5: Surrogate gradient using a sigmoid function (borrowed from [3])

2.3.4 Comparison with Standard ANNs

There are some key differences that set spiking neural networks apart from their analogical counterparts, the main ones being:

- Unlike ANNs, which always forward information after the activation function (even if that information be zero), information in SNNs should be understood as a binary process, the lack of spikes therefore representing lack of information (either because of irrelevance, or because of too poor excitation). This spike-driven nature of the data translates into sparse tensors within the models. In fact, sparsity itself can be enforced within the models with loss functions, therefore achieving ultra-light information communication within the network.

- When no data is being fed to the model, no processing of information is being performed at all, and therefore there is no associated energy cost.

- SNNs can be implemented on neuromorphic hardware, where sparsity can be directly implemented and exploited for lower energy consumption. In addition, neuromorphic chips work in almost-real time (contrarily to GPUs, working in discrete time), thus providing a higher temporal resolution and finesse.

- The inner potential of the neurons, in addition to their real-time information processing these networks achieve when deployed on dedicated hardware, mean that SNNs are tailored to processing natural sensory information, which can be understood as a continuous time series.

2.3.5 Application of SNNs

Due to their intrinsic temporal nature (i.e. intrinsic recurrence and memory thanks to the membrane potential), SNNs have been extensively researched for temporal tasks. In the computer vision field, SNNs have been widely used in a variety of classification tasks, with gesture recognition being the first challenge the community tackled, and still one of the most popular tasks to date (e.g. [99]). As far as regression algorithms are concerned, the prevalent task has been optical flow estimation either with hybrid [100, 101] or with fully-spiking architectures [102, 103, 104, 105], although other tasks have also been tackled (e.g. depth estimation from stereo event data using SNNs [9]). Spiking neural networks are starting to be successfully implemented in real-world applications, e.g. an energy-efficient SNN embarked on a drone and deployed on a Loihi chip, capable of estimating the drone's egomotion from event data and commanding it [106]. Additionally, other lines of research have focused on spiking architectures itself, often taking energy-efficiency constraints into consideration. These publications have either focused on network models as a whole [107] or on key network blocks, e.g. spiking residual blocks [108]. Finally, some methods have proposed adapting state-of-the-art memory units to spiking architectures, like the SpikGRU block proposed in [109].

Of course, the main driving motivation in SNN research is their energy gains when deployed on dedicated hardware, as opposed to the power-intensive GPUs used with standard ANNs. Many neuromorphic chips have thus appeared over recent years, from custom FPGAs developed in academic circles (e.g. [110]) to industrial solutions, such as the *Loihi* chip by Intel [111] or the *TrueNorth* chip by IBM [112]. The drive to exploit neuromorphic hardware has even led to the appearance of non-spiking hardware-friendly solutions that can be deployed on neuromorphic chips. Such is the case of Sigma-Delta networks [113], which have already been deployed on the *Loihi 2* chip for audio and video processing [114] and for

noise suppression [115].

Finally, a great part of research on spiking neural networks has focused in energy efficiency, since it is probably their main advantage when compared to their analogical counterparts. This research has mostly focused on sparsity as an indicator of energy efficiency: less spikes translate into less information being forwarded, and therefore into less neurons spiking at post-synaptic layers, thus making a more energy-efficient model overall. Nevertheless, this research has often been misleading, First of all, to take advantage of sparsity for energy efficiency, SNNs must be implemented on dedicated hardware. Indeed, implementing SNNs on GPUs does not improve energy efficiency even if sparse data is being forwarded, since the number of FLOPs does not change in function of the data being propagated. Meanwhile, on neuromorphic chips, the number of operations is instead measured in Multiply-accumulate (MAC) and Accumulate (AC) operations, which are spike-driven and can thus exploit sparsity for energy efficiency.[116, 117]. In fact, implementing SNNs on GPUs is more computationally costly than using an equivalent ANN, due to the increased memory cost associated to storing the membrane potentials, as well as the additional operations induced by the membrane leak and reset. As a result, SNNs should always be developed with their associated hardware in mind in order to claim energy-efficiency as one of their advantages over standard ANNs, and non-implementable operations (e.g. bilinear upsampling or in-network real-valued information) should therefore be avoided. Secondly, sparsity alone is not enough for a spiking neural network to achieve competitive energy savings, and even more so when considering that it is often associated to a decrease in performance. It has been demonstrated that energy efficiency is directly linked with memory access [118], where ANNs often outperform spiking models even when the latter are implemented on dedicated hardware. Consequently, although sparsity is a key player to achieve better energy efficiency, optimizing memory loss within the model (understood as the combination of hardware and software) becomes a necessity to further push the competitive advantage that SNNs can provide regarding energy consumption. Lastly, weight quantization [119], and even weight binarization [120] have been proposed as a way to tackle all of the limitations preventing SNNs from consistently achieving competitive results in power consumption: on the one hand, integer weights within a network allow it to modify the costlier MAC operations into successive AC operations, on the other hand, integer weights require less

bits to be stored than floating-point numbers, hence enabling better data transmission and easier memory access.

Chapter 3

Optical flow estimation from event-based cameras and spiking neural networks

The research results of this chapter have been published in CUADRADO, J., RANÇON, U., COTTEREAU, B. R., BARRANCO, F., and MASQUELIER, T. (2023). Optical flow estimation from event-based cameras and spiking neural networks. Frontiers in Neuroscience, 17, 1160034. [7] The code of the published model can be found on `https://github.com/J-Cuadrado/OF_EV_SNN`

In this chapter, we will present our results on optical flow estimation from event cameras using spiking neural networks, from the earliest stages of the project to the final model published in [7].

3.1 Materials and Methods

3.1.1 Project Overview

The first idea that we explored as a potentially feasible spiking optical flow estimator was a U-Net [121] inspired stateful model using the MVSEC dataset [8]. The main motivations behind these decisions were:

- The U-Net architecture is a well-established model within the literature for computer vision tasks, due to its feature-extraction capabilities.

In fact, the encoder half of the architecture may have a working mechanism not very different from the visual cortex's early stages, where higher-order visual features are extracted as we progress deeper into the system. In addition, this architecture has been extensively studied for similar tasks (e.g. [122, 102] to cite some of the most relevant publications). This model can theoretically achieve good results without exploding in trainable parameters (thanks to the use of simple convolutional layers) when compared to more sophisticated alternatives, which makes it also a desirable architecture for light-weight applications. Finally, this architecture can be turned into a neuromorphic-hardware-friendly version with very minor modifications, which lines up perfectly with the reasons encouraging the use of SNNs instead of their analogical counterparts.

- Spiking neurons are stateful neurons by default, since their inner membrane potential remains over time (albeit their leak value). Since optical flow is *a priori* a highly-temporal task, it becomes natural to think that sequentially treating data for virtually real-time optical flow estimation is an interesting route to pursue. Indeed, the influence of distant events in time would eventually be forgotten thanks to the membrane potential's leak, while more recent events would have a greater effect on the neuron potential's evolution. The network would therefore be trained by successively being fed event histograms (more details in Subsection 3.1.3), and the backwards propagation of a supervised loss via truncated back-propagation through time (modified from [123]). We will resort to the Spikingjelly [96] Python library for spiking neuron implementation and training, since it can be fully integrated with the PyTorch library and its automatic differentiation capabilities.

- As of the starting time of the project, the Multi Vehicle Stereo Event Camera dataset was incontestably the go-to dataset for depth and optical flow estimation from event data, and was therefore the chosen dataset to train and test our models without much hesitation.

The first draft of the project therefore consisted of sequential optical flow estimation, from event data, using a stateful spiking network. We will present the different parts of the project, as well as the decisions that were made, until the final iteration of the network was achieved.

3.1.2 Training Dataset

The first choice (and quite a relevant one) we were confronted with when starting to develop our model was the selection of a training dataset. However, it was a simple one, since at the time the only option for real-world scenarios was the Multi-Vehicle Stereo Event Camera dataset. While it had many advantages, i.e. the variety of different sequences and a great amount of labels, it was also limited in its quality, specially because of the optical flow ground-truth generation. It is important to note that optical flow is not a measurable magnitude, and it must instead be inferred from a combination of ego-motion (e.g. IMU Data) and depth data (obtained in this case from LIDAR measurements). This situation means that no optical flow ground-truth data is available in pixels where no LIDAR measurements are available. For the most part those pixels are related to near-infinite depths (e.g. sky or far horizon) where optical flow values are negligible, but the point-cloud data obtained from LIDAR also presents some holes where optical flow value is simply unknown. These pixels' optical flow value is set by default to NaN in the ground-truth, which is incompatible with our supervised learning procedure (more details in Subsection 3.1.6). This situation becomes particularly problematic when the vehicle changes course and starts moving backwards, since during a small time window the scene remains still, showing not only a lack of optical flow, but also a lack of events. We explored some potential solutions to overcome this difficulty:

- Our first idea was to set all of the problematic pixels' values to zero, which would allow us to train our model in a supervised way because all pixels would have a numeric optical flow value. Intuitively, this idea made sense because most of the NaN values come from the LIDAR not being able to measure depth, e.g. far away points such as the sky where the optical flow value is negligible. However, due to the point-cloud nature of LIDAR measurements, some of the NaN values are holes in the ground-truth where optical flow does exist, and setting these pixels' optical flow value to zero introduces discontinuities in the flow field that translate into poor learning.

- Next, we tried to fill the problematic holes in the ground-truth, so we could learn a more realistic optical flow map where zero-valued pixels would actually correspond to lack of luminance variations in the scene. To do so, we resorted to image area closing (see [124]

for a more in-depth explanation on mathematical transformations for image processing), a well-known technique for image smoothness which can be easily implemented using a variety of Python libraries (e.g. the `kornia.morphology.closing` method from the Kornia library or the `skimage.morphology.area_closing` method from the scikit-image library). While we expected this transformation to improve our accuracy, the effect was rather the opposite. Indeed, morphological closing has the side effect of blurring edge lines, making it impossible to predict sharp optical flow variations. Furthermore, since the holes in the ground truth have different sizes, it is very hard to account for all of them, either keeping some of them (and therefore making the use of this technique futile) or completely distorting the ground-truth into a smooth flow field.

The fact that none of our ideas were working to improve learning on MVSEC, as well as the publication of the DSEC dataset [4], which was quickly establishing itself as the go-to dataset for event-based driving scenarios, motivated our decision to switch to this new dataset. Below are listed the main differences between the two datasets (a rough comparison can be found on Table 3.1):

- First, the image resolution of the DSEC dataset's event camera (Prophesee PPS3MVCD) is significantly larger than the event camera used in the MVSEC dataset (DAVIS 346B). While this impacts image sharpness, greater resolution also translates into greater dataset overall size (0.3Mp vs. 0.1Mp translates into tensors three times larger).

- The ground-truth frequency is also different between both datasets: 10Hz for DSEC vs. 20Hz for MVSEC. However, this frequency does not impact the real prediction rate, since it will rather be imposed by the input events and how they are fed to the network (more details in Subsection 3.1.3). Moreover, the pre-treatment of the ground-truth performed in the DSEC dataset yields higher quality data, most notably in the lower presence of noise within the flow maps.

- In addition to providing optical flow maps to be used as ground-truth, DSEC also provides its users with ground-truth masks, i.e. pixels were the optical flow value is accurate. It is also only on those pixels that metrics are evaluated on their official, independent benchmark. While

	MVSEC		**DSEC**	
Image Size	260x346 px		480x640px	
Ground-truth Frequency	20Hz		10Hz	
	Num. Labels	Size (GB)	Num. Labels	Size(GB)
Outdoor Driving Day	18300	62.2	8173	128.7
Outdoor Driving Night	18240	59.6		
Indoor Flying	5360	15.2	-	-
Motorbike*	30000	42.7	-	-

Table 3.1: Dataset Comparison
(*Motorbike LIDAR data is not available on MVSEC dataset)

a similar approach can be used with MVSEC (creating a mask only where optical flow is known within the scene), it is a more lacking solution in this second case, since data may still be noisy. Still, it is the best solution we found, and the one we adopted when training on this dataset.

- The MVSEC dataset is richer than DSEC as far as the nature of its sequences is concerned, presenting not only car driving scenarios but also motorbike and even indoor flying. Moreover, there are frequent vehicle stops during the sequences of the MVSEC dataset that the DSEC dataset lacks.

- While not applicable for the task at hand, it is also interesting to point out that the MVSEC dataset provides the grayscale images of its sequences (obtained jointly with the event data by the DAVIS 346B camera), whereas the DSEC dataset provides RGB images (the use of these images further impacts the dataset size, but it is not taken into consideration for Table 3.1 since they are not used in the current study) taken from FLIR Blackfly S USB3 cameras facing forward in parallel to the event sensors.

To sum up, we chose the DSEC dataset as our training dataset due to its greater ground-truth quality, as well as an external benchmark that allows for fairer comparisons among models. To be able to compare ourselves with

pre-existing methods, though, we have also trained and evaluated our model on the MVSEC dataset, both on driving and on indoor-flying scenarios.

3.1.3 Input Event Representation

Having set on a training dataset, the next step to take was figuring out a way of feeding the event data into an artificial neural network. To achieve this, the first thing to do is understanding the output information that an event camera provides. As we have already explained in Subsection 2.2, event cameras trigger asynchronous events on pixels where the variation of the log luminance reaches a certain threshold. Each event e_i is usually presented in the following form:

$$e_i = (x_i, y_i, t_i, p_i) \tag{3.1}$$

- x_i and y_i are the *spatial* coordinates of the pixel where the event has been triggered (let us not forget that pixels in event cameras operate asynchronously)
- t_i is the event's timestamp, i.e., the moment when the event triggered. This parameter is in direct link with the camera's temporal resolution, which is usually in the range of the microsecond.
- Finally, p_i corresponds to the event's polarity, i.e., whether the event represents an increase ($p_i = +1$) or a decrease ($p_i = -1$) in the log luminance at location (x_i, y_i).

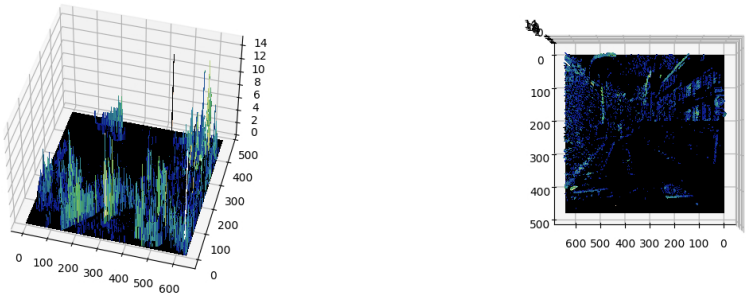
The information an event camera provides is therefore not an image-like tensor, but rather a sorted list of consecutive independent events. However, a discretization is needed in order to train a deep-learning model on a GPU. We decided to accumulate all of the events taking place within a given time window in an image-like input that would then be treated by our CNN, and we decided to call these tensors *event histograms*. We achieve this by creating a two-channel tensor (one channel for each polarity) of width W and height H equal to the event camera resolution. Then, we add 1 on the first channel at each pixel for each time a positive event has been triggered during the frame's time window in that pixel, and we perform the same operation on the second channel for each negative event. A pseudo-code example of the frame creation can be found in Algorithm 1. We decided to call these frames *event histograms* because they represent a two-dimensional,

two-channel event distribution within the visual scene, the same way a standard histogram would represent a data distribution in a simple plot. This approach to represent input event data, while extensively used within the literature (see Chapter 2 for some examples), is not the only possible option. We would also like to mention the input representation adopted in [125], where input events within the frame duration are binarily encoded, i.e., each pixel contains information about whether or not events occurred in it during the time window, but not about how many of them took place. While *a priori* more hardware-friendly than our integer encoding approach, it is in fact not needed to go this far, since many neuromorphic chips can handle integer information. Furthermore, we believe that the event count can be crucial information when estimating optical flow due to its temporal nature, since collapsing this information to binary frames would effective translate into severely hindering the temporal resolution of event cameras, thus questioning the pertinence of their use in such a task. Finally, there may be an argument for an event-driven approach, but it is not possible to pursue this approach on a GPU, and it would instead require on-chip learning in a dedicated device.

Algorithm 1 Event Histogram Creation

Inputs: $event\_list$, t_start, Δt
$event\_histogram \leftarrow zeros(C = 2, H, W)$
for $event$ **in** $event\_list$ **do**
    $(x, y, t, p) \leftarrow event$
    if $t \geq t\_start$ **and** $t < t\_start + \Delta t$ **then**
        if $p == +1$ **then**
            $event\_histogram[0, y, x] \leftarrow event\_histogram[0, y, x] + 1$
        else
            $event\_histogram[1, y, x] \leftarrow event\_histogram[1, y, x] + 1$
        end if
    end if
end for
Return: $event\_histogram$

The result of this process can be seen in Figure 3.1, where we have plotted one channel of an event histogram both in perspective (where the histogram-like nature of the tensor can be seen, with peaks representing



(a) Event accumulation at each pixel for a given time interval.

(b) Top view of the corresponding event frame.

Figure 3.1: Example of an input frame for 1 polarity.

event concentration) and in top view. Furthermore, we can see in the top view that events are heavily linked to contours (e.g. the windows on a building on the right side, or the zebra crossing on the bottom section of the scene), while regions with almost-constant luminance values (e.g. the sky or the pavement of the road) barely trigger events.

3.1.4 Network Architecture

Once the training dataset was chosen, we proceeded to develop our model's architecture. As we explained in Subsection 3.1.1, our first approach consisted of sequentially feeding event histograms to a U-Net-like network to statefully estimate optical flow. We would resort to leaky integrate-and-fire (LIF) neurons as our spiking units. This model would in theory be able to integrate temporal information thanks to the spiking neurons' membrane potential, while irrelevant information would be forgotten through the neurons' leak. The model would be trained with supervised learning using truncated back-propagation through time (TBPTT) and surrogate gradient learning (SGL). Optical flow would be represented by the inner potential of a final nonspiking neuron pool (leaky integrators with an infinite firing threshold). Finally, we would use separable convolutions [126, 127] everywhere in the model to ensure the lowest possible number of parameters,

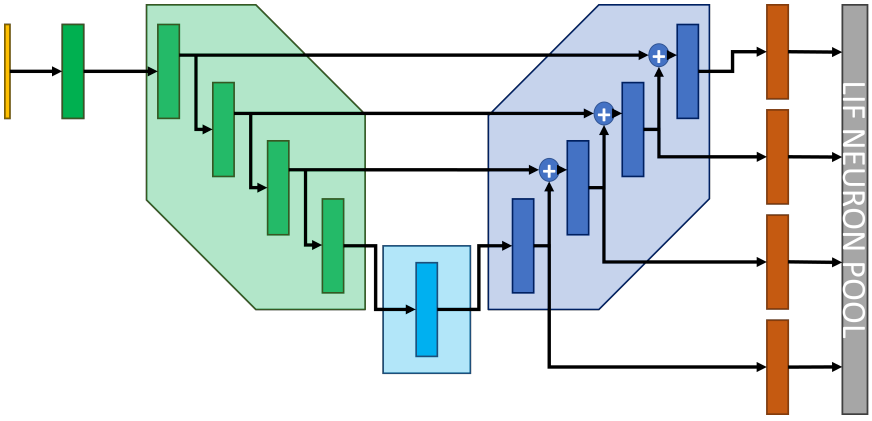


Figure 3.2: Network architecture: first model (draft)

speeding up training and improving generalization in the process. An example of our first architecture model can be found in Figure 3.2. The model, while heavily inspired by the EV-Flownet [122] architecture, presents one major difference: intermediate, lower-scale predictions are absent in our version of the network. This decision was made in order to remain as hardware-friendly as possible, since re-injecting intermediate predictions is not a spike-friendly operation supported by neuromorphic chips.

The main difference between our approach and most of the existing spiking models for optical flow estimation, either analog or spiking, is its hardware-friendliness. Indeed, to ensure that our model could one day be implemented on neuromorphic chips, all of the information within the network is encoded in binary tensors, and all of the input information is integer-natured. This means that we cannot resort to some of the techniques that have been used in the literature (e.g. feeding the last event timestamp at each pixel to the model for more informative input tensors, or re-injecting lower-scale predictions at the decoder for improved resolution), instead focusing our efforts on a fully-spiking, completely binary model that would exploit the temporal information of event data contained in events alone for accurate optical flow estimation.

The aforementioned approach made sense at first, since we believed temporal information to be key in optical flow estimation (as we would eventually demonstrate). Furthermore, lack of motion would translate

into absence of events, which would in turn naturally decrease the model's membrane potentials to zero provided that no biases were present within the model (achieved by removing the biases present in convolutional layers, as well as removing all instances of batch normalization within the model). However, this method never managed to produce sufficiently good results, and mostly converged to noisy predictions. We have long theorized about the possible reasons behind this phenomenon, and the conclusions we have reached are:

- First, in order to make TBPTT work as a training mechanism, we need to initialize the network to a point where transient optical flow values due to network *warming-up* have been fully forgotten. Furthermore, during training, we would like to use long sequences where both long- and short-term effects could be captured. However, since the computational graph has to be unrolled after the series of forward passes to perform back-propagation, the process soon becomes too costly for our hardware to handle.
- Second, the network's dynamics (represented by the different layers' leak factors) play a major role in accurate optical flow estimation. This situation is even more true for the very last neuron pool, responsible of optical flow predictions. The simplest iteration of the model would have constant leak factors within each layer, and they would be shared by all of the neurons present in a layer. Nevertheless, this is a very strict approach to learn optical flow dynamics. We therefore switched our spiking units into Parametric LIF neurons (PLIF, leaky integrate-and-fire spiking neurons with a learnable time constant). In this case, the leak factor of each layer is still shared by all of the neurons within the layer, but it is not fixed from the beginning: rather, it is initialized to a given factor, and learnt during training. Using these neurons, though, caused the time constants to explode, completely removing the leak and turning the spiking units into almost-perfect integrators. In the end, we believe that both of these approaches were lacking, and that shared time constants within a layer is too strict of a constraint. Indeed, the dynamics of optical flow being subtle, a more robust model would probably consist of heterogeneous leak factors, since the speed at which flow values change is usually larger at the scene's edges than in the center section. These time constants should

ideally be input-dependent, taking into account additional information such as event concentration, focus of motion, etc. (not explicitly in the input data, but reasonably easy for the model to extract). This approach is nonetheless too complex to implement on current neuromorphic devices, and we therefore decided to abandon it in order to stay as close to a real implementation as possible.

- Finally, the difficulty of generating such a high number of flow estimations at each timestep has to be acknowledged, and our approach was probably too naive to capture so many event interactions with such a simple model.

The task at hand having proven too ambitious for our set constraints, we decided to take a step back and simplify both the model and its learning. Using the knowledge we had acquired while developing *Stereospike*[9], we decided to try this approach for optical flow estimation. Our model would therefore switch into a stateless network where a full reset of the membrane potentials would be performed after each forward pass. This modification effectively means that our SNN is equivalent to a standard ANN where the non-linearities (usually represented by Rectified Linear Units) have been replaced by Heaviside step activation functions. While some may argue that this decision pulls us away from spiking neural networks, we strongly disagree, since our main focus has always been to develop a hardware-friendly algorithm, and this approach respects the constraints imposed by neuromorphic hardware. Moreover, both the leak and the reset have an energy cost for these chips, so depending on the targeted device the decision may be justified from an energy efficiency point of view. This approach presents a major limitation when compared to the previous one, though, because temporal context is lost as only one frame is fed at a time to the model. To overcome this situation, we decided to concatenate a series of event frames along the channel dimension, therefore obtaining an input tensor which consists of a suite of consecutive event histograms (this approach has also already been used in the literature, e.g. [125]). The model would thus consist of a 2-dimensional CNN estimating optical flow from a chunk of consecutive event frames.

We were optimistic about this approach, but in the end it did not pay off. We believe that the main issue with this approach was relegating temporal information to the very first convolutional layers, i.e., the layers

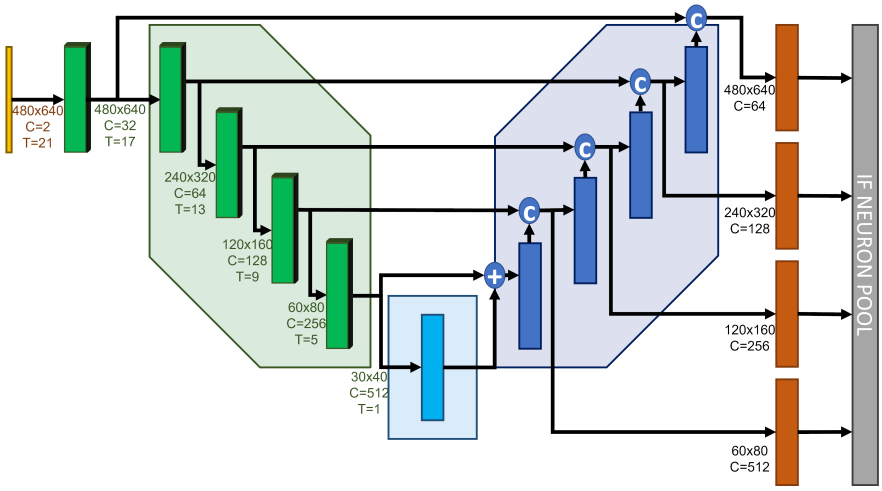


Figure 3.3: Network architecture: final model

representing the lowest number of parameters within the network. Due to the way convolutions work, temporal information would be shuffled after the first encoding step, and only the first layer would see "raw" continuity information. Since zero-input was now virtually impossible for our model, we tried including batch normalization in the architecture, due to its widespread use within the literature (even found to be crucial for certain tasks [125]), but the final result of this version of the network consisted of noisy predictions lacking any structure whatsoever. It became evident that, if we wanted to succeed in our task, we would need to be able to keep temporal information for as long as possible, giving the model time to correctly extract and interpret it.

The final iteration of the model, and the one that managed to achieve competitive results on both datasets, consists of a variation on the previous model, trying to incorporate temporal context into a stateless model. Inspired by Temporal Convolutional Networks (TCN) [128, 129], we tried to adapt their approach for a two-dimensional input (i.e. our event histograms). The main goal was to increase the network's bottleneck temporal effective receptive field [130] while keeping the temporal continuity along the different encoding stages.

Our final architecture can be found in Figure 3.3. It consists of a

U-Net-like architecture taking a three-dimensional tensor of shape *(B, C, T, H, W)* as its input, where:

- B represents the training batch size. In our experiments, we surprisingly found that B=1 yielded the best results for our method. While astonishing, it is the same result we obtained in [9] for depth estimation, and extensive ablation studies have led us to accept this batch size as the optimal solution.

- C is the number of input channels, one for each polarity. After a first encoding stage increases this dimension up to 32 channels while keeping the spatial resolution unchanged, each subsequent encoder doubles the number of channels until 512 channels are reached in the bottleneck.

- T represents the temporal dimension, i.e. the number of consecutive frames that the model is seeing at once. The original input consists of 21 consecutive event histograms of 9.1 milliseconds each (the optimal duration and number of frames we found, as we will present in Section 3.2). Each encoding stage decreases this dimension until it collapses to 1 right before the bottleneck (Figure 3.4 illustrates the temporal ERF evolution across the encoding stages). It is important to note that the passage into 3-dimensional convolutions is not "free", since the network requires longer training times. Moreover, while latency is not hindered because event histograms can be fed in a *sliding window* fashion into the final model, the number of floating-point operations (FLOPS) does severely increase. Still, this is the line of research that provided the best results, and thus the one we decided to pursue.

- Finally, H and W represent the tensor spatial resolution. Each encoding stage halves each of these dimensions, and each decoder stage doubles them until reaching a full-resolution tensor in the final layer.

Delving more in-depth into our network, there are a few details that set it apart from the most standard architectures. The first main difference is found in the encoder blocks, because downsampling is performed via a strided maximum pooling layer instead of a strided convolution. Both approaches were compared (see Subsection 3.2.4), and this is in fact the approach that yielded the best results. We believe that this result is due to they way maximum pooling works, propagating information as long as one

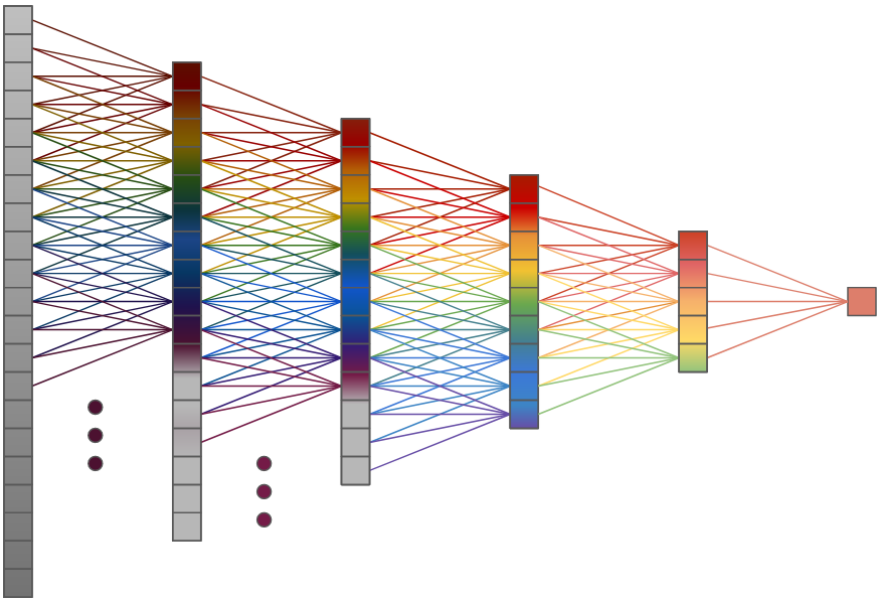
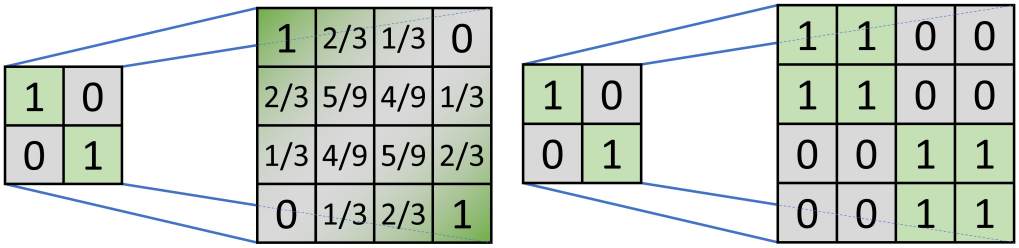


Figure 3.4: Temporal receptive field evolution along the network's encoder

single spike is present within the kernel's receptive field, therefore giving less importance to individual spikes in favor of broader spatial information. This method can also be implemented in neuromorphic hardware [131], and it is to the best of our knowledge the first time it has been used for dense regression with spiking neural networks. The other biggest difference we present against other similar architectures is nearest-neighbor upsampling in the decoder stages, opposed to bilinear upsampling (e.g. [102] or [122]). This choice is motivated by the fact that, unlike bilinear upsampling, nearest neighbor upsampling guarantees binary tensors after performing the upsampling operation (see Figure 3.5), thus being implementable on neuromorphic hardware. Finally, since our encoder is three-dimensional and our decoder is two-dimensional, tensor shapes do not match and regular skip connections are impossible. To overcome this issue, we only take the very last element along the T dimension, since it is the closest one to the present and therefore the most relevant one for optical flow estimation.

Lastly, there is an argument to be made about using more complex architectures, i.e. LSTMs or GRUs, or even attention-based models like transformers, since state-of-the-art publications usually include one or more



(a) Upsampled tensor via Bilinear Upsampling.

(b) Upsampled tensor via Nearest Neighbor Upsampling.

Figure 3.5: Upsampling techniques comparison.

of these blocks within their algorithms (see Section 3.2). While certainly interesting and intriguing, the problem with these blocks is that they are not implementable on neuromorphic hardware, and they were therefore out of the conversation from the very beginning given our set of constraints.

3.1.5 Spiking Neuron Model

Our initial approach to the problem led us to choose a LIF neuron as the desired spiking unit within out network, i.e. a memory unit whose membrane potential suffers an exponential decay over time (Equation 2.2) towards a certain resting potential (V_{rest}). Affected by the afferent connections, this inner potential can be increased (excitatory connections) or decreased (inhibitory connections). At some point, the potential will reach the neuron's firing threshold ($V_{threshold}$), causing the neuron to spike and resetting the membrane potential. The reset can be either a *hard reset*, i.e., setting the membrane potential to a reset value V_{rest}, or a *soft reset*, decreasing the membrane potential by a given delta.

As previously explained (see Subsection 3.1.4), our first attempts consisted of training a stateful SNN using truncated back-propagation through time. This approach would exploit the membrane leak as a forgetting mechanism for irrelevant, distant past information. Despite our best efforts, though, we were unable to achieve competitive results with this approach. Indeed, the noisy nature of event data greatly complicated

reaching competitive error metrics, and the high computation cost required by TBPTT was forcing us to choose a non-optimal compromise between frame duration and overall temporal context. We therefore decided to change our approach and exploit the insights we had gained while developing *StereoSpike* [9], using stateless units after each forward pass. Since the leak factor was no longer necessary, we switched the neuron model to the simpler integrate-and-fire units. While more limited in their performances, these neurons are also less computationally costly than their leaky counterparts, since the reset operation is easier to compute than the membrane update that the leak factor requires. Furthermore, temporal information has not been lost, since it has instead been transferred to an additional temporal dimension.

3.1.6 Supervised Learning Method

Loss Function

The final step towards developing our model was finding how to make it learn to predict optical flow, all while keeping a reasonable level of generalization. When measuring performance on optical flow estimation, there are many possible metrics to use [21], yet two of them stand out above the others: Average Endpoint Error (AEE, Equation 3.2) and Average Angular Error (AAE, Equation 3.3). In both equations, u represents the x-component of the optical flow, and v represents the y-component of the vector. In addition, $(\widetilde{u}, \widetilde{v})_{i,j}$ represents the estimated optical flow at location (i, j), and $(u, v)_{i,j}$ its ground-truth value.

$$AEE = \frac{1}{n} \sum_{i,j} \sqrt{(\widetilde{u}_{i,j} - u_{i,j})^2 + (\widetilde{v}_{i,j} - v_{i,j})^2} \tag{3.2}$$

$$AAE = \frac{1}{n} \sum_{i,j} arccos \left(\frac{\widetilde{u}_{i,j} \cdot u_{i,j} + \widetilde{v}_{i,j} \cdot v_{i,j}}{\sqrt{\widetilde{u}_{i,j}^2 + \widetilde{v}_{i,j}^2} \cdot \sqrt{u_{i,j}^2 + v_{i,j}^2}} \right) \tag{3.3}$$

- Average Endpoint Error represents the average of the error vector's modulo among all pixels.
- Average Angular Error represents the average angular offset between the estimated optical flow vector and its associated ground-truth. It

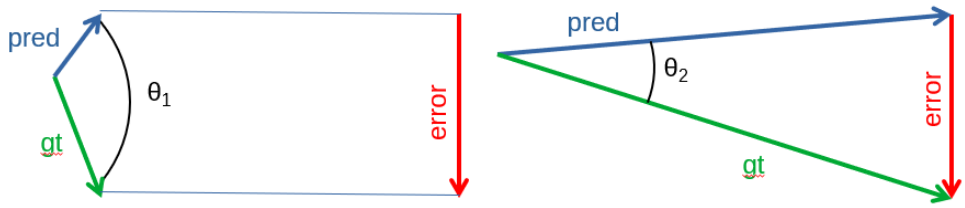


Figure 3.6: AEE vs. AAE: both cases have the same AEE, but the angular error is not the same.

is associated with the estimation's quality, since it account for the alignment between estimation and ground-truth.

Among the two of them, AEE is stronger as a metric, since a zero-valued AEE (i.e. a perfect estimation) would also translate into a zero-valued AAE, whereas the opposite is not truth. We therefore started by adopting it as our base loss function. Nevertheless, we soon found that it was limited in its performance as a loss function, since it naturally prioritizes pixels with a high-valued ground truth over the rest. This phenomenon can be easily seen in Figure 3.6, where two instances of the same AEE yield drastically different AAE. It became clear to us that, if we wanted to improve our predictions, we would have to ensure that the model was not only learning to decrease the absolute error between prediction and label, but also explicitly aligning both vectors.

The first intuition we had to align ground-truth and estimations was switching the AEE loss to an Average Relative Error (ARE), described in Equation 3.4. This function would account for the differences in absolute optical flow value by normalizing pixel-wise L2 error by the ground-truth's modulo (plus a small ϵ factor to ensure nonzero values in the denominator). Small errors in pixels with low-valued ground-truth would therefore be magnified, and the opposite would happen for high-valued ground-truth pixels.

$$ARE = \frac{1}{n}\sum_{i,j}\frac{\sqrt{(\widetilde{u}_{i,j}-u_{i,j})^2+(\widetilde{v}_{i,j}-v_{i,j})^2}}{\sqrt{u_{i,j}^2+v_{i,j}^2}+\epsilon} \tag{3.4}$$

Although we were optimistic about our new loss function, our efforts

did not pay off. In fact, this loss function led to noisy, unstable training results, where learning was severely impacted and only random predictions were obtained. We therefore decided to take a more direct approach and train our model with a two-term loss function, comprising a combination of the AEE and the AAE metrics. While the inclusion of the first term was straightforward, the angular term presented some challenges. First, if either the prediction or the ground-truth showed a zero value, there would be no way of computing the angle's cosine. To overcome this issue, we included a small positive ϵ in the cosine computation. Second, in order to calculate the angle, we needed to make sure that the cosine belonged to the interval $[-1, +1]$, but since the derivative of the arc-cosine function is infinite at its bounds, we decided to clamp its values in an approximation of the $]-1, +1[$ interval, that we achieved by offsetting the bounds by the same ϵ we had previously used. A draft of the angular loss function's algorithm can be seen in Algorithm 2.

Algorithm 2 Angular loss function

1: **Inputs**: $prediction,\ label,\ mask,\ \epsilon = 1e-5$

2: $n\_pixels = \sum mask$

3: $pred\_mod = \sqrt{prediction_x^2 + prediction_y^2}$
4: $label\_mod = \sqrt{label_x^2 + label_y^2}$

5: $dot\_product = prediction_x \cdot label_x + prediction_y \cdot label_y$

6: $cosine = \dfrac{dot\_product + \epsilon}{pred\_mod \cdot label\_mod + \epsilon}$
7: $cosine \leftarrow clamp(cosine, -1 + \epsilon, +1 - \epsilon)$

8: **Return**: $\dfrac{1}{n\_pixels} \sum [arccos(cosine) \cdot mask]$

Finally, when talking about optical flow estimation from event data, it is worth talking about photometric and smoothness losses, used for unsupervised learning (e.g. [102]).

- As far as photometric loss is concerned for event-based optical flow

estimation, it is used for accurate predictions wherever events have been triggered during the input event histogram time window, thus yielding sparse flow maps, and therefore not applicable for our task.

- Regarding the smoothness loss, we explored its inclusion, but the results we obtained consisted mostly of smooth maps with either fuzzy contours or lack of contours altogether. Despite our best effort, we did not manage to find a good trade-off between loss contribution and image sharpness, and we ultimately decided to abandon it for our model.

Data Augmentation

In addition to the supervised loss functions, we turned our attention to data augmentation, trying to prevent as much as possible the overfitting problem, inherent to supervised learning. We explored several options for event data augmentation, with varying degrees of success. The pseudocode for all of the different transformations we tested can be found in Appendix A.

- Random horizontal flip (Algorithm 3) is perhaps the easiest and most intuitive transformation we can perform on our data, since it generates ecological samples that keep the data structure. Its inclusion in the data augmentation pipeline translated into a significant improvement on the network's generalization capabilities for the task at hand.

- Random vertical flip (Algorithm 4) is also another widespread data augmentation technique in computer vision, with existing examples in the literature applying it to optical flow prediction (e.g. [102]). Nevertheless, it did prove detrimental for our model, since significantly changing the spatial data distribution while training led to poor evaluation metrics.

- Random rotation (Algorithm 5) was developed in parallel to our vertical flip code as yet another way to augment data. In the end, though, it was also abandoned, since it did not only present the same issues as the random vertical flips, but also modified the tensors' aspect ratio during training.

- Random event drop (Algorithm 6) was conceived due to the noisy nature of event data, since slight modifications to the input tensors may lead to more robust estimations. Random patches (Algorithm 7) follows the same philosophy, this time serving a twofold purpose. On the one hand, this method increase the likelihood of masking an area comprising events (event tensors presenting a certain level of data sparsity, random event drops often affect empty pixels and therefore have no effect). On the other hand, it allow us to simulate potential temporary occlusions to the event camera, which should in turn make the algorithm more robust to these situations. Both models slightly increased the overall performance, but the trade-off between improvement vs. increase in computational cost was not justifiable, and we therefore decided to not include them moving on with the project.

- Temporal mirroring (Algorithm 8) is the most complex of all the data augmentation techniques, since it does take into account the vector-like nature of the ground-truth in addition to the event camera's motion (impacting the event polarities). However, it is such a widespread technique that some datasets (e.g. DSEC) event provide the users with backwards data. This technique did not improve our test metrics either, but we are convinced that this is due to the datasets being heavily biased towards forward motion, and we remain positive that it is a transformation to include when working on more challenging and diverse datasets.

In the end, only random horizontal flip was kept as a data augmentation technique. Nevertheless, we are convinced that some of our methods may indeed be beneficial when training on a more challenging dataset with a wider variety of motion patterns.

3.1.7 Training Procedure and Technical Details

All of our calculations have been performed on either NVIDIA A40 GPUs belonging to the CerCo laboratory, or in Tesla V100-SXM2-16GB GPUs belonging to the French regional public supercomputer CALMIP, owned by the Occitanie region and that we were allowed to use under the allocation 2022-p22020.

Regarding our training samples on DSEC, we performed the following data division:

- Trainin split:
  - zurich_city_01_a
  - zurich_city_02_a
  - zurich_city_02_c
  - zurich_city_02_e
  - zurich_city_05_a
  - zurich_city_05_b
  - zurich_city_06_a
  - zurich_city_07_a
  - zurich_city_09_a
  - zurich_city_10_a
  - zurich_city_10_b
  - zurich_city_11_a
  - zurich_city_11_c
- Validation split:
  - thun_00_a
  - zurich_city_02_d
  - zurich_city_03_a
  - zurich_city_08_a
  - zurich_city_11_b

This data split was performed in order to ensure that around 75% of the available data would be used during training, while the remaining 25% was used in evaluation. For the final submission to the DSEC test benchmark, all of the available samples were used in training. For training and evaluation on the MVSEC dataset, we used the data splits proposed in [132] and [133].

3.2 Results

Having presented the choices we had to make to achieve our final model, we will now proceed to show the different iterations we went through until reaching the final architecture. We will also demonstrate that our model reaches good levels of performance on both the DSEC and the MVSEC datasets, showing competitive results with a fraction of the number of parameters when compared to other existing methods in the literature. We will continue by presenting the ablation studies we performed in our model. Finally, we will briefly talk about potential simplifications we explored

for real-life implementation, targeting the SPLEAT neuromorphic chip in particular [110].

3.2.1 Model Optimization

Kernel Size

Ever since their introduction in the famous *Attention is all you need* paper [47], transformers established themselves as the go-to architecture for many deep-learning tasks, eventually taking over the field of computer vision as well. This was mainly due to two facts: first, unlike recurrent neural networks, they can be easily parallelized; second, their attention mechanism can capture distant dependencies, be it in time or space. Yet, during the past few years we have seen the rise in popularity of convolutional neural networks once again. Indeed, it has been shown that sufficiently large kernel sizes can manage to capture long-distance dependencies within a tensor (similarly to transformers), and competitive results have been achieved with kernel sizes as large as 31x31 [134] and even 51x51 [135] thanks to efficient kernel reparametrization. Furthermore, the development of new learning techniques such as dilated convolutions with learnable spacings [136] has allowed the development of ultra-large spatial kernels with a sparse weight distribution, whose spacings within the kernel are learned through back-propagation.

Seeking easiness of implementation, we decided to constrain our model to a standard CNN. Still, there was the question of which size the spatial kernel should have without hindering performance. After a thorough study, we concluded that a kernel size of 7x7 was optimal for our task at hand, since larger kernels simply increased learning time without significant accuracy improvements. This results is in line with the findings of [134], which shows that standard Python libraries struggle to optimize kernel sizes larger than seven, and it is therefore the size we decided to use in our model.

After finding the optimal spatial kernel size, we turned our attention to the temporal dimension. The convolution along the temporal axis can be seen as a learnable delay among the frames affected by the kernel, and finding the optimal kernel size was thus key to ensuring accurate temporal feature extraction. Since we wanted the temporal dimension to collapse to

Frame duration	AEE (px/s)	AAE (rad)
4.5ms	1.41	0.129
9ms	**1.10**	0.094
18ms	1.19	**0.087**

Table 3.2: Optimization results on event histogram duration

one when reaching the residual block (by using unstrided convolutions along the temporal dimension), larger kernel sizes naturally translated into longer input sequences (more input frames fed at once to the model), consequently representing a longer overall temporal context. We tested our model for kernel sizes of 3 (11 input histograms, 100ms of temporal context), 5 (21 input histograms, 192ms of temporal context) and 7 (31 input histograms, 282ms of temporal context). In the end, we found 5 to be the optimal kernel size along the temporal dimension, i.e. using event information further in the past than 192ms did not improve the evaluation metrics.

Temporal Context

Having found 21 input event histograms to be the optimal amount of frames to use for our chosen configuration, we proceeded to question ourselves on whether 9ms was the optimal histogram duration. A larger frame duration would translate into further-in-the-past temporal information, sacrificing temporal resolution in the process. On the contrary, shorter event histograms would contain finer temporal information, but this information would be constrained to a temporal context much closer to the present. To check the effect of input histogram timespan with a fixed number of input histograms (21, as found during our aforementioned study), we tested our base architecture (Figure 3.3) on frames comprising 4.5ms, 9ms and 18ms. The results we obtained for this study have been gathered on Table 3.2,

Our results show that 9ms is in fact the optimal frame duration for our architecture. Indeed, 18ms frames do manage to capture long-term dependencies along the temporal axis, but the lower temporal resolution does hinder the model's performance to some extent. On the other hand, finer input histograms do not manage to properly translate into accurate optical flow estimations due to the lack of overall temporal context, thus

Architecture	AEE (px/s)	AAE (rad)	Num. Params. (x 1e6)
1 res + sum	1.18	1.101	1.1
1 res + cat	**1.10**	**0.094**	1.2
2 res + sum	1.15	0.097	1.7
2 res + cat	1.16	1.109	1.8

Table 3.3: Optimization results on network architecture

showing the worst performance metrics.

Residual Blocks and Skip Connections

Finally, we questioned ourselves about the network architecture itself. Trying to stay as close to neuromorphic hardware as possible, the model had to be as light as possible to facilitate implementation. Among all of the blocks present in the network, the most expensive ones in terms of computational requirements were by far the residual blocks, since each one of them represented roughly 600.000 trainable parameters (i.e. 50% of the total parameters of the base model). Nevertheless, the standard practice within the literature is to include two residual blocks in U-Net-like architectures, so we decided to verify if the associated increase in the number of parameters translated into better predictions. In addition, we tested whether our model should concatenate tensors in its skip connections, or if simply summing them would be enough, since concatenation allows information to be treated separately, but at the cost of more trainable parameters. The results of all of these studies can be found in Table 3.3.

We have found that using one single residual block, in conjunction with concatenations in the skip connections, yields the best results for our combination of temporal context/histogram resolution. We believe that this result is justified by two reasons:

- As we have previously stated, concatenation allows the skip connection to spread information along channels, therefore having much richer information for the decoders to extract (potentially even treating channels separately if necessary).
- Integrating two residual blocks in the architecture seems to make

the model more prone to overfitting, showing worse results on the evaluation metrics.

3.2.2 Model Evaluation on the DSEC Dataset

Having set on a final architecture for our SNN optical flow estimator, we now proceeded to test it on the most relevant state-of-the-art datasets, both comprising ecological data recorded with real event cameras. The first dataset we tested our model on was the DSEC dataset [4], available at `https://dsec.ifi.uzh.ch/`. This dataset is becoming the go-to comparison benchmark for disparity and optical flow estimation, not only because of its superior ground-truth quality, but also because of its official benchmark, providing an equal playing field for all models alike. Thus, we decided to train our definitive architecture on all of the available data for a total of 100 epochs, and then submit our best result to the official benchmark. The results we obtained, as well as a comparison to other existing publications, can be found on Table 3.4. Please note that the DSEC benchmark is very dynamic, and it is possible to submit anonymously and remove the submissions at will. We have therefore only included in our comparison table the results that can be cited, but we invite the readers to consult `https://dsec.ifi.uzh.ch/uzh/dsec-flow-optical-flow-benchmark/` to see all the most up-to-date submissions.

Our results comfortably place us in the middle of the table, at a reasonable distance from the top performers, but also outclassing some of the most complex models, establishing our model as the best-performing spiking neural network on the task. Furthermore, we achieve this result not only through a hardware-friendly approach (potentially leading to significant energy savings when correctly deployed on a dedicated chip), but also at a fraction of our competitors' trainable parameters, making us the lightest model among all of the published alternatives.

Our results are also good as far as qualitative estimations are concerned, which is shown on Figure 3.7. This image contains three samples of optical flow estimations obtained with our model: in order, each one of them represents the optical flow ground-truth, the masked prediction (valid pixels only) and the unmasked estimation of the flow field. In addition, Figure 3.7d shows the colormap used for optical flow representation (the

Model	**AEE (px/s)**	**AAE (deg)**	**Num. Params (M)**
IDNet [81]	**0.72**	2.72	2.5
TMA [137]	0.74	**2.68**	6.9
E-Flowformer [62]	0.76	**2.68**	n/a
ADMFlow [63]	0.78	2.84	7.0
E-RAFT [138]	0.79	2.85	5.3
Ours	1.71	6.34	**1.2**
VSA-SM [139]	2.22	8.86	N/A
TamingCM [77]	2.33	10.56	31.4
MultiCM [78]	3.47	14.0	n/a
RTEF [140]	4.88	10.82	n/a

Table 3.4: Comparison with the state-of-the art, obtained from `https://dsec.ifi.uzh.ch/uzh/dsec-flow-optical-flow-benchmark/`

optical flow vectors are encoded as Lab images, where the luminance channel represents the absolute magnitude of the flow, and the a and b channels the x and y components respectively). It can be seen that our method achieves accurate predictions within the scene, with particular good results when estimating the focus of expansion. Furthermore, our method is able to identify artifacts within the scene (e.g. vertical posts on Figure 3.7a or traffic signals on Figure 3.7c). This final result was somewhat unexpected, since masked supervised training is more similar to learning an optical flow point-cloud than it is to learning an overall flow distribution, and it demonstrates the generalization capabilities of our model.

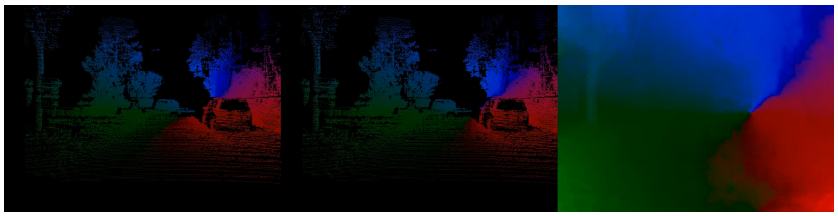
3.2.3 Model Evaluation on the MVSEC Dataset

For the sake of easiness of comparison with most of the existing models within the literature, in addition to analyzing our model's generalization capabilities, we also tested our architecture on the MVSEC dataset [8], available online at `https://daniilidis-group.github.io/mvsec/`. Using the data splits introduced in Subsection 3.1.7, we tested our model for indoor flying and outdoor driving scenarios.

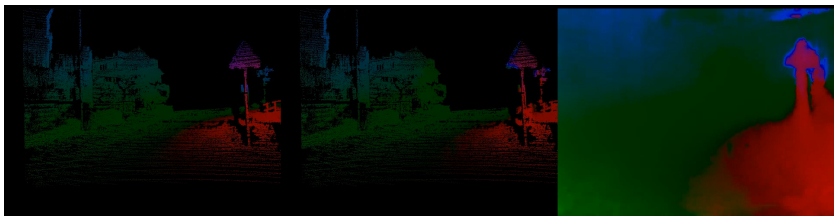
Wanting to keep our training philosophy, we were bound to develop a ground-truth mask akin to the one we used when training on the DSEC dataset. To achieve this, we only considered as valid pixels those showing an optical flow value greater than a threshold on both of their components. We set the threshold to $thr = 10^{-5}$ to ensure keeping as many pixels as possible. To keep our training pipeline consistent, we also evaluated the optical flow estimations on valid pixels only.

We started by testing our model on indoor flying sequences (Table 3.5). After several test, we found that the best results were obtained when the architecture was initialized to a set of weights previously learnt on DSEC. We therefore initialized the weights with a model obtained after training for 35 epochs on our DSEC train subsplit, and proceeded to optimize them on the indoor flying sequences of the MVSEC dataset. The results on Table 3.5 show that we achieve near-state-of-the-art performance (best result in bold, runner-up underlined), beating all but one of the previously published methods on two out of three splits and overall performance, and placing

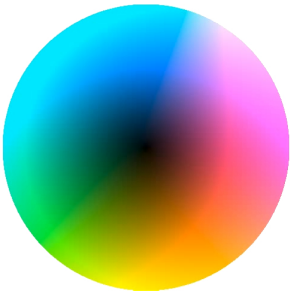
(a) Optical flow discontinuities due to vertical artifacts within the visual scene.



(b) The silhouette of the leftmost tree can be perceived on the unmasked optical flow map.



(c) Traffic signs clearly distinguishable on the right side



(d) Optic Flow Encoding

Figure 3.7: Example predictions of our best architecture on our validation set (ground-truth, estimation and masked estimation).

ourselves as the best spiking neural network for optical flow estimation on this task.

Next, we proceeded to evaluate our architecture on MVSEC's outdoor driving sequences, training on *outdoor_day_2* and testing on *outdoor_day_1*. Although we were optimistic after finding out our performance levels on the indoor flying sequences, our enthusiasm would soon be tempered:

- First, we struggled to stabilize the training, getting noisy results with no real structure for quite a long time. In fact, the training data itself consists only of a nine-minute sequence subject to high-frequency vibrations (see [122]), which translates into noisy data that greatly complicates learning.

- Second, unlike for the previous scenario, we found that network initialization was actually detrimental for the final result in this case, despite the sequences being apparently more similar in nature.

We believe that both of this issues are related to the noisy nature of the data in general, and to the presence of the car's dashboard in particular. Indeed, unlike the sequences in the DSEC dataset, the car's dashboard makes part of the visual scene in the MVSEC dataset. This part of the car being reflective, it triggers an enormous amount of events not linked to motion, and therefore being detrimental to optical flow estimation. Eventually, we managed to stabilize our training results by masking the corresponding section both in the input tensors and in the ground-truth. We are convinced that this manual fix should not raise major concerns, since the dashboard does not move relative to the event camera and therefore optical flow there should consistently be zero, but we acknowledge that even after this engineering solution our results for this scenario remain far from the top-performing publications in the literature.

While somewhat disappointing, the results on the MVSEC outdoor sequences are nevertheless not discouraging, because:

1. when optimizing our network architecture, and specially the spatial kernel size, we were targeting event histograms and optical flow predictions of 480x640 pixels of image resolution. Nevertheless, MVSEC presents a much lower-resolution ground-truth (only 260x346

Model	**Split 1**	**Split 2**	**Split 3**	**AEE sum**
EV-FlowNet [122]	1.03	1.72	1.53	4.28
Zhu et. al.[141]	0.58	1.02	0.87	2.47
Spike-FlowNet [100]	0.84	1.28	1.11	3.23
Back to Event Basics$_{Evf}$ [142]	0.79	1.40	1.18	3.37
Back to Event Basics$_{Fire}$ [142]	0.97	1.67	1.43	4.07
XLIF-EV-FlowNet [102]	0.73	1.45	1.17	3.35
XLIF-FireNet [102]	0.98	1.82	1.54	4.34
Orchard et. al. [143]	0.83	1.22	0.97	3.02
Fusion-FlowNet [144]	<u>0.56</u>	0.95	0.76	2.27
Adaptive-SpikeNet (ANN) [104]	0.84	1.59	1.36	3.79
Adaptive-SpikeNet (SNN) [104]	0.79	1.37	1.11	3.27
FSFN$_{FP}$ [145]	0.82	1.21	1.07	3.10
FSFN$_{HP-ADC}$ [145]	0.85	1.29	1.13	3.27
Shiba et. al. [78]	**0.42**	**0.60**	**0.50**	**1.52**
VSA-SM [139] (dt = 1)	0.57	0.91	0.69	2.17
<u>Ours</u>	0.58	<u>0.72</u>	<u>0.67</u>	<u>1.97</u>

Table 3.5: Performance comparison on the MVSEC dataset (indoor flying sequences): per-sequence AEE (px/s).

Model	outdoor_day1 AEE (px/s)
EV-FlowNet [122]	0.49
Zhu et. al. [141]	0.32
ECN$_{masked}$ [146]	**0.30**
Spike-FlowNet [100]	0.49
Back to Event Basics$_{Evf}$ [142]	0.92
Back to Event Basics$_{Fire}$ [142]	1.06
XLIF-EV-FlowNet [102]	0.45
XLIF-FireNet [102]	0.54
Fusion-FlowNet [144]	0.59
Adaptive-SpikeNet (best ANN) [104]	0.48
Adaptive-SpikeNet (best SNN) [104]	0.44
FSFN$_{FP}$ [145]	0.51
FSFN$_{HP-ADC}$ [145]	0.48
Shiba et. al. [78]	**0.30**
VSA-SM [139] (dt = 1)	0.46
Ours	0.85

Table 3.6: Performance comparison on the MVSEC dataset (outdoor sequences).

pixels), thus potentially having a different optimal kernel size associated with its resolution.

2. when optimizing our overall temporal context and event histogram duration, we did so for a specific event camera with a given configuration. However, since the MVSEC dataset was filmed with a different camera, potentially with different dynamics, a new optimization may be required to find its optimal temporal representation.

3. when optimizing the training pipeline (learning rate, optimizer, scheduler, etc.), we designed it for 480x640 pixels of ground-truth resolution and a certain signal-to-noise ratio present in the DSEC dataset (this ratio, although unknown, can be safely assumed to be constant within the data due to the lack of variety across sequences). The MVSEC dataset does not only present lower-resolution ground-truth, but also sequences of different nature, and each one of them with its own signal-to-noise ratio that would require its own hyperparameter tuning.

We therefore believe that we have demonstrated the generalization capabilities of our model, and this belief is supported by the evidence shown in Table 3.5. In addition, when trying to accomplish the same task (optical flow estimation for driving scenarios) with a different EC, our base architecture is able to estimate optic flow from scratch, without undergoing any modification to target this new condition.

3.2.4 Ablation Studies

Several ablation studies were performed on our architecture and training procedure, which we will hereby discuss. In order not to clutter this section with so many different graphs, we have decided to gather them in Appendix B, where they can be easily consulted. All of the results presented in this section have been obtained on the DSEC dataset, using the train/validation split proposed in Subsection 3.1.7.

Downsampling: Pooling vs. Convolution

Although maximum pooling as a downsampling technique is a hardware-friendly operation, and even if we got good results when using this technique, the fact remains that it is not a widespread method of tensor downsampling, since convolutions are the go-to operation for achieving lower-resolution tensors. We therefore performed an ablation study, modifying the architecture to be fully-convolutional instead of using maximum pooling as a downsampling method. The result (Figure B.3) shows that indeed pooling performs better for our task. We believe that this result is due to the fact that, by using pooling, we somewhat remove the importance of individual spikes in favor of more "global" information, i.e. whether any spikes were triggered within the pooling kernel's receptive field. However, it is also important to note that, since implementing pooling layers in our network also requires us to perform unstrided convolutions, the computational cost in terms of number of operations is higher with respect to a fully-convolutional architecture.

Encoding: Three-dimensional vs. Two-dimensional

Next, we proceeded to ablate the use of three-dimensional convolutions instead of regular bidimensional layers. Indeed, albeit our conviction that they were beneficial for the model, they also meant a great training time increase, so their use should be properly justified. To make this comparison, we trained an equivalent model where all the temporal information had been concatenated along the channel dimension, and compared it to our proposed architecture. The results, shown in Figure B.4, clearly show that explicitly handling the temporal context is indeed beneficial for network performance, since higher-order information is sequentially concatenated respecting its transient nature, thus allowing for better temporal feature extraction.

Loss Function

We have also studied different loss function combinations and their effect on the final model's performance. Specifically, we have paid attention to three study cases: training the model only on error modulo (i.e. using AEE as the loss metric), explicitly enforcing both error modulo and angular error

(i.e. a combination of AEE and AAE), and training the model on relative error only (implicitly imposing both AEE and AAE at the same time). Eventually, ARE loss was abandoned due to lackluster performance levels, so our attention turned towards the angular loss term and its influence on estimation accuracy.

Our studies show that simply enforcing minimum error modulo is not enough to achieve competitive results on the optical flow estimation task, due to the reasons already explained in Subsection 3.1.6 - Loss Function. Additionally, introducing a penalization term in the loss function seems to allow the network to better understand the optical flow structure within the visual scene without explicitly enforcing it, as we can clearly see on Figure 3.7c where obstacles are easily identified even after training on masked flow maps. Furthermore, the network trained with a combination of AEE and AAE reaches a lower AEE than an equivalent model trained only on error modulo penalization, suggesting that this second approach gets the model trapped in a local minimum. The angular loss term would therefore make the optimization landscape easier to traverse, allowing for better performance levels.

Polarities: Single-channel vs. Two-channel

Another ablation study we performed was trying to combine the effects of both polarities in a single channel at the level of the event histogram, therefore caring about an overall per-pixel event count rather than about physical luminance variations. This idea was motivated by the fact that objects of different color can yield drastically opposed event patterns (see Figure 3.8) that are nonetheless not linked to relative motion, but to the object's color instead. The toy example we show here is illustrative of real-world situations, since differently-colored cars would present a different event front evolution potentially associated to equal relative motion. By combining both polarities in a single channel, we could potentially get rid of this effect, instead accounting only for the event front displacement within the scene.

This ablation study shows that, in spite of our previous concerns, combining polarities is detrimental for our current task. Nevertheless, we

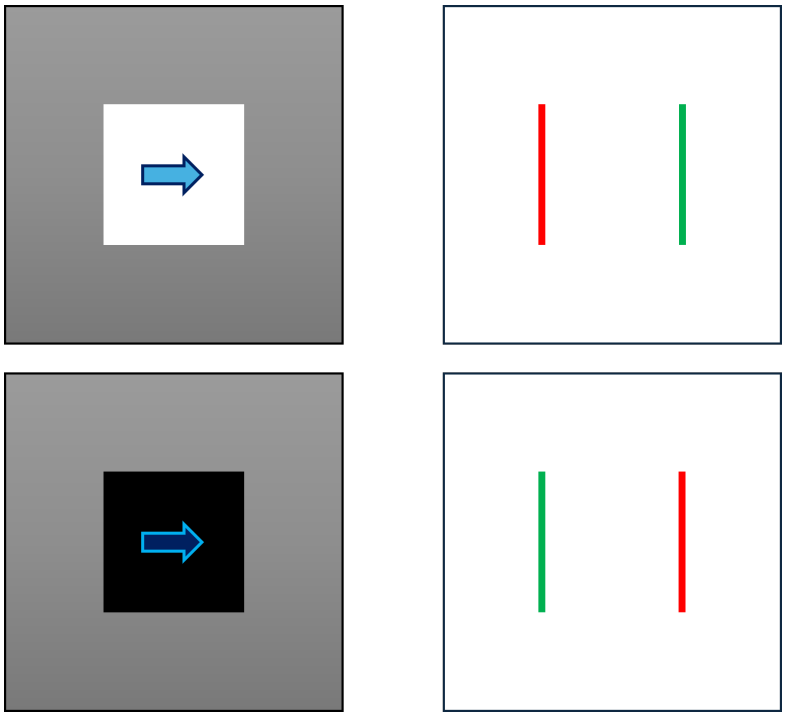


Figure 3.8: Objects of different color can yield opposite event patterns, even if they present the exact same motion.

believe this result to be caused by the lack of variety within the dataset, therefore always causing one-way event patterns. Furthermore, there is information to be gained from split polarities, since event cameras can be calibrated using different thresholds for positive and negative events, therefore leading to different dynamics for each of the polarity channels.

Effect of Stereo Vision

In order to improve the model's performance, we also explored the possibility of exploiting stereo vision for better optical flow estimation. The rationale behind this approach was fairly natural, since optical flow is a magnitude which is heavily intertwined with depth, and stereo vision has been proven to be a major player in depth/disparity estimation. We therefore proceeded to train a model using the data obtained from the left and right cameras of the DSEC dataset (instead of only the left one that we used for our monocular model), and proceeded to compare it with our base model. Much to our surprise, we found stereo vision not only to not improve our results, but rather to be detrimental to the model's performance. We believe this situation to be due to optical flow being mostly a temporal task, thus relying less on spatial information for accurate estimations. Furthermore, our network architecture has been explicitly developed to better extract temporal features, which may be hindering the implementation of stereo inputs, since the information that is expected to be extracted from them relies on spatial discrepancies. Additionally, the inclusion of an additional input was creating a heavier model both in terms of operations and of trainable parameters, so we ultimately decided to abandon this line of research.

Post-residual Skip Connection

We finally tested the role of the first post-residual skip connection in our architecture. Indeed, a keen reader will have already remarked that it is the only skip connection that consists of a tensor addition instead of a concatenation. The origin of this difference lies in a simple bug, since we forgot to modify that skip connection while iterating on the model's architecture. Thus, when we realized that it was different from all of the

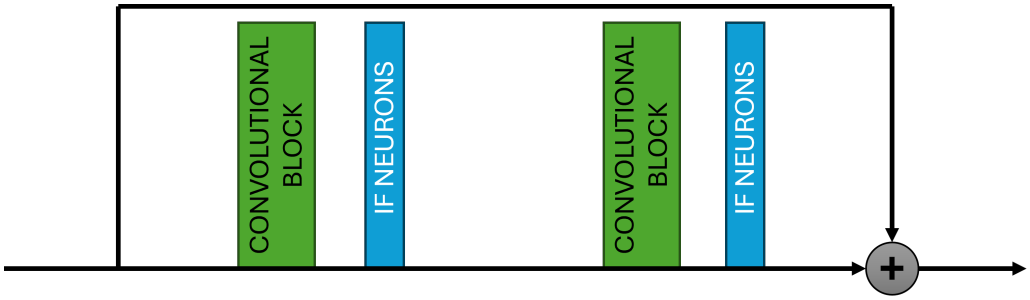


Figure 3.9: Residual Block Architecture

other ones, it was quite late in the optimization process, and we therefore needed to make sure that it was not detrimental to our overall performance. Nevertheless, we needn't worry, for the ablation study showed that indeed this skip connection worked better for our task. We believe that this situation is due to the architecture of the residual block itself (Figure 3.9), which already implicitly includes a sum skip connection between the input and the output. Thus, an additional sum skip connection is equivalent to applying a gain of two to the residual's input with respect to the pre-addition output, whereas a concatenation would have to juggle two skip connections of different nature at the same time. Since resorting to sums for this very first skip connection also reduces the number of parameters (we are at the stage of the network handling the most channels), we decided to validate this architecture and to keep it as it was first conceived.

3.2.5 Simplification for Real-world Implementation

Before moving on to the second work package of the project, regarding sensor fusion for optical flow estimation, we performed one last study concerning the physical implementation of our model in a neuromorphic chip. Indeed, although our approach was hardware friendly and theoretically deployable on neuromorphic hardware, technological limitations might prevent it from achieving a real-world implementation. To find out how implementable our model really was, we decided to target the SPLEAT [110] FPGA, and started to work on model adaptations.

After visiting the LEAT laboratory in University Côte d'Azur (Nice), we were informed of the limitations that the latest versions of the chip still presented:

- While our model was extremely light parameter-wise, it was still too large for on-chip deployment.
- SPLEAT does not currently support 3D convolutions
- Pooling is not yet implemented on the SPLEAT FPGA.
- The SPEAT hardware does not yet support residual connections, but they are working towards their implementation.

The two latter limitations were not insurmountable, since one of them was even a work in progress, but the two former represented a bottleneck for on-chip implementation. It was obvious that simplifying the model was a necessity to implement it on SPLEAT, and that the model should be fully bidimensional as far as convolutions were concerned. In other words, we would have to find a way to implement our ERF evolution along the encoder with two-dimensional convolutions, ensuring that temporal information was kept for as long as possible without explicitly including the additional temporal dimension.

To do this, we developed a model that was able to keep sequential information in the channel dimension, making a former/latter split for each downsampling stage thanks to grouped convolutions. We decided to call this method Time-Aware Grouped Convolutions (TAGC), since overall temporal information would not be combined until the very last encoding stage. Figure 3.10 illustrates this method, that we will now explain. Let us assume that the overall temporal context consists of a time window of timespan Δt, and let us assume that 512 different channels are present just before our residual block, and 256 before the deepest encoder. In addition, let us assume that we are trying to estimate optical flow at time t. Our goal is to constrain the information concerning the time interval $[t-\Delta t, t-\Delta t/2[$ within the first 128 channels of the pre-encoder tensor, and the information concerning $[t-\Delta t/2, t[$ in the second half. By sequentially applying this methodology to each of the encoder blocks, we see that a

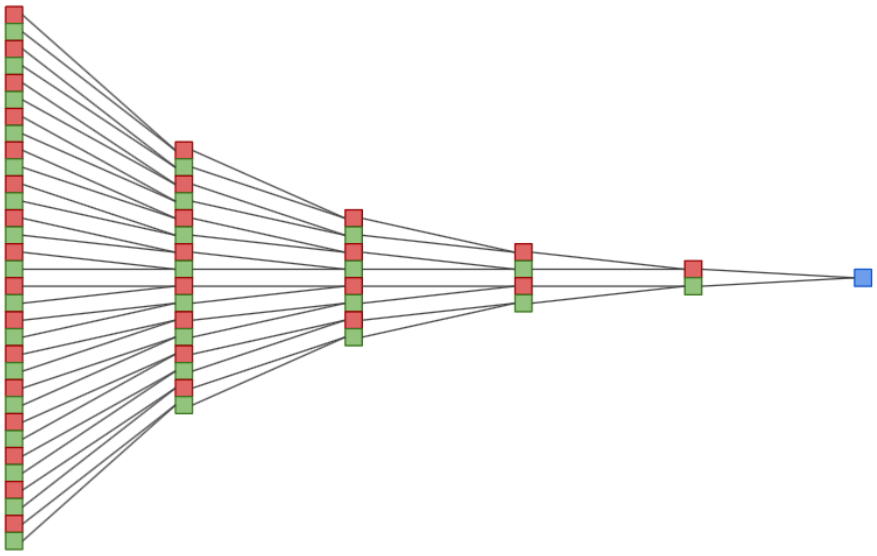


Figure 3.10: TAGC - Temporal context evolution along the network's encoder

five-stage architecture would need a total of 32 input histograms, each of them with two polarity channels. By using sequentially smaller group sizes, we can achieve a temporal separation akin to the one obtained for our three-dimensional model, but constrained to the channel dimension, since implementing grouped convolutions prevents temporal information from being completely shuffled after the very first stage. Furthermore, the use of grouped convolutions has the side benefit of reducing the number of trainable parameters present within the network.

Although the idea was promising, the results were disappointing. The network was able to achieve stable learning, but the estimations were extremely noisy, and no structure was to be found in them. We made a thorough study, including strided convolutions vs. pooling as downsampling strategies, residual blocks vs. the absence of them, and additive vs. concatenation as skip connections, but in every single case the predictions were blurry at best (these ablation results have been included in Appendix B.3). We believe that this situation may be in part due to the lower number of parameters (see Table 3.7), and in part due to using the same frame length as in our previous studies (i.e. 9ms, since we did not optimize the frame duration for this task). Moreover, our studies showed that pooling

Model	Num. Params.	Best AEE (valid)
Base model (Figure 3.3)	1.2M	1.10
TAGC: sum skip + res. block	995K	1.24
TAGC: cat skip w/o res. block	510K	1.36
TAGC: sum skip w/o res.block	420K	1.37

Table 3.7: TAGC model comparison

downsampling and residual blocks greatly improved model performance which, although still far from our base model, was pushing us further away from the FPGA's capabilities. At the light of these discouraging results, and wanting to continue our works on the second half of the PhD project, we decided to abandon this line of research. Nevertheless, we still think that results could be improved after optimizing the training procedure, and we believe that lower-resolution optical flow predictions may benefit of this approach, since the amount of predictions per timestamp is significantly lower and therefore less neurons are needed to guarantee sufficient network expressivity.

Chapter 4

Optical Flow from Event- and Frame-based Sensor Fusion using Spiking Neural Networks

This chapter will introduce our first results on sensor fusion for optical flow estimation using SNNs. We will be presenting the main idea of the project, the challenges we have encountered, and our preliminary results. Despite the fact that the current state of the fusion architecture is far from our desired performance levels, and even though it may eventually be found that image data does not greatly contribute towards accurate bimodal optical flow estimation, our most recent results remain promising, and we are still convinced that there is a way to improve optical flow estimation from event- and frame-based data combined.

4.1 Materials and Methods

4.1.1 Project Overview

After having successfully developed an optical flow estimation SNN, we proceeded to tackle the 4^{th} level of study (L4) of the DeepSee project, i.e. the fusion of frame-based and event-based sensors for better optical flow estimation. Our drive to pursue this line of research is motivated by the low spatial resolution that EBS often provide (e.g. 480x640 pixels for the

Prophesee Gen3.1 cameras [55] used in the DSEC dataset [4]). Meanwhile, frame-based sensors often show higher spatial resolution, thus being potentially complementary and beneficial for computer vision problems with high accuracy constraints.

Perhaps the most relevant publication to our task is Fusion-FlowNet [144], which performs optical flow estimation using a hybrid SNN-ANN architecture from event data and images combined, although other applications for EBS-FBS fusions exist (e.g. [147]). Of course, event and frame fusion is not the only line of research that has ever been explored, and a plethora of different domains have exploited fusion to improve their performance (e.g. EBS and radar fusion for drone navigation [148]). Sensor fusion, however, is not a novel research topic. For instance, it has been extensively investigated for autonomous driving applications, due to the high safety constraints associated with these vehicles and the high levels of redundancy usually sought, and proof of this interest is the number of surveys [149] and reviews [150] on the subject. Additional efforts have been made towards finding the best fusion strategies for machine learning [151] and deep learning [152] applications, which have allowed more complex fusion strategies to arise (e.g. attention bottlenecks [153]).

The idea of using a bimodal input to better estimate optical flow appears as quite natural, since event sensors often provide gray-scale images in addition to the event stream (e.g. the Davis 346 camera [53] used to record the MVSEC [8] dataset), and even datasets using cameras without this functionality often provide the users with synchronized frame-based data (e.g. DSEC dataset). Although EBSs are widely renowned for their high temporal resolution, they often lack in the spatial domain. Meanwhile, FBSs provide high levels of spatial resolution, but at the cost of a lower data latency. Thus, by fusing both modalities, a network should theoretically be able to optimize the extraction of temporal and spatial information for better performance. Of course, the integration of an additional modality would also translate into an increase in required computation power and in estimation latency, but they should be compensated by the improvement of the quality of the estimations.

Using the knowledge we have gained during our previous study, our goal is once again to develop a fully-spiking, hardware-friendly model, this time

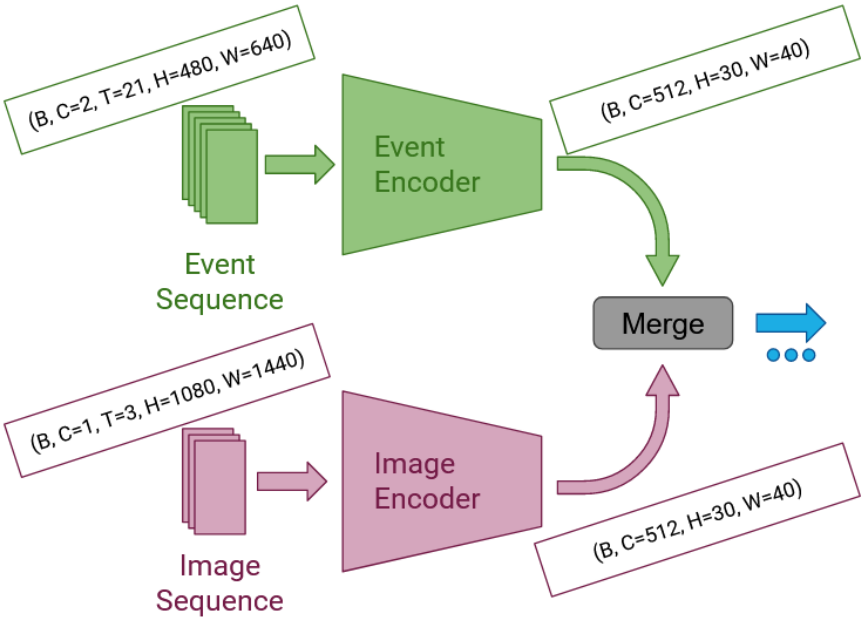


Figure 4.1: Draft of EBS and FBS fusion.

exploiting two inputs of different nature to boost the model's accuracy. To achieve this goal, we intend to develop a dual-encoder architecture, fusing both modalities at the bottleneck level and combining the information of both modalities from that point onward (Figure 4.1). The training dataset, input event representation and spiking neuron model remain unchanged, and can be consulted in Section 3.1. Additionally, the loss function will largely remain unchanged, although the relative contributions of each of the loss terms may be tuned for this new task.

4.1.2 Fusion Strategy

As soon as we started to work on combining frame- and event-based data into a single model, we encountered two main challenges, both linked to the differences in resolution these sensors present:

- On the one hand, the discrepancy in temporal resolution (virtually real-time for event-data vs. discrete time for frames) forces us to make a choice: either we keep the sliding window introduced in Chapter

3, thus meaning that the relative shift between the present (i.e. the current optical flow estimation) and and the last (most recent) input image will inevitably be variable, or we switch the sliding window time span to the frame rate of the FBS, therefore increasing the model's latency but keeping a constant temporal context for each estimation. In the end, we decided to follow the final approach, since the frame-rate of the DSEC dataset (20fps) appears to us as fast enough for the task at hand.

- On the other hand, the difference in spatial resolution (480x640 px for the ECS vs. 1080x1440 for the FBS in the DSEC dataset, i.e. 2.25 greater spatial resolution for the FBS along each axis) enormously complicates the fusion itself, both at bottleneck level and at each decoder step via the skip connections.

We explored different potential solutions to overcome the challenge that nonmatching spatial resolutions poses. Our first idea was to use nonmatching convolutions along the image encoder, i.e. applying different parameters to each convolution to ensure matching spatial resolution at the bottleneck stage. However, this approach is not optimal, because:

1. There is not a straightforward combination of parameters that achieves the desired tensor resolution at bottleneck level.

2. This approach does not allow to naturally incorporate skip connections in the model.

Indeed, the initial ratio between the image tensors and the event histograms being 2.25 times larger along each dimension, it is impossible to achieve a perfect multiple of the dimensions after just a single convolution, thus requiring multiple stages of downsampling and resizing along the encoder. Since this approach was clearly not working, we decided to slightly modify the model's philosophy, collapsing the output of each encoder into a feature vector. By performing this operation, the tensors could easily be concatenated even if they contained a different number of elements. Nevertheless, this approach also presented its challenges:

1. First, using latent vectors in the residual block (and therefore fully-connected layers instead of convolutions) translated into an unacceptable increase of the number of trainable parameters. We tried

to mitigate this by using smaller feature vectors (by adding additional encoders), but the model was still unable to perform properly.

2. Second, upsampling into images from latent vectors is not a straightforward process. Indeed, using latent vectors instead of feature maps at the bottleneck dimension meant that, at the post-residual stage of the model, the locality of the information had been lost in favor of a more global representation of the information (in other words, the ERF of each pixel after the residual block comprises all of the input pixels of both modalities, both in the temporal and in the spatial dimensions). Therefore, when resized into a tensor shape and upsampled, the information was too mixed, and it was hard for the model to extract the relevant relationships within the data.

3. Finally, the implementation of skip connections is still not solved with this approach.

In the end, this approach was also abandoned due to the many challenges it posed, aggravated by the fact that skip connections were also not implementable in the model. At this point we only had two choices to continue with this study: either we abandoned our desire to implement skip connections altogether, or we found a more "natural" way to implement them in the network. Completely removing them from the architecture was never an option, since the model was deep enough for vanishing gradients to be a concern, and the removal of skip connections would most likely translate into a loss of spatial details. Another potential solution could be to add intermediate convolutions or reshaping blocks between the image encoder outputs and the decoder inputs, but this caused an increase in the number of parameters and was a bit too close to a manual fix, in the same line that nonmatching convolutions were.

In the end, the best result we found was perhaps the simplest one we could have come up with: instead of using the 1080x1440 pixel tensors as input for the FBS encoder, we scale them to 960x1280 px images. By performing this transformation, we preserve the image's aspect ratio, all while ensuring that a first encoding stage with strided convolutions achieves matching resolutions for the FBS and the EBS sides of the encoder when using a stride value of 2 (with appropriate padding and no dilation). Although this approach somewhat hinders the spatial resolution of the FBS

(1.23Mp vs. 1.56Mp on the original images), it is the simplest possible approach, and the one that allows us to freely modify the architecture later on, and it is therefore the one we decided to adopt for our model.

4.1.3 Image Input Representation

Images in the DSEC dataset are provided as RGB arrays. However, we wish to extract optical flow information linked to motion itself, so color is not considered to be crucial data for our task. We have therefore started by converting the colored images into grayscale data using the following equation:

$$grayscale = 0.2990 \cdot red + 0.5870 \cdot green + 0.1140 \cdot blue \tag{4.1}$$

By applying this transformation at each pixel, we get an image which should preserve the structural spatial information, but where color has been removed. Afterwards, as explained in the previous subsection, images are reshaped from 1080x1440 px into 960x1280 px tensors (i.e. exactly twice larger than the event histograms along each spatial dimension) for ease of implementation.

We also decided to once again re-use our TCN-inspired architecture on the image-side, i.e., using three-dimensional convolutions along a time dimension until it naturally collapses to one in order to increase the temporal ERF while preserving a sense of continuity. We decided to use three consecutive grayscale images as our image input because:

- A pair of consecutive images would not be sufficient for the model to estimate second-order derivatives.
- We that no more images are necessary, since the temporal information ought to be extracted by the event encoder. In fact, the images of the DSEC dataset have been recorded using FLIR BFS-U3-16S2C-CS cameras, able to record RGB video at a frame rate of 20Hz. The temporal context of our image sequence thus consists of 100ms of information, which is far from the temporal context provided by the event encoder (192ms), but acceptable since temporal features are not the focus of the new encoder.

In hindsight, using 5 input images instead of 3 might have been a more relevant choice, specially when considering the future event-only vs. image-only comparisons that we will perform later in this chapter. Indeed, it is complicated to assess both of each modalities' appropriateness for optical flow estimation by comparing each of them when both inputs represent a different overall temporal context, and although we remain convinced that event data is probably the better choice due to the temporal finesse of their information, a more thorough study could be done to assess if the superiority of event data is indeed linked to temporal resolution or rather to overall temporal context. Nevertheless, the results we have obtained using 5 input images don't show a significant improvement over exploiting only 3 images for bimodal optical flow estimation, so we believe that all of the reflections that we will present in this chapter remain reasonable and acceptable.

Moreover, due to the physical layout of the different sensors used to record the dataset's sequences (Figure 4.2), there is a slight spatial offset between the FBS and the EBS data (see Figure 4.3). Furthermore, while slight, this misalignment is not exactly constant among sequences, meaning that each image-event pair would require a different correction depending on the sequence they belong to. Despite correction parameters being provided by the dataset, we have opted not to perform the correction on the input data because:

1. Introducing a correction stage in the model would require more computational resources, which we are actively trying to limit as much as possible. However, a CNN should in theory be able to learn how to correct this situation itself.

2. By forcing the network to learn how to treat different misalignments, we are virtually performing a kind of data augmentation during the training, thus making the model more robust to potential life-cycle misalignments due to real-life relative displacements between both cameras.

The difference in field of view (FOV) and associated lens distortion also play a role in data misalignment, since frame-based data (Figure 4.3a) shows a wider, more distorted scene than the associated event histogram (Figure 4.3b). Although the wider FOV of the frame-based camera means that we could have cropped the images instead of downsampling them in order to

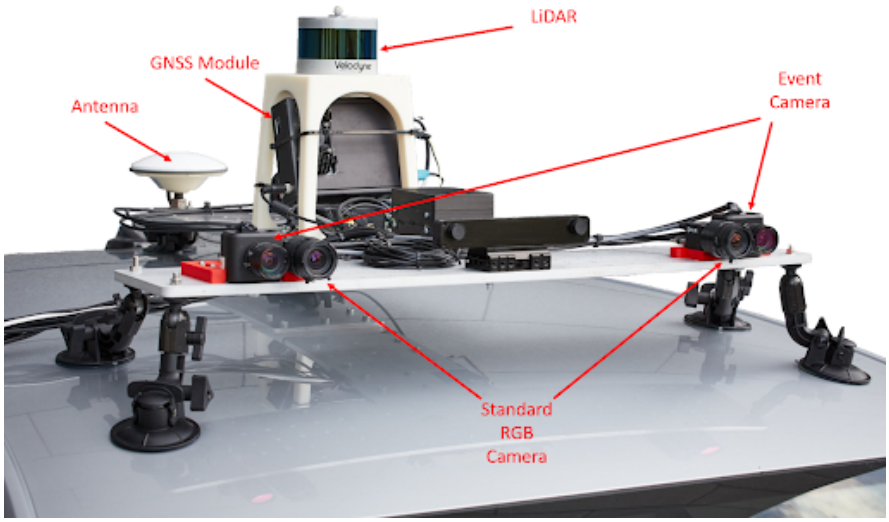
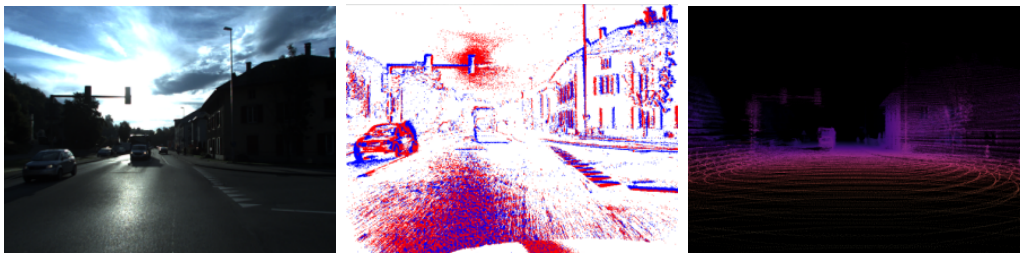


Figure 4.2: Camera rig of the DSEC dataset (picture borrowed from [4]).

make their dimensions match the event camera resolution, the problem of the slight distortion and of the alignment mismatch would always persist. We therefore decided to keep the images as they are prior to downsampling, and to once again let the network learn the relevant information by itself.

4.1.4 Network Architecture

Figure 4.4 shows the final iteration of the FBS/EBS fusion model that we have developed. It consists of two encoders, one handling 21 consecutive event histograms 9ms apart from one another, and another one handling 3 consecutive downscaled grayscale images for a total temporal context of 100ms. Every encoder stage but the first one (which increases the number of channels to 32) doubles the number of channels, until 512 channels are achieved at the bottleneck level per modality. Both inputs are downsampled via 3d convolutions until the temporal dimension collapses: for the event encoder, a temporal kernel size of 5 is used, therefore collapsing at the bottleneck level; for the image encoder, a kernel size of 2 is used, therefore collapsing after the second encoding stage. Both encoders' outputs are



(a) FBS frame (b) EBS frame (c) Ground-truth frame

Figure 4.3: Data provided by the DSEC dataset: images, events and masked ground-truth (pictures borrowed from [4]).

concatenated along the channel dimension before the residual block, and remain together from that point onwards. The decoder upsamples these tensors using nearest neighbor upsampling and separable convolutions, halving the number of channels after each stage. In addition to the previous decoder output, skip connections are included from the encoder side, using the closest-to-present information at the stages where the temporal dimension has not yet collapsed. Finally, each decoder output is upsampled into a tensor of shape $(C = 2, H = 480, W = 640)$, and all of these results contribute to the final optical flow estimation.

Although we explored many different architectures with a plethora of iterations, the final model turned out to be as similar as possible to our optical flow estimator using only event data (Figure 3.3). This choice is not by accident, and while simplicity has played a role in the choice of the architecture, it has not been the main driving force in this direction. Indeed, skip connections have heavily influenced the architectural choice, but challenges related to learning have also pushed us towards this model. In reality, although all of the model iterations we had tested so far were able to learn how to estimate optical flow to some degree, the results were lackluster at best, often consisting of inconsistent, incoherent optical flow maps. By implementing a new architecture which is in essence an update of our previous model, we are able to initialize all event-related weights to the values obtained with our previous model, thus exploiting the expertise already acquired. A discussion of the influence of network initialization on performance can be found in Section 4.2, since this is a result we found relatively late into the development of the fusion algorithm.

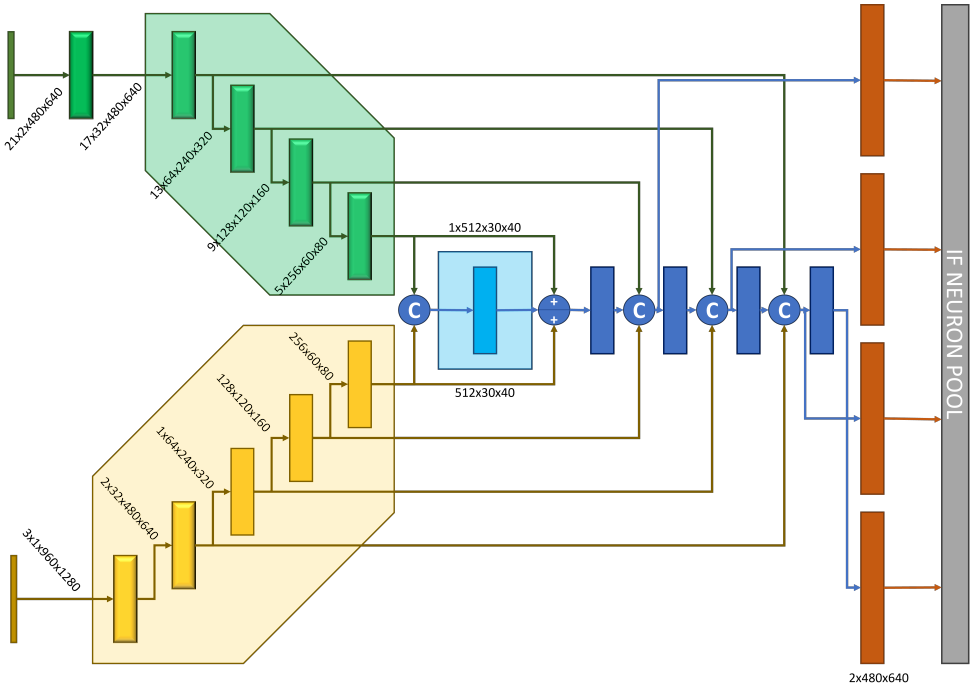


Figure 4.4: Sensor fusion network architecture

A relevant remark to our fusion architecture would be its increased number of parameters when compared to its event-only counterpart. Indeed, this new model amounts to a total of 3.3 millions of trainable parameters, mostly due to the fact that fusing modalities before the bottleneck doubles the number of channels at the "heaviest" part of the network. However, it can be seen in Table 3.4 that this value remains below most of the other available alternatives, with only IDNet [81] showing better results both in performance levels and in number of parameters, although the computational cost it requires is increased by the additional deblurring step.

4.1.5 Training Procedure and Technical Details

All of the calculations that we have performed to study the feasibility of sensor fusion for enhanced optical flow estimation have taken place on either the NVIDIA A40 GPUs belonging to the CerCo laboratory, or in the four NVIDIA A100 GPUs installed in CerCo's recently acquired HPC cluster.

Only results on DSEC are available for this study. The data split that we have used to obtain all of our results is the same one that we introduced in Subsection 3.1.7, to ensure fair comparisons among the different iterations of the model that we have studied and our baseline architecture.

4.2 Results

We will now present all of the insights we have gained concerning EBS and FBS fusion for optical flow estimation. Although we have not yet achieved competitive results for our fusion model, we believe that our results so far are steps in the right direction for successful sensor fusion using deep SNNs.

4.2.1 Exploiting Dual Inputs

We started this project by focusing our efforts into making the model exploit both modalities during training. Indeed, we remarked that models naively trained from scratch either on event-data only, on image-data only or on

both event and image data yielded similar results, with the fusion model actually presenting the worst metrics of the three. It was clear that we would have to guide the model towards exploiting bimodal data for better optical flow estimation.

Our first reflex was turning into neuroscience, specifically to the global workspace theory [154]. In a nutshell, this theory proposes a shared latent space for different modalities, where the same information is encoded into the global workspace at the same location regardless of its origin (modality). Since we wanted to exploit two inputs of different nature to improve optical flow performance, we thought that some inspiration might be taken from this domain. In particular, we looked at their training procedure, and specially at how they manage to make the model exploit the different modalities. This result is often achieved by randomly masking one of the input modalities during training, thus achieving unimodal and multimodal samples during the training procedure. Because of this alternating lack of information, the model is forced to adapt and to be able to exploit both modalities to achieve a good information encoding. Although this approach did indeed improve performance metrics on our validation set, the results did not manage to beat our previous best metrics, achieved with the event-only model. We believe that this was due to the different goals that both methods are trying to achieve:

- The global workspace seeks to encode the same information into the same latent space regardless of its origin, i.e. regardless of which modality it comes from. In other words, it is heavily linked to information redundancy, extracting the same features from different inputs and discarding the irrelevant information.
- Our approach focuses instead on making the most out of both modalities at the same time, i.e. exploiting the temporal finesse that event cameras can provide and mix it with the superior spatial resolution that frame-based sensors can offer. As such, even if we merge modalities before the bottleneck, our goal is to extract different features from each of the modalities (ideally keeping most of the temporal information restrained to the event encoder and most of the spatial dependencies to the image encoder), and then exploit this richer information for better optical flow estimations both qualitatively and in metrics.

Our goal consisted of exploiting both modalities at the same time to extract richer information, rather than using them as a complement of one another for more robust feature extraction. We therefore needed the model to rely on both modalities at the same time, since we wanted to prevent it from focusing on one of them all while neglecting the other one. In the end, we decided to solve this issue by implementing a reasonably aggressive dropout layer after the modality fusion, right before the bottleneck. This layer should create a sufficient number of holes for the network to need both modalities in order to compensate for this lack of information, thus exploiting both modalities at the same time. On the other hand, both modalities would always be present, and different information could be extracted from them both, contrary to the global workspace approach where the alternation in input modality forced the network to redundantly encode the same information from both modalities (this hypothesis is further supported by the similar metrics we obtained for both unimodal studies).

4.2.2 Network Initialization

Despite our best efforts, we were struggling to improve our previous best results. Indeed, although the model was able to learn how to estimate optical flow from both modalities, and even if results improved epoch after epoch until the learning procedure's stagnation, we were not managing to improve our event-only network's performance.

Given the difficulties we were encountering, we decided to try giving the fusion network a head-start. To this end, we initialized all event-related weights within the network to the values obtained for our event-only model. For the remaining weights, we explored different alternatives following the chronology described below:

1. Our first intuition was to set all the remaining weights to zero before training. In theory, this initialization would only allow performance to improve during training, since before training the model is exactly equivalent to our previously published model. However, this approach did not work, and training procedures often crashed as soon as they were launched. After a thorough examination, we discovered that weights were exploding due to poorly-defined gradients, since

zero-initialization everywhere was preventing gradient information from flowing during the backward pass.

2. Since initializing the whole model to zero was not an option, we decided to initialize all of the unknown weights to a small epsilon value, and to train from that point onwards. This starting point should be equivalent to zero initialization (provided that the firing threshold of the spiking units was not reached, which could be achieved with a sufficiently small epsilon value), but it should in theory allow gradients to flow and weights to be updated. Nevertheless, this initialization did not work either, since we were unable to find a good compromise between epsilon value and performance at training time 0:

   - On the one hand, too large values of epsilon allowed information to easily flow, but they were very far from our desired initial metrics.
   - On the other hand, too little values of epsilon fell into the same ill-defined backward information that the exactly zero initialization showed.

   Although we spent a significant amount of time researching this initialization, we were unable to find an intermediate epsilon value showing a good compromise between initialization and gradient flowing, and we ultimately abandoned this approach.

3. Trying to achieve a less aggressive initialization, we tried randomly initializing all of the weights without a known value except for the very first convolutional layers of the image encoder. The philosophy behind this approach was really simple: since the first encoding stage would output a zero-valued tensor (due to the lack of biases in the convolutional layers), the initialized model would be equivalent to the event-only network regardless of the initialization of the later stages. While this approach allowed us to successfully complete the training process without crashing, the results were lackluster, very close to the event-only network but never significantly improving them. After careful examination, we discovered that the image-related weights were not being updated during the training, which was preventing the model from exploiting the additional modality. Indeed, an exactly-zero initialization on the very first convolutional layer, although it did in theory allow backwards propagation of gradients, it also made the

forward weight update impossible, since only a zero value was known to the model. We were probably on the right track, but there was still an additional modification to include in our initialization process.

4. The final iteration on network initialization, and the one that showed the most promising results, consisted of a mixture of all of the previous ideas. In this case, we decided to randomly initialize the network, and then fixed the known weights to the values obtained for our event-only model. To make sure that we were close to our departure point, though, we initialized the weights corresponding to the very first image encoder to a uniform distribution centered around zero and spanning over a small epsilon value ($\epsilon = 1e{-}3$ for each of the layers within the separable convolution), effectively silencing most of the image input, but keeping gradient information flowing within the model.

Although this approach seemed to work at first, even beating our previous best results on our validation split, results were consistently bad when submitting to the official test benchmark, even worse than the departure point. To try to understand what was happening within the model, we decided to test it on event-only and image-only input after bimodal training. Our idea was quite straightforward: if both modalities were helping and contributing to optical flow estimation, both of the unimodal results should yield worse metrics than the fusion model. The result we obtained, though, was the opposite of what we were expecting: the event-only model presented the best performance results, followed by the fusion model. It became apparent that the model was not exploiting the image side: it was instead oscillating around the initial value trying to mitigate the impact of image input on the overall performance. We believe that this result is due to the initialization: by initializing to our best event-only model, we are most likely starting close to a local minimum, therefore making it extremely hard for the bimodal network to take advantage of both modalities.

The final step in our study on network initialization consisted of verifying whether a bimodal network trained from scratch would be able to exploit both modalities, regardless of how good the metrics of this model may be. To do this, we trained a model using our standard event input and 5 input images instead of the usual three, so both encoders would represent an equal time span of information for the sake of a fairer comparison. These results are shown in Table 4.1, where it can be seen that, when starting

Initialization	AEE (px/s)		
	Bimodal	**Event-only**	**Image-only**
Pretrained + Silenced Images	1.961	1.919	14.348
Random Initialization	2.988	3.131	14.731

Table 4.1: Per-input network performance comparison

from a random initialization, both modalities contribute to optical flow estimation. However, even in that case, the model seems to naturally favor the event information over the image tensors, suggesting that the temporal finesse associated to event data is more important for accurate optical flow estimation than the larger spatial resolution of frame-based images.

4.2.3 Next Steps

Despite not yet having succeeded in exploiting a bimodal input for enhanced performance, we have nonetheless managed to exploit both inputs at the same time, and we believe we are taking steps in the right direction. Nevertheless, images might not be needed for optical flow estimation altogether, or at least not when event data is available, and we have not yet fully abandoned this possible outcome either.

As far as potential future modifications are concerned, the first approach we would like to test consists of normalizing the image tensors, i.e. restricting their values between 0 and 1 instead of 0 and 255. Indeed, this technique would bring the data distributions between event tensors (with per-pixel values often ranging between 1-20) and image tensors closer together, which could improve network performance. In the same vein, network modifications could be implemented to account for different data distributions across modalities, mainly in the form of different firing threshold values for each of the inputs.

A different approach would consist of pretraining also the image side, and initializing all image-related weights to these values. By initializing both modalities to pretrained values, we would ensure that they would depart from an equal playing field for joint optical flow estimation. However, this initialization would not ensure the departure metrics to be those of the

unimodal networks, and we remain skeptical of its efficacy.

Another line of research would consist of exploiting temporal information contained on an image chunk instead of blindly feeding the images to the network. We are currently exploring the implementation of adaptive filters [155] on the image input before the encoding stages, whose trainable parameters should theoretically be able to learn the appropriate balance between sustained and transient information. We have managed to obtain minimal gains with this approach over our base models, but so far they are not significant enough to establish the AdapTrans filter as a beneficial layer for our model. Furthermore, should the adaptive filter indeed improve network performance, it would raise new questions that should be addressed with further studies, for example:

- What is the interest of transient image information when using event cameras, which themselves are able of yielding only transient information?
- Would event data remain a necessity if transient information can be easily obtained from higher-resolution sensors?

Again, all of these questions would only arise should the adaptive filter work, but they are nonetheless considerations worthy of being taken into account.

Finally, concerns about bimodality itself and its implementation should also be addressed. On the one hand, we might be trying to wrongly exploit bimodality, using a too naive of an approach to achieve it. However, this fusion strategy is rather standard in the literature, so we are not leaning towards this hypothesis. On the other hand, sensor fusion might simply not be needed for optical flow estimation, since the temporal finesse that event data can provide with could potentially be the key factor to achieve performing results instead of the increased spatial resolution that frame-based cameras can provide with. Furthermore, FBSs are extremely efficient in capturing information like texture and color, which are often considered to be irrelevant for optical flow estimation. Consequently, the fact that fusion models have not managed to beat the event-only network might not be as surprising as initially thought, and this result should not be interpreted as a failure but rather as a confirmation that fine temporal

information is the key to accurate optical flow estimation. This idea seems to be further supported by the recent publication of [156], where the authors acknowledge that only minor improvements were achieved when using dual data instead of event-only information, and only after strenuous fine tuning studies, suggesting that there might be a minimal gain to be achieved from frame based images when used for bimodal optical flow estimation.

Chapter 5

Conclusions

5.1 Optical flow estimation from event-based cameras and spiking neural networks

5.1.1 Discussion

The main takeaway from our studies on optical flow estimation using EBS data with spiking neural networks is the key role that temporal information plays in accurate prediction, both in terms of overall temporal context and temporal finesse (i.e. the global input duration and the event histogram temporal resolution respectively). In addition, we have shown that temporal information handling is equally important for accurate optical flow estimation, introducing a novel stateless model that exploits three-dimensional convolutions (akin to delay lines) to increase the model's effective receptive field in the temporal domain. Furthermore, by using depthwise and pointwise separable convolutions, we manage to reduce the number of parameters by a factor of up to ten when compared to other existing models in the literature, and we ensure that our model is at least theoretically implementable on dedicated neuromorphic chips by respecting hardware-friendliness constraints.

Our three-dimensional network has proven its efficacy in estimating optical flow for a variety of scenarios (outdoor driving on the DSEC Dataset and indoor flying on the MVSEC Dataset), and our combination of modulo and angular losses ensures that the scene's structure is learnt despite not

being esplicitly enforced (see Figure 3.7). These results place us as the best SNN for optical flow estimation from event data, and we believe that they also help to further establish event cameras as the go-to sensor for optical flow estimation, specially when coupled with spiking neural networks.

5.1.2 Perspectives

Future lines of research regarding optical flow estimation from event data using spiking neural networks should focus on improving the model by bringing it closer to neuromorphic hardware. Indeed, while our model is hardware friendly and therefore theoretically implementable on dedicated chips, there gap between our model size and existing chips remains a bit too large. Some potential improvements to close this gap could therefore be:

- Exploiting the intrinsic recurrence of spiking neurons, developing a stateful model capable of optical flow estimation. By handling temporal information through the neurons' membrane potential instead of with delay lines (implemented as 3d convolutions), we would achieve a lighter model both in terms of size (number of parameters) and required computational power (2d convolutions are easier operations to implement), thus increasing the model's maximum throughput and improving its deployability on dedicated hardware. However, the input-dependent dynamics to introduce within the network make this approach extremely challenging, and therefore potentially too time consuming and too complex to be worthy of consideration.

- Another potential line of research could consist of swapping the 3d encoding of temporal information by a learnable delay line (1d convolution) along the temporal dimension. Were temporal information to be handled all at once, this technique would translate into a bi-dimensional architecture, improving network's size and throughput.

In addition, some steps could be taken towards energy optimization:

- in terms of energy consumption, sparsity could be enforced within the model, which would help further exploit the energy efficiency capabilities associated to spiking neural networks. However, too much sparsity can translate into worse performances, and an in-depth study

should be performed to ensure that it is not detrimental to the network's metrics.

- in terms of memory requirements, steps should be taken towards weight quantization. Indeed, integer weights require less memory to be stored and less computational power overall (MAC operations can be replaced by AC), and are standard in SNN hardware.

5.2 Optical Flow from Event- and Frame-based Sensor Fusion using Spiking Neural Networks

5.2.1 Discussion

Our preliminary studies on sensor fusion for improved optical flow estimation have left us with a lukewarm sensation. Although we have been able to develop models that jointly exploit images and events for optical flow regression, none of them have managed to overthrow our event-only spiking neural network. These results seem to indicate that image tensors might not be able to compete with the temporal finesse that event data can provide with, thus acting as a fine-tuning contribution to the event-based prediction, but greatly complicating the network training due to increased parameters. Indeed, some of our results have managed to surpass our previous best score on our own validation split, but they have systematically fell short when evaluated on DSEC's test benchmark. Furthermore, these results seem to show a preference for events, silencing image data while improving on our previous best results.

Another possible explanation of why event and image fusion has not yet yielded sufficiently accurate results, specially when testing hybrid models on uni-modal input on the test set, might be linked to the nature of the sequences themselves. While the test set presents a reasonably balanced data distribution, including urban and road sequences in day, night, dusk and dawn conditions, the training data is heavily skewed towards day urban sequences. Whereas this situation does not hinder the performance of event data, mostly due to the high dynamic range that event cameras present, as well as the lack of information on absolute brightness in event data, the same

is not true for frame-based images. Indeed, different luminosity conditions translate into different input data distribution, and fast brightness variations can cause the sensors to saturate, thus hindering data expressivity and overall performance. We have so far naively tested some data augmentation techniques on image data to account for this phenomenon, consisting of mean brightness variations both by an absolute (constant offset) and a relative (multiplicative factor) quantity. The results are not yet conclusive, but we are optimistic about the effect that this kind of data augmentation might have on overall network performance.

5.2.2 Perspectives

The first line of research to pursue for optical flow estimations from EBS and FBS data combined pertains to the network's training itself. To exploit bimodality, both inputs seem to need to depart from an equal playing field. Furthermore, even from scratch, the model seems to prefer events over images. To test this hypothesis, we trained a bimodal network from a random initialization, and tested it on both bimodal, event-only and image-only inputs. Our results show that the best performance is achieved with dual input, closely followed by event-only data, and finally image-only input (see the second line of Table 4.1). This result seems to point towards our initial idea that temporal information alone is more than enough for accurate optical flow estimation. However, it is hard to believe that images play virtually no role in optical flow prediction, and our preliminary results obtained with bimodal networks trained from scratch seem support this theory.

On the other hand, temporal information seems to be key in optical flow computation, as demonstrated by the fact that randomly initialized networks seem to lean towards exploiting event information, using images as a fine-tuning mechanism more than as a true contribution to optical flow estimation. Although this phenomenon may be linked to the data distribution, as explained in the previous subsection, it may also prove that spatial resolution is not key for optical flow estimation, and that it is the combination of overall temporal context and frame temporal finesse that counts (however, it is difficult to asses whether or not this lack of significance is linked to the spatial resolution of the optical flow estimations, which is

the same as that of the DVS sensor). We have tried using more images to include a larger temporal context in the image encoder, so it would not be hindered when compared to the event encoder, but our preliminary results show that using more than 3 input images does not significantly increase the model's accuracy. We therefore believe that future efforts should be focused on exploiting temporal images also from the image side of the network, for example by using adaptive filters able to extract transient information. However, we remain aware that even with transient information, images might not be able to compete with the temporal finesse that event data can provide, and events might therefore be enough (and possibly the default choice) for optical flow computation using deep learning models.

Bibliography

[1] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[2] Anirban Nandi, Heinz Schättler, Jason Ritt, and Shinung Ching. Fundamental limits of forced asynchronous spiking with integrate and fire dynamics. *The Journal of Mathematical Neuroscience*, 7, 10 2017.

[3] Ali Rasteh, Florian Delpech, Carlos Aguilar-Melchor, Romain Zimmer, Saeed Bagheri Shouraki, and Timothée Masquelier. Encrypted internet traffic classification using a supervised spiking neural network. *Neurocomputing*, 503:272–282, 2022.

[4] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 2021.

[5] OpenAI. Introducing ChatGPT.

[6] OpenAI. DALL·E: Creating images from text.

[7] Javier Cuadrado, Ulysse Rançon, Benoit R Cottereau, Francisco Barranco, and Timothée Masquelier. Optical flow estimation from event-based cameras and spiking neural networks. *Frontiers in Neuroscience*, 17:1160034, 2023.

[8] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018.

[9] Ulysse Rançon, Javier Cuadrado-Anibarro, Benoit R Cottereau, and Timothée Masquelier. Stereospike: Depth learning with a spiking neural network. *IEEE Access*, 10:127428–127439, 2022.

[10] James J Gibson. The perception of the visual world. 1950.

[11] Benoit R. Cottereau, Andrew T. Smith, Samy Rima, Denis Fize, Yseult Héjja-Brichard, Luc Renaud, Camille Lejards, Nathalie Vayssière, Yves Trotter, and Jean-Baptiste Durand. Processing of Egomotion-Consistent Optic Flow in the Rhesus Macaque Cortex. *Cerebral Cortex*, 27(1):330–343, 01 2017.

[12] C. J. Duffy and R. H. Wurtz. Sensitivity of mst neurons to optic flow stimuli. i. a continuum of response selectivity to large-field stimuli. *Journal of Neurophysiology*, 65(6):1329–1345, 1991. PMID: 1875243.

[13] C. J. Duffy and R. H. Wurtz. Sensitivity of mst neurons to optic flow stimuli. ii. mechanisms of response selectivity revealed by small-field stimuli. *Journal of Neurophysiology*, 65(6):1346–1359, 1991. PMID: 1875244.

[14] Charles J. Duffy. Mst neurons respond to optic flow and translational movement. *Journal of Neurophysiology*, 80(4):1816–1827, 1998. PMID: 9772241.

[15] Frank Bremmer, Jean-René Duhamel, Suliann Ben Hamed, and Werner Graf. Heading encoding in the macaque ventral intraparietal area (vip). *European Journal of Neuroscience*, 16(8):1554–1568, 2002.

[16] Reuben H. Fan, Sheng Liu, Gregory C. DeAngelis, and Dora E. Angelaki. Heading tuning in macaque area v6. *Journal of Neuroscience*, 35(50):16303–16314, 2015.

[17] Velia Cardin and Andrew T. Smith. Sensitivity of Human Visual and Vestibular Cortical Regions to Egomotion-Compatible Visual Stimulation. *Cerebral Cortex*, 20(8):1964–1973, 12 2009.

[18] Valentina Sulpizio, Alice Teghil, Sabrina Pitzalis, and Maddalena Boccia. Common and specific activations supporting optic flow processing and navigation as revealed by a meta-analysis of neuroimaging studies. *Brain Structure and Function*, pages 1–25, 2024.

[19] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[20] Kwon Byung-Ki, Kim Sung-Bin, and Tae-Hyun Oh. The devil in the details: Simple and effective optical flow synthetic data generation. *arXiv preprint arXiv:2308.07378*, 2023.

[21] Simon Baker, Daniel Scharstein, James P Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92:1–31, 2011.

[22] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pages 611–625. Springer, 2012.

[23] Jonas Wulff, Daniel J Butler, Garrett B Stanley, and Michael J Black. Lessons and insights from creating a synthetic optical flow benchmark. In *Computer Vision–ECCV 2012. Workshops and Demonstrations: Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*, pages 168–177. Springer, 2012.

[24] Joel Janai, Fatma Guney, Jonas Wulff, Michael J Black, and Andreas Geiger. Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3597–3607, 2017.

[25] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[26] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[27] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[28] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[29] Rachid Deriche, Pierre Kornprobst, and Gilles Aubert. Optical-flow estimation while preserving its discontinuities: A variational approach. In *Asian Conference on Computer Vision*, pages 69–80. Springer, 1995.

[30] Joachim Weickert, Andrés Bruhn, Thomas Brox, and Nils Papenberg. *A survey on variational optic flow methods for small displacements*. Springer, 2006.

[31] Andrés Bruhn, Joachim Weickert, Christian Feddern, Timo Kohlberger, and Christoph Schnorr. Variational optical flow computation in real time. *IEEE Transactions on Image Processing*, 14(5):608–615, 2005.

[32] Naveen Onkarappa and Angel D Sappa. Laplacian derivative based regularization for optical flow estimation in driving scenario. In *Computer Analysis of Images and Patterns: 15th International Conference, CAIP 2013, York, UK, August 27-29, 2013, Proceedings, Part II 15*, pages 483–490. Springer, 2013.

[33] Wenbin Li and Darren Cosker. Video interpolation using optical flow and laplacian smoothness. *Neurocomputing*, 220:236–243, 2017.

[34] Wenbin Li, Darren Cosker, Matthew Brown, and Rui Tang. Optical flow estimation using laplacian mesh energy. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2435–2442, 2013.

[35] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[36] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *2010 IEEE computer*

society conference on computer vision and pattern recognition, pages 2432–2439. IEEE, 2010.

[37] F. Raudies. Optic flow. *Scholarpedia*, 8(7):30724, 2013. revision #149632.

[38] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *proceedings of the IEEE international conference on computer vision*, pages 1744–1752, 2017.

[39] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Flow-grounded spatial-temporal video prediction from still images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 600–615, 2018.

[40] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020.

[41] Abrar Anwar. Deephhd: Learning helmholtz-hodge decomposition to predict optical flow. *online report*, 2020.

[42] Shuaicheng Liu, Kunming Luo, Ao Luo, Chuan Wang, Fanman Meng, and Bing Zeng. Asflow: Unsupervised optical flow learning with adaptive pyramid sampling. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(7):4282–4295, 2021.

[43] Ji-il Park, Yeongseok Lee, Eungyo Suh, Hyunyong Jeon, Kuk-Jin Yoon, and Kyung-soo Kim. Improvement of optical flow estimation by using the hampel filter for low-end embedded systems. *IEEE Robotics and Automation Letters*, 6(4):7233–7239, 2021.

[44] Hao Shi, Yifan Zhou, Kailun Yang, Xiaoting Yin, and Kaiwei Wang. Csflow: Learning optical flow via cross strip correlation for autonomous driving. In *2022 IEEE intelligent vehicles symposium (IV)*, pages 1851–1858. IEEE, 2022.

[45] Shili Zhou, Ruian He, Weimin Tan, and Bo Yan. Samflow: Eliminating any fragmentation in optical flow with segment anything model.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7695–7703, 2024.

[46] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[48] Dongdong Hou and Gan Sun. Optical flow estimation with foreground attention guided network. In *Proceedings of the 2021 5th International Conference on Innovation in Artificial Intelligence*, pages 42–48, 2021.

[49] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European conference on computer vision*, pages 668–685. Springer, 2022.

[50] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[51] Misha A Mahowald and Carver Mead. The silicon retina. *Scientific American*, 264(5):76–82, 1991.

[52] Carver Mead. Neuromorphic engineering: In memory of misha mahowald. *Neural Computation*, 35(3):343–383, 2023.

[53] DAVIS346 — inivation 2024-04-26 documentation. https://docs.inivation.com/hardware/current-products/davis346.html.

[54] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6713–6719. IEEE, 2019.

[55] Gen3.1 VGA Sensor — Metavision SDK Docs 4.5.2 documentation. https://docs.prophesee.ai/stable/hw/sensors/gen31.html.

[56] Welcome to Metavision SDK — Metavision SDK Docs 4.5.2 documentation. https://docs.prophesee.ai/stable/index.html.

[57] SilkyEvCam (VGA) - CenturyArks Co., Ltd. https://centuryarks.com/en/silkyevcam-vga/.

[58] Smart Event Based Camera Vision Cam EB - imago-technologies. https://imago-technologies.com/smart-event-based-camera/.

[59] Triton Industrial Camera - LUCID Vision Labs. https://thinklucid.com/triton-gige-machine-vision/, July 2018.

[60] Sony to release two types of stacked event-based vision sensors with the industry's smallest 4.86μm pixel size for detecting subject changes only delivering high-speed, high-precision data acquisition to improve industrial equipment productivity. https://www.sony-semicon.com/en/news/2021/2021090901.html.

[61] Shoushun Chen and Menghan Guo. Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1682–1683. IEEE, 2019.

[62] Yijin Li, Zhaoyang Huang, Shuo Chen, Xiaoyu Shi, Hongsheng Li, Hujun Bao, Zhaopeng Cui, and Guofeng Zhang. Blinkflow: A dataset to push the limits of event-based optical flow estimation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3881–3888. IEEE, 2023.

[63] Xinglong Luo, Kunming Luo, Ao Luo, Zhengning Wang, Ping Tan, and Shuaicheng Liu. Learning optical flow from event camera with rendered dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9847–9857, 2023.

[64] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3586–3595, 2020.

[65] Huachen Fang, Jinjian Wu, Leida Li, Junhui Hou, Weisheng Dong, and Guangming Shi. Aednet: Asynchronous event denoising with spatial-temporal correlation among irregular data. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1427–1435, 2022.

[66] Gregor Lenz, Serge Picaud, and Sio-Hoi Ieng. A framework for event-based computer vision on a mobile device. *arXiv preprint arXiv:2205.06836*, 2022.

[67] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[68] Xu Zheng, Yexin Liu, Yunfan Lu, Tongyan Hua, Tianbo Pan, Weiming Zhang, Dacheng Tao, and Lin Wang. Deep learning for event-based vision: A comprehensive survey and benchmarks. *arXiv preprint arXiv:2302.08890*, 2023.

[69] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1964–1980, 2019.

[70] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 415–431. Springer, 2020.

[71] Luna Gava, Marco Monforte, Massimiliano Iacono, Chiara Bartolozzi, and Arren Glover. Puck: Parallel surface and convolution-kernel tracking for event-based cameras. *arXiv preprint arXiv:2205.07657*, 2022.

[72] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking.

In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[73] Kaixuan Zhang, Kaiwei Che, Jianguo Zhang, Jie Cheng, Ziyang Zhang, Qinghai Guo, and Luziwei Leng. Discrete time convolution for fast event-based stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8676–8686, 2022.

[74] Inigo Alonso and Ana C Murillo. Ev-segnet: Semantic segmentation for event-based cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[75] Sami Barchid, José Mennesson, and Chaabane Djéraba. Bina-rep event frames: A simple and effective representation for event-based cameras. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3998–4002. IEEE, 2022.

[76] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: Loss functions for event-based vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12280–12289, 2019.

[77] Federico Paredes-Vallés, Kirk YW Scheper, Christophe De Wagter, and Guido CHE De Croon. Taming contrast maximization for learning sequential, low-latency, event-based optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9695–9705, 2023.

[78] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Secrets of event-based optical flow. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVIII*, pages 628–645. Springer, 2022.

[79] Stephen J Maybank, Sio-Hoi Ieng, Davide Migliore, and Ryad Benosman. Optical flow estimation using the fisher–rao metric. *Neuromorphic Computing and Engineering*, 1(2):024004, 2021.

[80] Wachirawit Ponghiran, Chamika Mihiranga Liyanagedera, and Kaushik Roy. Event-based temporally dense optical flow estimation

with sequential neural networks. *arXiv preprint arXiv:2210.01244*, 2022.

[81] Yilun Wu, Federico Paredes-Vallés, and Guido CHE de Croon. Rethinking event-based optical flow: Iterative deblurring as an alternative to correlation volumes. *arXiv preprint arXiv:2211.13726*, 2022.

[82] Yaozu Ye, Hao Shi, Kailun Yang, Ze Wang, Xiaoting Yin, Yaonan Wang, and Kaiwei Wang. Towards anytime optical flow estimation with event cameras. *arXiv preprint arXiv:2307.05033*, 2023.

[83] Nikola Zubić, Mathias Gehrig, and Davide Scaramuzza. State space models for event cameras. *arXiv preprint arXiv:2402.15584*, 2024.

[84] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[85] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[86] Romain Brette. Philosophy of the spike: rate-based vs. spike-based theories of the brain. *Frontiers in systems neuroscience*, 9:151, 2015.

[87] Milad Mozafari, Saeed Reza Kheradpisheh, Timothée Masquelier, Abbas Nowzari-Dalini, and Mohammad Ganjtabesh. First-spike-based visual categorization using reward-modulated stdp. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6178–6190, 2018.

[88] Lina Bonilla, Jacques Gautrais, Simon Thorpe, and Timothée Masquelier. Analyzing time-to-first-spike coding schemes: A theoretical approach. *Frontiers in Neuroscience*, 16, 2022.

[89] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in neuroscience*, 13:457850, 2019.

[90] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural networks*, 14(6-7):715–725, 2001.

[91] R. G. Morris. D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949. *Brain Research Bulletin*, 50(5-6):437, 1999.

[92] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[93] Haibing Wang, Zhen He, Tengxiao Wang, Junxian He, Xichuan Zhou, Ying Wang, Liyuan Liu, Nanjian Wu, Min Tian, and Cong Shi. Triplebrain: A compact neuromorphic hardware core with fast on-chip self-organizing and reinforcement spike-timing dependent plasticity. *IEEE Transactions on Biomedical Circuits and Systems*, 16(4):636–650, 2022.

[94] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

[95] Saeed Reza Kheradpisheh, Maryam Mirsadeghi, and Timothée Masquelier. Spiking neural networks trained via proxy. *IEEE Access*, 10:70769–70778, 2022.

[96] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[97] Christian Pehle and Jens Egholm Pedersen. Norse - A deep learning library for spiking neural networks, January 2021. Documentation: https://norse.ai/docs/.

[98] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[99] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

[100] Chankyu Lee, Adarsh Kumar Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 366–382. Springer, 2020.

[101] Shubham Negi, Deepika Sharma, Adarsh Kumar Kosta, and Kaushik Roy. Best of both worlds: Hybrid snn-ann architecture for event-based optical flow estimation. *arXiv preprint arXiv:2306.02960*, 2023.

[102] Jesse Hagenaars, Federico Paredes-Vallés, and Guido De Croon. Self-supervised learning of event-based optical flow with spiking neural networks. *Advances in Neural Information Processing Systems*, 34:7167–7179, 2021.

[103] Yisa Zhang, Hengyi Lv, Yuchen Zhao, Yang Feng, Hailong Liu, and Guoling Bi. Event-based optical flow estimation with spatio-temporal backpropagation trained spiking neural network. *Micromachines*, 14(1):203, 2023.

[104] Adarsh Kumar Kosta and Kaushik Roy. Adaptive-spikenet: Event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics. *arXiv preprint arXiv:2209.11741*, 2022.

[105] Kenneth Chaney, Artemis Panagopoulou, Chankyu Lee, Kaushik Roy, and Kostas Daniilidis. Self-supervised optical flow with spiking neural networks and event based cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5892–5899. IEEE, 2021.

[106] F. Paredes-Vallés, J. J. Hagenaars, J. Dupeyroux, S. Stroobants, Y. Xu, and G. C. H. E. de Croon. Fully neuromorphic vision and control for autonomous drone flight. *Science Robotics*, 9(90):eadi0591, 2024.

[107] Byunggook Na, Jisoo Mok, Seongsik Park, Dongjin Lee, Hyeokjun Choe, and Sungroh Yoon. Autosnn: Towards energy-efficient spiking neural networks. In *International Conference on Machine Learning*, pages 16253–16269. PMLR, 2022.

[108] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.

[109] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Investigating current-based and gating approaches for accurate and energy-efficient spiking recurrent neural networks. In *International Conference on Artificial Neural Networks*, pages 359–370. Springer, 2022.

[110] Nassim Abderrahmane, Benoît Miramond, Erwann Kervennic, and Adrien Girard. Spleat: Spiking low-power event-based architecture for in-orbit processing of satellite imagery. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2022.

[111] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[112] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[113] Peter O'Connor and Max Welling. Sigma delta quantized networks. *arXiv preprint arXiv:1611.02024*, 2016.

[114] Sumit Bam Shrestha, Jonathan Timcheck, Paxon Frady, Leobardo Campos-Macias, and Mike Davies. Efficient video and audio processing with loihi 2. In *ICASSP 2024-2024 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pages 13481–13485. IEEE, 2024.

[115] Jonathan Timcheck, Sumit Bam Shrestha, Daniel Ben Dayan Rubin, Adam Kupryjanow, Garrick Orchard, Lukasz Pindor, Timothy Shea, and Mike Davies. The intel neuromorphic dns challenge. *Neuromorphic Computing and Engineering*, 3(3):034005, 2023.

[116] Manon Dampfhoffer, Thomas Mesquida, Emmanuel Hardy, Alexandre Valentian, and Lorena Anghel. Leveraging sparsity with spiking recurrent neural networks for energy-efficient keyword spotting. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[117] Edgar Lemaire, Loïc Cordone, Andrea Castagnetti, Pierre-Emmanuel Novac, Jonathan Courtois, and Benoît Miramond. An analytical estimation of spiking neural networks energy efficiency. In *International Conference on Neural Information Processing*, pages 574–587. Springer, 2022.

[118] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Are snns really more energy-efficient than anns? an in-depth hardware-aware study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(3):731–741, 2022.

[119] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7308–7316, 2019.

[120] Charlotte Frenkel, Jean-Didier Legat, and David Bol. Morphic: A 65-nm 738k-synapse/mm ˆ2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. *IEEE transactions on biomedical circuits and systems*, 13(5):999–1010, 2019.

[121] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015:*

18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pages 234–241. Springer, 2015.

[122] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*, 2018.

[123] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[124] Edward Dougherty. *Mathematical morphology in image processing*, volume 1. CRC press, 2018.

[125] Loïc Cordone, Benoît Miramond, and Philippe Thierion. Object detection with spiking neural networks on automotive event data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[126] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[127] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[128] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks: A unified approach to action segmentation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 47–54. Springer, 2016.

[129] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[130] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon,

and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[131] Ramashish Gaurav, Bryan Tripp, and Apurva Narayan. Spiking approximations of the maxpooling operation in deep snns. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[132] Stepan Tulyakov, Francois Fleuret, Martin Kiefel, Peter Gehler, and Michael Hirsch. Learning an event sequence embedding for dense event-based deep stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1527–1537, 2019.

[133] Alex Zihao Zhu, Yibo Chen, and Kostas Daniilidis. Realtime time synchronized event-based stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 433–447, 2018.

[134] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975, 2022.

[135] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022.

[136] Ismail Khalfaoui-Hassani, Thomas Pellegrini, and Timothée Masquelier. Dilated convolution with learnable spacings. In *11th International Conference on Learning Representations (ICLR 2023)*, 2023.

[137] Haotian Liu, Guang Chen, Sanqing Qu, Yanping Zhang, Zhijun Li, Alois Knoll, and Changjun Jiang. Tma: Temporal motion aggregation for event-based optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9685–9694, 2023.

[138] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *International Conference on 3D Vision (3DV)*, 2021.

[139] Hongzhi You, Yijun Cao, Wei Yuan, Fanjun Wang, Ning Qiao, and Yongjie Li. Vector-symbolic architecture for event-based optical flow. *arXiv preprint arXiv:2405.08300*, 2024.

[140] Vincent Brebion, Julien Moreau, and Franck Davoine. Real-time optical flow for vehicular perception with low-and high-resolution event cameras. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15066–15078, 2021.

[141] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 989–997, 2019.

[142] Federico Paredes-Vallés and Guido CHE de Croon. Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3446–3455, 2021.

[143] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. Efficient neuromorphic signal processing with loihi 2. In *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259. IEEE, 2021.

[144] Chankyu Lee, Adarsh Kumar Kosta, and Kaushik Roy. Fusion-flownet: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6504–6510, 2022.

[145] Marco Paul E Apolinario, Adarsh Kumar Kosta, Utkarsh Saxena, and Kaushik Roy. Hardware/software co-design with adc-less in-memory computing hardware for spiking neural networks. *arXiv preprint arXiv:2211.02167*, 2022.

[146] Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A Yorke, and Yiannis Aloimonos. Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors. In *2020 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 5831–5838. IEEE, 2020.

[147] Wen Yang, Jinjian Wu, Jupo Ma, Leida Li, Weisheng Dong, and Guangming Shi. Learning for motion deblurring with hybrid frames and events. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1396–1404, 2022.

[148] Ali Safa, Tim Verbelen, Ilja Ocket, André Bourdoux, Hichem Sahli, Francky Catthoor, and Georges Gielen. Fusing event-based camera and radar for slam using spiking neural networks with continual stdp learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2782–2788. IEEE, 2023.

[149] Amr Mohamed, Jing Ren, Moustafa El-Gindy, Haoxiang Lang, and AN Ouda. Literature survey for autonomous vehicles: sensor fusion, computer vision, system identification and fault tolerance. *International Journal of Automation and Control*, 12(4):555–581, 2018.

[150] Jamil Fayyad, Mohammad A Jaradat, Dominique Gruyer, and Homayoun Najjaran. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, 20(15):4220, 2020.

[151] Ramon F Brena, Antonio A Aguileta, Luis A Trejo, Erik Molino-Minero-Re, and Oscar Mayora. Choosing the best sensor fusion method: A machine-learning approach. *Sensors*, 20(8):2350, 2020.

[152] Valentin Vielzeuf, Alexis Lechervy, Stéphane Pateux, and Frédéric Jurie. Multilevel sensor fusion with deep learning. *IEEE sensors letters*, 3(1):1–4, 2018.

[153] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *Advances in neural information processing systems*, 34:14200–14213, 2021.

[154] Rufin VanRullen and Ryota Kanai. Deep learning and the global workspace theory. *Trends in Neurosciences*, 44(9):692–704, 2021.

[155] Ulysse Rançon, Timothée Masquelier, and Benoit R Cottereau. A general model unifying the adaptive, transient and sustained properties of on and off auditory neural responses. *bioRxiv*, pages 2024–01, 2024.

[156] Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Dense continuous-time optical flow from event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Appendix A

Data Augmentation Functions

Algorithm 3 Random Horizontal Flip

Inputs: $events$, $label$, $mask$, p_flip
if $p\_flip \leq random\ x\ drawn\ from\ U(0,1)$ **then**
    $events \leftarrow hflip(events)$
    $label \leftarrow hflip(label)$
    $label_x \leftarrow (-1) \cdot label_x$
    $mask \leftarrow hflip(mask)$
end if
Return: $events$, $label$, $mask$

Algorithm 4 Random Vertical Flip

Inputs: $events$, $label$, $mask$, p_flip
if $p\_flip \leq random\ x\ drawn\ from\ U(0,1)$ **then**
    $events \leftarrow hflip(events)$
    $label \leftarrow hflip(label)$
    $label_y \leftarrow (-1) \cdot label_y$
    $mask \leftarrow hflip(mask)$
end if
Return: $events$, $label$, $mask$

Algorithm 5 Random Rotate

Inputs: $events$, $label$, $mask$, p_rotate
if $p\_rotate \leq random\ x\ drawn\ from\ U(0,1)$ **then**
    $p1 = random\ x\ drawn\ from\ U(0,1)$
    if $p1 <= 0.5$ **then**
        $events \leftarrow rotate(events, -90)$
        $label \leftarrow rotate(label, -90)$
        $label \leftarrow flip\_channels(label)$
        $label[1] \leftarrow (-1) \cdot label[1]$
        $mask \leftarrow rotate(mask, -90)$
    else
        $events \leftarrow rotate(events, +90)$
        $label \leftarrow rotate(label, +90)$
        $label \leftarrow flip\_channels(label)$
        $label[0] \leftarrow (-1) \cdot label[0]$
        $mask \leftarrow rotate(mask, -90)$
    end if
end if
Return: $events$, $label$, $mask$

Algorithm 6 Random Event Drop

Inputs: $events$, p_drop, min_drop_rate, max_drop_rate
if $p\_drop \leq random\ x\ drawn\ from\ U(0,1)$ **then**
    $q1 \leftarrow x\ drawn\ from\ U(0,1)$
    $q2 \leftarrow (min\_drop\_rate - max\_drop\_rate) \cdot q1 + max\_drop\_rate$
    $event\_mask \leftarrow [random\_like(events) > q2]$
    $events \leftarrow events * event\_mask$
end if
Return: $events$

Algorithm 7 Random Patch

Inputs: $events$, p_patch, min_patch_size, max_patch_size, max_patch
$(H, W) \leftarrow size(events)$
if $p\_patch \leq random\ x\ drawn\ from\ U(0,1)$ **then**
    $n\_patches \leftarrow random\_choice(max\_patch)$
    for n **in** $range(n\_patches)$ **do**
        $size \leftarrow random\_choice(min\_patch\_size, max\_patch\_size)$
        $x\_start \leftarrow random\_choice(W - size)$
        $y\_start \leftarrow random\_choice(H - size)$
        $events[y\_start : y\_start + size, x\_start : x\_start + size] \leftarrow 0$
    end for
end if
Return: $events$

Algorithm 8 Random Temporal Mirror

Inputs: $events$, $prev_label$, $next\_label$, $prev_mask$, $next\_mask, p\_mirror$
if $p\_mirror \leq random\ x\ drawn\ from\ U(0,1)$ **then**
    $events \leftarrow flip\_channels(events)$
    $label \leftarrow (-1) * prev\_label$
    $mask \leftarrow prev\_mask$
else
    $events \leftarrow events$
    $label \leftarrow next\_label$
    $mask \leftarrow next\_mask$
end if
Return: $events$, $label$, $mask$

Appendix B

Network optimization and Ablation studies

B.1 3D Network - Optimization

B.1.1 Temporal Context

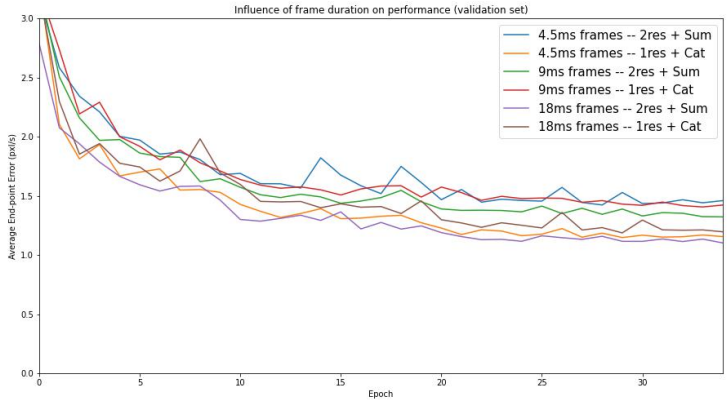


Figure B.1: Temporal context optimization (frame duration comparison)

B.1.2 Residual Blocks and Skip Connections

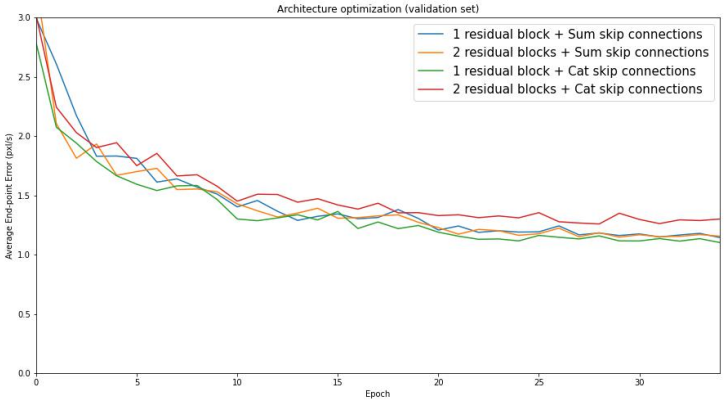


Figure B.2: Architecture optimization - Skip connections and residual blocks

B.2 3D Network - Ablation Studies

B.2.1 Downsampling: Pooling vs. Convolution

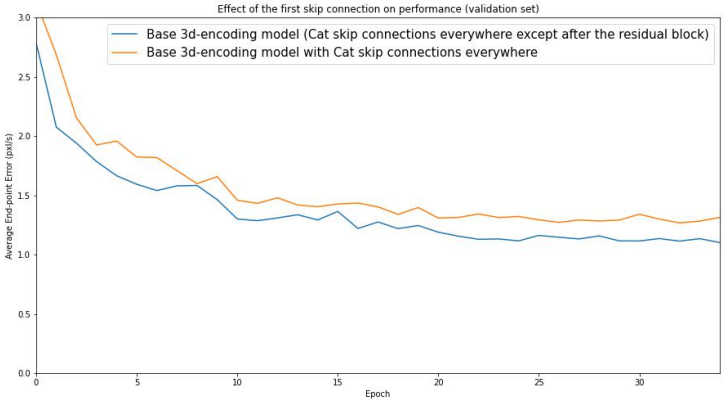


Figure B.3: Downsampling strategies comparison: Max. pooling vs. Strided Convolutions

B.2.2 Encoding: Three-dimensional vs. Two-dimensional

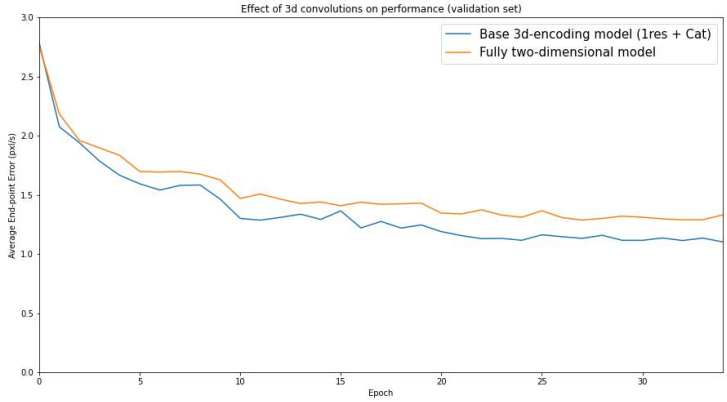


Figure B.4: Bidimensional vs. tridimensional encoders: effect on performance

B.2.3 Loss Function

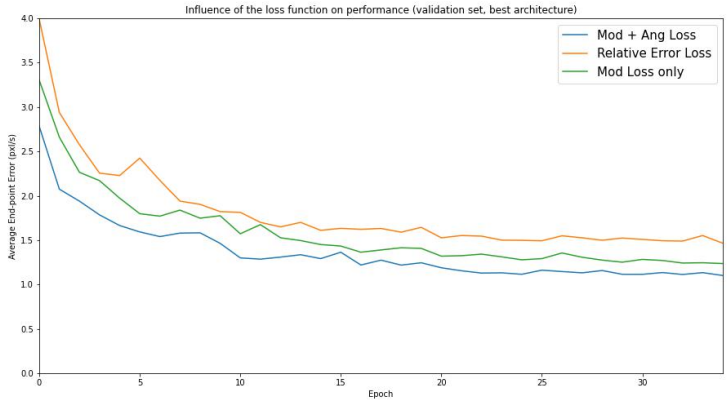


Figure B.5: Effect of loss function on network accuracy

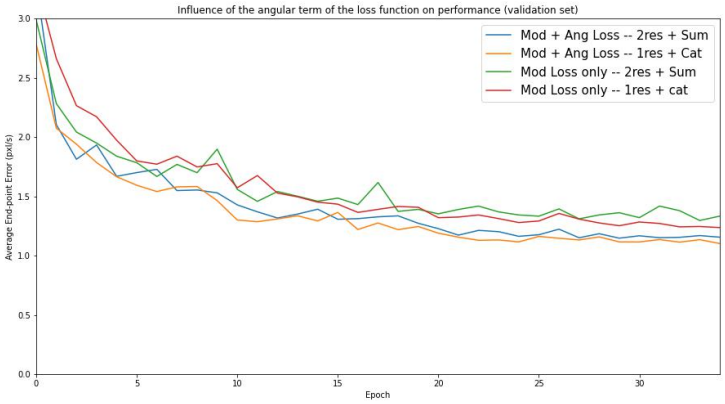


Figure B.6: Influence of the angular loss on model network accuracy

B.2.4 Polarities: Single-channel vs. Two-channel

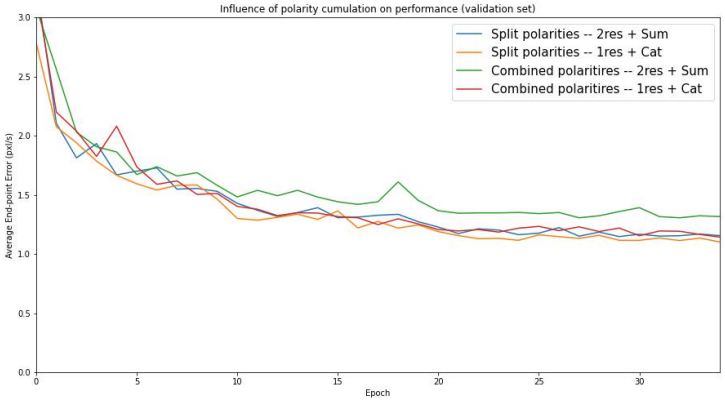


Figure B.7: Split vs. Combined polarities: effect on network accuracy

B.2.5 Effect of Stereo Vision

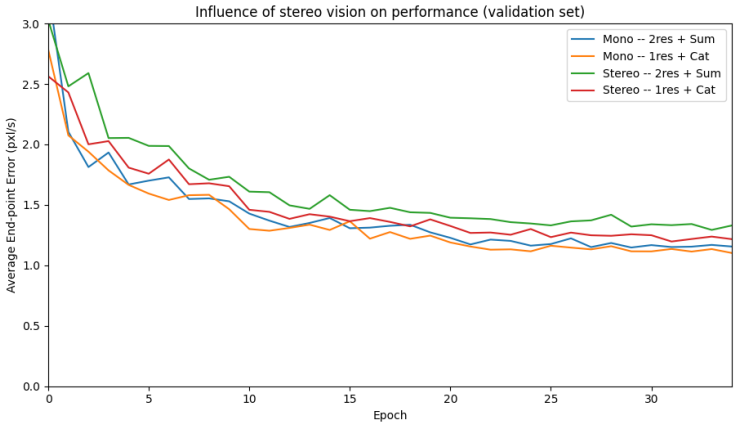


Figure B.8: Influence of stereo vision on model accuracy

B.2.6 Post-residual Skip Connection

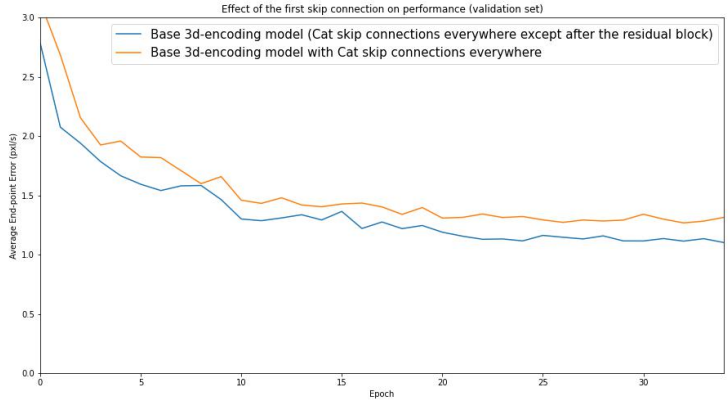


Figure B.9: Effect of post-residual skip connection on performance (best architecture).

B.3 2D Network (TAGC)- Ablation Studies

B.3.1 Downsampling: Pooling vs. Convolution

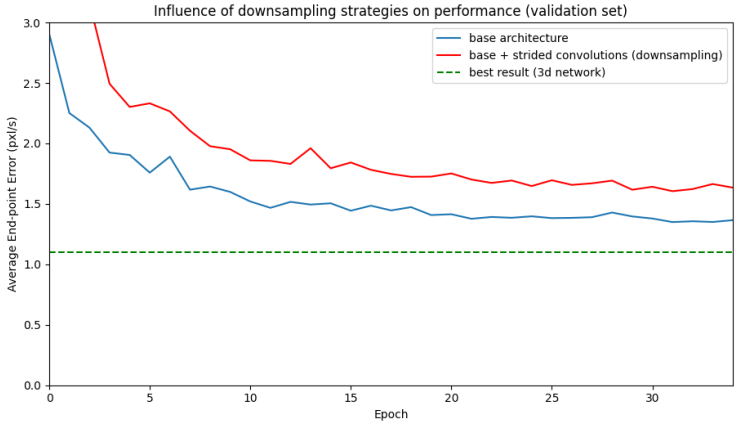


Figure B.10: TAGC: Influence of downsampling strategy on performance.

B.3.2 Residual Blocks

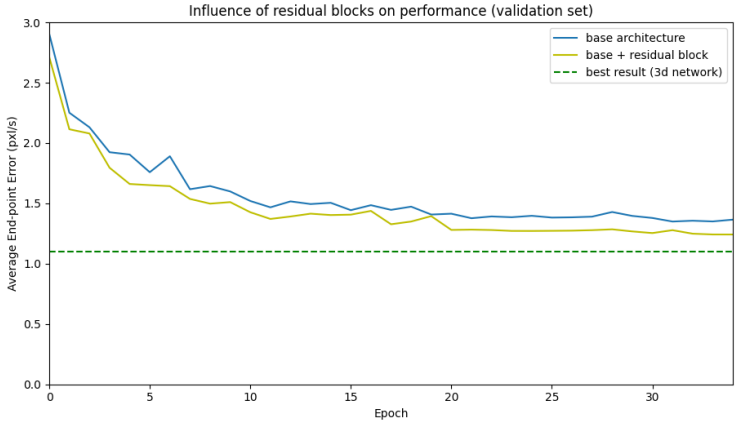


Figure B.11: TAGC: Influence of residual block on performance.

B.3.3 Skip Connections

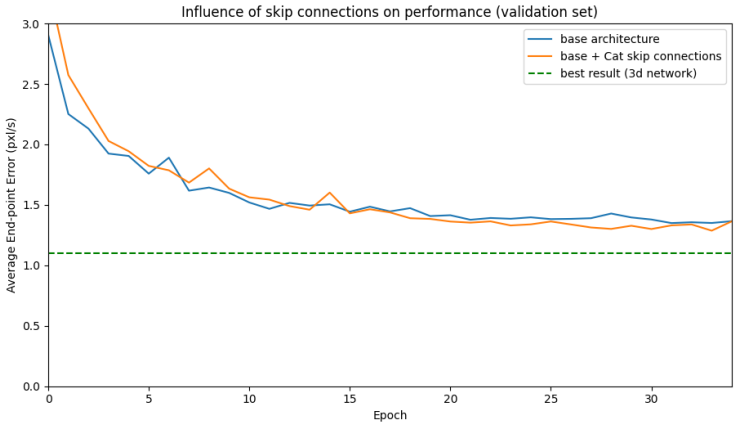


Figure B.12: TAGC: Influence of skip connections on performance.

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

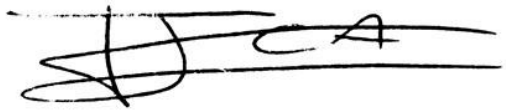
SERMENT DES DOCTEURS/ THE DOCTORAL OATH

“En présence de mes pairs.

Parvenu(e) à l’issue de mon doctorat en Informatique, et ayant ainsi pratiqué, dans ma quête du savoir, l’exercice d’une recherche scientifique exigeante, en cultivant la rigueur intellectuelle, la réflexivité éthique et dans le respect des principes de l’intégrité scientifique, je m’engage, pour ce qui dépendra de moi, dans la suite de ma carrière professionnelle quel qu’en soit le secteur ou le domaine d’activité, à maintenir une conduite intègre dans mon rapport au savoir, mes méthodes et mes résultats.”

« In the presence of my peers.

With the completion of my doctorate in Computer Science, in my quest for knowledge, I have carried out demanding research, demonstrated intellectual rigour, ethical reflection, and respect for the principles of research integrity. As I pursue my professional career, whatever my chosen field, I pledge, to the greatest of my ability, to continue to maintain integrity in my relationship to knowledge, in my methods and in my results. »



Javier CUADRADO ANÍBARRO

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Titre : Estimation du flux optique à partir des données issues des capteurs événementiels avec des réseaux de neurones impulsionnels

Mots clés : apprentissage machine, vision basée sur des événements, bio-inspiration, réseaux de neurones impulsionnels

Résumé : Ces travaux de thèse ont pour but le développement d'un modèle de vision artificielle prenant des données issues d'une caméra basée sur des événements pour l'estimation du flux optique dans la scène visuelle.
Ce projet repose donc sur trois piliers : premièrement, le flux optique, grandeur associée à la fois à la profondeur et au déplacement relatif entre l'observateur et le monde autour de lui, et qui représente la distribution du champ de vitesses dans la scène visuelle. Cette grandeur, d'importance clé pour la navigation des animaux, est de plus en plus recherchée en vue de ses potentielles applications dans de nouvelles technologies telles que les voitures autonomes.
Deuxièmement, les caméras basées sur des événements (EC), un type de capteur bio-inspiré dont les pixels fournissent une information sur les variations locales de la luminosité de façon asynchrone et en temps réel : une augmentation locale de la luminosité déclencherait un événement positif au pixel concerné, tandis qu'une diminution de la luminosité déclencherait un événement négatif. C'est en effet cette information temporelle qui rend les EC aussi intéressantes en tant que capteur pour la vision artificielle, et notamment pour des tâches temporelles telle que l'estimation du flux optique. Ce type de capteurs présente certains avantages par rapport aux caméras conventionnelles : latence inférieure, gamme dynamique très large et efficacité énergétique. Ces avantages sont mitigés par une résolution spatiale inférieure, ainsi que par un manque d'information sur la luminosité absolue dans la scène.
Finalement, les réseaux de neurones impulsionnels (SNN) sont des algorithmes d'apprentissage profond composé de neurones bio-inspirées. D'une part les neurones impulsionnels sont dotés d'une mémoire au niveau de leur potentiel de membrane, d'autre part l'information n'est propagée que quand ce potentiel dépasse un certain seuil. Ces neurones sont caractérisés par leur efficacité énergétique quand déployés sur des puces dédiées, car une absence d'information se traduit dans une absence d'impulsions, et par conséquent dans une très faible consommation énergétique. Ainsi, la triade flux optique – EC – SNN se présente comme un système fermé et cohérent : des capteurs liés au mouvement (avec absence d'information si le déplacement relatif est nul), avec des neurones bio-inspirés à propriétés similaires, et tout cela appliqué pour l'estimation d'une grandeur essentiellement liée à la temporalité du mouvement.
Nos travaux ont commencé par le développement d'un SNN capable d'estimer le flux optique à partir des données issues des EC. Finalement, nos efforts ont abouti à une publication en mai 2023 dans Frontiers en Neuroscience, introduisant le meilleur SNN de l'état de l'art pour cette tâche. Notre modèle, disponible dans https://github.com/J-Cuadrado/OF_EV_SNN, présente un SNN sans mémoire capable d'exploiter la temporalité des données évènementielles grâce à des convolutions le long de la dimension temporelle pour une représentation améliorée de cette information au niveau du bottleneck en augmentant son champ récepteur effectif.
Nous avons continué nos travaux avec une étude sur l'intégration des images en niveau de gris dans un modèle plus complexe pour améliorer les performances du modèle. Bien que nos résultats aient été encourageants jusqu'à présent, nous n'avons pas encore réussi à systematiquement battre nos meilleurs résultats. Ainsi, il s'avère que l'information contenue dans des images standard pourrait en effet ne pas être nécessaire pour l'estimation du flux optique. Par conséquent, nous allons poursuivre nos travaux sur cette ligne de recherche, soit pour vérifier cette hypothèse, soit pour effectivement améliorer les performances de notre modèle grâce à l'exploitation d'une information bimodale.

Title: Optical flow estimation from event-based data using spiking neural networks

Key words: machine learning, event-based vision, bio-inspiration, spiking neural networks

Abstract: This PhD project seeks to develop an artificial vision model exploiting event data for optical flow estimation within the visual scene.
The project lies on three main axis: first, optical flow, a magnitude linked at the same time to depth and to relative movement between the observer and the world around it. This magnitude represents the velocity field distribution within the visual scene, and it is key for navigation in living animals. It is currently being heavily researched because of its potential applications in new technologies such as autonomous cars.
Second, event cameras are a bio-inspired sensor with asynchronous pixels yielding real-time information on local luminosity variations: brightness increments trigger positive events, and negative events are caused by luminosity decrements. It is indeed this temporal information that makes event cameras appropriate for computer vision tasks, specially for temporal tasks such as optical flow estimation. This type of camera presents a number of advantages over their frame-based counterparts: lower latency, higher dynamic range and lower energy consumption. This advantages are somewhat mitigated by a lower spatial resolution, as well as by the lack of information on absolute brightness within the scene.
Finally, spiking neural networks (SNNs) are consist of deep learning algorithms using bio-inspired neurons. Spiking neurons show memory capabilities inherent to their membrane potential, and they only propagate information when this potential surpasses a certain value (firing threshold). These neurons are characterized by their energy efficiency when deployed on dedicated hardware, since a lack of input information translates into an absence of spikes within the model (provided that there are no biases in the model's layers), and therefore into a very low energy consumption. The triad optical flow – event vision – spiking neural networks is therefore a closed and coherent system: movement-linked sensors (with lack of events for static scenes), processed with bio-inspired neurons showing the same behavior, and all of this applied to the estimation of a temporal magnitude such as optical flow.
Our research works started with the development of a SNN capable of estimating optical flow from event data. Our results were published in may 2023 in Frontiers in Neuroscience, introducing the best SNN in the literature for this task. Our model, which can be found on https://github.com/J-Cuadrado/OF_EV_SNN, consists of a stateless SNN able to exploit temporal information from event data via convolutions along a temporal dimension for a better representation of temporal encoding at the bottleneck level, increasing its effective receptive field.
We have continued our research by seeking to integrate grayscale images into a more complex architecture to improve the model's performance. Although our results are encouraging, we have so far been unable to consistently beat our previous best results. Thus, is possible that the information present in frame-based sequences might not be needed for optical flow estimation when coupled with event data. We are therefore going to continue pursuing this line of research, either to test this hypothesis, or to eventually succeed in improving the model's metrics thanks to the exploitation of bimodal information.