ABSTRACT

Title of Dissertation: BIO-INSPIRED MOTION PERCEPTION: FROM GANGLION CELLS TO AUTONOMOUS VEHICLES

Chethan Mysore Parameshwara
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Yiannis Aloimonos and Dr. Cornelia Fermüller
Neuroscience and Cognitive Science Program

Animals are remarkable at navigation, even in extreme situations. Through motion perception, animals compute their own movements (egomotion) and find other objects (prey, predator, obstacles) and their motions in the environment. Analogous to animals, artificial systems such as robots also need to know where they are relative to structure and segment obstacles to avoid collisions. Even though substantial progress has been made in the development of artificial visual systems, they still struggle to achieve robust and generalizable solutions. To this end, I propose a bio-inspired framework that narrows the gap between natural and artificial systems.

The standard approaches in robot motion perception seek to reconstruct a three-dimensional model of the scene and then use this model to estimate egomotion and object segmentation. However, the scene reconstruction process is data-heavy and computationally expensive and fails to deal with high-speed and dynamic scenarios. On the contrary, biological visual systems excel in the aforementioned difficult situation by extracting only minimal information sufficient for motion

perception tasks. I derive minimalist/purposive ideas from biological processes throughout this thesis and develop mathematical solutions for robot motion perception problems.

In this thesis, I develop a full range of solutions that utilize bio-inspired motion representation and learning approaches for motion perception tasks. Particularly, I focus on egomotion estimation and motion segmentation tasks. I have four main contributions: 1. First, I introduce NFlowNet, a neural network to estimate normal flow (bio-inspired motion filters). Normal flow estimation presents a new avenue for solving egomotion in a robust and qualitative framework. 2. Utilizing normal flow, I propose the DiffPoseNet framework to estimate egomotion by formulating the qualitative constraint in a differentiable optimization layer, which allows for end-to-end learning. 3. Further, utilizing a neuromorphic event camera, a retina-inspired vision sensor, I develop 0-MMS, a model-based optimization approach that employs event spikes to segment the scene into multiple moving parts in high-speed dynamic lighting scenarios. 4. To improve the precision of event-based motion perception across time, I develop SpikeMS, a novel bio-inspired learning approach that fully capitalizes on the rich temporal information in event spikes.

BIO-INSPIRED MOTION PERCEPTION: FROM GANGLION CELLS TO AUTONOMOUS VEHICLES

by

Chethan Mysore Parameshwara

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Dr. Cornelia Fermüller, Co-Advisor
Professor Guillermo Gallego
Professor Behtash Babadi
Dean's Representative:
Professor Daniel Butts

Dedication

To my family and friends.

Acknowledgments

I would like to thank my advisors Prof. Yiannis Aloimonos and Dr. Cornelia Fermüller, for introducing me to the world of Bio-inspired Perception and giving me the opportunity to work at the intersection of Neuroscience and Robotics. Their continuous guidance and feedback enabled me to develop a more profound and fundamental perspective on research problems. They are the innovative thinkers I look up to during my graduate school journey and beyond. I will always be grateful for their impact on my career and personal life.

I would also like to thank my doctoral committee members, Prof. Guillermo Gallego, Prof. Behtash Babadi, and Prof. Daniel Butts, for agreeing to serve on my dissertation committee and for sparing their invaluable time reviewing the manuscript. I am also grateful for having the chance to know and work with some amazing people in the NACS, CS, Robotics, and UMIACS. I would like to thank Pam Komarek, Tom Hurst, Ania Picard, and Janice M. Perrone.

My colleagues at the Perception and Robotics Group (PRG) have enriched my graduate life in many ways and deserve a special mention. I would like to thank Nitin J. Sanket and Chahat Deep Singh for collaborating on multiple projects and inspiring me to dive deep and fully commit to every research endeavor. In addition, I feel fortunate to work alongside the brilliant researchers in the PRG lab: Kanishka Ganguly, Snehesh Shrestha, Anton Mitrokhin, Gokul Hari, Simin Li, Ashwin V. Kuruttukulam, Max Morrison, Rohith Jayarajan, Levi Burner, Zhiyuan Hua, Alex Ecins, Yezhou Yang, Xiaomin Lin, Michael Maynord, Matthew Evanusa, Francisco Barranco,

Chengxi Ye, Neal Anwar, Chinmaya Devaraj, Peter Sutor, Behzad Sadrfaridpour, Pyone Thaht Win, Jingxi Chen, and Vishaal Kanna Sivakumar. In addition to some of my best collaboration experiences, I will never forget our discussions and the fun we had, both within and outside of the lab.

I owe my deepest thanks to my family - my mother, Meenakumari C., and father, Parameshwara G. N., who have always stood by me and guided me through my career and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them. I would like to thank the smartest and kindest person I am blessed to be with, Sindhura Venkatesh; thank you for being with me along the Ph.D. journey and helping me in every step.

I also feel lucky to have become friends with many awesome people during my years in College Park. Aditya Earanky, Kiran Yakkala, Prudhvi Gurram, Gilad Nave, Eunice Jallah, Perri Smith, Anna Tinnemore, Aounon Kumar, Süleyman Aslan, Star Kim, Raveesh Mayya, Shraddha Karanth, Deepthi Raghunandan, Ankan Bansal, Karthick Abhinav, Rajeev Ranjan, Aquia Richburg, Nicole Catanzarite, Jo Shoemaker, Peter Gaskins, Ben Rickles, thank you all for being there for me, being good friends and housemates, and helping me remain balanced.

I would like to acknowledge financial support from William Hodos Dissertation Assistantship, Graduate School Research Fellowship, Northrop Grumman Corporation, Sony Prophesee, Samsung Electronics, National Science Foundation (NSF), NVIDIA, and Intel for all the research work discussed herein.

It is impossible to remember all, and I apologize to those I have inadvertently left out. Lastly, thank you all, and thank God!

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The advent of automation has pushed for the coexistence of humans and artificial systems such as Robots and Self-driving cars. Robots utilize a plethora of sensors for navigating safely and with agility. Among various sensor modalities, vision is the most important cue for encoding navigation information necessary for sensory-motor control or scene understanding. Therefore, a significant amount of attention in AI and Robotics has been devoted to building robust artificial visual navigation systems. Many solid mathematical frameworks and practical solutions have developed [11]; however, they still struggle to perform in everyday environments and adverse weather conditions. One approach to improve perception capabilities is to mimic biology, as animals can navigate complex environments even at extreme speeds. Perhaps bio-inspired computer vision could unlock the full potential of these autonomous systems. This thesis proposes bio-inspired solutions to motion perception problems under challenging environments, namely high speed, changing lighting conditions, and unseen/novel environments.

From primitive insects to advanced humans, one can observe that animals are excellent at solving motion perception problems, such as localization and obstacle avoidance, that are imperative for survival. Even miniature beings, such as bees (Fig. 1.1), are able to forage for food and dodge predators with minimal resources [12]. The bulk of evidence [13] suggests that biological beings follow the purposive or functional approach [14] by utilizing only minimal

Figure 1.1: The motion perception in the nature. The images the depicts an example for, Left: localization or egomotion, Right: obstacle avoidance. For original video see https://www.youtube.com/c/FrederickDunnPhoto

information required for the perceptual task at hand. In this thesis, I derive inspiration from biology and inculcate the minimalist philosophy right from data acquisition until motion inference.

1.1 The Chicken or the Egg dilemma: Egomotion and IMO Segmentation

At the core of visual navigation is Structure from Motion (SfM), i.e. the computations for estimating the image motion, egomotion/3D camera motion, and scene structure. The motto of these approaches is to recover the scene properties through reconstruction via multiple images. Earlier algorithms [15–17] developed "scene independent" constraints (e.g. the epipolar constraint [18] or depth positivity constraint [19]) to obtain egomotion, which subsequently is used in scene reconstruction [20]. More recently, the SLAM framework has been adopted [21], where depth, 3D motion, and image measurements are estimated together using probabilistic algorithms. These methods, however, are computationally very heavy. Furthermore, classic SfM and SLAM struggle to deal with the uncertainty posed by highly dynamic scenarios, where both camera and objects are independently moving.

A significant problem still unsolved in dynamic scenarios is Independent Moving Object

(IMO) segmentation. IMO segmentation is an important step in obstacle avoidance (throughout this thesis, motion segmentation and IMO segmentation are used interchangeably). In order to do IMO segmentation, we need to enforce constraints on the image motion, which in turn depends on the egomotion. In the case of the stationary camera, the motion field on the image is only caused by moving objects. Still, in the case of a moving camera, the motion field is generated by the combined effect of camera motion, structure, and the independent motion of objects. Isolating the contribution of each of these three factors is needed to solve for independent motion completely. Thus, space-time geometry has to be considered early in the problem to solve them together. Along with the IMO segmentation problem, standard approaches also have difficulty in handling challenging scenarios such as high-speed, low-light, and novel environments.

1.2 Minimalist Philosophy: Purposive Vision

The problem of IMO segmentation has traditionally been thought of in a reconstructionist approach. This approach aims to first reconstruct a three-dimensional geometric description of the world from one or more images and quantitatively recover the properties of the objects in the scene that are relevant to the given task. For example, the IMO segmentation task is addressed as a byproduct of solving the SfM task. This approach leads to extracting redundant information that may not be absolutely required for the completion of a task. On the other hand, the evolution of biological beings has led to the development of dedicated visual modules to extract/process only required information. Light-sensitive cells in the retina come in two main types: rods (receptive to low light) and cones (sensitive to color and bright light). Because of rods' and cones' different functions, some nocturnal animals (such as lab mice) tend to have retinas rich in rods. On the other

hand, diurnal animals like the tree squirrel and tree shrew have higher levels of cone receptors [22]. These purposive functionalities in the biological beings probably gave rise to superior visual capabilities.

Similar to biological beings, it turns out that we can achieve many highly nontrivial visual tasks in navigation with limited resources. This fresh perspective of minimalist philosophy is termed Purposive Vision [23]. Inspired by the purposive vision, I developed bio-inspired motion representation and learning capabilities to solve egomotion and IMO segmentation problems.

1.3 Bio-inspired Motion Representation

1.3.1 Normal Flow

When a robot is in continuous motion, its visual system extracts low-level representation from moving images, and the representation is termed as optical flow field [24, 25]. In practice, it is challenging to extract the optical flow field since it depends on the 3D motion (translation and radial directions of a camera) and the structure of the scene in view. However, normal flow, the component perpendicular to the edges, is the only component of the optical flow that is well defined on the basis of local information [26] (Fig. 1.2). This is the well-known aperture problem (Fig. 1.3). Most reconstructionist approaches estimate the optical flow early in the process. However, many theoretical and empirical findings suggest that initial local motion measurements in biological beings are far away from an optical flow representation [27, 28].

Experiments conducted on primates suggested that there exist neural pathways dedicated to processing dynamic visual motion information in the so-called “where” pathway or dorsal stream [29]. The dorsal visual stream begins in the striate/primary visual cortex (V1), extends

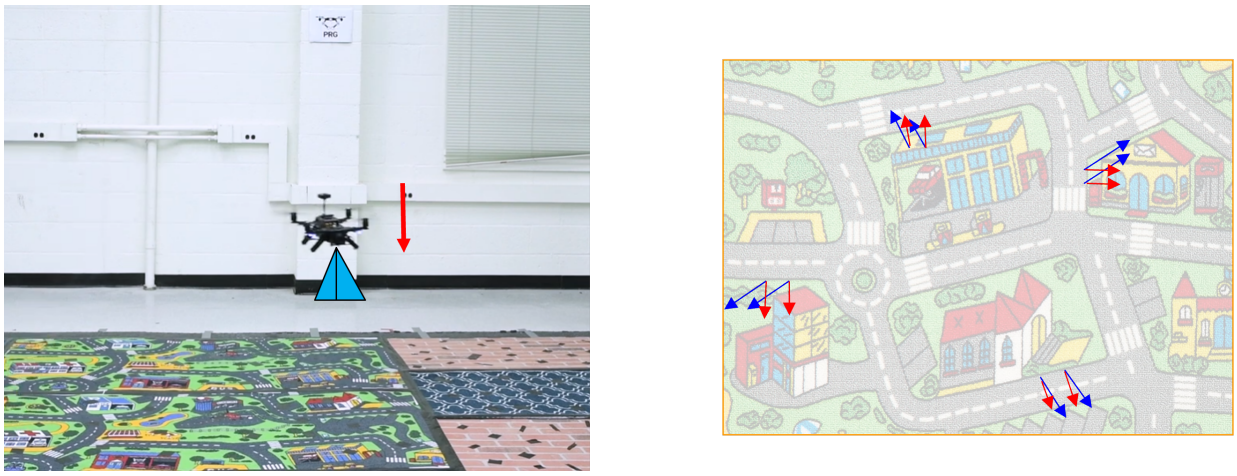

Figure 1.2: The comparison of normal flow vs. optical flow fields. Left: A drone with a down-facing camera is landing vertically on a carpet. Right: Red arrows indicate normal flow, and blue arrows indicate optical flow. The normal flow is the projection of optical flow on the component perpendicular to the edges.

through several extrastriate areas (areas V3, V3A, middle temporal area MT/V5, medial superior temporal area MST), and terminates in higher areas of the parietal and temporal lobes [30]. MT/V5 and MST are classically considered the key motion regions of the dorsal visual stream, being strongly responsive to visual stimuli in motion and showing selectivity for the direction. A recent neuro-imaging study [31] demonstrates that translation and radial direction of optic flow stimuli are signaled independently by neural activity in areas V5/MT and V3A, respectively. This dissociation reveals the existence of separate processing pathways to analyze different attributes of optic flow. It supports that optical flow is estimated in later stages of motion processing rather than early.

Following the inspiration from neural pathways, a better computational approach [32,33] has been developed. Early on in the pipeline, we compute a rough estimate of image motion (normal flow), then we can compute the egomotion and estimate a more accurate optical flow, structure, and IMO segmentation. Despite the advantages of the normal flow formulation, the computational methods to estimate normal flow have not been robust enough to allow deployment in the wild.

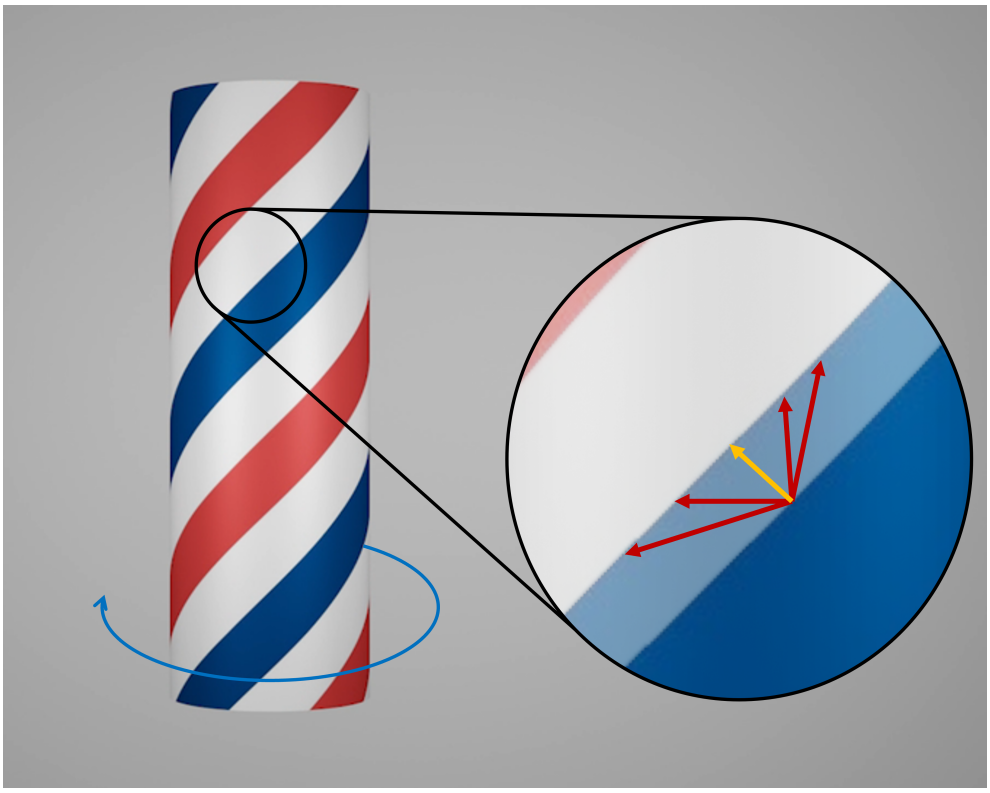

Figure 1.3: The illustration of aperture problem using Barbers' Pole. Line feature observed through a small aperture. It is not possible to determine exactly where each point has moved to. However, normal flow (yellow arrow), the optical flow (red arrow) component perpendicular to the line feature can be computed.

The recent approach [34] in this computational paradigm are limited to simple scenarios. In this thesis, I present a scalable and robust approach for normal flow estimation [35] to aid egomotion estimation in more generic and challenging driving/flying scenarios.

1.3.2 Neuromorphic Event Sensors

Traditionally, cameras acquire visual information through images. The image generation process is used initially only for the purposes of visualization. However, there is no necessity for a robot to acquire a complete image and extract motion cues necessary for navigation afterward. This

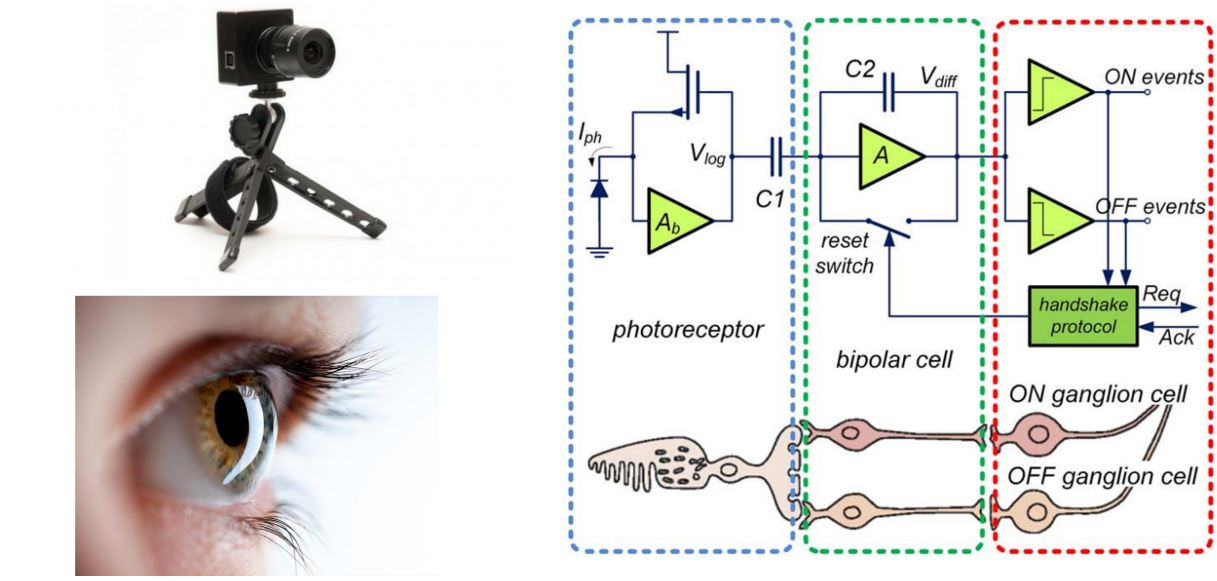

Figure 1.4: Three-layer model of a human retina and corresponding event sensor pixel circuitry [4]

sequential acquiring and processing of visual data is inefficient and is a bottleneck for high-speed navigation tasks.

On the contrary, biological visual systems are more efficient in image acquisition. The retina of humans is a multilayered neural network lining the back hemisphere of the eyeball, where the acquisition and the first stage of processing of the visual information happens. The retina converts spatio-temporal information contained in the incident light from the visual scene into spike trains and patterns, output and conveyed to the visual cortex by retinal ganglion cells, whose axons form the fibers of the optic nerve [4]. In summary, retinal layers can be grouped into four main processing stages: photoreception; transmission to bipolar cells; transmission to ganglion cells; and transmission along the optic nerve (refer Fig1.4).

Inspired by the retinal layers, a new class of sensors called *Event* sensors [36, 37] have been developed (Fig. 1.4). The event camera does not record image frames, but instead - the changes of lighting occurring independently at every pixel. Each of these changes is transmitted

asynchronously and is called an Event, also referred to as Spike. In most cases, brightness changes occur due to the underlying motion of the sensor or the objects in the scene. Hence, due to this unique property, event sensors are suitable for motion perception tasks. The neuromorphic design of event sensors allows the sensor to accommodate a large dynamic range of lighting conditions and provides high temporal resolution and low latency, which are imperative for real-time robot applications.

In recent years, event sensors have attracted a lot of attention from academia and industry. Event sensors are used for object tracking [38, 39], surveillance and monitoring [40], and object/gesture recognition [41, 42], depth estimation [43, 44], optical flow estimation [45, 46], and Simultaneous Localization and Mapping (SLAM) [47–49]. In this thesis, the contributed approaches [50, 51] capitalize on the unique properties of event cameras in the development of low-latency, low-power motion perception solutions.

1.4 Bio-inspired Learning

1.4.1 Learning from Geometry

Recent progress in deep neural networks can overcome the limitations of classical approaches through end-to-end learning from data. Many such learning-based egomotion and IMO segmentation methods [52, 53] have been proposed. Out of multiple learning paradigms, supervised approaches are more popular and have shown promising results on publicly available datasets. However, the major drawback is getting labels for the training data. The label annotation process is laborious and time-consuming. Another disadvantage of a supervised model is that it overtrains on the scene

properties and struggles to generalize to novel/unseen scenarios.

On the other hand, humans and animals develop a rich, structural understanding of the world through the past visual experience without continuous label information. This inspires us to seek an answer to a question: is it possible to train a deep neural network in a self-supervised fashion without annotated labels? This is particularly possible in motion perception tasks due to the geometrical relation between egomotion, 3D structure, and moving objects. Traditional geometrical constraints such as epipolar [18] and cheirality [11] can be utilized to derive annotation-less supervisory signals.

Geometry-based self-supervision approaches [54–57] have paved a new research direction. Furthermore, recent advances in Differentiable Programming [58–60], facilitated new opportunities to formulate geometric relations as a constrained optimization problem and embed them in an end-to-end learning pipeline. Throughout this thesis, I showcase geometry-based self-supervised learning approaches [35, 50, 61] for egomotion and IMO segmentation problems.

1.4.2 Spiking Neural Networks

Artificial neural networks are the de facto learning architectures for a wide range of motion perception problems. The focus of much recent research has been improving accuracy, while other considerations such as the energy and computational resource during the learning process have been of secondary interest [62]. However, the problem lies in deploying large-scale models on resource constraint robots and IoT systems for real-time performance. Over the last few years, multi-disciplinary research efforts have been conducted in this direction. One approach is to look deeper into neuroscience research. The biological brain is known for its excellent resource

management and extreme energy efficiency [63].

Inspired by the brain, an alternative version of Artificial Neural Networks (ANNs) called Spiking Neural Networks (SNNs) aims to replicate living neurons' dynamical system aspects. In contrast to standard ANNs which are essentially networks of complex functions, SNNs are comprised of networks of neurons modeled as differential equations, inherently encode temporal data, and offer low power and highly parallelizable computations similar to biological systems. Furthermore, they can deliver predictions whose confidence incrementally changes with the availability of input data. These networks are intrinsically suitable for directly processing asynchronous irregular data from event-based cameras without pre-processing events.

There has been a renewed interest in using SNNs to process data directly from event-based visual sensors since the sensor produces spike-like activity that fits well with SNN neurons. Applications of SNNs in this domain include classification problems [64] such as digit recognition [65], object recognition [41], gesture recognition [66], and optical flow [67,68]. Recent development of neuromorphic processors such as the Intel Loihi [69] has led to the deployment of SNNs on hardware [67, 70, 71]. Utilizing SNNs, I propose an energy-efficient IMO segmentation approach [51] for event sensor data.

1.5 Research Objectives

To overcome the limitations of traditional motion perception techniques in challenging and resource constrained scenarios, I propose a suite of minimalist bio-inspired solutions. Throughout my doctoral research, I focus on the problem of egomotion estimation and IMO segmentation. Each of my works has a philosophical ideology of purposive vision with a practical application of

immediate impact. I have four research objectives:

- Robust and fast extraction of low-level motion representation (normal flow) in real-world scenarios.
- Motion analysis with neuromorphic event sensors in high-speed and variable lighting conditions.
- Induce geometrical knowledge in the learning mechanism to improve robustness and cross-scenario generalization.
- Implement motion perception approaches in energy-efficient learning frameworks.

1.6 Contributions

This thesis considers the full scope of objectives discussed above and contributes the following algorithms

- *NFlowNet* to estimate normal flow in a deep learning framework. This estimated robust normal flow is useful for applications requiring computationally efficient solutions for navigation tasks in computer vision and robotics (refer to Chapter 2).
- *DiffPoseNet* for estimation of pose from normal flow and qualitative constraint (cheirality or depth positivity). The qualitative constraints are formulating as a differentiable layer, which allows for end-to-end learning in a self-supervised fashion (refer to Chapter 2).
- *0-MMS* for monocular IMO segmentation using event sensors, combining bottom-up deep feature tracking and top-down motion compensation into a unified pipeline (refer to Chapter 3).
- *SpikeMS* demonstrate an encoder-decoder spiking neural network architecture for the problem of IMO segmentation using the event sensors camera as input (refer to Chapter 4).

Next, I briefly introduce each chapter with its key contributions and experimental results.

1.6.1 Chapter 2

Brief Description: Based on [35], we propose *DiffPoseNet* (Fig. 1.5), a novel direct approach for ego-motion estimation in an end-to-end differentiable learning framework. Current deep neural network approaches for camera pose estimation rely on scene structure for 3D motion

estimation, but this decreases the robustness and thereby makes cross-dataset generalization difficult. In contrast, classical approaches to structure from motion estimate 3D motion utilizing optical flow and then compute depth. Their accuracy, however, depends strongly on the quality of the optical flow. To avoid this issue, direct methods have been proposed, which separate 3D motion from depth estimation, but compute 3D motion using only image gradients in the form of normal flow. In this chapter, we introduce a network *NFlowNet*, for normal flow estimation which is used to enforce robust and direct constraints. In particular, normal flow is used to estimate relative camera pose (egomotion) based on the cheirality (depth positivity) constraint. We achieve this by formulating the optimization problem as a differentiable cheirality layer, which allows for end-to-end learning of camera pose. We perform extensive qualitative and quantitative evaluation of the proposed *DiffPoseNet*'s sensitivity to noise and its generalization across datasets. We compare our approach to existing state-of-the-art methods on KITTI, TartanAir, and TUM-RGBD datasets.

1.6.2 Chapter 3

Brief Description: This chapter is based on [50]. We present *0-MMS* (Fig. 1.6) framework for IMO Segmentation in challenging and novel scenarios using event sensors. Without prior knowledge of the object structure and motion, the IMO segmentation problem is very challenging due to the plethora of motion parameters to be estimated while being agnostic to motion blur and occlusions. Event sensors, because of their high temporal resolution, and lack of motion blur, seem well suited for addressing this problem. We propose a solution to multi-object motion segmentation using a combination of classical optimization methods along with deep learning

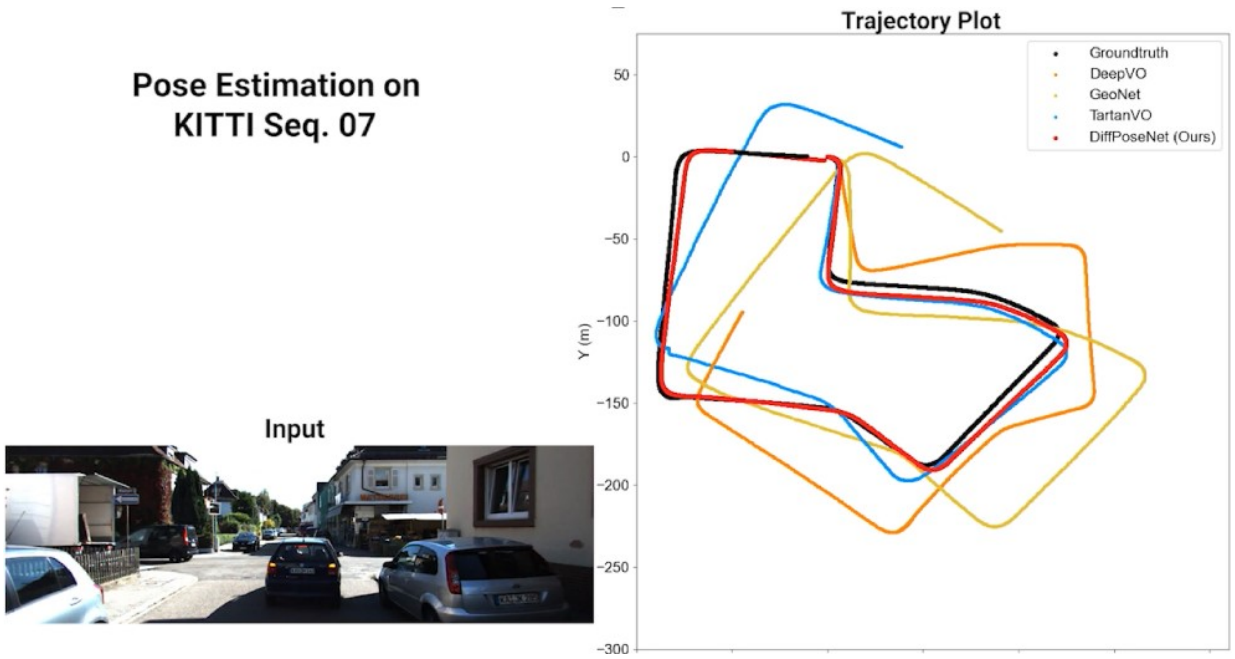

Figure 1.5: *DiffPoseNet*: Egomotion estimation with normal flow and learning from geometry paradigm.

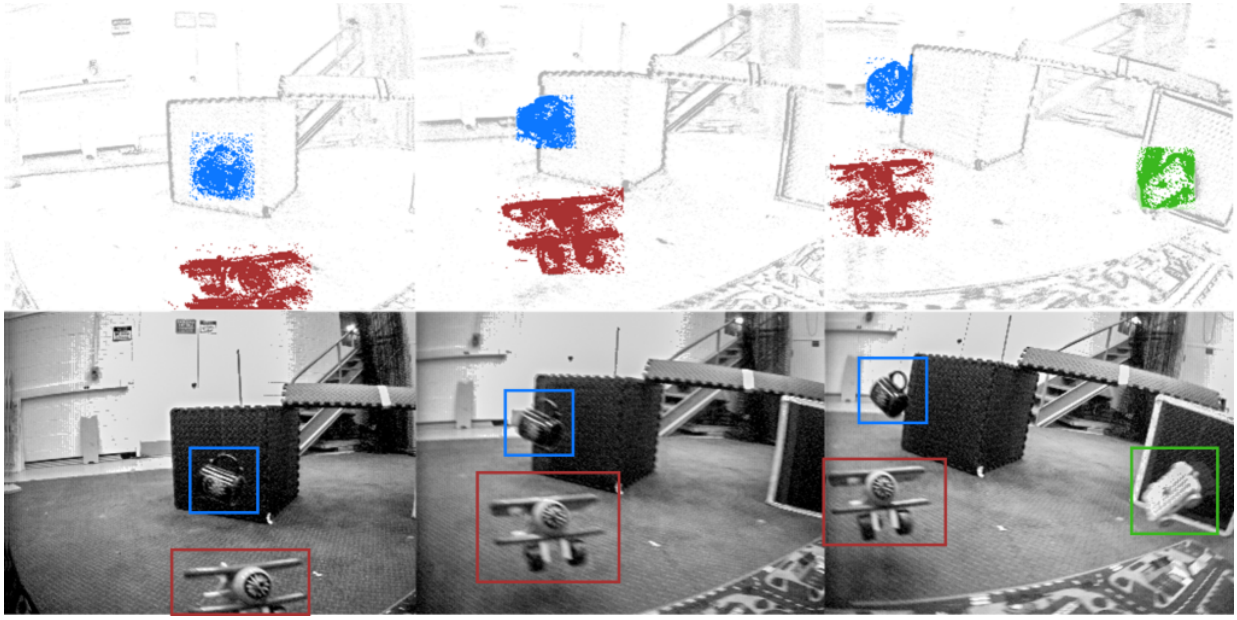

Figure 1.6: *0-MMS*: IMO Segmentation with a monocular event camera on an EV-IMO dataset sequence.

and does not require prior knowledge of the 3D motion and the number and structure of objects. Using the events within a time-interval, the method estimates and compensates for the global rigid motion. Then it segments the scene into multiple motions by iteratively fitting and merging models using as input tracked feature regions via alignment based on temporal gradients and contrast measures. The approach was successfully evaluated on both challenging real-world and synthetic scenarios from the EV-IMO, EED and MOD datasets.

1.6.3 Chapter 4

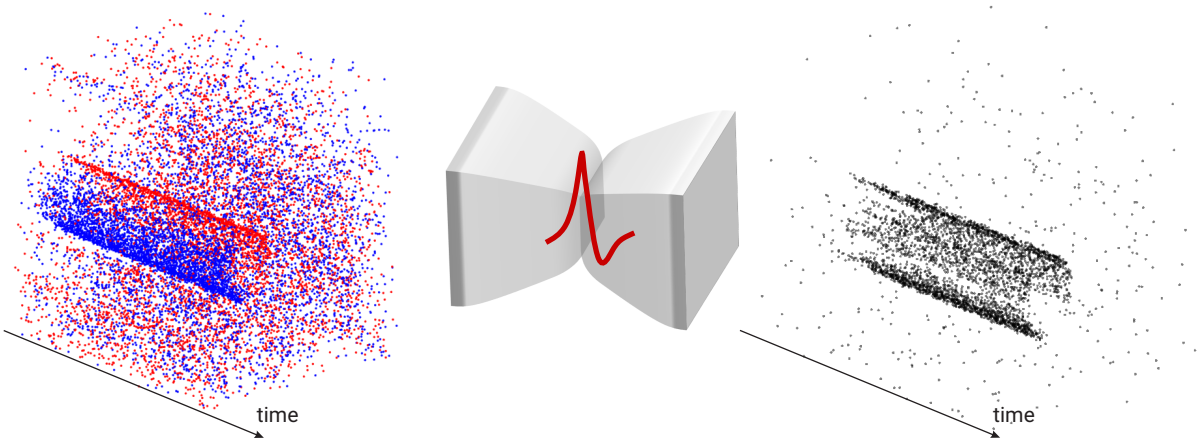

Figure 1.7: *SpikeMS*: Event-based motion segmentation pipeline using a deep spiking neural network.

Brief Description: In this chapter, we present the Spiking Neural Networks (SNN) for the IMO segmentation problem utilizing event sensor data as input. This chapter is based on [51] work. SNNs are called so-called third generation of neural networks which attempt to more closely match the functioning of the biological brain. They inherently encode temporal data, allowing for training with less energy usage and can be extremely energy efficient when coded on neuromorphic hardware. In addition, they are well suited for tasks involving event-based

sensors, which match the event-based nature of the SNN. However, SNNs have not been as effectively applied to real-world, large-scale tasks as standard Artificial Neural Networks (ANNs) due to the algorithmic and training complexity. To exacerbate the situation further, the input representation is unconventional and requires careful analysis and deep understanding. We propose *SpikeMS* (Fig. 1.7), the first deep encoder-decoder SNN architecture for the real-world large-scale problem of motion segmentation using the event-based DVS camera as input. To accomplish this, we introduce a novel spatio-temporal loss formulation that includes both spike counts and classification labels in conjunction with the use of new techniques for SNN backpropagation. In addition, we show that *SpikeMS* is capable of *incremental predictions*, or predictions from smaller amounts of test data than it is trained on. This is invaluable for providing outputs even with partial input data for low-latency applications and those requiring fast predictions. We evaluated *SpikeMS* on challenging synthetic and real-world sequences from EV-IMO, EED and MOD datasets and achieving results on a par with a comparable ANN method, but using potentially 50 times less power.

1.6.4 Chapter 5

Finally, in this chapter, I summarize the key contributions of my work and discuss its scope. I also provide a few potential future directions to address these problems.

1.6.5 List of Publications

1.6.5.1 First Author

[1] **Parameshwara, C. M.**, Hari, G., Fermüller, C., Sanket, N. J., Aloimonos, Y., "DiffPoseNet: Direct Differentiable Camera Pose Estimation", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022*.

Project Webpage: https://prg.cs.umd.edu/DiffPoseNet

[2] **Parameshwara, C. M.***, Li, S.*, Fermüller, C., Sanket, N. J., Evanusa, M. S., Aloimonos, Y., "SpikeMS: Deep Spiking Neural Network for Motion Segmentation", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021* (* equal contribution).

Project Webpage: https://prg.cs.umd.edu/SpikeMS

[3] **Parameshwara, C. M.**, Sanket, N. J., Singh, C. D., Fermüller, C., Aloimonos, Y., "0-MMS: Zero-Shot Multi-Motion Segmentation With A Monocular Event Camera", *IEEE International Conference on Robotics and Automation (ICRA), 2021*.

Project Webpage: https://prg.cs.umd.edu/0-MMS

[4] **Parameshwara, C. M.***, Sanket, N. J.*, Singh, C. D., Kuruttukulam, A., Fermüller, C., Scaramuzza, D., Aloimonos, Y., "EVDodgeNet: Deep Dynamic Obstacle Dodging with Event Cameras", *IEEE International Conference on Robotics and Automation (ICRA), 2020* (* equal contribution).

Project Webpage: https://prg.cs.umd.edu/EVDodgeNet

1.6.5.2 Second/Third Author

[1] Sanket, N. J., Singh, C. D., **Parameshwara, C. M.**, Fermüller, C., de Croon, G.C.H.E., Aloimonos, Y., "EVPropNet: Detecting Drones By Finding Propellers For Mid-Air Landing And Following", *Robotics: Science and Systems (RSS), 2021*.

Project Webpage: https://prg.cs.umd.edu/EVPropNet

[2] Singh, C. D.*, Sanket, N. J.*, **Parameshwara, C. M.**, Fermüller, C., Aloimonos, Y., "NudgeSeg: Zero-Shot Object Segmentation by Repeated Physical Interaction", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021* (* equal contribution).

Project Webpage: https://prg.cs.umd.edu/NudgeSeg

[3] Mitrokhin, A., Fermüller, C., **Parameshwara, C. M.**, Aloimonos, Y., "Event-based Moving Object Detection and Tracking", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018*.

Project Webpage: https://prg.cs.umd.edu/BetterFlow

Chapter 2: DiffPoseNet: Direct Differentiable Camera Pose Estimation

In this chapter, utilizing bio-inspired motion representation and learning paradigm, we propose a novel direct approach for ego-motion estimation. We introduce a network *NFlowNet*, for normal flow estimation which is used to enforce robust and direct constraints. Further, we discuss *DiffPoseNet*, a differentiable framework to estimate relative camera pose based on the normal flow and cheirality (depth positivity) constraint.

2.1 Introduction

The ability to localize is imperative for applications in mobile robotics, and solutions based on vision are often the preferred choice because of size, weight, power constraints and the availability of robust localization methods. Many mathematical frameworks and deep learning approaches have been developed for the problem of visual localization [21, 72] under the umbrella of Visual Odometry (VO) or Simultaneous Localization and Mapping (SLAM). However, their performance is subpar for commonly encountered challenging conditions in-the-wild that involve changing lighting, scenes with textureless regions, and dynamic objects.

Classical approaches [73–76] for localization rely either on sparse feature correspondences between images or on the computation of dense motion fields (optical flow). One of the difficulties in optical flow estimation is bias due to noise [8, 77]. For example, if in a patch there are more

gradients in one direction than another, their estimated optical flow will be biased towards the dominant direction. Even though over the past decade many learning-based approaches have proposed to improve optical flow estimation, this behavior still persists in optical flow approaches. This is demonstrated in Fig. 2.1 and Fig. 2.2, which shows the errors produced by the normal flow algorithm presented in this chapter in comparison to three optical flow algorithms from the literature. As can be seen, all flow algorithms have large errors in regions of non-uniform gradient distributions.

To this end, the pioneers of the field remarked on the observation that the projection of optical flow on the image gradient direction is resilient to the bias and can be computed robustly. This projection is called the *normal flow*. Over the past few decades, a number of methods have been proposed that use the spatial gradient directly for 3D motion recovery. These methods are commonly called direct methods. In principle, such methods are robust and computationally cheaper than flow-based feature based approaches as they use the image brightness directly. However, despite the advantages of the normal flow formulation, the computational methods to estimate normal flow have not been robust enough to allow deployment in the wild. Thus, optical flow has been the go-to representation for ego-motion estimation, supported in recent years by the high accuracy and speed of deep learning algorithms [78–80]. To improve the robustness of camera pose estimation (ego-motion), we propose the first normal flow network *NFlowNet*.

Further, to estimate pose independent of scene structure from normal flow, direct approaches utilize minimal constraints. When optical flow or correspondence are available, pose is estimated using the depth-independent epipolar constraint. However different from these 2D measurements, normal flow is 1D and thus depth cannot be eliminated from the equations relating it to scene geometry and 3D motion. Not making assumptions about the scene structure, the only constraint

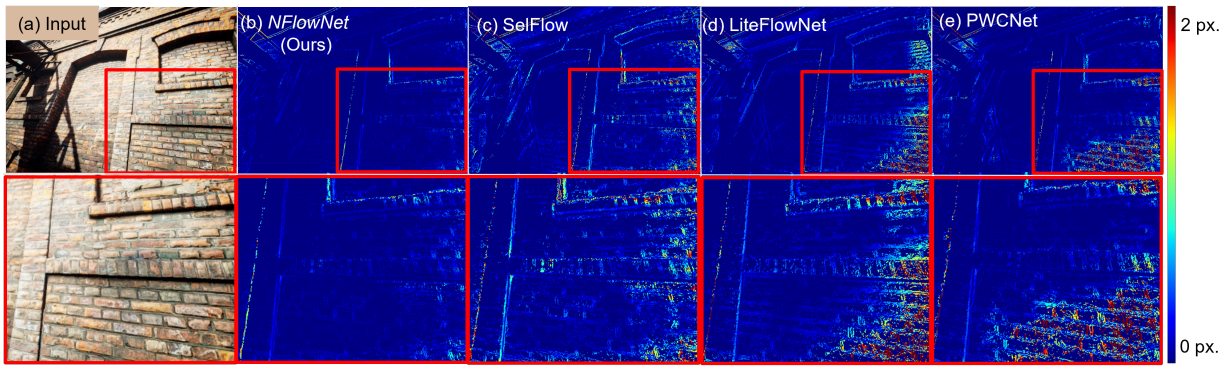

Figure 2.1: Top Row: Projected Endpoint Error (PEE) map of *NFlowNet* compared to three different optical flow approaches (SelFlow [5], LiteFlowNet [6], and PWC-Net [7]). Bottom Row: enlarged endpoint error map of the region highlighted in red in the top row. The optical flow is due to the camera undergoing translation parallel to the wall. It has large errors in regions of non-uniform gradient directions. Notably, on the bricks of rectangular shape, there are many more (vertical) gradients due to the horizontal edges than (horizontal) gradients due to vertical edges, and this causes erroneous flow estimation. Similarly, on the edges of the niches, where there is only one gradient direction, there is error.

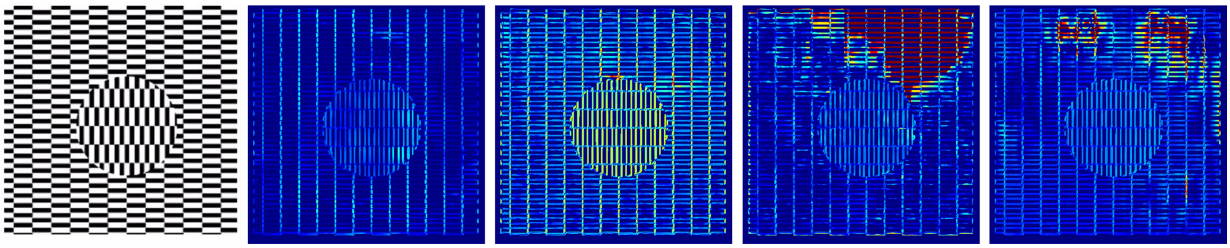

Figure 2.2: Projected Endpoint Error (PEE) map on Ouchi Illusion [8]. In the illusion, the central disk seems to float above the checkered background when moving the eyes around while viewing the figure. For this experiment, we simulated the eye motion by randomly warping the illusion image.

that can be imposed on the scene, is the depth positivity [81] or cheirality constraint [82]. Cheirality states that the scene has to be in front of the camera for it to be imaged, and thus the depth has to be positive. This constraint when enforced on normal flow can be utilized to estimate camera pose without making assumptions on scene depth or shape. Since the cheirality condition is

an inequality constraint and hence is not differentiable, until recently it was not possible to employ it in a deep learning pipeline. To this end, we utilize the differentiable programming paradigm [58] implemented with the implicit differentiation [59] framework to reformulate the cheirality optimization into a differentiable layer and hence train our pose network in an end-to-end fashion.

In this work, we design a novel normal flow network *NFlowNet* and couple it with a differentiable cheirality layer for robust pose estimation. Our contributions (in the order for ease of understanding) can be summarized as follows:

- We introduce a network *NFlowNet* to estimate normal flow. This estimated robust normal flow, beyond this chapter, is useful for applications requiring computationally efficient solutions for navigation tasks in computer vision and robotics.
- We formulate the estimation of pose from normal flow using the cheirality (or depth positivity) constraint as a differentiable optimization layer.
- Extensive qualitative and quantitative experimental results highlighting the robustness and cross-dataset generalizability of our approach without any fine-tuning and/or re-training.

2.2 Related Work

2.2.1 Normal Flow and Camera Pose

Several works have developed direct methods that use normal flow for pose estimation. The idea is that normal flow can be interpreted as "the projection of optical flow on the gradient direction." Thus, given a normal flow vector, the optical flow is constrained to a half-plane [83]. If the 3D motion is only due to translation, this constrains the focus of expansion, i.e., the intersection of the translation axis with the image plane [81] to a half plane. Based on this concept, [84] proposed different algorithms to solve for the case of translation only, and [85] analyzed the method's stability in the presence of small rotations. Modeling the scene as piece-wise planar, [86] solved for 3D motion and calibration [87], and [88] added constraints for combining multiple flow fields. Not making depth assumptions, [19] developed constraints on the sign of normal flow, which geometrically separate the rotational and translational flow components and can be implemented as pattern matching. Others proposed techniques for separating 3D motion components by searching for lines in the image, where certain 3D motion components cancel out [89, 90]. Recently [34] modeled the cheirality constraint by approximating it with a smooth function, which allowed the use of modern optimization techniques. The method first solves for 3D motion from normal flow, and then refines using a regularization defined on depth. Experimental results demonstrate that the proposed pipeline outperforms other flow based approaches. Inspired by these findings, we follow a similar pipeline, but we develop the constraints within a neural network approach.

2.2.2 Learning-based Camera Pose Estimation

Early studies of learning-based camera pose (VO) models [91] [92] [93] were mainly focused on supervised learning approaches modelled as either absolute pose/relative pose regression problems. However, these methods require real-world ground truth poses which is often difficult to obtain. In order to alleviate the need of ground truth data, self-supervised VO was proposed. SfMLearner [94] learns depth and pose simultaneously by minimizing photometric loss between warped and input image. [95] and [96] extend this idea to joint estimation of pose, depth and optical flow. Learning-based models suffer from generalization issues when tested on images from a new environment. Most of the VO models are trained and tested on the same dataset. Most recently, TartanVO [97] addresses generalization issues by incorporating the camera intrinsics directly into the model and training with a large amount of data. Recent advances in differentiable optimization layers (or differentiable programming) [58, 59] has enabled a new generation of generalizable pose learning approaches. [98] embeds an Epipolar geometric constraint into a self-supervised learning framework via bi-level optimization of camera pose and optical flow. BlindPnP [60] embeds geometric model fitting algorithms (PnP algorithm, RANSAC) into implicit differentiation layers. All the above work focus on reconstructing the structure (either through network or optimization) and/or optical flow correspondences for estimating camera motion. In our work, we address robust pose estimation by depending only on structure-less cheirality constraints and normal flow. Finally, we utilize the concepts of best of both-worlds to enable speedup using data prior and novel data generalization from mathematical optimization.

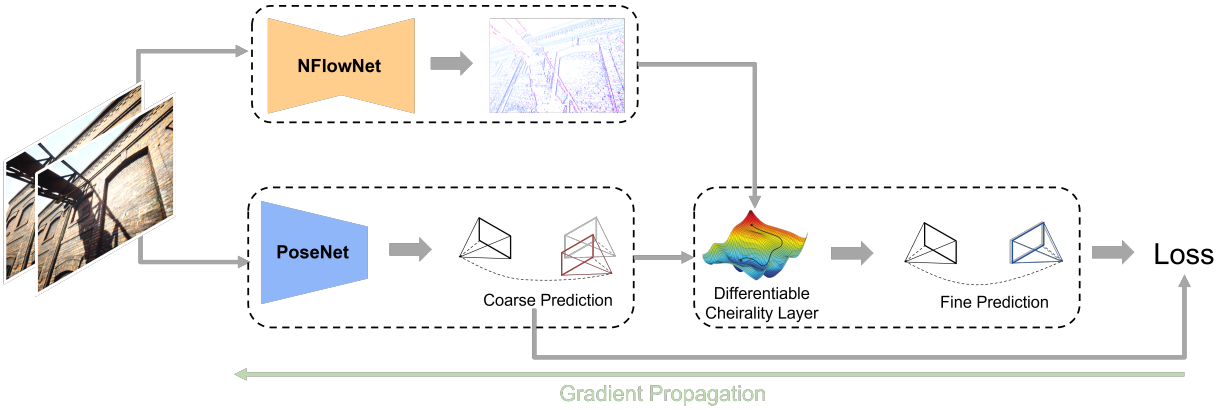

Figure 2.3: Overview of our proposed *DiffPoseNet* framework. Our network starts with a novel normal flow estimation network *NFlowNet* and a first coarse pose estimate. Next, fine pose is estimated using the proposed differentiable cheirality layer.

2.3 Overview of Proposed Approach

The network architecture is illustrated in Fig. 2.3. It consists of *NFlowNet*, a network for estimating normal flow (Section 2.4), which then is used for self-adaptive camera pose estimation (or odometry estimation) (Section 2.5). The camera pose estimation proceeds in three steps. First we *initialize* the PoseNet using supervised training with successive images as input (Sec. 2.5.1). Next, pose is estimated using differentiable optimization by embedding a cheirality constraint defined on normal flow (Sec. 2.5.2). This way, simliar to the classic SfM approach, pose is *estimated independent of depth*. In a last step, the pose in PoseNet is refined through a self-supervised refinement loss by using the pose estimates from the *cheirality constraint* to minimize the error in normal flow.

2.4 *NFlowNet* for Normal Flow Prediction

The first step in motion analysis from video is to compute an image motion representation. Most approaches either detect and track distinct features or compute gradient-based optical flow. The latter is estimated by assuming the intensity $I_{\mathbf{x}}$ (or a function of the intensity) at a point $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ to remain constant over a short time interval δt. This is referred to as the brightness constancy constraint [82]:

$$I(x, y, t) = I(x + u\delta t, y + v\delta t, t + \delta t) \tag{2.1}$$

Here u and v refers to the image pixel motion. Approximating Eq.(2.1) with a first order Taylor expansion, we obtain Eq.(2.2):

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v = -\frac{\partial I}{\partial t} \tag{2.2}$$

We call the component of the flow along the gradient direction, the *normal flow*. Denoting the spatial gradients as $\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$ and the flow as $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^T$, the normal flow vector $\mathfrak{n}$ (a 2d vector) is defined as:

$$\mathfrak{n} = \frac{(\mathbf{u} \cdot \nabla I)}{||\nabla I||_2}\nabla I \tag{2.3}$$

Using the brightness constancy constraint (eq. 2.1), the normal can be computed directly from the spatial and temporal image derivatives, I_t, as $\mathfrak{n} = \frac{-I_t}{||\nabla I||_2}\nabla I$. Since this constraint alone is not sufficient to determine two-dimensional image motion, additional constraints need to be introduced. Traditionally, variational methods combining multiple global smoothness assumptions,

have been the dominant approach for optical flow estimation, and more recently they have been replaced by deep learning algorithms. However, these methods tend to perform subpar on regions with little features or repeated texture due to the strong reliance on the training dataset, and in boundary regions due to oversmoothing, especially when the size of the flow varies significantly in the image. Computing normal flow solely on spatio-temporal derivatives, is unreliable and prone to errors. Hence, we propose a novel Normal flow network called *NFlowNet*.

We use an encoder-decoder convolutional neural network, which we train in a supervised way. Given an image pair, normal flow describes the pixel motion parallel to the image derivatives. To learn normal flow, we utilize the TartanAir dataset. Specifically, we utilize Eq. 2.3 to compute ground-truth normal flow. We train the*NFlowNet* supervised using the l_2 loss between our network predictions $\tilde{\mathbf{n}}$ and ground truth $\hat{\mathbf{n}}$, i.e.,

$$\underset{\tilde{\mathbf{n}}}{\operatorname{argmin}} \|\hat{\mathbf{n}} - \tilde{\mathbf{n}}\|_2 \tag{2.4}$$

In the experimental section (Sec. 2.7), we show that *NFlowNet* generalizes to the real-world and other datasets without any fine-tuning or re-training.

2.5 Self-Adaptive Pose Estimation from Normal Flow

We use a deep network to regress relative poses, that is, the 3D rigid motion of the camera between subsequent time steps and denoted as $\mathbf{P}_t^{t+1}$. The conversions between absolute poses and relative poses is explained next.

If the absolute pose at time t is given by $\begin{bmatrix} \mathbf{T}_t & R_t \end{bmatrix}^T$, where $\mathbf{T}_t \in \mathbb{R}^{3\times 1}$ and $R_t \in SO(3)$. The relative pose between t and $t+1$ under a linear velocity assumption is denoted as $\begin{bmatrix} \mathbf{V} & \mathbf{\Omega} \end{bmatrix}^T$ and is given by

$$\mathbf{V} = \frac{\mathbf{T}_{t+1} - \mathbf{T}_t}{dt}; \quad \mathbf{\Omega}_\times = \frac{\text{logm}\left(R_t^T R_{t+1}\right)}{dt} \tag{2.5}$$

Here, dt is the time increment between t and $t+1$, logm is the matrix logarithm operator and $\mathbf{\Omega}_\times$ converts the vector $\mathbf{\Omega}$ into the corresponding skew symmetric matrix

$$\mathbf{\Omega}_\times = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \tag{2.6}$$

2.5.1 PoseNet for Initializing Pose Estimation

In the first stage, we learn coarse relative poses using a CNN+LSTM. The feature representations learned by the CNN layers are passed to the LSTM for sequential modelling. We use a supervised l_2 loss between the ground truth ($\widehat{\mathbf{P}} = \begin{bmatrix} \widehat{\mathbf{V}} & \widehat{\mathbf{\Omega}} \end{bmatrix}^T$) and predicted poses ($\widetilde{\mathbf{P}} = \begin{bmatrix} \widetilde{\mathbf{V}} & \widetilde{\mathbf{\Omega}} \end{bmatrix}^T$). Here, $\mathbf{V}$ and $\mathbf{\Omega}$ represent the translational and rotational parts of the pose. The orientation is represented in $X - Y - Z$ Euler Angles. Denoting as λ a weighting parameter, we solve the following optimization using backpropagation:

$$\underset{\tilde{\mathbf{P}}}{\text{argmin}} \left(\|\widehat{\mathbf{V}} - \widetilde{\mathbf{V}}\|_2^2 + \lambda \|\widehat{\mathbf{\Omega}} - \widetilde{\mathbf{\Omega}}\|_2^2 \right) \tag{2.7}$$

2.5.2 Differentiable Cheirality Layer for Fine Pose Estimation

To enable self-supervised learning of continuous pose (given a initialization value), we propose to utilize the cheirality constraint or depth positivity constraint, which states that all world points have to be in front of the camera, i.e., have positive depth. This condition has been classically used in structure from motion problems to disambiguate the physically possible camera poses from the set of computed solutions. The main reason for utilizing the cheirality rather than the acclaimed epipolar constaint or scene planarity constraint is due to the minimalism in the assumptions. Since in our formulation, depth positivity is enforced using the normal flow, which in-turn makes minimal assumptions about the scene structure, our formulation generalizes to novel scenes with remarkable accuracy (Sec. 2.6.2.2).

Let us mathematically define the constraints. Denoting the magnitude of the normal flow (a scalar) at pixel $\mathbf{x}$ as $n_{\mathbf{x}}$ and the direction of the image gradient as $\mathbf{g}_{\mathbf{x}}$ (a unit vector), we have

$$n_{\mathbf{x}} = \|\mathfrak{n}\|_2 = \frac{1}{Z_{\mathbf{x}}}(\mathbf{g}_{\mathbf{x}} \cdot A)\mathbf{V} + (\mathbf{g}_{\mathbf{x}} \cdot B)\mathbf{\Omega}, \tag{2.8}$$

where, $\mathbf{V}$ denotes the constant translational velocity and $\mathbf{\Omega}$ the rotational velocity $\mathbf{\Omega}$ in a small time instant, and $Z_{\mathbf{x}}$ is the depth of the point under consideration. Intuitively, $\mathbf{V}, \mathbf{\Omega}$ warp the flow field and $Z_{\mathbf{x}}$ scales the flow field, and the matrices A and B determine how the motion flow field is projected onto the image plane due to translational and rotational velocities respectively and are given by

$$A = \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} \tag{2.9}$$

$$B = \begin{bmatrix} xy & -(x^2+1) & y \\ (y^2+1) & -xy & -x \end{bmatrix} \tag{2.10}$$

Let us consider in eq. (2.8) the two components of normal flow: the translational component, which depends on depth, and the rotational component, which is independent of depth. If we subtract the rotational component from both sides, we obtain

$$n_{\mathbf{x}} - (\mathbf{g}_{\mathbf{x}} \cdot B)\mathbf{\Omega} = \frac{1}{Z_{\mathbf{x}}}(\mathbf{g}_{\mathbf{x}} \cdot A)\mathbf{V}. \tag{2.11}$$

We can enforce that the left hand side (the derotated normal flow) and the right hand side (the translational component) must have same sign. Since the depth ($Z_{\mathbf{x}}$) is positive, the following product, which we denote as $\rho_{\mathbf{x}}(\mathbf{V}, \mathbf{\Omega})$, is positive, i.e.,

$$\rho_{\mathbf{x}}(\mathbf{V}, \mathbf{\Omega}) = ((\mathbf{g}_{\mathbf{x}} \cdot A)\,\mathbf{V}) \cdot (n_{\mathbf{x}} - (\mathbf{g}_{\mathbf{x}} \cdot B)\,\mathbf{\Omega}) > 0 \tag{2.12}$$

To arrive at an objective function to be used in an optimization, we can model the cheirality constraint by passing $\rho_{\mathbf{x}}$ through a smooth function, such as the ReLU function [34]. Since, the deep learning pipeline requires the function to be twice differentiable, we choose the GELU function, a smooth approximation of the ReLu function. Denoting the negative GELU function as $\mathcal{R}$, and denoting the average over all $\mathbf{x}$ values as $\mathbb{E}$, we then obtain the following minimization,

for the estimation of relative camera pose:

$$\underset{\{\mathbf{V},\mathbf{\Omega}\}}{\operatorname{argmin}} \mathbb{E}\left(\mathcal{R}(\rho_{\mathbf{x}}(\mathbf{V},\mathbf{\Omega}))\right) \tag{2.13}$$

In the stage of re-estimating motion, we simply use constraint (2.13) in an optimization implemented by a robust Quasi-Newton optimization algorithm. We solve the optimization sequentially, in one step for V and in the other for Ω, because $\rho_{\mathbf{x}}(\mathbf{V},\mathbf{\Omega})$ is bilinear in these parameters. The initial estimate comes from the PoseNet estimate in Sec. 2.5.1. In our implementation we use the L-BFGS algorithm [99]. These steps (Sec. 2.5.1 and 2.5.2) are performed in the forward pass in the network.

2.5.3 Self-Supervised Refinement

Let us denote the coarse pose obtained from our *PoseNet* as $\widetilde{\mathbf{P}}_c = \begin{bmatrix} \widetilde{\mathbf{V}}_c & \widetilde{\mathbf{\Omega}}_c \end{bmatrix}$. This is further refined by the Cheirality layer and we denote this refined pose as $\widetilde{\mathbf{P}}_r = \begin{bmatrix} \widetilde{\mathbf{V}}_r & \widetilde{\mathbf{\Omega}}_r \end{bmatrix}$. Now we use $\widetilde{\mathbf{P}}_r$ to refine our *PoseNet*'s coarse pose to obtain a more accurate prediction of pose.

The final self-adaptive pose estimation is performed as a bi-level minimization in the network [59], in which an upper-level problem is solved subject to constraints imposed by a lower-level problem and is formally defined next.

$$\begin{aligned} &\underset{\widetilde{\mathbf{P}}_c}{\operatorname{argmin}} \mathbb{E}\left(n_{\mathbf{x}} - \mathbf{g}_{\mathbf{x}} \cdot \left(\left(\frac{n_{\mathbf{x}} - (\mathbf{g}_{\mathbf{x}} \cdot B)\widetilde{\mathbf{\Omega}}_r}{(\mathbf{g}_{\mathbf{x}} \cdot A)\widetilde{\mathbf{V}}_r}\right) A\widetilde{\mathbf{V}}_c - B\widetilde{\mathbf{\Omega}}_c\right)\right) \\ &\text{subject to} \quad \underset{\widetilde{\mathbf{P}}_r}{\operatorname{argmin}} \mathbb{E}\left(\mathcal{R}(\rho_{\mathbf{x}}(\widetilde{\mathbf{V}}_c, \widetilde{\mathbf{\Omega}}_c))\right) \end{aligned} \tag{2.14}$$

The lower-level problem (second row) of Eq. (2.14) enforces the cheirality constraint to obtain pose $\widetilde{\mathbf{P}}_r$, which then is used to compute the normal flow error in the upper-level loss function (first row). The upper-level loss enforces the consistency between the normal flow from *NFlowNet* and that computed using the motion parameters $\widetilde{\mathbf{P}}_c$ with the implicit depth term expressed by the motion parameters $\widetilde{\mathbf{P}}_r$.

In practice, we back-propagate through the upper-level to refine poses $\widetilde{\mathbf{P}}_\mathbf{c}$ using supervision from $\widetilde{\mathbf{P}}_\mathbf{r}$ through the lower-level. Using implicit differentiation, all that is computed from the lower layer is the gradient, and this step is agnostic to the optimizer used. Specifically, we derive $\frac{\partial \widetilde{\mathbf{P}}_r}{\partial \widetilde{\mathbf{P}}_c}$, which is computed from the product of second order derivatives. (The interested reader is referred to [59], eq. (15) for details.) It is important to note that we rely on the generalizability of *NFlowNet*, hence it is not fine-tuned.

2.6 Experiments

2.6.1 Implementation Details

2.6.1.1 Datasets

We use eight environments from the TartanAir [1] dataset (`amusement`, `oldtown`, `neighborhood`, `soulcity`, `japanesealley`, `office`, `office2`, `seasidetown`) for training and two environments (`abandonedfactory` and `hospital`) for testing our *NFlownet* network. For odometry evaluation, we use the Tartan challenge test data [1]. We also conduct extensive experiments on the KITTI Odometry [3] and the TUM-RGBD [2] datasets to evaluate the robustness and generalization performance of our proposed system.

2.6.1.2 Networks and Optimization Layer

For the *NFlowNet* we use an encoder-decoder architecture based on EVPropNet [100] to directly regress sparse normal flow. The encoder contains residual blocks with convolutional layers and the decoder contains residual blocks with transpose convolutional layers. We choose the number of residual and transposed residual blocks as 2 and the expansion factor (factor by which the number of neurons are increased after every block) as 2. We backpropagate the gradients using a mean squared loss computed between groundtruth and predicted normal flow as given in Eq. 2.4. We used the Adam optimizer to train our network with a learning rate of 10^{-4} and batch size of 8 for 400 epochs.

Our *PoseNet* architecture is inspired from [91] and uses the VGG-16 encoder for the CNN stage [101] and two LSTM layers each with 250 hidden units for the recurrent layer-stage. We initially train this model with a subset of the TartanAir data for 30 epochs to obtain a coarse estimate to initialise the Cheirality Layer. We use the Adam optimizer and set a fixed learning rate of 10^{-5}. We consider sequences of six consecutive image frames and a batch size of eight while training. During test time, we use only two consecutive image frames to estimate the relative camera pose between images I_t and I_{t+1}.

For the optimization layer we use the L-BFGS [102] solver. The line search function was set to strong Wolfe [103], the number of iterations were set to 100, and the gradient norms were clipped to 100. We initialised the optimizer with coarse predictions provided by our *PoseNet*. Our overall system is implemented in Python 3.7 and PyTorch 1.9.

2.6.1.3 Training and Testing Procedure

The whole training schedule consists of three stages. First, we only train *NFlowNet* and *PoseNet* in a supervised manner using the training strategies mentioned above. Then we freeze *NFlowNet* and jointly train the *PoseNet* with the cheirality layer in an self-supervised fashion via the refinement loss. The self supervised training is carried out for 120 epochs using four Nvidia P6000 GPUs. For the cheirality layer, the stopping criteria is when the objective function is below 10^{-20} or the number of iterations exceeds 300.

During testing, our final pose predictions are obtained by passing *PoseNet* priors through the cheirality layer along with the predictions from *NFlowNet*. Due to our self-supervised refinement training, the prior *PoseNet* estimates help in faster convergence of the cheirality optimization process (takes less than 5 iterations). Overall, NFlowNet takes 15 ms and the coarse PoseNet takes 8 ms. The Cheirality layer takes an average of 8 ms per iteration to refine the estimates. The inference time is obtained for an image resolution of 320×640 using a Nvidia 2070 MaxQ GPU.

2.6.1.4 Evaluation Metrics

To evaluate our *NFlowNet* and to compare against other optical flow networks, we project the optical flow obtained from the state-of-the-art approaches on the gradient direction, and we measure the pixel error. As normal flow is defined only along the gradient direction, we utilize the Projection Endpoint Error (PEE), an analog to the Average Endpoint Error (AEE) error metric proposed in [104]. The PEE between the projected optical flow ($\widetilde{\mathbf{u}}$) and the groundtruth ($\hat{\mathbf{n}}$) is defined as:

$$\text{PEE} = \left\| \hat{\mathbf{n}} - \frac{\nabla I}{||\nabla I||_2^2} \cdot \widetilde{\mathbf{u}} \right\| \tag{2.15}$$

To evaluate the regressed relative poses from our model, we use Absolute Trajectory Error and Relative Pose Error metrics.

2.6.2 Analysis of Experimental Results

2.6.2.1 Accuracy of Normal Flow

In the first case study, we quantitatively evaluate *NFlowNet*. We compare our network with optical flow methods of various flavours: (a) supervised (PWC-Net [7], LiteFlowNet [6]), and (b) self-supervised (SelFlow [5]).

In Table 2.1, we present a quantitative evaluation of normal flow. We trained our *NFlowNet* and fine-tuned optical flow networks with TartanAir (8 environments). We demonstrate the PEE error on the first four sequences of the environments `abandonedfactory` and `hospital`. *NFlowNet* performs better than the optical flow networks by up to 6×. By learning normal flow, we constrain the network to focus on prominent features (edges, textures) rather than dense correspondences in textureless regions. Through this "attention-like" mechanism, *NFlowNet* performs better than its optical flow counterparts with upto 46× smaller model size.

We also tested our NFlowNet against supervised flow networks (including RAFT [105]) and on the TartanAir and KITTI datasets. Referring to Table 2.2, we observe that directly regressing normal flow via NFlowNet performs better in comparison to projecting the produced SOTA flow predictions onto the image gradient. The weaker performance of the compared networks may be due to optical flow bias because of the correlation layer in RAFT, PWC-Net, and LiteFlowNet.

Since the correlation layer performs multiplicative patch comparisons between feature maps which imitates classical matching of correspondences, it is likely to suffer from the aperture problem [106].

Table 2.1: PEE (Projection Endpoint Error) ↓ of different state-of-the-art methods as compared to of our *NFlowNet*.

Method	abandonfactory				hospital				Num.
	000	001	002	003	000	001	002	003	param.(M) ↓
LiteFlownet [6]	2.56	1.82	1.93	2.15	3.17	2.68	2.45	1.93	5.37
PWC-Net [7]	1.23	0.95	1.64	1.48	2.35	2.92	2.28	1.47	8.75
UnFlow [107]	1.35	1.15	1.75	1.56	2.21	1.76	1.83	1.07	126.90
SelFlow [5]	**0.67**	0.73	**0.52**	0.64	1.91	0.51	0.73	**0.65**	5.11
NFlowNet (Ours)	0.72	**0.54**	0.57	**0.63**	**0.82**	**0.44**	**0.57**	0.71	**2.72**

Table 2.2: PEE (Projection Endpoint Error) ↓ of different state-of-the-art methods as compared to of our *NFlowNet*.

Method	TartanAir		KITTI 2015	Num.
	abandon	hospital		param.(M) ↓
LiteFlownet [6]	2.12	2.57	3.94	5.37
PWC-Net [7]	1.32	2.25	3.18	8.75
RAFT [105]	0.83	1.25	1.52	5.3
NFlowNet (Ours)	0.61	0.64	1.37	2.72

2.6.2.2 Comparison of Pose Estimation

In this section we compare our *DiffPoseNet* framework with various state-of-the-art camera pose estimation approaches. These approaches can be broadly be classified into: (a) pure deep learning models, (b) pure geometric constraint based models, and (c) hybrid deep learning models that incorporate some form of geometric constraints in the learning pipeline.

We present the Absolute Trajectory Error (ATE) of our model on the TartanAir challenge data in the `MH000-007` sequences and compare it to the results of TartanVO and ORB-SLAM in Table 2.3. We outperform both methods in this experiment by up to 3.4×.

We present the ATE for the TUM-RGBD sequences (`360`, `desk`, `desk2`, `rpy`, `xyz`) in Table 2.4. We achieve competitive results, because TUM RGBD is difficult for monocular vision methods due to rolling shutter, motion blur and large rotation. This is where pure geometric constraints show a massive advantage. We believe our work can further be improved by compensating for rolling shutter in the differential layer and we see this as a potential future research direction. It is important to stress that our method was trained on the TartanAir dataset and tested directly on other datasets to highlight cross-dataset generalization without any fine-tuning or re-training.

We also present the relative pose errors, specifically, the average translational RMSE drift (t_{rel} in %) and average rotational RMSE drift (r_{rel} in °/100m) for comparison in the KITTI dataset sequences `06`, `07`, `09` and `10` (See Fig. 2.5). The error metrics are computed on a trajectory of length of 100–800 m. In Table 2.5 we compare our model (*DiffPoseNet*) with (a) pure deep learning models (TartanVO [97], GeoNet [95], UnDeepVO [108], DeepVO [91], Wang et al. [109]), (b) pure geometric constraint based approaches (ORB-SLAM [110], VISO2-M [111]) and with (c) BiLevelOpt [98], a method implementing the epipolar constraint via a differentiable layer. The test sequences were selected such that they do not overlap with the training sets in the deep learning models used for comparison. Note that, similar to TartanVO, our training is performed only on TartanAir and do not perform any fine-tuning or re-training on KITTI dataset. Regardless, we perform competitive to other approaches that are trained/fine-tuned on similar data, thus demonstrating our cross-dataset generalization.

Table 2.5 also presents our ablation study, comparing different configurations in our pipeline on KITTI. 'Ours (no-SS)' denotes the results of the coarse PoseNet fine-tuned on KITTI ground-truth poses. 'Ours (no-CL)' denotes DiffPosenet without cheirality layer refinement during inference time, and 'Ours (CL 1-iter)' denotes DiffPoseNet with the refinement restricted to only one iteration of the cheirality layer. 'Ours (OF)' denotes the results of DiffPoseNet where the normal flow is computed by projecting optical flow on image gradients, instead of NFlowNet. We noticed that 'Ours' and 'Ours (OF)' are capable of outperforming learning approaches, particularly when there are rotation errors, which is notable because learning methods have shown higher rotation errors compared to geometric methods.

We infer from the above results (also see Fig. 2.6) that deep learning models usually outperform classical geometric constraint based methods in translation errors (t_{rel}), which can be attributed to the scale drift issue. This sometimes might be solved by performing expensive global bundle adjustment and loop closure. However, we are not using a loop closing procedure in the models in our experiments. Models with differentiable optimizer layers, like *DiffPoseNet* (Ours) and BilevelOpt [98], achieve the best of both worlds, with lower relative rotation errors competitive with geometric methods like ORB-SLAM.

Table 2.3: ATE (m) ↓ on the MH sequences of TartanAir [1] dataset.

Methods	000	001	002	003	004	005	006	007
ORB-SLAM [110]	**1.30**	**0.04**	2.37	2.45	-	-	21.47	2.73
TartanVO [97]	4.88	0.26	2.00	0.94	1.07	3.19	**1.00**	2.04
Ours	2.56	0.31	**1.57**	**0.72**	**0.82**	**1.83**	1.32	**1.24**

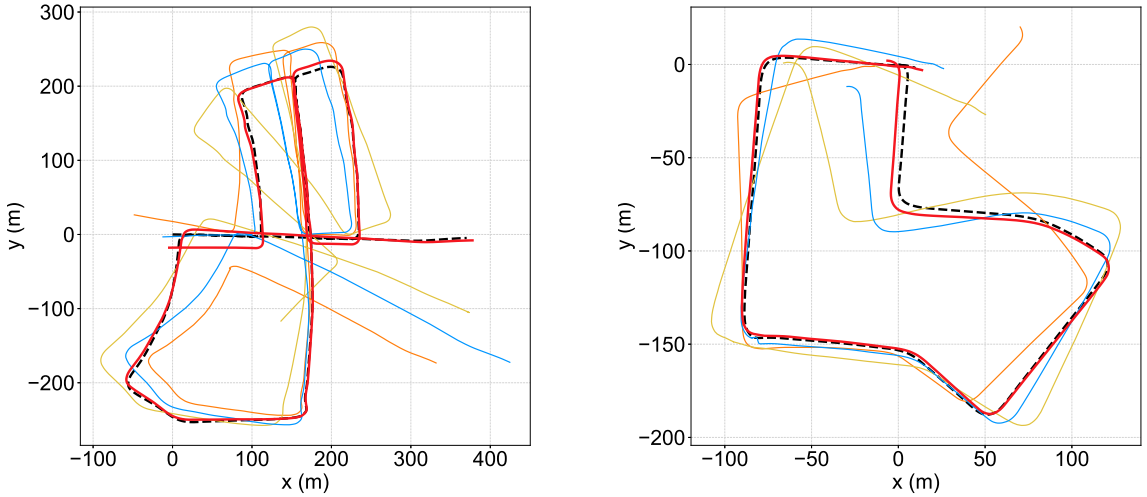

Figure 2.4: Qualitative comparison of trajectories between our *DiffPoseNet* and other state-of-the-art approaches on the KITTI dataset sequences `Seq.05` (left) and `Seq.07` (right)

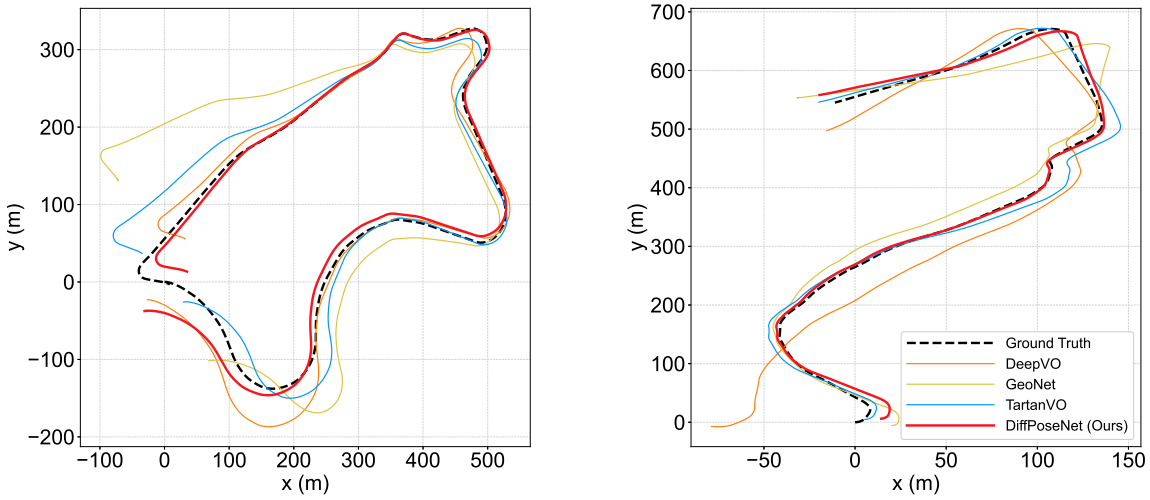

Figure 2.5: Qualitative comparison of trajectories between our *DiffPoseNet* and other state-of-the-art approaches on the KITTI dataset `Seq.09` (left) and `Seq.10` (right)

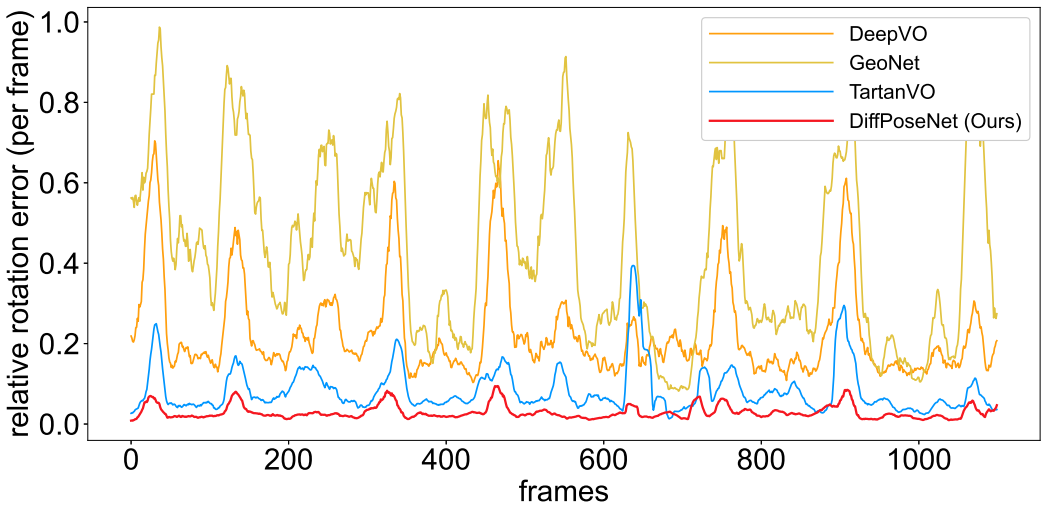

Figure 2.6: Comparison between our model and pure learning based VO on relative rotation error (in °/frame) from `KITTI - 07`.

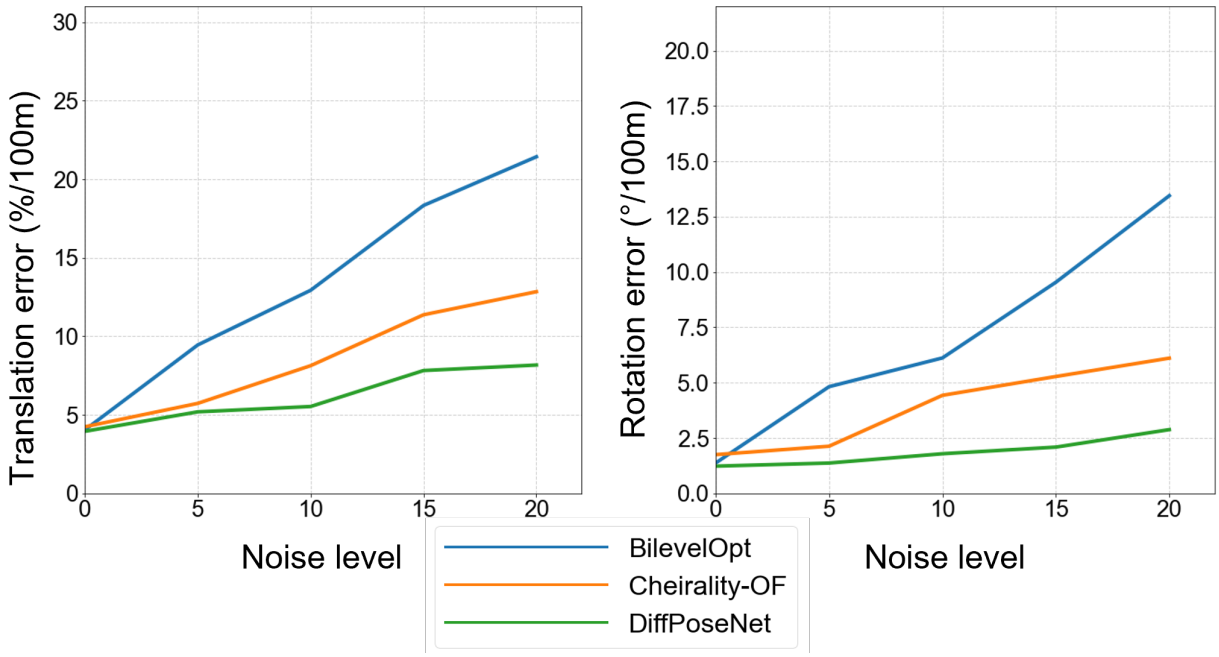

Figure 2.7: Robustness evaluation of normal flow representation for pose estimation on `KITTI-10`.

Table 2.4: ATE (m) ↓ on the TUM-RGBD [2] benchmark.

Methods	360	desk	desk2	rpy	xyz
ORB-SLAM2 [112]	-	**0.016**	0.078	-	**0.004**
DeepTAM [113]	**0.116**	0.078	0.055	0.052	0.054
TartanVO [97]	0.178	0.125	0.122	**0.049**	0.062
Ours	0.121	0.101	**0.053**	0.056	0.048

Table 2.5: Relative Pose Error (t_{rel} and r_{rel}) ↓ results of various Pose estimation methods on KITTI [3] dataset. Note that **bold** represents the best result and underline represents the second best.

Method	06		07		09		10	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
DeepVO [91]	5.42	5.82	3.91	4.60	-	-	8.11	8.83
Wang et al. [109]	-	-	-	-	8.04	1.51	6.23	0.97
UnDeepVO [108]	6.20	1.98	**3.15**	2.48	-	-	10.63	4.65
GeoNet [95]	9.28	4.34	8.27	5.93	26.93	9.54	20.73	9.04
TartanVO [97]	4.72	2.95	4.32	3.41	6.03	3.11	6.89	2.73
BiLevelOpt [98]	-	-	-	-	4.36	0.69	4.04	1.37
ORB-SLAM [110]	18.68	**0.26**	10.96	**0.37**	15.3	**0.26**	**3.71**	**0.3**
VISO2-M [111]	7.34	6.14	23.61	19.11	4.04	1.43	25.2	3.84
Ours (no-SS)	5.23	3.15	4.83	3.92	7.12	4.31	8.33	3.74
Ours (no-CL)	4.23	2.43	4.28	2.76	5.13	3.18	5.89	2.98
Ours (CL-1-iter.)	3.19	2.03	4.13	2.53	4.72	1.71	4.82	2.57
Ours (OF)	3.03	2.08	3.89	2.13	4.24	0.72	4.12	1.56
Ours	**2.94**	1.76	4.06	2.35	**4.02**	0.51	3.95	1.23

2.6.2.3 Robustness of Pose Estimation

For most camera pose estimation approaches the performance of the algorithm is also governed by external factors such as lighting, weather, and sensor noise. These external factors often lead to errors in motion fields and cause pose estimation to fail or diverge. In this case study we present a robustness analysis by injecting noise into the motion fields.

We study the robustness of the normal flow based cheirality layer against the epipolar layer [98]. We artificially inject errors in the normal flow and optical flow and evaluate our framework's performance under these conditions. The error is modelled as additive uniform noise $\mathcal{U}(\epsilon)$, where $[-\epsilon, \epsilon]$ is the bound on noise. We induce this noise to both normal flow and optical flow only on the gradient direction where normal flow is well-defined to make the comparison fair.

Fig. 2.7 shows the relative pose errors, t_{rel} and r_{rel}, over ϵ values $\{0, 5, 10, 15, 20\}\%$. Here, BiLevelOpt refers to the pose estimation using the epipolar constraint layer [98] with optical flow as input. "Cheirality-OF" refers to the pose estimation via the cheirality layer using normal flow obtained by projecting SelFlow optical flow predictions on the gradient directions (we choose SelFlow as it has the best performance among all optical flow methods used in this chapter). Observe that, our *DiffPoseNet* is more resilient to noise and fails "more gracefully" compared to other methods. We owe this robustness to the lack of strong constraints used in our approach as compared to other methods that either rely on strong features or strong photometric consistency. We believe that carefully crafted optimization problems can lead to robust pose estimation neural networks that generalize well to novel datasets while being robust to noise, a capability rarely seen in most state-of-the-art methods [114, 115].

2.7 Conclusion

In this work, we combined the best of both worlds from classical direct camera pose estimation approaches and deep learning taking advantage of differentiable programming concepts. Specifically, we addressed the problem of estimating relative camera pose using a sequence of images. To achieve this, we introduced the *DiffPoseNet* framework. As part of this framework,

we introduced a normal flow network called *NFlowNet*, that predicts accurate motion fields under challenging scenarios and is more resilient to noise and bias. Furthermore, we proposed a differentiable cheirality layer that when coupled with *NFlowNet* can estimate robust and accurate relative camera poses.

A comprehensive qualitative and quantitative evaluation was provided on the challenging datasets: TartanAir, TUM-RGBD and KITTI. We demonstrated the efficacy, accuracy and robustness of our method under noisy scenarios and cross-dataset generalization without any fine-tuning and/or re-training. Our approach outperforms the previous state-of-the-art approach. Particularly, NFlowNet can output accurate motion fields at up to $6\times$ the speed with up to $46\times$ smaller model size. We believe this will open a new direction for camera pose estimation problems.

Chapter 3: 0-MMS: Zero-Shot Multi-Motion Segmentation With A Monocular Event Camera

In this chapter, we focus on the application of neuromorphic event sensors for the problem of IMO Segmentation in challenging and novel scenarios. We propose *0-MMS*, a solution to multi-object IMO segmentation using a combination of classical optimization methods along with deep learning and does not require prior knowledge of the 3D motion and the number and structure of objects.

3.1 Introduction

Navigation is a fundamental competence of life with visual motion estimation as its beating heart [116–118]. Even though motion estimation has seen a tremendous advancement in the last few decades, dynamic object motion is usually addressed by outlier rejection schemes as a part of the mature structure from motion and SLAM pipelines [119]. Though, this can provide an initial dynamic object segmentation, further processing for each segment relies on some prior information (commonly appearance/structure/recognition).

To exacerbate the scenario further, classical imaging cameras often fail in dynamic scenarios (moving objects) due to motion blur and low light scenarios. To this end, drawing inspiration from nature, neuromorphic engineers developed a sensor called *Dynamic Vision Sensor* (DVS) [120]

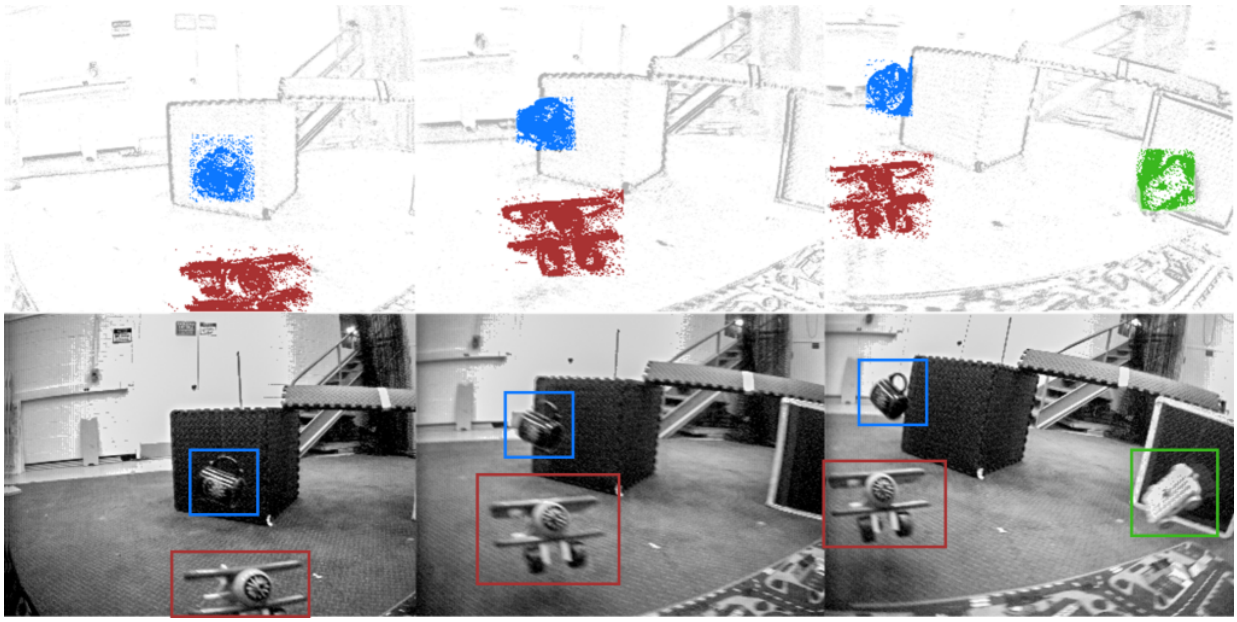

Figure 3.1: Multi-Motion Segmentation with a monocular event camera on an EV-IMO dataset sequence. Top Row: The event frames are color-coded by cluster membership. The corresponding grayscale frames are shown in the bottom row. Bounding boxes on the images are color coded with respect to the objects for reference. *Note that grayscale images are not used for computation and are provided for visualization purposes only*.

which records the asynchronous temporal changes in the scene in the form of a stream of events, rather than the conventional image frames. This gives an unparalleled advantage in-terms of temporal resolution, low latency, and low band-width signals. Such event data is tailor-made for motion segmentation because of the disparity in event density at object boundaries.

In this chapter, we present a method to detect moving objects by inferring their motion using a monocular event camera; we call this *multi-motion segmentation*. We formally define the problem statement and our contributions next.

3.1.1 Problem Formulation and Contributions

We address the following question: *How do you cluster the scene into background and Independently Moving Objects (IMOs) based on motion using data from a moving monocular event camera?*

Given an event volume $\mathcal{E}$, we find and cluster the events based on 2D motion. We over-segment the scene with the help of feature tracks and then merge clusters based on the motion models and a contrast score. Each cluster is represented by a four parameter motion model (denoting the similarity transformation/warp) $\Theta = \{\Theta_x, \Theta_y, \Theta_z, \Theta_\theta\}$ which represents the 2D translation (Θ_x, Θ_y), divergence and in-plane rotation, respectively [121]. To speed up computation, we propagate these motion models until a cluster keyslice is invoked. A summary of our contributions is given below (Sample outputs are shown in Fig. 3.7):

- A novel cluster splitting and merging approach for monocular event-based multi-motion segmentation without prior knowledge of scene geometry (zero-shot) and a number of objects.
- New open-source multi-motion segmentation dataset and benchmark MOD++ including extensive motion stratification and including challenging collision/exploding sequences.
- Speeding up computation using motion propagation and introduction of cluster keyslices.

3.1.2 Related Work

There has been extensive progress in the field of event-based motion segmentation in the past decade for different scenarios at variable scene complexity. Earlier works focused on the case of a static camera, where the events are generated by the moving objects, and a simple clustering scheme can provide motion segmentation [122–125]. The next mark up in complexity is the case of a moving camera, where event alignment is computed for the whole scene [121, 126, 127] (also called sharpness or contrast measure [128, 129]) and the parts of the scene which are misaligned give the motion segmentation of IMOs [121] using a simple thresholding algorithm. The results were further improved by [130], who used an Expectation-Maximization scheme to obtain better segmentation. Our work is closely related to [130] with the same underlying philosophy: using motion compensation for clustering but adds robustness in long term segmentation using feature tracking and cluster splitting and merging. We also introduce motion propagation and the concept of cluster keyslices to speed-up the entire procedure.

Finally, a few approaches used machine learning [131] to learn object contours and border-ownership information via a structured random forest, which they demonstrated for segmentation. [61, 132] demonstrated a combination of supervised and unsupervised CNN learning using deblurring/event-alignment in the cost function, and [133] designed a graph convolutional neural network for supervised motion segmentation, that uses as input event volumes over extended time periods.

3.2 Preliminaries

3.2.1 Data From An Event Camera

A traditional camera records frames at a fixed frame rate by integrating the number of photons for the chosen shutter time for all pixels *synchronously*. In contrast, an event camera only records the polarity of logarithmic brightness changes *asynchronously* at each pixel. If the brightness at time t of a pixel at location $\mathbf{x}$ is given by $I_{t,\mathbf{x}}$ an event is triggered when $\| \log\left(I_{t+\delta t,\mathbf{x}}\right) - \log\left(I_{t,\mathbf{x}}\right) \|_1 \geq \tau$. Here, δt is a small time increment and τ is a threshold which will determine the trigger of an event (τ is set at the driver level as a combination of multiple parameters). Each event outputs the following data: $\mathbf{e} = \{\mathbf{x}, t, p\}$, where $p = \pm 1$ denotes the sign of the brightness change. We'll denote events in a spatio-temporal window as $\mathcal{E}_t = \{e_i\}_{i=1}^{N}$ (N is the number of events) and we'll refer to $\mathcal{E}$ as event slice/stream/cloud/volume.

3.2.2 Model Fitting For Contrast Maximization

Processing event cloud is generally computationally very expensive and to speed up the processing we use a projection function. The projection of $\mathcal{E}$ leads to a "blurry" image, and a number of methods for measuring this blurriness to achieve event-cloud alignment (also called contrast maximization or motion compensation or deblurring) have been developed [128, 129, 134]. The output of the alignment is an event frame denoted as $\mathcal{E}_t$. In particular, we utilize the method proposed in [121] to estimate model parameters Θ to maximize the alignment $\mathcal{E}$ by minimizing temporal gradients $\nabla\mathcal{T}$. Here $\mathcal{T}\left(\mathcal{E}\right) = \mathbb{E}\left(t_{\mathbf{x}} - t_0\right)$, $\mathbb{E}$ is the expectation/averaging operator, $t_{\mathbf{x}}$ denotes the time value at location $\mathbf{x}$, and t_0 is the initial time of the temporal window. Formally, we

solve the following optimization problem: $\text{argmin}_{\Theta} \|\nabla \mathcal{T}\|_2$, where ∇ denotes the spatial gradient operator. Note that the projection function denoted by $\mathcal{W}$ is also called a warping function and refers to the creation of $\mathcal{E}_t$ using parameters Θ.

3.2.3 Tracklets Using Point Tracker

We rely on obtaining tracklets (feature tracks across multiple event frames) as an input to multi-motion segmentation. Over the past few years, robust feature extraction and tracking approaches for event data have been proposed [127, 135–138], however most of the robust methods are relatively slow or not open-source or use conventional intensity images. Hence, we adapt SuperPoint [139] (previously used on grayscale images) for extracting tracklets $\mathbb{T}_t$ from a set of consecutive N event frames $\{\mathcal{E}_{t+\Delta t \times i} | i \in [0, N-1]\}$. We found the SuperPoint tracker to be robust and generalizable over a wide range of scenarios without any fine tuning. The SuperPoint tracker runs in the backend (we call this Tracker backend) on a First-In First-Out (FIFO) buffer of consecutive N event slices.

3.3 Proposed Approach

3.3.1 Overview

The proposed solution comprises of two steps: 1. *Split and Merge* summarized in Algorithm 1 and illustrated in Fig. 3.2. 2. *Motion Propagation and Cluster keyslices* summarized in Algorithm 2.

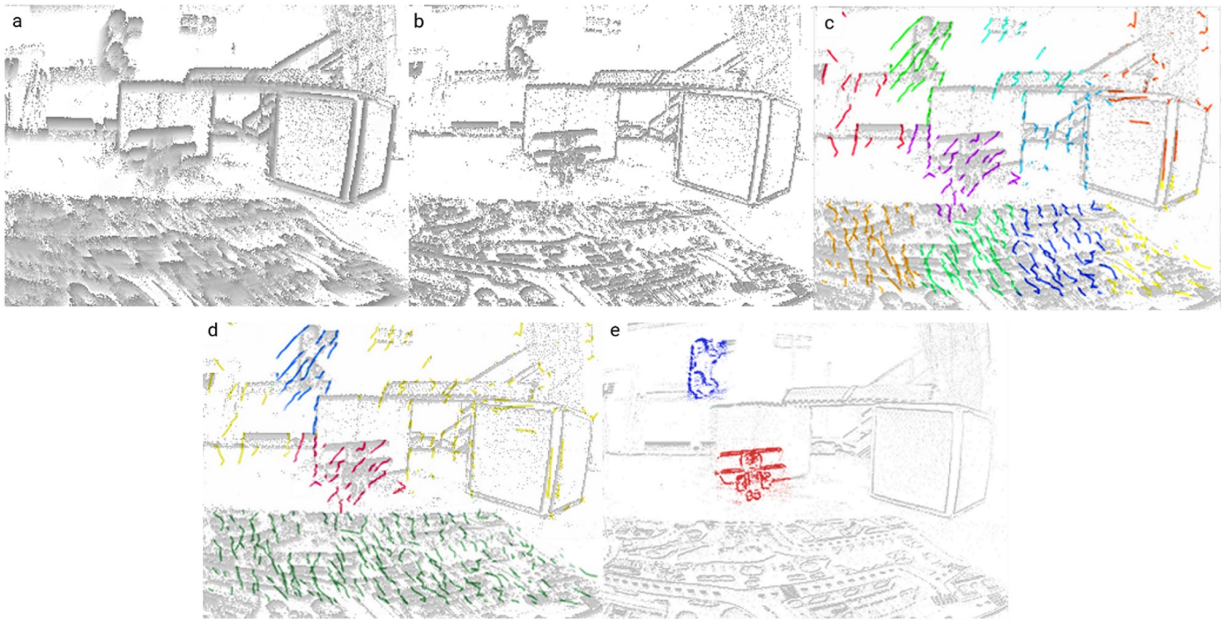

Figure 3.2: Overview of the proposed pipeline on a sequence from EV-IMO dataset (a) Projection of the raw event cloud $\mathcal{E}_t$ without motion compensation, (b) Projection of event cloud after global motion compensation ($\mathcal{E}_t$), (c) Sparse tracklets $\mathbb{F}_t$ extracted on compensated event cloud, (d) Merged feature clusters based on contrast and distance metrics ($\{\mathbb{C}_t\}$), (e) Output of the pipeline is the cluster of events. The cluster membership is color coded where gray color indicating background cluster.

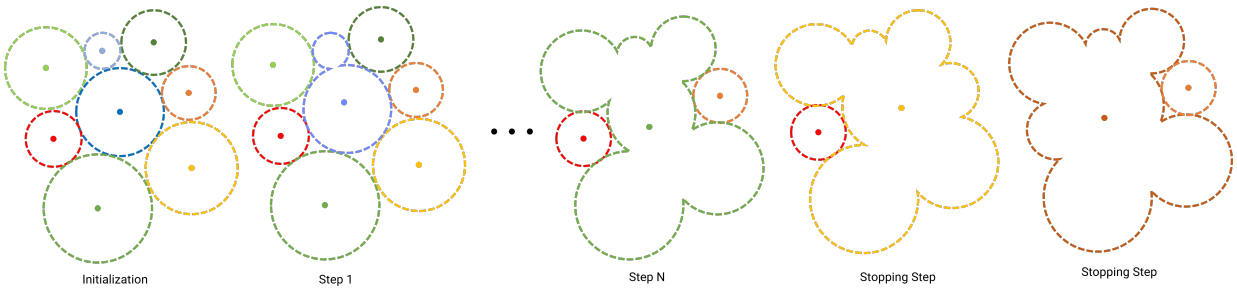

Figure 3.3: Iterative model fitting and merging approach. The colors indicate the average temporal gradient for that particular cluster. (Blue indicates a low value and red indicates a high value). We stop merging whenever the average temporal gradient increases drastically after a merging step since this indicates a merger of the background with an IMO cluster.

Algorithm 1: Splitting and Merging

Data: Tracklets $\mathbb{T}_t$, Event Stream $\mathcal{E}_t$, Num. oversegments K
Result: Clusters $\{\mathbb{C}_t\}$, Cluster Models $\{\Theta_t\}$, Segmentation Masks $\{\mathcal{S}\}$
`Splitting`;
$\{\widetilde{\mathbb{C}}_t\}$ = k-Means($\mathbb{T}_t$, K);
`Merging`;
$\{\widetilde{\Theta}_t\}$ = ClusterModelFit($\mathcal{E}_t$, $\{\widetilde{\mathbb{C}}_t\}$);
while *Stopping Criterion and All Clusters Visited* **do**
  if $\mathcal{C}_{k,j}\mathcal{D}_{k,j}^{-1} > \zeta$ *(Merging Criterion)* **then**
    $\{\widetilde{\mathbb{C}}_t\}, \{\widetilde{\Theta}_t\}$ = MergeClusters($\mathcal{E}, \{\widetilde{\mathbb{C}}_t\}, \{\widetilde{\Theta}_t\}$) Updated Clusters and Motion Models;
  end
end
$\{\mathbb{C}_t\}$ = $\{\widetilde{\mathbb{C}}_t\}$ Final Clusters;
$\{\Theta_t\}$ = $\{\widetilde{\Theta}_t\}$ Final Models;
$\mathcal{S}$ = ConvexHull($\{\mathbb{C}_t\}$) Final Dense Segmentation;

Algorithm 2: Motion Propagation And Cluster Keyslices

Data: Tracklets $\mathbb{T}_t$, Event Stream $\mathcal{E}_t$, Clusters $\{\mathbb{C}_{t-1}\}$, Cluster Models $\{\Theta_{t-1}\}$
Result: Clusters $\{\mathbb{C}_t\}$, Cluster Models $\{\Theta_t\}$
foreach *Cluster* i **do**
  $\mathcal{E}_t^i = \mathcal{W}(\mathcal{E}_t^i, \mathbb{C}_{t-1}^i, \Theta_{t-1}^i)$ Cluster Event frame;
end
if $\mathbb{E}(\mathcal{C}(\mathcal{E}_t^i)\ \forall i) > \alpha$ *Scene Contrast Measure;*
then
  foreach *Cluster* i **do**
    if $\mathcal{C}(\mathcal{E}_t^i) > \chi$ **then**
      Keep Propagation;
      no new cluster keyslice required;
    else
      Split and Merge on current cluster (Algorithm 1);
      New cluster keyslice;
    end
  end
else
  Split and Merge on entire scene (Algorithm 1);
  New cluster keyslice for all clusters (scene);
end

3.3.2 Notations

Before we explain the algorithm, we need to define the notations used formally. Tracklets $\mathbb{T}_t^i$, event stream $\mathcal{E}_t^i$, clusters $\mathbb{C}_t^i$, cluster models Θ_t^i at time t for the i^{th} cluster. If a superscript is omitted, the quantity represents the entire data available at time t. Contrast function $\mathcal{C}_{k,j}$ and distance function $\mathcal{D}_{k,j}$ are defined between two clusters k and j. The user chosen thresholds are ζ (merging criterion), λ (stopping criterion), α (scene contrast measure), and χ (propagation threshold).

3.3.3 Split And Merge

Splitting: Here, the tracklets from the backend are clustered into k clusters ($k >>$ Num. of objects) using k-Means clustering for its simplicity and speed (similar to [140, 141]). If a prior on the number of objects or a bound is known, it can be trivially incorporated to choose k. We denote these clusters as $\mathbb{C}_t$.

Merging: Since the splitting method oversegments the scene, we need to merge the clusters to obtain motion segmentation for Independently Moving Objects (IMOs) and the background. The cluster merging is based on a similarity measure that depends on the contrast match (warping a cluster with the model from another cluster and measuring the contrast) and distance between centroids of the clusters (refer Fig. 3.3). We define contrast and distance functions $\mathcal{C}_{k,j}$ and $\mathcal{D}_{k,j}$ respectively as follows: $\mathcal{C}_{k,j} = \mathbb{E}\left(\|\text{Var}\left(\mathcal{E}(\delta\mathcal{E}_j|\Theta_k)\right)\|_1\right)$ and $\mathcal{D}_{k,j} = \|\mathbf{C}_k - \mathbf{C}_j\|_2$ where k, j are the cluster numbers, $\delta\mathcal{E}_j$ represents the event volume for cluster j and $\mathbf{C}_k$ denotes the centroid of cluster k. This formally entails solving the following optimization problem: $\text{argmax}_j \ \mathcal{C}_{k,j}\mathcal{D}_{k,j}^{-1}$

which simultaneously maximizes the contrast and minimizes the distance. The larger value of $\mathcal{C}_{k,j}\mathcal{D}_{k,j}^{-1}$ ensures a higher probability that motion model of j^{th} cluster is similar to that of k^{th} cluster (as applying this motion model on $\delta\mathcal{E}_j$ yields maximum contrast). Hence, both clusters belong to one. This step is iteratively performed per cluster (where merging happens with every neighboring cluster using breath first search) until a stopping criterion has been reached. The entire process is repeated until all the clusters have been visited. The stopping criterion is explained next.

After each merging operation, we compute the motion model $\Theta_{k,j}$ of the merged clusters by minimizing the temporal gradients $\nabla\mathcal{T}$. Intuitively, when two clusters are merged, the combined motion model captures the average motion of the two clusters thereby slightly increasing the average temporal gradients. Further, whenever a moving object cluster is merged with the background or different IMO cluster the average temporal gradient increases drastically. Hence, a difference in temporal gradient at every step i ($\mathbb{E}\left(\|\nabla\mathcal{T}_i\|_2\right)$) with respect to the initial step ($\mathbb{E}\left(\|\nabla\mathcal{T}_0\|_2\right)$) is computed. If at any step the difference in temporal gradient is large, we terminate the current merge and continue to the next iteration until all the clusters have been visited. This is mathematically described by $\|\mathbb{E}\left(\|\nabla\mathcal{T}_i\|_2\right) - \mathbb{E}\left(\|\nabla\mathcal{T}_0\|_2\right)\|_1 \geq \lambda$, where λ is some constant threshold.

After split and merge has been performed, we obtain the final clusters $\{\mathbb{C}_t\}$ ($\{\}$ indicates a set of clusters and each cluster can be indexed with a superscript, i.e., $\mathbb{C}_t^i$ for i^{th} cluster) and motion models per cluster Θ_t^i where i indexes the cluster number. Optionally, we obtain the dense segmentation by taking the convex hull of the sparse feature points in each cluster, which is denoted as $\mathcal{S}$. Refer to Algorithm 1 for a summary of split and merge methods.

3.3.4 Motion Propagation

Optimizing the parameters Θ at every time step to obtain $\mathcal{E}_t$ from $\mathcal{E}_t$ is computationally exorbitant. Inspired by classical tracking pipelines, we propagate motion models from previous to current time slice assuming linear event trajectories. The propagation is achieved using tracklets, and we validate the propagation quality using the following contrast function $\mathcal{C}_t = \mathbb{E}\left(\|\text{Var}\left(\mathcal{E}(\delta\mathcal{E}_t|\Theta_{t-1})\right)\|_1\right)$. Here, $\mathcal{C}_t$ measures the deviation from the current optimal motion model with respect to the motion model of the previous time slice.

3.3.5 Speeding-Up Computation Using Cluster Keyslices

Classical Visual Odometry pipeline utilizes the concept of keyframes to speed-up computation based on certain conditions, this avoids performing bundle adjustment on every frame whilst maintaining good accuracy. We employ a similar strategy of keyframes for event slices, which we call a *keyslice*, for re-clustering based on contrast function $\mathcal{C}_t$. We apply the contrast function at different levels starting from the entire scene to each clusters separately. Depending on the following two measures the split and merge procedures are performed either per cluster or on the entire scene: 1. Cluster contrast score, 2. Scene contrast score. The cluster contrast score is defined by $\mathcal{C}_t$ when applied to a single cluster during motion propagation and the average of all cluster scores is called the scene contrast score. Refer to Algorithm 2 for a summary of motion propagation and cluster keyslicing.

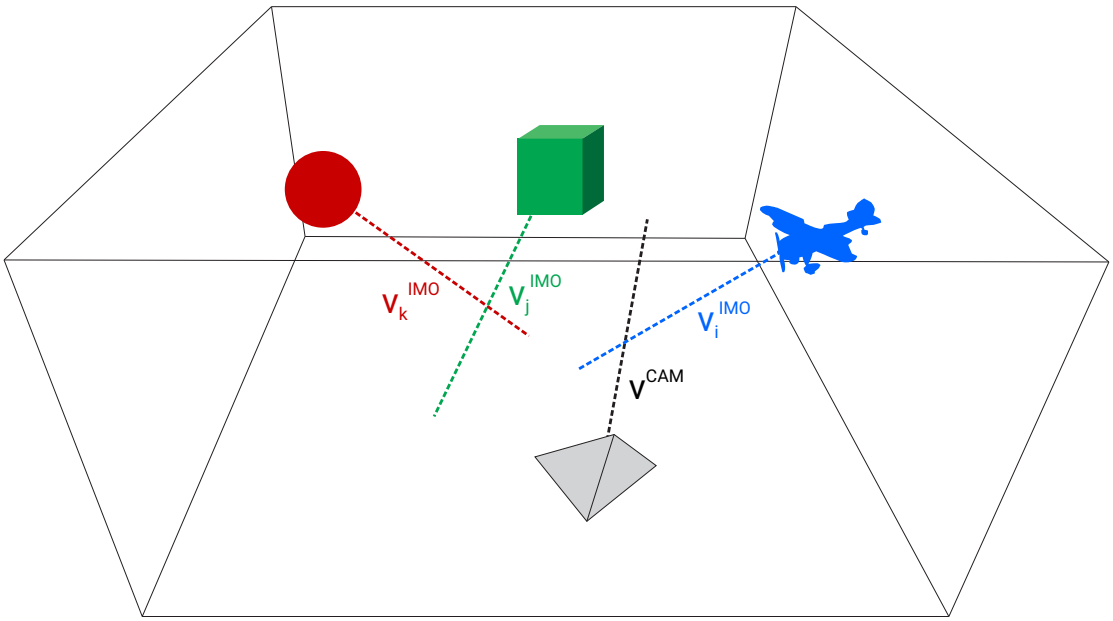

Figure 3.4: Velocity vectors v^{CAM} and v_i^{IMO} used for data stratification in the MOD++ dataset.

3.4 MOD++ Dataset/Benchmark

Currently, three main datasets exist for IMO segmentation using event cameras, namely, Extreme Event Dataset (EED) [121] and EV-IMO [132] featuring real data and the synthetic Moving Object Dataset (MOD) [61]. However none of the datasets have data stratification based on camera and/or object motion and/or velocities. To this end, we extend the MOD dataset presented in [61] which we call MOD++ to add additional synthetic sequences (Refer to Table 3.1). This data stratification is explained next. Let the instantaneous velocity of the center of mass of the camera and IMOs be denoted as v^{CAM} and v_i^{IMO}, where i is the IMO index (Fig. 3.4). Now consider the angle and relative-magnitude between two vectors (a, b) denoted by θ and η respectively and defined as follows: $\theta\,(a,b) = \cos^{-1}\left(\frac{a\cdot b}{\|a\|\|b\|}\right)$ and $\eta\,(a,b) = \frac{\|a\|}{\|b\|}$. Also, let the instantaneous rotational velocity around it's principal axes be denoted by ω where the superscripts and subscripts have the same meaning as that of linear velocities. We classify the sequences as follows: 1.

Different linear velocities: Here we set the instantaneous rotational velocity close to zero, i.e., $\left\|\omega^{\text{CAM}}\right\| \approx 0$ and $\left\|\omega_i^{\text{IMO}}\right\| \approx 0$. We classify the motions based on relative angle and speed. If $\theta\left(v^{\text{CAM}}, v_i^{\text{IMO}}\right) \in [0, 45]^{\circ} \forall i$ we call this sequence small angle. If $\theta\left(v^{\text{CAM}}, v_i^{\text{IMO}}\right) \in [60, 120]^{\circ} \forall i$ we call this sequence medium angle and if $\theta\left(v^{\text{CAM}}, v_i^{\text{IMO}}\right) \in [140, 180]^{\circ} \forall i$ we call this sequence large angle. (Note that we wrap the angles in the range [0, 180] in our case).

If $\eta\left(v^{\text{IMO}}, v_i^{\text{CAM}}\right) \in [0.5, 3.0)\ \forall i$ we call this sequence slow. If $\eta\left(v^{\text{IMO}}, v_i^{\text{CAM}}\right) \in [3.0, 7.0)\ \forall i$ we call this sequence medium and if $\eta\left(v^{\text{IMO}}, v_i^{\text{CAM}}\right) \in [7.0, 10.0]\ \forall i$ we call this sequence fast.

If $\left\|\omega^{\text{CAM}} - \omega_i^{\text{IMO}}\right\| \in [0, 5]^{\circ} s^{-1} \forall i$ we call this sequence slow rotation. If $\left\|\omega^{\text{CAM}} - \omega_i^{\text{IMO}}\right\| \in [25, 30]^{\circ} s^{-1} \forall i$ we call this sequence medium rotation and if $\left\|\omega^{\text{CAM}} - \omega_i^{\text{IMO}}\right\| \in [90, 100]^{\circ} s^{-1} \forall i$ we call this sequence fast rotation.

To make it easy to identify the sequence we use the following naming convention: `SeqSEQNUM_ATTR1_ATTRN` where `SEQNUM` is the sequence number which will determine the scene setup (room walls and objects with texture), `ATTR1` to `ATTRN` are modifiers which specify speed and/or rotation classifications. We use the following modifiers: `AS, AM, AL` for small, medium and large angles, `SS, SM, SL` for small, medium and large linear speeds, `RS, RM, RL` for small, medium and large rotational speeds. For eg., `Seq4_AM_SL_RS` would be the fourth sequence with medium angles, large linear speeds and small rotational speeds.

Additionally, we also provide two challenge sequences for researchers to evaluate their algorithm on: `Cube` and `Cup`. The `Cube` sequence is two cubes (a smaller cube on top of a larger cube) falling on the ground and breaking into smaller non-cube pieces. The `Cup` sequence is a bullet hitting a cup and shattering it into smaller fragments of different shapes.

Table 3.1: Overview Of Related Datasets.

	MOD++	EV-IMO [132]	EED [121]	MOD [61]
Year	2020	2019	2018	2019
Data-type	Simulated	Real	Real	Simulated
Camera	Sim. DAVIS346C	DAVIS346C	DAVIS240B	Sim. DAVIS346C
Data	Events RGB Images @ 1000 Hz 6-DoF Camera Poses @ 1000 Hz 6-DoF IMO Poses @ 1000 Hz IMO Bounding Boxes @ 1000 Hz IMO Masks @ 1000 Hz Optical Flow @ 1000 Hz Depth @ 1000 Hz	Events Grayscale Images @ 40 Hz 6-DoF Camera Poses @ 200 Hz 6-DoF IMO Poses @ 200 Hz IMO Masks @ 40 Hz	Events Grayscale Images @ 20 Hz IMO Bounding Boxes @ 20 Hz	Events RGB Images @ 1000 Hz 6-DoF Camera Poses @ 1000 Hz 6-DoF IMO Poses @ 1000 Hz IMO Bounding Boxes @ 1000 Hz IMO Masks @ 1000 Hz
Poses Ground Truth (Acc.)	Blender® Engine (Sub. mm)	12 × Vicon® Vantage V8 Cameras (≈ 1mm)	–	Blender® Engine (Sub. mm)
IMO Bounding Boxes (Masks) Ground Truth	Blender® Engine for both	Scanned 3D Objects projected using Ground Truth Pose	Hand-labelled (–)	Blender® Engine for both
Num. Sequences	43	30	5	7
Num. Unique Objects (Max. Number of Objects in frame)	12 (9)	4 (3)	7 (3)	9 (3)
Num. Backgrounds	11	5	5	9
Challenging Scenes	Exploding and Breaking objects	Fast camera motion	Extreme illumination	–
Data Stratification	Velocity Direction Velocity Magnitude IMO Rotation Magnitude	–	–	–

3.5 Experiments and Results

We evaluate our approach on publicly available real and synthetic datasets. We demonstrate our approach's performance both qualitatively and quantitatively employing two different metrics based on the availability of groundtruth information.

3.5.1 Detection Rate

For datasets which provide timestamped bounding boxes for the objects, we consider the prediction as success when the estimated bounding box fulfills two conditions; (1) it has a overlap of more than at least 50% with the ground-truth bounding box, (2) the area of intersection with the ground-truth box is higher than the intersection with outside area. We can formulate the metric as:

$$\text{Success} \quad \text{if} \quad \mathcal{D} \cap \mathcal{G} > 0.5 \quad \text{and} \quad (\mathcal{D} \cap \mathcal{G}) > (\neg\mathcal{G} \cap \mathcal{D}) \tag{3.1}$$

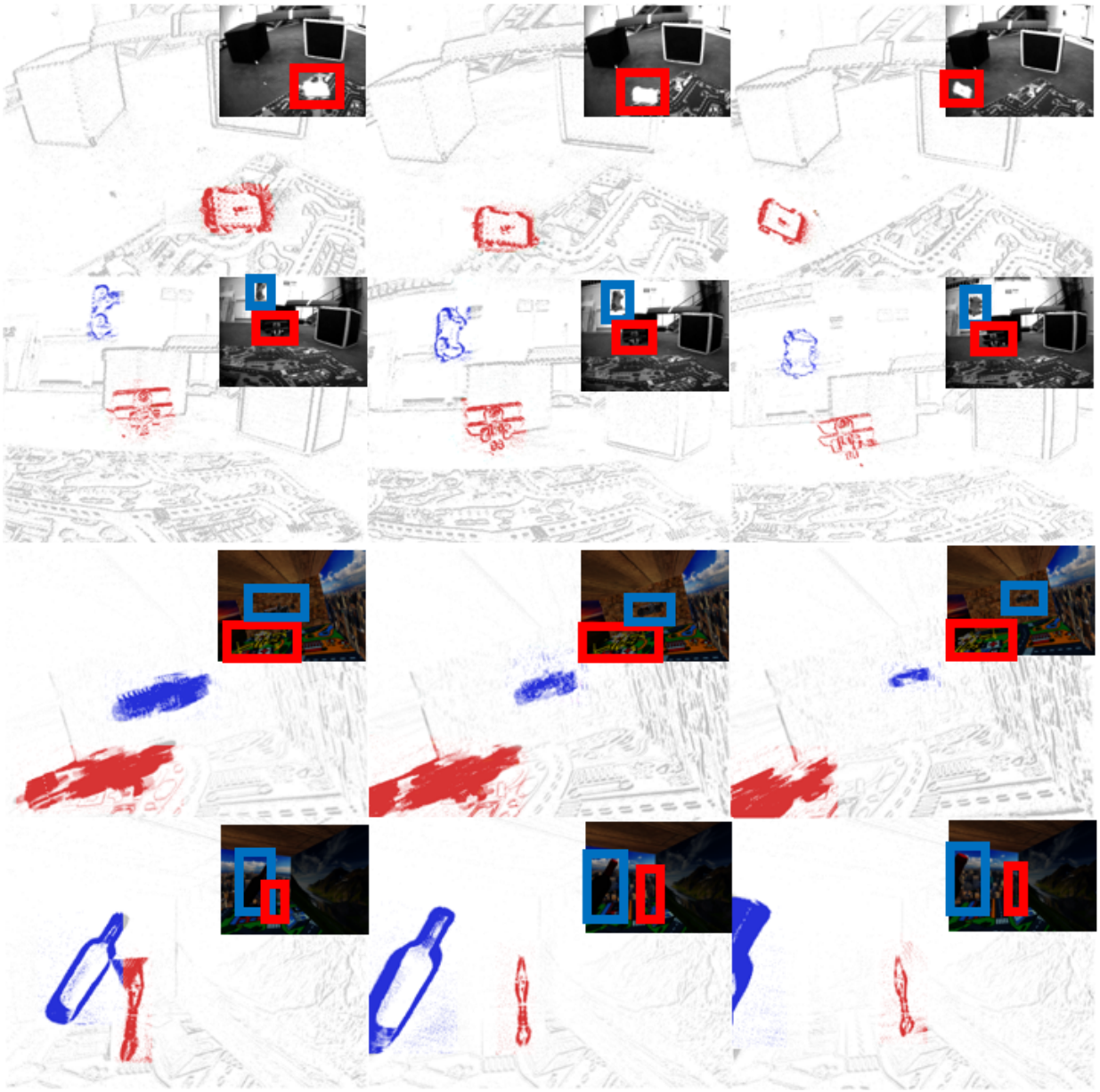

Figure 3.5: Qualitative Evaluation of our method on three datasets. Top two rows: EV-IMO dataset, Bottom two rows: MOD dataset. Insets show the corresponding grayscale/RGB images for reference. The cluster membership is color coded where gray color indicates background cluster. Bounding boxes on the images are color coded with respect to the objects for reference.

where $\mathcal{D}$ is the predicted mask and $\mathcal{G}$ is the ground truth mask. We evaluate our pipeline's performance on all the datasets using this metric. We obtain the bounding box for our method by obtaining the convex hull on the cluster of events. For comparison purpose we evaluate the

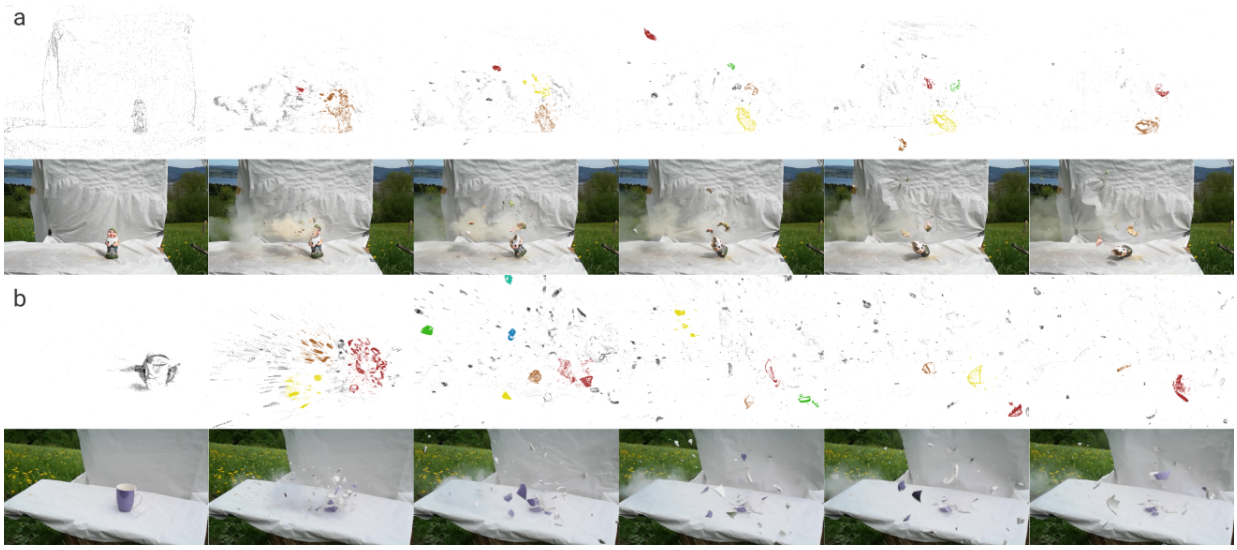

Figure 3.6: Multi-Motion Segmentation on the real sequences from [9]. (a) `Gnome shooting`: Gnome statue getting shot by a bullet, (b) `Mug shooting`: Mug getting shot by a bullet. The event frames are colored by cluster membership with gray showing background cluster. Note that the corresponding RGB frames *are not used for computation and are provided for visualization purposes only*.

performance of [121] using the same metric on all the three datasets. For datasets with more than one sequence, we compute the average of each model's performance on individual sequences.

Table 3.2: Comparison with state-of-the-art using the detection rate for different sequences of MOD++.

Method	Detection Rate (DR in %) ↑								Speed ↑ (MEv/s)	Speed×Avg. DR ↑
	AS_SM_RS	AM_SM_RS	AL_SM_RS	AM_SS_RS	AM_SL_RS	AL_SS_RS	AL_SS_RM	AL_SS_RL		
Mitrokhin *et al.* [121]	35.24	32.29	38.12	28.78	43.28	24.56	32.29	39.65	5.41	185.47
EVDodgeNet [61]	42.25	46.94	53.23	37.81	61.72	46.13	43.50	52.38	10.01	480.42
k-Means (k=5)	44.89	47.73	49.35	40.17	59.52	42.19	45.71	55.83	1.07	51.54
k-Means (k=10)	60.36	64.87	59.27	47.73	65.78	48.92	54.74	58.47	1.02	58.66
k-Means (k=20)	56.1	62.28	58.01	45.25	61.25	44.71	49.37	54.91	0.98	52.90
Ours (No Propagation)	**70.13**	73.29	**72.58**	**65.80**	**85.76**	**66.24**	**72.90**	**79.02**	0.83	60.77
Ours	69.52	**73.94**	71.27	63.58	84.21	64.93	71.18	78.37	**1.16**	**83.67**

3.5.2 Intersection Over Union (IoU)

IoU is one the most common and henceforth the most standard measure to evaluate and compare the performance of different segmentation methods. IoU is given by:

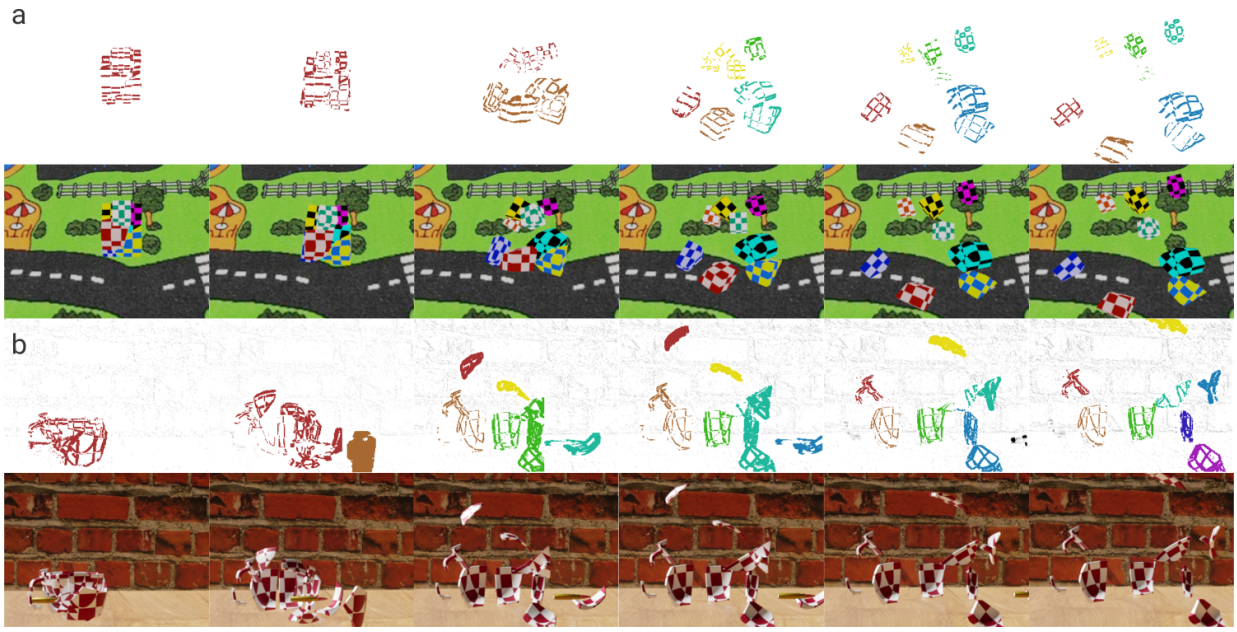

Figure 3.7: Challenge sequences from the proposed MOD++ dataset. (a) `Cube` breaking into smaller pieces by falling, and (b) `Cup` getting shot by a bullet.

$$IoU = (\mathcal{D} \cap \mathcal{G}) \,/\, (\mathcal{D} \cup \mathcal{G})$$

Our method outputs a cluster of events which are associated with an object. For the purpose of comparison we convert the sparse mask to a dense mask by assigning all the points lying inside the cluster as the same value.

3.5.3 Discussion of Results

Table 3.2 reports results on our proposed MOD++ dataset. We pick eight scenarios with different relative (camera and IMO) velocity direction, velocity magnitude and rotation magnitude. We illustrate the merits of split and merge, and motion propagation through extensive ablation studies and compare with previous approaches. Our approach outperforms others by at least $\sim$10% . Executing split and merge at every step offers better accuracy than propagating motion

Table 3.3: Comparison with state-of-the-art using detection rate for EED, MOD, EV-IMO datasets.

Method	Detection rate for dataset (%) ↑			Speed ↑
	EED	MOD	EV-IMO	(MEv/s)
Mitrokhin *et al.* [121]	88.93	70.12	48.79	5.41
Stoffregen *et al.* [130]	93.17	-	-	0.64* (N_l=10)
Ours	**94.2**	**82.35**	**81.06**	1.16

* Results taken directly from [130]

models (evident for more challenging scenarios like slow relative motion, large/small relative velocity direction and/or small rotation magnitude). However, motion propagation offers a speed-up without a significant loss of performance ($\sim$1%). Even though simple thresholding [121] and the deep learning-based approach [61] offer better speed-up and Speed$\times$Avg. DR (a metric proposed in [142]), their accuracies are almost 2-3$\times$ lower than our approach and is not reliable in challenging scenarios. We leave the speeding-up of our approach using deep learning to enable deployment on mobile robots for future work.

Table 4.2 reports the result of our method in comparison with two state-of-the-art IMO detection methods [121], [130] using only a monocular event camera. Our method outperforms the previous methods by up to $\sim$12% detection rate. Specifically, we outperform [121] by a large margin (up to $\sim$32%) on all the three datasets. Our approach is about 2$\times$ faster than [130] because of motion propagation while maintaining similar/slightly better accuracy on EED.

Table 3.4 reports the comparison with two deep learning methods for IMO segmentation [132] and [61] using the IoU metric. [61] was trained on the MOD dataset and is tested here on the EV-IMO dataset without any fine-tuning/re-training. We outperform [61] and [132] (which was trained on EV-IMO) on the EV-IMO dataset.

Fig. 4.3 shows qualitative results of our approach on the two datasets (top two rows show results for the EV-IMO dataset and the last two rows show results for the MOD dataset). Gray

Table 3.4: Comparison with state-of-the-art using IoU for EV-IMO.

Method	IoU
EV-IMO [132]	77.00*
EVDodgeNet [61]	65.76
Ours	**80.37**

*Results taken directly from [132] in which `boxes` and `wall` are used for training.

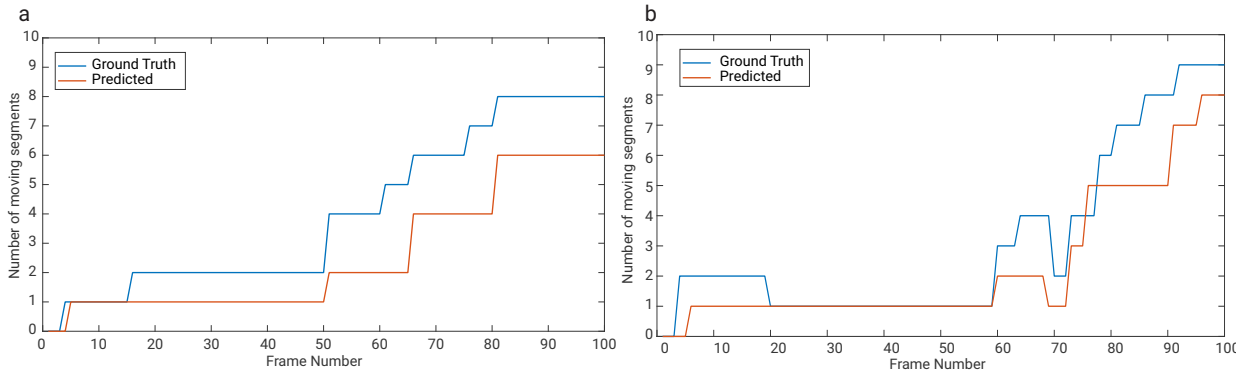

Figure 3.8: Number of moving segments vs. frame number (time) on challenge sequences of MOD++: (a) `Cube`, (b) `Cup`.

areas in the event images show the background cluster and red/blue colored regions show the differently segmented IMOs. The outputs show the robustness of our approach to shape, size and speed of the objects and in-variance with respect to camera motion. Also, note that the objects are sometimes very hard to detect in the corresponding grayscale/RGB frames in Fig. 4.3 motivating the use of event cameras for IMO detection using motion cues.

We obtain the ground truth IMO by counting ground truth labels with IoU overlap $\leq$0.2. The graph shows robustness of our approach with increasing number of moving segments. The predicted number of segments closely matches the ground truth. Segmenting solely based on motion with a monocular event-camera is ambiguous in challenging scenarios and results could be improved with incorporation of depth and appearance information in our split and merge step which we believe is the logical next step for future work.

Figs. 3.6a and 3.6b show the output of our method for challenging sequences of `Gnome`

`shooting` and `Mug shooting` from [9] showing that our method performs well even on real sequences with a large number of objects.

Fig. 3.8 shows performance of our algorithm with the respect to number of moving segments across time on challenge sequences of MOD++ dataset i.e., `Cube` and `Cup` sequence (shown in Figs. 3.7a and 3.7b).

Our algorithm runs on a hybrid CPU and GPU system (i7 CPU and NVIDIA Titan Xp GPU). Model fitting and feature extractions are run on GPU in parallel. Even though our core algorithm runs fast, the bottleneck is in the memory transfer to and from the GPU. *The complexity of our approach is linear in the number of clusters, events and frequency of cluster keyslices initiation.* Table 3.2 shows the speed of our algorithm in comparison with other approaches in Million Events per second (MEv/s). Our motion propagation and keyslicing provides a speed-up of upto 40% without compromising accuracy.

3.6 Conclusions

We presented a method for multi-motion segmentation using data from a monocular event camera. Our approach works by splitting the scene into smaller motions and then iteratively merging them based on a contrast measure. To our knowledge, this is the first approach for monocular independent motion segmentation which combines a bottom-up feature tracking and top-down motion compensation into a unified pipeline. We further speed up our method by using the concept of motion propagation and cluster keyslices.

A comprehensive qualitative and quantitative evaluation is provided on three challenging event motion segmentation datasets, namely, EV-IMO, EED and MOD showcasing the robustness

of our approach. Our method outperforms the previous state-of-the-art approaches by upto $\sim$12% detection, thereby achieving the new state-of-the-art on the three aforementioned datasets. To accelerate further research in this area, we present and open-source a new benchmark dataset MOD++ which includes challenging scenes such as cube breaking and a mug getting shot by a bullet along with extensive data stratification in-terms of camera and object motion, velocity magnitudes, direction and rotational speeds. We achieve 73.21% detection rate on MOD++ which is 2 to 3$\times$ higher than the state-of-the-art methods.

Chapter 4: SpikeMS: Deep Spiking Neural Network for Motion Segmentation

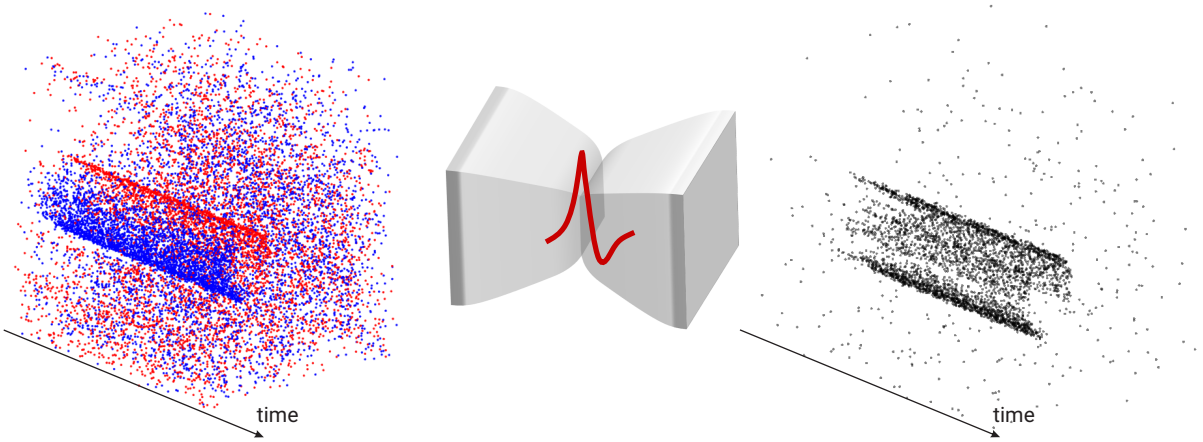

Figure 4.1: Event-based motion segmentation pipeline using a deep spiking neural network. Left to right: Event stream input represented as red (brightness increase) and blue (brightness decrease), representation of the proposed encoder-decoder spiking neural network called *SpikeMS* and the output predicted spike containing only the region of moving object(s).

In this chapter, we present the Spiking Neural Networks (SNN) for the IMO segmentation problem utilizing event sensor data as input. We propose *SpikeMS*, the first deep encoder-decoder SNN architecture for the problem of IMO segmentation using the event sensor as input. To accomplish this, we introduce a novel spatio-temporal loss formulation that includes both spike counts and classification labels in conjunction with the use of new techniques for SNN backpropagation.

4.1 Introduction

The animal brain is remarkable at perceiving motion in complex scenarios with high speed and extreme energy efficiency. Inspired by the animal brain, an alternative version of Artificial Neural Networks (ANNs) called Spiking Neural Networks (SNNs) aim to replicate the *dynamical system* aspects of living neurons. In contrast to standard ANNs which are essentially networks of complex functions, SNNs are comprised of networks of neurons modeled as differential equations, and inherently encode temporal data and offer low power and highly parallelizable computations. Furthermore, they possess the capability to deliver predictions whose confidence scale with the availability of input data [143, 144]. These low-power and low-latency properties are of great use to real-world robotics applications such as self-driving cars or drones, which demand fast responses during navigation in challenging scenarios [61].

Until recently, SNNs have been restricted to simple tasks and small datasets due to instability in learning regimes [145]. Recent development in new spike learning mechanisms [146, 147] has made it possible to design SNNs for real-world robotics applications. This coupled with neuromorphic processors such as Intel ® Loihi [69] and IBM TrueNorth [148] along with neuromorphic sensors such as DVS [120] and ATIS [149] have made it possible for producing real-world prototypes, drastically enhancing the appeal of such technologies.

In this work, we propose a deep SNN architecture called *SpikeMS* for the problem of motion segmentation using a monocular event camera. We consider the data from event sensors as they are well-suited for motion segmentation (due to the disparity in event density at object boundaries) and are a natural fit for SNNs (due to their temporal nature). We will now formally define the problem statement and our main contributions.

4.1.1 Problem Formulation and Contributions

We address the following question: *How do you learn to segment the scene into background and foreground (moving objects) using a Spiking Neural Network from the data of a moving monocular event camera?*

Our spiking neural network, *SpikeMS*, takes the event stream as input and outputs predictions of each event's class association as either foreground (moving object) or background (moving due to camera motion).

The model learns to distinguish between the spatio-temporal patterns of moving objects and the background. To the best of our knowledge, this is the first end-to-end deep encoder-decoder SNN. In particular, we evaluate our network on the task of motion segmentation using event input.

The main contributions of the chapter are given below:

- A novel end-to-end deep encoder-decoder Spiking Neural Network (SNN) framework for motion segmentation from event-based cameras.
- Demonstration of "early" evaluation of the network (at low latency), which we call *Incremental Predictions*, for imprecise but fast detection of moving objects for variable-sized integration windows.

4.2 Related Work

4.2.1 Spiking Neural Network Weight Learning Rules

While the concept of a spiking neuron has been around for a few decades [150], their progress has been bounded by the difficulty in training due to the ubiquitous vanishing gradient

problem for deep neural networks. In SNNs, the neurons output pulses that are non-differentiable, rendering attempts at directly applying the backpropagation algorithm non-trivial. Early attempts at training SNNs revolved around more biologically plausible Hebbian-style mechanisms [151] that only involve *local* updates, such as *Spike Time Dependent Plasticity* (STDP) [152], avoiding gradient issues. Work in this field continues to this day, with results [153–155] demonstrating utilities of STDP in training deep SNNs. Early attempts at incorporating backpropagtion into SNNs involved first training a traditional ANN, and then transferring the learned weights to an SNN [143]. Recent methods, which we build off of here, allow the SNN to be directly trained through backpropagation by finding a surrogate, continuous value function that roughly correlates for the spike activity [146, 147, 156].

4.2.2 SNNs for Visual Tasks and Event Data

There has been a renewed interest in using SNNs to process data directly from event-based visual sensors, such as the DVS since the sensor produces spike-like activity that fits well with SNN neurons. Applications of SNNs in this domain include classification problems [64] such as digit recognition [65], object recognition [41] gesture recognition [66], and optical flow [67, 68]. Recent development of neuromorphic processors such as the Intel Loihi [69] has lead to the deployment of SNNs on hardware [67, 70, 71].

Closest related to our work, in [145] recently a neural architecture of multiple layers has been designed. A six-layer neural network (five convolutional and one pooling layer) is used to learn with supervision to regress the three parameters of camera rigid rotation. In Lee *et al.* [157] a deep hybrid encoder decoder architecture was designed for self-supervised optic flow estimation.

The encoding layers are SNN with the backpropagation learning employing the approximation of [156], and the residual and decoding layers are ANN with the self-supervised loss computed from the images of a combined DVS and image sensor (DAVIS).

Event-based cameras have been recognized as a promising sensor for the problem of segmentation and detection of independently moving objects, as the event stream carries essential information about the movement of object boundaries [125]. Classical approaches [39,50,130] treat motion segmentation as a geometric problem and model it as an artifact of motion compensation of events. In ANN approaches, the input representation is formed by binning the events within a time-interval and convert to an image-like frame based structure [61, 132] the so called "event-frames". Our approach is similar to [133], but rather than creating event-frames, a sampled version of the event stream is fed directly into the network, taking advantage of the SNN's temporal nature in conjunction with the temporal nature of the event stream.

4.3 SpikeMS Architecture

4.3.1 Event Camera Input

A traditional camera records frames at a fixed frame rate by integrating the number of photons for the chosen shutter time for all pixels *synchronously* (in a global shutter camera). In contrast, an event camera only records the polarity of logarithmic brightness changes *asynchronously* at each pixel, resulting in asynchronous packets of information containing the pixel location and the time of the change known as an *event*. If the brightness at time t of a pixel at location $\mathbf{x}$ is given by $I_{t,\mathbf{x}}$ an event is triggered when

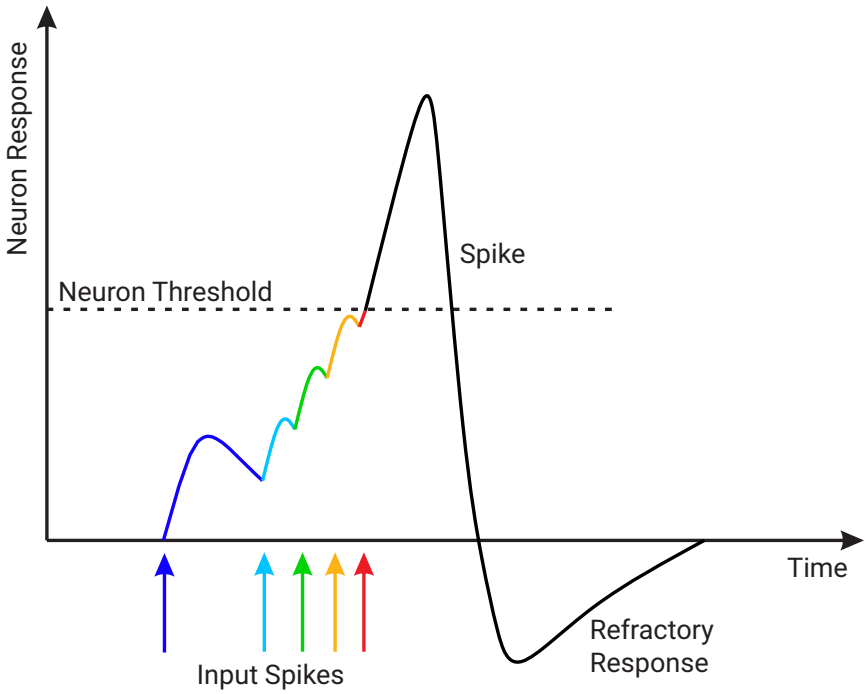

Figure 4.2: Depiction of the dynamical activity of a spiking neuron. The neuron receives input coming either from the data or lower layers (shown here as colored arrows), which generate bumps in the membrane voltage; we refer to this voltage in the chapter as $u(t)$. If the voltage $u(t)$ exceeds a threshold ϑ, shown here as the dotted line, the neuron outputs a spike, and then enters a refractory phase where it is less likely to fire another spike for a short time. Computationally, this spiking after passing a threshold amounts to feeding $u(t)$ through the spike function f_s. The effect that incoming pulses have on the voltage, and the extent of the refractory response, is governed in the Spike Response Model (SRM) [10] via the ε and ν kernels respectively (See Section 4.3.2 for more detail).

$$\| \log\left(I_{t+\delta t,\mathbf{x}}\right) - \log\left(I_{t,\mathbf{x}}\right) \|_1 \geq \tau \tag{4.1}$$

where δt is a small time increment and τ is a trigger threshold. Each event outputs the following data: $\mathbf{e} = \{\mathbf{x}, t, p\}$, where $p = \pm 1$ denotes the sign of the brightness change. We will denote the event stream in a spatio-temporal window as $\mathcal{E}\left(t, t+\delta t\right) = \{e_i\}_{i=1}^{N}$ (N is the number of events). We refer to it as event slice/stream/cloud/volume or spike train interchangeably.

4.3.2 Spiking Neuron Model

Spiking Neurons, unlike traditional *rate-encoding* neurons (commonly used neurons in standard ANNs), implicitly encode time in their formulation. They are modeled loosely after neurons in the brain, following the pioneering work by Hodgkin-Huxley [150] which laid the groundwork for differential equation modeling of neuronal activity. We utilize the Spike Response Model (SRM) which similar to all spiking neuron models, sums up incoming voltage from pre-synaptic neurons, but contains two filters: a filter that accounts for the neuron's *self-refractory* response denoted as ν, and a *spike response kernel* that accounts for the integration of incoming pre-synaptic pulses denoted as ε. For a given neuron i at timestep t, the update of the neuron's synaptic potential dynamics takes the form of:

$$\begin{aligned} u_i(t) &= \Big(\sum_j w_j (\varepsilon * s_j) \Big) + (\nu * s) \\ &= \mathbf{w}^\top \mathbf{a} + (\nu * s) \end{aligned} \tag{4.2}$$

for all incoming weight connections from pre-synaptic neurons $1, ..., j$, where $a(t) = (\varepsilon * s)(t)$, $s_i(t)$ is an input spike train in a neuron and $*$ denotes the convolution operator. An output spike is generated whenever $u(t)$ reaches the spiking threshold ϑ (the dotted line in Fig. 4.2)

The motivation for using SRM neuron types is that it inexpensively models the refractory behavior of neurons without having to run multiple differential equation solvers, as seen in other models.

Specific choices of ε and ν reduce the SRM equations to a LIF neuron [158]. Here, we use the formulation from [145]:

$$\varepsilon(t) = \frac{t}{\tau_s} e^{1-\frac{t}{\tau_s}} \mathcal{H}(t) \tag{4.3}$$

$$\nu(t) = -2\vartheta e^{1-\frac{t}{\tau_r}} \mathcal{H}(t) \tag{4.4}$$

where $\mathcal{H}$ is the Heaviside function, and τ_s and τ_r are the spike response and refractory time constants.

The activity of the neurons is then propagated forward through the layers of the network, in the same manner as an ANN. The feed-forward weight matrix $\mathbf{W}^{(l)} = [\mathbf{w}_1, ..., \mathbf{w}_{N_{l+1}}]$ for a given layer l with N_l neurons is applied to the activity resulting from the spike response kernel, added to the refractory activity and then thresholded. Thus, for all layers l in the network, the activity is forward-propagated as:

$$a^{(l)}(t) = (\varepsilon_d * s^{(l)})(t) \tag{4.5}$$

$$u^{(l+1)}(t) = \mathbf{W}^{(l)} a^{(l)}(t) + (\nu * s^{(l+1)}(t)) \tag{4.6}$$

$$s^{(l+1)}(t) = f_s(u^{(l+1)}(t)) \tag{4.7}$$

where f_s is the thresholding function, $\mathbf{W}^{(l)}$ is the forward weight matrix for layer l, and ε_d is the spike response kernel with delay, as in [147]. The input to the network, $s^{(0)}$, is the event data over the learning window.

4.3.3 Network Architecture

SpikeMS utilizes an end-to-end deep Spiking Neural Network, in contrast to many recent models [157] that use a hybrid combination of spiking and rate-encoding layers. To the best of our knowledge, *SpikeMS* is the first end-to-end spike trained deep encoder-decoder network for large scale tasks such as motion segmentation.

SpikeMS consists of a traditional hourglass-shaped layer structure of an autoencoder, with larger layers progressively encoded to smaller representations, which are then decoded back to the original size. We use three encoder layers followed by three decoder layers. The first three convolutional layers perform spatial downsampling with a stride of 2 and kernel size of 3×3. The output of the first encoder layer contains 16 channels, and each encoder layer after doubles the number of channels. The last three decoder layers perform spatial upsampling using transposed convolution, with a stride of 2 and kernel size of 3x3. Decoder layers 4 and 5 each halve the number of channels. The last layer (6) outputs the predicted spikes of the moving object(s) using 2 channels, representing positive and negative event polarities.

4.3.4 Spatio-Temporal Loss

We propose a novel loss formulation which takes full advantage of the spatio-temporal nature of the event data. Our loss function consists of two parts: a binary cross entropy loss $\mathcal{L}_{\text{bce}}$ and spike loss $\mathcal{L}_{\text{spike}}$.

The binary cross-entropy loss $\mathcal{L}_{\text{bce}}$ is computed by comparing the predicted temporal spike

projection to the ground truth temporal spike projection.

$$\mathcal{L}_{\text{bce}} = -\left(1_f \log\left(\hat{\mathcal{E}}_f\right) + 1_b \log\left(\hat{\mathcal{E}}_b\right)\right) \tag{4.8}$$

Where, the spike projection $\mathcal{E}$ is obtained as: $\mathcal{E}(\mathbf{x}) = \sum_t \mathcal{E}(\mathbf{x})$. Such a projection converts a spike train into a real-valued output, encoding the frequency of spikes. And the groundtruth foreground and background labels are denoted as 1_f and 1_b respectively.

The spike loss $\mathcal{L}_{\text{spike}}$ is derived from the Van-Rossom distance [159] and measures the distance between two binary spike trains [160]. $\mathcal{L}_{\text{spike}}$ preserves the temporal precision of the event stream. The ground truth spike labels are generated by applying a binary mask to the event cloud input $\mathcal{E}$, i.e., masking all non-moving-object events (background events) as 0, and keeping intact the events that correspond to the moving object. $\mathcal{L}_{\text{spike}}$ is given by

$$\mathcal{L}_{\text{spike}} = \sum_{t=0}^{t+\delta t} \left(\hat{\mathcal{E}}(t, t+\delta t) \circ 1_f - \tilde{\mathcal{E}}(t, t+\delta t)\right)^2 \Delta t \tag{4.9}$$

where $\circ$ denotes the Hadamard product and 1_f denotes the mask of foreground spikes.

The overall loss $\mathcal{L}_{\text{total}}$ is given by

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{bce}} + \lambda \mathcal{L}_{\text{spike}} \tag{4.10}$$

where λ is a weighting factor. The error is backpropagated through the network using SLAYER [147].

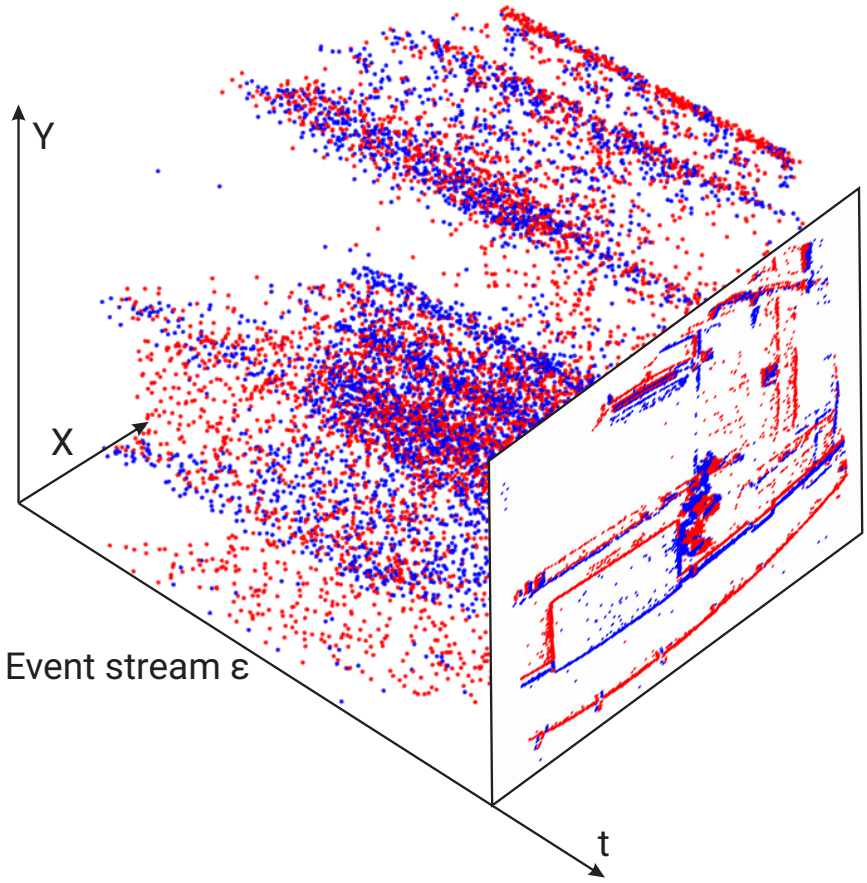

Figure 4.3: Representation of event stream $\mathcal{E}$ and its corresponding projection (event frame). Note that, only event streams are fed to *SpikeMS*. The event frame is shown only for clarity purposes.

4.3.5 Simulation of SNNs on GPU

SNNs are continuous dynamical systems which can process input event streams asynchronously. However, for our experiments, we simulate the SNN network on a GPU. To achieve this, we need to discretize the event data at fixed time steps. To balance the trade-off between accuracy and resource availability [145], we restrict the simulation time step to one millisecond. We train our SNNs with fixed simulation time window/width/steps Δt_{train} of 10ms for all experiments. However, to test the out-of-domain temporal performance, we test our predictions on simulation time steps

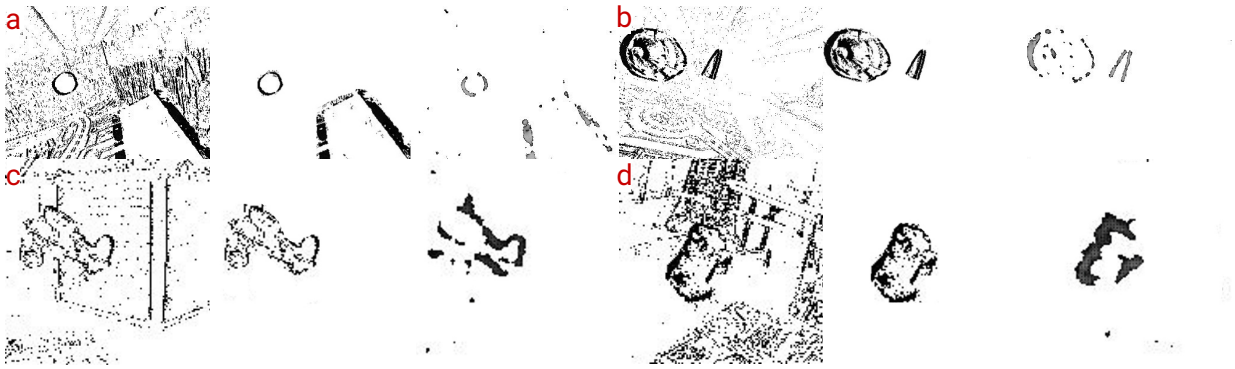

Figure 4.4: Qualitative Evaluation of our approach on two datasets. Top row (a and b): MOD dataset, Bottom row (c and d): EV-IMO dataset. Each sample includes, (left to right) event stream, groundtruth, and network prediction. Here, we show event projections for clarity purposes but *SpikeMS* predicts spatio-temporal spikes.

Δt_{test} of 1ms to 25ms. To fit the event inputs into fixed time steps, the multiple events with the same polarity and spatial location within a timeframe are represented as a single binary event. This downsampling collapses all events within the window into a single event. The simulated network is trained with the publicly available PyTorch implementation of SLAYER [147].

4.4 Experiments and Results

We evaluate our approach on publicly available synthetic and real datasets. We demonstrate performance of *SpikeMS* both qualitatively and quantitatively by employing the Intersection over Union (IoU) and Detection Rate (DR) metrics [50].

4.4.1 Overview of Datasets

We use the publicly available MOD [61] and EV-IMO [132] datasets for training and evaluating the motion segmentation predictions. MOD [61] is a synthetic dataset specifically targeted for learning based motion segmentation approaches. The simulated data contains objects

moving in an indoor room-like environment with randomized wall textures, static/dynamic objects and the object/camera trajectories. EV-IMO [132] contains monocular event-based camera data captured in a lab environment with challenging scenarios (multiple objects moving in random trajectories and varying speeds). EV-IMO contains five different sequences (`boxes`, `floor`, `wall`, `table`, and `fast`) which were collected using the DAVIS 346 camera.

4.4.2 Quantitative Results

Table 4.1: Quantitative Evaluation using IoU (%) ↑ metric on EV-IMO and MOD datasets.

Method	EV-IMO										MOD	
	`boxes`		`floor`		`wall`		`table`		`fast`			
	100	20	100	20	100	20	100	20	100	20	100	20
EV-IMO† [132]	70±5		**59±9**		78±5		79±6		67±3		-	
EVDodgeNet [61]	**67±8**		61±6		72±9		70±8		60±10		75±12	
GConv† [133]	81±8	60±18	79±7	55±19	83±4	80±7	57±14	**51±16**	74±17	**39±19**	-	-
PointNet++ [161]	71±22	80±15	68±18	76±10	75±19	74±20	62±28	68±23	24±10	20±6	74±13	**67±15**
Ours ($\mathcal{L}_{bce}$)	57±11	59±7	56±9	46±12	62±8	62±9	51±12	45±12	42±13	36±13	62±11	63±7
Ours ($\mathcal{L}_{spike}$)	45±4	52±7	49±8	44±6	53±15	47±11	43±15	37±4	41±6	35±4	55±11	55±8
Ours ($\mathcal{L}_{bce} + \mathcal{L}_{spike}$)	61±7	**65±8**	**60±5**	53±16	65±7	63±6	52±13	**50±8**	45±11	**38±10**	68±7	**65±5**

†Results taken directly from [133]

We compare our method against state-of-the-art ANNs (both 2D and 3D) and the results are given in Table 4.1. In particular, 2D ANNs (EV-IMO [132], EVDodgeNet [61]) are trained with inputs consisting of event-frames computed by accumulating (or projecting) events on a plane. In contrast, 3D ANNs (GConv [133] and PointNet++ [161]) are trained directly on the event cloud $\mathcal{E}$. We evaluate ANN-2D with event frames integrated with a time width Δt of 25ms. ANN-3D approaches are evaluated with two time widths Δt of 20ms and 100ms similar to [133].

During evaluation, *SpikeMS* is tested at Δt_{test} = 100ms and Δt_{test} = 20ms (trained at Δt_{train}=10ms) for a fair comparison with ANNs-3D. Table 4.1 provides the mean IoU results on multiple sequences of the EV-IMO and MOD datasets. We observe that the performance of

SpikeMS is comparable to ANN-2D and ANN-3D approaches in all cases. However, note that the ANN-2D and ANN-3D perform better in the domain they are trained in as compared to *SpikeMS* and we speculate this is because of more stable training procedures in ANNs. This points to a direction of future work for SNNs of proposing better training methodologies.

In Table 4.1, we also compare our results when trained on different loss functions. We observe that the proposed spatio-temporal loss formulation performs better than just using the spike loss or crossentropy loss as it utilizes the information from both spatial and time domains together.

Finally, we also compare *SpikeMS* with classical hand-crafted methods in Table 4.2 and we see that, our SNN approach outperforms most hand-crafted methods whilst being deployable directly on neuromorphic hardware. This would lead to huge power savings when deployed on a robot.

4.4.3 Incremental Prediction

We test the network's capability to perform *incremental predictions* evaluating the network at different testing discretized window sizes, with Δt_{test} ranging from 1ms to 25 ms, while keeping the training window fixed at Δt_{train} of 10ms. This experiment tests for the out-of-domain generalization performance of *SpikeMS*.

All predictions made at $\Delta t < 10$ms can be considered as *incremental predictions* (See Fig. 4.5), since the testing window is smaller than the training window. This is particularly important for robotics applications since one can filter these incremental predictions to get close to the accuracy of the model with long time predictions but with a lower latency. For example, we

can filter predictions (we use a linear Kalman filter [162] for filtering) of 3ms to obtain up to ∼64% of the accuracy of 10ms predictions, but with 70% less latency which might be required for time-critical controllers. We also experiment with values greater than 10ms, where we examine whether longer integration windows yield more accurate results.

Fig. 4.5 shows the plot of accuracy (IoU) versus the duration of input spikes during simulation, considered for prediction. We observe that the prediction accuracy increases over time with the occurrence of more spikes, but critically, that the SNN is able to output reasonable predictions from less spikes. As shown in SNNs outperform ANN-2D and ANN-3D at early stages with less amount of data. We observe that ANN-3D outperforms ANN-2D since it is trained with temporal augmentation techniques as proposed in [133]. Note that the SNN does not rely on temporal augmentations for incremental predictions rather utilizes dynamic nature of spiking architecture. This demonstrates how well *SpikeMS* generalize outside the temporal domain.

4.4.4 Qualitative Results

Fig. 4.4 shows qualitative results of our approach on the two datasets. For each example the input, moving object groundtruth, and network prediction are shown. Note that, we show the event projections for clarity purposes but the network input and outputs are the event cloud/spikes. We can observe that the network output predictions are similar to the ground truth for the moving objects in the presence of significant background variation and motion dynamics.

Fig. 4.6 shows the performance of *SpikeMS* on real-world event streams, again with significant background variations and patterns. These results demonstrate the capability of *SpikeMS* to generalize to different environments without any retraining or fine tuning of the

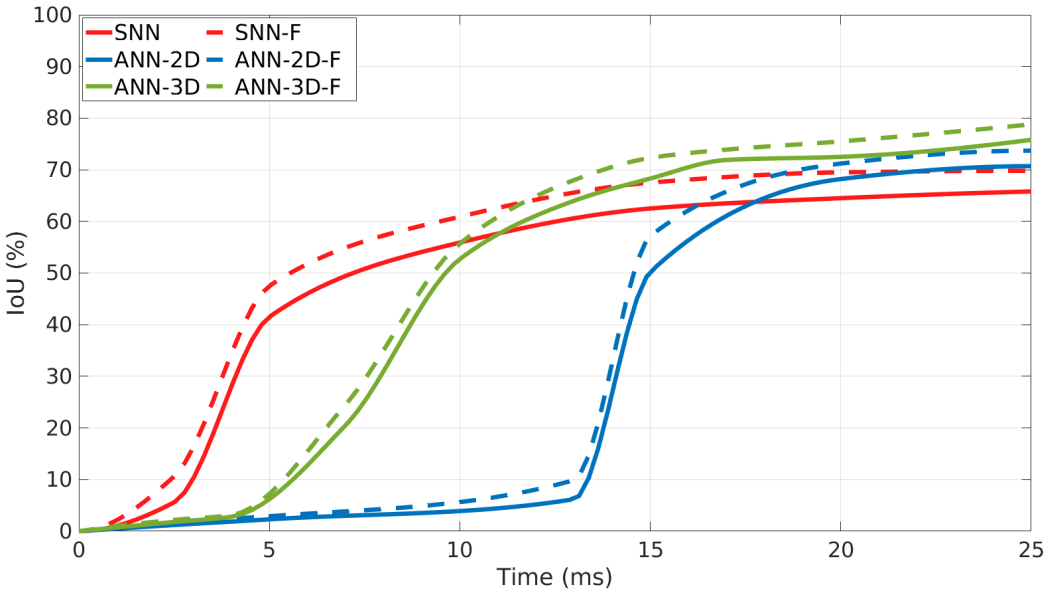

Figure 4.5: Incremental Prediction: Segmentation accuracy vs. input spike window length in milliseconds for various simulation time width Δt. SpikeMS is able to achieve good accuracy significantly faster than ANNs, given smaller input data. The dashed lines represent accuracy improvement after employing a filtering (See Sec. 4.4.3).

network.

4.4.5 Power Efficiency

We further analyze the benefits of *SpikeMS* compared to a fully ANN architecture with respect to power consumption. It is important to note that the main power consumption benefits occur when the SNN is implemented on a neuromorphic hardware such as the Intel® Loihi [69], where the network *only* consumes power when there is a spike. Hence, the power consumption depends on the mean spike activity of the incoming data and the number of synaptic operations. In contrast, ANNs perform dense matrix operations without exploiting the event sparsity. Thus, in anticipation of deploying *SpikeMS* on the new generation of neurmorphic chips, we demonstrate

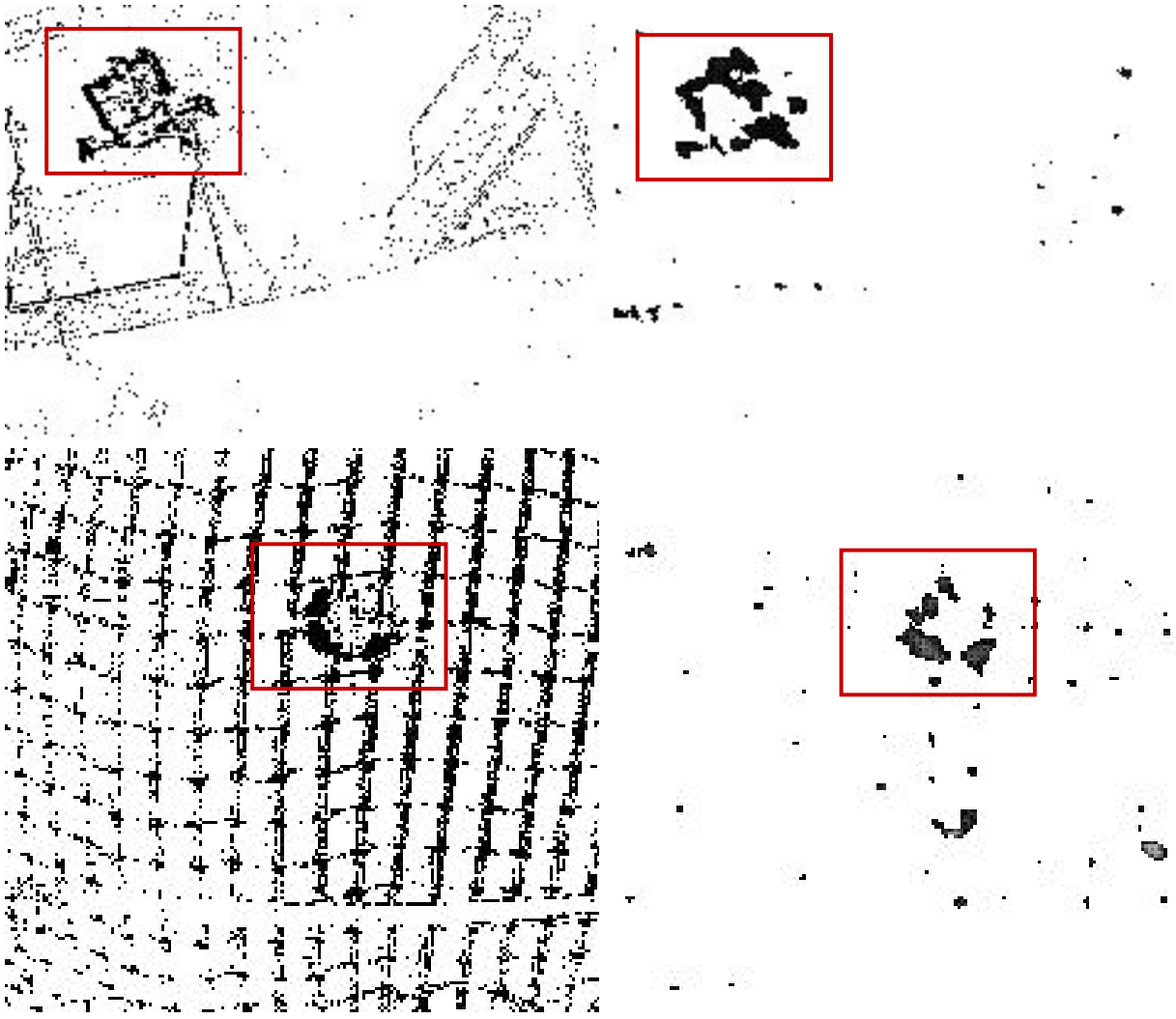

Figure 4.6: Results showing motion segmentation generalization without fine-tuning or re-training on real world data. *SpikeMS* is able to segment the moving object from the scene even in the presence of substantial background noise. The objects in the red bounding box are the true moving objects. Top row: A fast drone approaching a moving event camera. Bottom row: Moving object behind netted background.

Table 4.2: Comparison with state-of-the-art classical approaches for EED, MOD, EV-IMO datasets.

Method	Detection Rate (%) ↑		
	EED	MOD	EV-IMO
Mitrokhin *et al.* [39]	88.93	70.12	48.79
Stoffregen *et al.* [130]	93.17	-	-
0-MMS [50]	**94.2**	**82.35**	**81.06**
Ours	**91.5**	**68.82**	**65.14**

Table 4.3: Performance Metrics for EV-IMO and MOD datasets.

Method	EV-IMO					MOD
	`boxes`	`floor`	`wall`	`table`	`fast`	
Num. Operations ($\times 10^8$)	0.42	0.34	0.52	0.38	0.53	0.83
Energy benefit ($\times$)	116.19	143.53	93.84	128.42	92.07	58.80

the power savings by comparing the number of operations by a metric proposed in [157].

Table 4.3 provides the average number of synaptic operations in SNNs along with a conservative estimate of the energy benefit when compared to an ANN-2D. We can observe that SNNs require a significantly lower number of synaptic operations and power as compared to ANNs.

4.5 Conclusion

We presented a deep encoder-decoder Spiking Neural Network for a large-scale problem and demonstrated our architecture on the task of motion segmentation using data from a monocular event camera. Our novel spatio-temporal loss formulation takes full advantage of the spatio-temporal nature of the event data. We demonstrated the unique ability of our network, *SpikeMS*, for incremental prediction and showed its capability to generalize across a range of temporal intervals without explicit augmentation. A comprehensive qualitative and quantitative evaluation

was provided using synthetic and real-world sequences from the EV-IMO, EED and MOD datasets. It was shown that *SpikeMS* achieves performance comparable to an ANN method, but with 50$\times$ less power consumption.

Chapter 5: Conclusion and Future Work

The thesis presented a suite of minimalist solutions for motion perception problems. Notably, we address the egomotion and IMO segmentation problems in challenging scenarios such as high speed, changing lighting conditions, and unseen/novel environments. Furthermore, we have established the mathematical foundation for combining bio-inspired motion representation (normal flow and event sensors) and learning (geometry-based and spiking network) frameworks to address the problem mentioned above. In this chapter, I lay out the key contributions of my work and discuss its scope. I also provide a few potential future directions to address these problems.

Chapter 2 addressed the problem of estimating egomotion (camera pose) using a sequence of images. We introduced a normal flow network called *NFlowNet*, which predicts accurate motion fields under challenging scenarios and is more resilient to noise and bias. Furthermore, we proposed *DiffPoseNet* framework, whose primary goal is to improve robustness and cross-data generalization of egomotion. This is achieved by changing the order of processes classically used in the structure from motion (SfM) and postponing depth estimation and regularization to later stages in the pipeline. We utilized normal flow from *NFlowNet* and estimated relative camera pose based on the cheirality (depth positivity) constraint. We formulated cheirality constraint as a differentiable optimization layer to enable end-to-end learning of camera pose. A comprehensive

qualitative and quantitative evaluation was provided on the challenging datasets: TartanAir, TUM-RGBD, and KITTI. We demonstrated our method's efficacy, accuracy, and robustness under noisy scenarios and cross-dataset generalization without any fine-tuning and/or re-training. Our approach outperforms the previous state-of-the-art approaches. Particularly, *NFlowNet* can output accurate motion fields at up to 6× the speed with up to 46× smaller model size.

The currently pretrained posenet takes the role of "initialization" in the optimization pipeline. Since we are proposing *DiffPoseNet* as a generic framework, one could replace posenet with better pose estimators in future work. Furthermore, the current approach focuses on static scenes (with no moving objects) only. In the future, dynamic scenes can be addressed by including the differentiable IMO segmentation module in *DiffPoseNet*.

Until recently, classical normal flow estimation methods were unreliable and prone to noise when deployed in real-world scenarios. Our *NFlowNet* is the first deep learning approach to robustly estimate normal flow and to showcase the application via *DiffPoseNet* framework. Utilizing the new normal flow estimation approach, we could implement additional purposive and qualitative visual frameworks [14] to improve robust and cross-scenario generalization. As these approaches are fast and require less computational resources than traditional approaches, we can deploy them in real-time autonomous vehicle applications.

Recent advances in differentiable programming have paved a new research direction to induce geometrical priors into end-to-end learning. This fusion is beneficial in reducing the dependency on large deep neural networks and thereby minimizing power requirements. Differentiable programming is not only limited to visual sensors but also can enable next-generation sensor fusion research, similar to [163]. Furthermore, from the bio-inspired perspective, we can build cognitive architecture (inspired by the brain's "physics engine" [164]) for sensorimotor

integration in real-time robotics applications.

Chapter 3 introduced *0-MMS*, a method for IMO segmentation using data from a monocular event sensor. Our approach works by splitting the scene into smaller motions and then iteratively merging them based on a contrast measure. *0-MMS* combines bottom-up feature tracking and top-down motion compensation into a unified pipeline. We further accelerated our method by using the concept of motion propagation and cluster key slices. The approach was successfully evaluated on both challenging real-world and synthetic scenarios from the EV-IMO, EED, and MOD datasets. It outperformed the state-of-the-art detection rate by 12%, achieving a new state-of-the-art average detection rate of 81.06%, 94.2%, and 82.35% on the aforementioned datasets. To enable further research and systematic evaluation of multi-motion segmentation, we present a new open-source dataset/benchmark called MOD++, which includes challenging sequences and extensive data stratification in camera and object motion, velocity magnitudes, direction, and rotational speeds.

One of the limitations of *0-MMS* was the manual tuning of parameters for spitting and merging operations. This can be resolved by an automatic choice of parameters via model selection [165]. Another major issue with *0-MMS* is the assumption of rigidly moving segments. However, this assumption would be constraining when the object considered is non-rigid, such as a human or animal. One could solve this issue by considering non-rigid motion as a composition of many piece-wise rigid motions. Another approach would be to model non-rigid motion directly by fitting event surfaces. Even though the current method is relatively faster than existing approaches, it is not real-time. The approach needs acceleration via GPU parallelization to enable usage on mobile robots.

Throughout our *0-MMS* work, we focused only on monocular camera solutions. Overall performance of our approach can be improved by fusing additional sensors such as IMU and stereo event camera, similar to [49, 166]. However, one of the drawbacks of using additional event sensors would be increased computational requirements. As early event cameras such as the DVS128 had a resolution of 128×128 pixels, it was not challenging to deploy multiple DVS128 for robotic applications. But, the recent release of large resolution sensors from Samsung [167], Prophesee [168], Celepixel [169] now go up to 1280×960 pixels. The sensor design improvements definitely helped in the deployment of event sensors in real-world autonomous driving [170], flying [100], and computational photography [9] applications. Using multiple large-resolution events sensors would be very challenging in resource-constrained scenarios. Recent initial efforts in event compression mechanism [171] have shown promising results for an image reconstruction task. The future compression approaches can be specifically designed for motion perception problems for a real-time fusion of multiple event sensors.

Chapter 4 presented *SpikeMS*, a deep encoder-decoder Spiking Neural Network (SNNs) for a large-scale problem, and demonstrated our architecture on the task of motion segmentation using data from a monocular event camera. Traditionally, SNNs are aimed at closely matching the functioning of the biological brain. However, due to algorithmic and training complexity, SNNs have not solved the highly complex tasks that standard Artificial Neural Networks (ANNs) excel at. To accomplish this, we introduced a novel spatio-temporal loss formulation that includes both event spike counts and classification labels in conjunction with new techniques for SNN backpropagation. In addition, we demonstrated the unique ability of our network, *SpikeMS*, for incremental prediction, or predictions from smaller amounts of test data than it is trained on.

Providing outputs even with partial input data proves invaluable for low-latency applications and fast predictions.

A comprehensive qualitative and quantitative evaluation was provided using synthetic and real-world sequences from the EV-IMO, EED, and MOD datasets. It was shown that *SpikeMS* achieves performance comparable to an ANN method but with 50$\times$ less power consumption. Also, the incremental prediction feature demonstrated the *SpikeMS*'s capability to generalize across a range of temporal intervals without explicit augmentation.

The current *SpikeMS* pipeline is trained with ground-truth labels in a supervised fashion. As getting labels is arduous and time-consuming, a self-supervised variant of *SpikeMS* would be beneficial for real-time robot deployment. In this work, we simulated the SNN dynamical processes on a GPU for evaluation purposes. A recent development in neuromorphic computing has enabled us to bypass the simulation process and directly deploy on the neuromorphic processors [69, 148]. Neuromorphic processors have given a new direction to spiking neural network research by allowing real-time deployment. The tight coupling of sensors and processors would enable the realization of bio-inspired and bio-plausible learning approaches such as Vector Symbolic Architectures (VSA) [172] or HD computing [173].

The thesis has opened up a new line of research in applying a minimalist philosophy. The proposed approaches have been formulated in generic frameworks to enable possible extension beyond the scope of this thesis. This work will help put bio-inspired approaches at the forefront of autonomous vehicle research.

Appendix A: Moving Object Dataset (MOD)

This chapter presents the details of MOD dataset generation. MOD is a simulated dataset that was used for evaluating *0-MMS* and *SpikeMS* in Chapter 3 and Chapter 4.

Extensive and growing research on visual (inertial) odometry or SLAM have lead to the development of a large number of datasets. Recent adaption of deep learning to solve these aforementioned problems have fostered the development of large scale datasets (large amount of data). However, most of these datasets are built with the fundamental assumption of static scenes in mind and as a manifestation of which moving or dynamic objects are often not included in these datasets [174–176].

To this end, we propose to use synthetic scenes for generating "unlimited" amount of training data with one or more moving objects in the scene. We accomplish this by adapting and proliferating the simulator presented in [175]. To incubate generalization to novel scenes and to utilize the algorithm trained on simulation directly in the real world, we create synthetic moving objects which vary significantly in their texture, shape and trajectory. We also choose random textures for the walls of the 3D room in which objects will move about.

To generate data, we randomize wall textures, objects and object/camera trajectories to obtain seven unique configurations out of which one is exclusively used for test of generalization on more complex structures. Each configuration has a room with three objects moving as shown

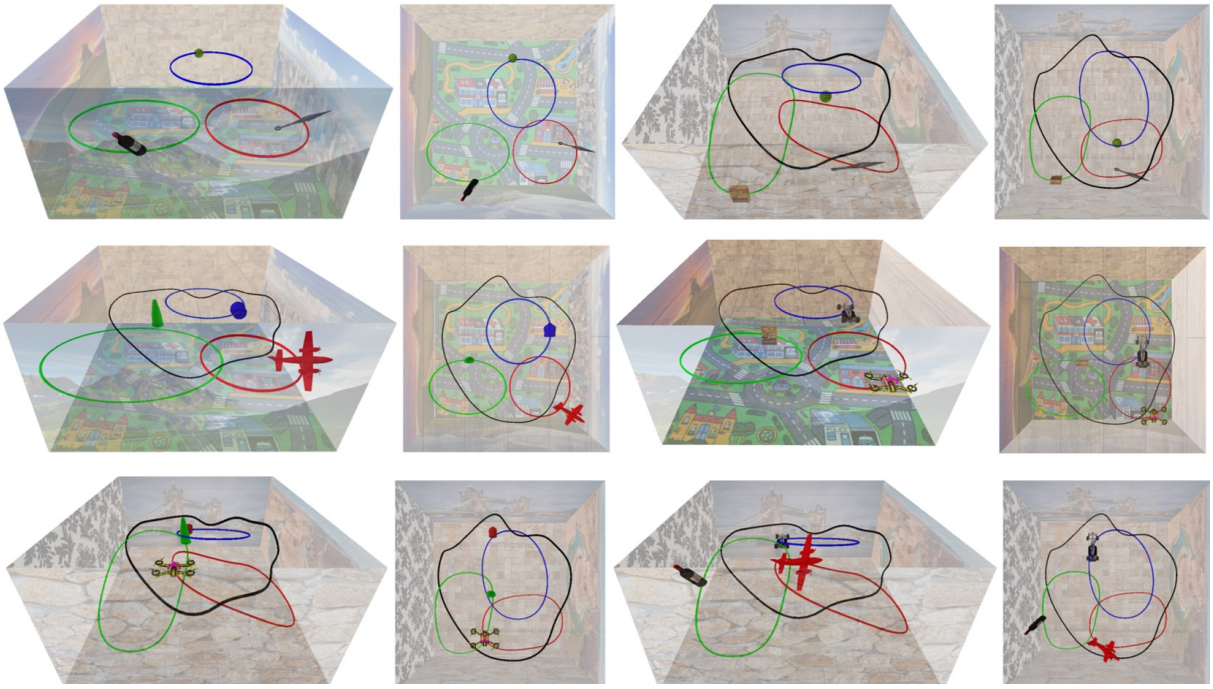

Figure A.1: Various Scene setups used for generating data.

in Fig. A.1. Images are rendered at 1000 frames per second at a resolution of 346×260 and a field of view of 90° for each configuration. Using these images, events are generated following the approach described in [175]. Later event frames $\mathcal{E}$ are generated with three different integration times δt of $\{1, 5, 10\}$ ms. Details about the room, lighting and objects are given next.

A.1 Simulated Environment

Each room is of size $10 \times 10 \times 5$ m and has random textures on all the walls. These random textures consist of different patterns, colors and shapes. These textures mimic those which occur in real-world indoor and outdoor environments such as skyscrapers, flowers, landscape, bricks, wood, stone and carpet. Each room contains seven light sources inside it for uniform illumination.

The camera is moved inside the 3D room on trajectories such that almost all possible combinations of rotation and translation are obtained. This is aimed at replicating the movement

Figure A.2: Moving objects used in our simulation environment. Left to right: ball, cereal box, tower, cone, car, drone, kunai, wine bottle and airplane. Notice the variation in texture, color and shape.

Figure A.3: Random textures used in our simulation environment

which could be encountered on a real quadrotor.

We have three Independently Moving Objects (IMOs) in each room. Each object is unique in color, shape, texture and size. The objects are chosen to range from simple shapes and textures to complex ones. The objects chosen are ball, cereal box, tower, cone, car, drone, kunai, wine bottle and airplane. The trajectories of the objects are chosen such that many different combinations of relative pose between the camera and the objects are encountered. Also, the objects are moving ten times faster than the camera simulating objects being thrown at a hovering or a slow moving (drifting) quadrotor. The wall textures and moving objects are shown in Figs. A.2 and A.3 respectively.

A.2 Dataset Content

The dataset comprise of images and events along with ground truth for object segmentation, optical flow, object poses, camera pose, and depth. All the data is stored in the form of ROS bags files which are compressed by default and each bag file consuming up to 5 gigabytes. To help with the data pre-possessing, we provide the development kit for easy data handling.

A.3 Evaluation Metrics

A.3.1 IMO Segmentation

For the IMO segmentation, we present three simple evaluation metrics based on standard literature [117]. Detection Rate is defined as number of detections which have an overlap of more than τ.

$$\text{DR} = \mathbb{E}\left(\frac{\overline{\overline{\mathcal{D} \cap \mathcal{G}}}}{\overline{\overline{\mathcal{G}}}} \geq \tau\right) \times 100\%$$

Here, $\overline{\overline{x}}$ denotes the cardinality of a set x, $\mathcal{G}, \mathcal{D}$ denote the set of ground truth and detected pixels respectively. Detection rate penalizes for false positives, false negatives and missed detections.

To measure false positives we defined a metric called Average False Positive (AFP) and is given by

$$\text{AFP} = \mathbb{E}\left(\frac{\overline{\overline{\neg\mathcal{D} \cap \mathcal{G}}}}{\overline{\overline{\mathcal{G}}}}\right) \times 100\%$$

Similarly, false negatives are given by a metric called Average False Negative (AFN) and is given by

$$\text{AFN} = \mathbb{E}\left(\frac{\overline{\overline{\mathcal{D} \cap \neg\mathcal{G}}}}{\overline{\overline{\mathcal{G}}}}\right) \times 100\%$$

A.3.2 Optical Flow

Optical Flow is defined as the image pixel velocity between two camera views. It can be thought of as the projection of the real 3D velocity on the image plane between two camera views. Optical flow can be computed in multiple-ways, either by using depth and pose of the camera between two views or by matching features in the image frame. The former method is often much more accurate and is the method employed in this dataset for generating optical flow ground truth.

Let the images/event frames be captured at times t and $t+1$ with a pose ${}^{\mathbf{W}}[\mathbf{R},\mathbf{T}]_{\mathbf{C},t_0}$ and ${}^{\mathbf{W}}[\mathbf{R},\mathbf{T}]_{\mathbf{C},t_1}$ respectively. Here, $\mathbf{W},\mathbf{C}$ denotes the world and camera coordinate frames respectively. Assuming that the linear and angular velocity is constant between t_0 and t_1, we can compute the linear and angular velocities by numerical differentiation.

$$\mathbf{v}_{t_0}^{t_1} = \frac{\left({}^{\mathbf{W}}\mathbf{T}_{\mathbf{C},t_1} - {}^{\mathbf{W}}\mathbf{T}_{\mathbf{C},t_0}\right)}{\delta t} \tag{A.1}$$

$$\lfloor \omega_{t_0}^{t_1} \rfloor_\times = \frac{\text{logm}\left({}^{\mathbf{W}}\mathbf{R}_{\mathbf{C},t_0}{}^{T}\,{}^{\mathbf{W}}\mathbf{R}_{\mathbf{C},t_1}\right)}{\delta t} \tag{A.2}$$

Here $\delta t = (t_1 - t_0)$, logm denotes matrix logarithm, $\lfloor \omega_{t_0}^{t_1} \rfloor_\times$ represents the skew-symmetric matrix from of the vector $\omega_{t_0}^{t_1}$ given as follows.

$$\lfloor \omega \rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Finally, the ground truth optical flow $\hat{\dot{\mathbf{x}}}$ is computed as follows.

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} \\ 0 & \frac{-1}{Z} & \frac{y}{Z} \end{bmatrix} \mathbf{v}_{t_0}^{t_1} + \begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \omega_{t_0}^{t_1} \tag{A.3}$$

Here. Z is the depth at the pixel location $\mathbf{x} = (x, y)$ and is computed as the average of depths at times t_0 and t_1, i.e., $Z_{\mathbf{x}} = \mathbb{E}\left(Z_{\mathbf{x},t_0}, Z_{\mathbf{x},t_1}\right)$. Note that the ground truth optical flow computed above is based on the assumption that the movement in the image is only due to camera motion/ego-motion, hence this optical flow is generally called *rigid flow*. However, in our dataset, there are a lot of independently moving objects where the above flow calculation will be wrong. Hence, we compute the flow of pixels corresponding to IMOs separately. To compute the optical flow for the i^{th} IMO's visible pixels denoted by $\mathcal{F}_i$, we have to compute relative linear and angular velocities between the coordinate frame of the camera $\mathbf{C}$ and each of the IMO $\mathbf{F}_i$. In our case, the simulator gives us the object pose in world frame (with respect to the object's geometric center), i.e., ${}^{\mathbf{W}}[\mathbf{R}, \mathbf{T}]_{\mathbf{F}_i,t}$. Also, all the objects are rigid (non-deformable), hence with the information of relative object pose we can compute relative 3D movement of each and every point on the object and hence finally obtain the optical flow for $\mathcal{F}_i$. To obtain the transformation between $\mathbf{C}$ and $\mathbf{F}_i$, we use the following transformation.

$${}^{\mathbf{W}}\mathbf{H}_{\mathbf{C},t} = {}^{\mathbf{F}_i}\mathbf{H}_{\mathbf{W},t}\, {}^{\mathbf{W}}\mathbf{H}_{\mathbf{C},t} = \left({}^{\mathbf{W}}\mathbf{H}_{\mathbf{F}_i,t}\right)^{-1} {}^{\mathbf{W}}\mathbf{H}_{\mathbf{C},t}$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}_{3\times 1} & 1 \end{bmatrix}$$

Obtaining, optical flow of IMO pixels denoted by $\dot{\hat{\mathbf{x}}}_{\mathcal{F}_i}$ is as simple as replacing coordinate frame $\mathbf{W}$ by $\mathbf{F}_i$ in Eqs. A.1, A.2 and A.3. From now on, optical flow computed for background

pixels (non-foreground pixels) denoted by $\mathcal{B}$ is denoted by $\hat{\dot{\mathbf{x}}}_{\mathcal{B}}$ or simply $\hat{\dot{\mathbf{x}}}$ (This is given by Eqs. A.1, A.2 and A.3). Again depth for $\mathcal{F}_i$ is computed as the average depths at the two time instants just like that for $\mathcal{B}$.

Now, that we have ground truth optical flows for both background and IMOs, we can present a metric for quantitative evaluation. The first metric is called Average Endpoint Error or AEE.

$$\text{AEE} = \mathbb{E}\left(\|\hat{\dot{\mathbf{x}}} - \widetilde{\dot{\mathbf{x}}}\|_2\right)$$

Here, $\widetilde{\dot{\mathbf{x}}}$ is the estimated/predicted optical flow. We compute AEE for background and IMO pixels separately and is denoted as $\text{AEE}_{\mathcal{B}}$ and $\text{AEE}_{\mathcal{F}_i}$ respectively.

For non-geometric computations using optical flow such as tracking on the image plane, it is generally sufficient to have accurate flow direction rather than accurate flow magnitude. A metric to quantitatively evaluate only the flow direction is called Average Angular Error or AAE.

$$\text{AAE} = \mathbb{E}\left(\cos^{-1}\left(\frac{\hat{\dot{\mathbf{x}}} \cdot \widetilde{\dot{\mathbf{x}}}}{\|\hat{\dot{\mathbf{x}}}\|_2 \|\widetilde{\dot{\mathbf{x}}}\|_2}\right)\right)$$

Similar to AEE, we compute AAE separately for background and IMO pixels and is denoted as $\text{AAE}_{\mathcal{B}}$ and $\text{AAE}_{\mathcal{F}_i}$ respectively.

In addition to computing AEE and AAE we also compute the percentage of outliers in each case. This is computed as follows for the error metric AEE and AAE separately.

$$\text{AEE}_{\text{outlier}}\left(\epsilon,\delta\right)=\mathbb{E}\left(1\left(\text{AEE}\left(\hat{\mathbf{x}},\widetilde{\mathbf{x}}\right)\geq\min\left(\epsilon,\delta\hat{\mathbf{x}}\right)\right)\right)$$

$$\text{AAE}_{\text{outlier}}\left(\epsilon,\delta\right)=\mathbb{E}\left(1\left(\text{AAE}\left(\hat{\mathbf{x}},\widetilde{\mathbf{x}}\right)\geq\epsilon\right)\right)$$

where ϵ denotes the maximum acceptable error tolerance, δ denotes the maximum acceptable percentage error tolerance and 1 denotes the indicator function. $\text{AEE}_{\text{outlier}}$ and $\text{AAE}_{\text{outlier}}$ are computed separately for $\mathcal{B}$ and $\mathcal{F}_i$ and are denoted by $\text{AEE}_{\text{outlier},\mathcal{B}}$, $\text{AEE}_{\text{outlier},\mathcal{F}_i}$ and $\text{AAE}_{\text{outlier},\mathcal{B}}$, $\text{AAE}_{\text{outlier},\mathcal{F}_i}$ respectively.

A.3.3 Depth and Pose

Though the main aim of the dataset is to facilitate monocular IMO segmentation and optical flow, we also release per-pixel depth to enable learning based algorithms to learn segmentation based on depth disparity or border ownership methods. We also present pose of the camera to enable Visual Odometry or Ego-motion algorithm development.

If the ground truth and predicted depth at a pixel location $\mathbf{x}$ are given by $\hat{Z}_{\mathbf{x}}$ and $\widetilde{Z}_{\mathbf{x}}$, we defined the depth loss with the following three metrics defined below based on standard literature [177, 178].

$$\text{SILog} = \mathbb{E}\left(\|\mathcal{LD}_{\mathbf{x}}\|_2^2\right) - \mathbb{E}\left(\|\mathcal{LD}_{\mathbf{x}}\|_1\right)^2$$

$$\text{RMSE}_{\text{lin}} = \sqrt{\mathbb{E}\left(\|\mathcal{D}_{\mathbf{x}}\|_2^2\right)}$$

$$\text{RMSE}_{\text{log}} = \sqrt{\mathbb{E}\left(\|\mathcal{LD}_{\mathbf{x}}\|_2^2\right)}$$

$$\mathcal{D}_{\mathbf{x}} = Z_{\hat{\mathbf{x}}} - Z_{\widetilde{\mathbf{x}}}$$

$$\mathcal{LD}_{\mathbf{x}} = \log\left(Z_{\hat{\mathbf{x}}}\right) - \log\left(Z_{\widetilde{\mathbf{x}}}\right)$$

SILog, RMSE_{lin} and RMSE_{log} denote the Scale Invariant Logarithmic error, Linear Root Mean Square Error and Logarithmic Root Mean Square Error respectively.

For evaluating pose/odometry quantitatively, we adopt two error metrics from literature [179] defined next. Let the sequence of poses denote a trajectory. For eg., if pose at time t is given by $\mathbf{H}_t \in \text{SE(3)}$, then the trajectory from t_0 to t_1 is given by $\mathcal{H} = \{\mathbf{H}_t \mid \forall t \in [t_0, t_1]\}$. Now, let $\hat{\mathcal{H}}, \widetilde{\mathcal{H}}$ denote the ground truth and predicted pose sequence or trajectory. The first error metric is called Relative Pose Error or RPE which measures the amount of drift, particularly useful for odometry algorithms. The RPE and it's RMSE is given as follows.

$$\text{RPE}_t = \left(\hat{\mathcal{H}}_{t_0}^{-1}\hat{\mathcal{H}}_{t_1}\right)^{-1}\left(\widetilde{\mathcal{H}}_{t_0}^{-1}\widetilde{\mathcal{H}}_{t_1}\right)$$

$$\text{RPE}_{\text{RMSE}} = \sqrt{\mathbb{E}\left(\|\text{trans}\left(\text{RPE}_t\right)\|_2^2\right)}$$

Here $\text{trans}\left(\text{RPE}_t\right)$ refers to the translational component of RPE at time t.

For measuring global consistency of trajectories of SLAM systems, Absolute Trajectory

Error or ATE is used and is defined as follows.

$$\text{ATE}_t = \hat{\mathcal{H}}_t^{-1} \mathbf{S} \widetilde{\mathcal{H}}_t$$

$$\text{ATE}_{\text{RMSE}} = \sqrt{\mathbb{E}\left(\| \text{trans}\left(\text{ATE}_t \right) \|_2^2 \right)}$$

Here $\mathbf{S}$ refers to the least squares best fit rigid body transformation between $\hat{\mathcal{H}}$ and $\widetilde{\mathcal{H}}$ which includes scale, rotation and translation.

Bibliography

[1] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. 2020.

[2] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] Christoph Posch, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Tobi Delbruck. Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014.

[5] Pengpeng Liu, Michael R. Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *CVPR*, 2019.

[6] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8981–8989, 2018.

[7] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. 2018.

[8] Cornelia Fermüller, Robert Pless, and Yiannis Aloimonos. The ouchi illusion as an artifact of biased flow estimation. *Vision Research*, 40(1):77–95, 2000.

[9] 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*, 2019.

[10] Wulfram Gerstner. Time structure of the activity in neural network models. *Physical review E*, 51(1):738, 1995.

[11] Richard Hartley Andrew Zisserman. Multiple view geometry in computer vision. 2004.

[12] M Srinivasan, Shaowu Zhang, M Lehrer, and TS Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199(1):237–244, 1996.

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

[14] John Aloimonos. Purposive and qualitative active vision. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume 1, pages 346–360. IEEE, 1990.

[15] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[16] Roger Y Tsai and Thomas S Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on pattern analysis and machine intelligence*, (1):13–27, 1984.

[17] Cornelia Fermüller and Yiannis Aloimonos. Observability of 3d motion. *International Journal of Computer Vision*, 37(1):43–63, 2000.

[18] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[19] Cornelia Fermüller. Passive navigation as a pattern recognition problem. *International Journal of Computer Vision*, 14(2):147–158, 1995.

[20] Cornelia Fermüller, LoongFah Cheong, and Yiannis Aloimonos. Visual space distortion. *Biological Cybernetics*, 77(5):323–337, 1997.

[21] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 6:1052–1067, 2007.

[22] Martin-Paul Agbaga, Dana K Merriman, Richard S Brush, Todd A Lydic, Shannon M Conley, Muna I Naash, Shelley Jackson, Amina S Woods, Gavin E Reid, Julia V Busik, et al. Differential composition of dha and very-long-chain pufas in rod and cone photoreceptors. *Journal of lipid research*, 59(9):1586–1596, 2018.

[23] J. Aloimonos. Purposive and qualitative active vision. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume i, pages 346–360 vol.1, 1990.

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

[25] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc 7th Intl Joint Conf on Artificial Intelligence (IJ CAI)*, 1981.

[26] Y. Aloimonos. *Visual Navigation: From Biological Systems To Unmanned Ground Vehicles*. Computer Vision Series. Taylor & Francis, 2013.

[27] Semir Zeki. *A vision of the brain.* Blackwell scientific publications, 1993.

[28] Jonathan Chey, Stephen Grossberg, and Ennio Mingolla. Neural dynamics of motion grouping: From aperture ambiguity to object speed and direction. *JOSA A*, 14(10):2570–2594, 1997.

[29] Mortimer Mishkin, Leslie G Ungerleider, and Kathleen A Macko. Object vision and spatial vision: two cortical pathways. *Trends in neurosciences*, 6:414–417, 1983.

[30] Sabrina Pitzalis, Patrizia Fattori, and Claudio Galletti. The functional role of the medial motion area v6. *Frontiers in behavioral neuroscience*, 6:91, 2013.

[31] Samantha L Strong, Edward H Silson, André D Gouws, Antony B Morland, and Declan J McKeefry. Differential processing of the direction and focus of expansion of optic flow stimuli in areas mst and v3a of the human visual cortex. *Journal of neurophysiology*, 117(6):2209–2217, 2017.

[32] Cornelia Fermüller, Robert Pless, and Yiannis Aloimonos. Families of stationary patterns producing illusory movement: insights into the visual system. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 264(1383):795–806, 1997.

[33] Cornelia Fermuller, Tomas Brodsky, and Yiannis Aloimonos. Motion segmentation: a synergistic approach. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 226–231. IEEE, 1999.

[34] Francisco Barranco, Cornelia Fermüller, Yiannis Aloimonos, and Eduardo Ros. Joint direct estimation of 3d geometry and 3d motion using spatio temporal gradients. *Pattern Recognition*, 113:107759, 2021.

[35] Chethan M Parameshwara, Gokul Hari, Cornelia Fermüller, Nitin J Sanket, and Yiannis Aloimonos. Diffposenet: Direct differentiable camera pose estimation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[36] 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.

[37] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008.

[38] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor. *Frontiers in neuroscience*, 7:223, 2013.

[39] 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.

[40] Martin Litzenberger, Bernhard Kohn, Ahmed Nabil Belbachir, Nikolaus Donath, Gerhard Gritsch, Heinrich Garn, Christoph Posch, and Stephan Schraml. Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor. In *2006 IEEE intelligent transportation systems conference*, pages 653–658. IEEE, 2006.

[41] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. Hfirst: A temporal approach to object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2028–2040, 2015.

[42] Jun Haeng Lee, Tobi Delbruck, Michael Pfeiffer, Paul KJ Park, Chang-Woo Shin, Hyunsurk Ryu, and Byung Chang Kang. Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE transactions on neural networks and learning systems*, 25(12):2250–2263, 2014.

[43] Paul Rogister, Ryad Benosman, Sio-Hoi Ieng, Patrick Lichtsteiner, and Tobi Delbruck. Asynchronous event-based binocular stereo matching. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):347–353, 2011.

[44] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, 126(12):1394–1414, 2018.

[45] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–417, 2013.

[46] Alex Zihao Zhu and Liangzhe Yuan. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Robotics: Science and Systems*, 2018.

[47] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[48] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2016.

[49] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018.

[50] Chethan M Parameshwara, Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. 0-mms: Zero-shot multi-motion segmentation with a monocular event camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9594–9600. IEEE, 2021.

[51] Chethan M Parameshwara, Simin Li, Cornelia Fermüller, Nitin J Sanket, Matthew S Evanusa, and Yiannis Aloimonos. Spikems: Deep spiking neural network for motion segmentation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3414–3420. IEEE.

[52] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[53] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[54] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, 2017.

[55] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[56] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *CoRR*, abs/1803.02276, 2018.

[57] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

[58] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[59] Stephen Gould, Richard Hartley, and Dylan John Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[60] Dylan Campbell, Liu Liu, and Stephen Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *European Conference on Computer Vision*, pages 244–261. Springer, 2020.

[61] Nitin J. Sanket, Chethan M. Parameshwara, Chahat Deep Singh, Ashwin V. Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras, 2019.

[62] Simon Davidson and Steve B Furber. Comparison of artificial and spiking neural networks on digital hardware. *Frontiers in Neuroscience*, 15:345, 2021.

[63] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.

[64] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks. In *Proceedings of the 31st annual ACM symposium on applied computing*, pages 293–298, 2016.

[65] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[66] 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.

[67] G. Haessig, A. Cassidy, R. Alvarez, and G. Benosman, R.and Orchard. Spiking optical flow for event-based sensors using ibm's truenorth neurosynaptic system, 2018.

[68] Federico Paredes-Vallés, Kirk YW Scheper, and Guido CHE de Croon. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE transactions on pattern analysis and machine intelligence*, 42(8):2051–2064, 2019.

[69] 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.

[70] Germain Haessig, Xavier Berthelon, Sio-Hoi Ieng, and Ryad Benosman. A spiking neural network model of depth from defocus for event-based neuromorphic vision. *Scientific reports*, 9(1):1–11, 2019.

[71] Alpha Renner, Matthew Evanusa, and Yulia Sandamirskaya. Event-based attention and tracking on neuromorphic hardware. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1709–1716. IEEE, 2019.

[72] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & Automation Magazine*, 18(4):80–92, 2011.

[73] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[74] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, pages 25–36. Springer, 2004.

[75] Hui Ji and Cornelia Fermüller. Better flow estimation from color images. *EURASIP Journal on Advances in Signal Processing*, 2007:1–9, 2007.

[76] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2010.

[77] Cornelia Fermüller, David Shulman, and Yiannis Aloimonos. The statistics of optical flow. *Computer Vision and Image Understanding*, 82(1):1–32, 2001.

[78] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470, 2017.

[79] Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 557–572. Springer, 2020.

[80] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4161–4170, 2017.

[81] S Negahdaripour. Direct methods for structure from motion. *Ph. D Thesis, Mechanical Engineering, MIT*, 1986.

[82] Bernhard P. Wrobel. Multiple view geometry in computer vision. *Künstliche Intell.*, 15:41, 2001.

[83] Yiannis Aloimonos and Zoran Duric. Estimating the heading direction using normal flow. *International Journal of Computer Vision*, 13(1):33–56, 1994.

[84] Berthold KP Horn and EJ Weldon. Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76, 1988.

[85] David Sinclair, Andrew Blake, and David Murray. Robust estimation of egomotion from normal flow. *International Journal of Computer Vision*, 13(1):57–69, 1994.

[86] Tomas Brodsky, Cornelia Fermüller, and Yiannis Aloimonos. Shape from video. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 2146–2146. IEEE Computer Society, 1999.

[87] Tomáš Brodskỳ and Cornelia Fermüller. Self-calibration from image derivatives. *International Journal of Computer Vision*, 48(2):91–114, 2002.

[88] Hui Ji and Cornelia Fermuller. A 3d shape constraint on video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1018–1023, 2006.

[89] César Silva and José Santos-Victor. Robust egomotion estimation from the normal flow using search subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1026–1034, 1997.

[90] Ding Yuan, Miao Liu, and Hong Zhang. Direct ego-motion estimation using normal flows. In *2013 2nd IAPR Asian Conference on Pattern Recognition*, pages 310–314. IEEE, 2013.

[91] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[92] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.

[93] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564, 2017.

[94] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020.

[95] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[96] W. Zhao, S. Liu, Y. Shu, and Y. Liu. Towards better generalization: Joint depth-pose learning without posenet. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9148–9158, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[97] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. *arXiv preprint arXiv:2011.00359*, 2020.

[98] Shihao Jiang, Dylan Campbell, Miaomiao Liu, Stephen Gould, and Richard Hartley. Joint unsupervised learning of optical flow and egomotion with bi-level optimization. In *2020 International Conference on 3D Vision (3DV)*, pages 682–691. IEEE, 2020.

[99] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. L-bfgs-b: a limited memory fortran code for solving bound constrained optimization problems. *Technique Report Rep*, 1994.

[100] Nitin J Sanket, Chahat Deep Singh, Chethan M Parameshwara, Cornelia Fermüller, Guido CHE de Croon, and Yiannis Aloimonos. EVPropNet: Detecting Drones By Finding Propellers For Mid-Air Landing And Following. In *Robotics: Science and systems (RSS) conference 2021*. Robotics: Science and Systems, 2021.

[101] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[102] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[103] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[104] Bas J. Pijnacker Hordijk, Kirk Y. W. Scheper, and Guido C.H.E. de Croon. Vertical landing for micro air vehicles using event-based optical flow. *J. Field Robotics*, 35:69–90, 2018.

[105] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[106] Simon Schrodi, Tonmoy Saikia, and Thomas Brox. What causes optical flow networks to be vulnerable to physical adversarial attacks. *arXiv preprint arXiv:2103.16255*, 2021.

[107] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, New Orleans, Louisiana, February 2018.

[108] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291, 2018.

[109] Xiangwei Wang, Daniel Maturana, Shichao Yang, Wenshan Wang, Qijun Chen, and Sebastian Scherer. Improving learning-based ego-motion estimation with homomorphism-based losses and drift correction. In *Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 970 – 976, November 2019.

[110] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[111] Shiyu Song, Manmohan Chandraker, and Clark C. Guest. High accuracy monocular sfm and scale correction for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):730–743, 2016.

[112] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[113] H. Zhou, B. Ummenhofer, and T. Brox. Deeptam: Deep tracking and mapping. In *European Conference on Computer Vision (ECCV)*, 2018.

[114] Adithya Prem Anand, H Gokul, Harish Srinivasan, Pranav Vijay, and Vineeth Vijayaraghavan. Adversarial patch defense for optical flow networks in video action recognition. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1289–1296, 2020.

[115] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael Black. Attacking optical flow. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2404–2413, 2019.

[116] Cornelia Fermüller. Navigational preliminaries. *Active Perception*, 1:103–150, 1993.

[117] Nitin J Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermüller, and Yiannis Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. volume 3, pages 2799–2806. IEEE, 2018.

[118] Nitin J Sanket, Chahat Deep Singh, Varun Asthana, Cornelia Fermüller, and Yiannis Aloimonos. Morpheyes: Variable baseline stereo for quadrotor navigation. *arXiv preprint arXiv:2011.03077*, 2020.

[119] Huai-Jen Liang, Nitin J Sanket, Cornelia Fermüller, and Yiannis Aloimonos. Salientdso: Bringing attention to direct sparse odometry. *IEEE Transactions on Automation Science and Engineering*, 16(4):1619–1626, 2019.

[120] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128, 120 db 15 μ s latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008.

[121] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018.

[122] M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schon, B. Kohn, and H. Garn. Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. In *IEEE 12th Digital Signal Processing Workshop and 4th IEEE Signal Processing Education Workshop*, pages 173–178, 2006.

[123] Alejandro Linares-Barranco, Francisco Gómez-Rodríguez, Vicente Villanueva, Luca Longinotti, and Tobi Delbrück. A usb3. 0 fpga event-based filtering and tracking framework for dynamic vision sensors. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2417–2420. IEEE, 2015.

[124] Abhishek Mishra, Rohan Ghosh, Jose C Principe, Nitish V Thakor, and Sunil L Kukreja. A saccade based framework for real-time motion segmentation using event based vision sensors. *Frontiers in neuroscience*, 11:83, 2017.

[125] Francisco Barranco, Cornelia Fermüller, and Eduardo Ros. Real-time clustering and multi-target tracking using event-based sensors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5764–5769, 2018.

[126] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2):632–639, 2017.

[127] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based visual inertial odometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5816–5824, July 2017.

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

[129] Timo Stoffregen and Lindsay Kleeman. Event cameras, contrast maximization and reward functions: An analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[130] Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. Event-based motion segmentation by motion compensation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7244–7253, 2019.

[131] Francisco Barranco, Cornelia Fermüller, and Yiannis Aloimonos. Bio-inspired motion estimation with event-driven sensors. In *International Work-Conference on Artificial Neural Networks*, pages 309–321. Springer, 2015.

[132] Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbruck. EV-IMO: motion segmentation dataset and learning pipeline for event cameras. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2019.

[133] Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermuller, and Yiannis Aloimonos. Learning visual motion segmentation using event surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14414–14423, 2020.

[134] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018.

[135] Hochang Seok and Jongwoo Lim. Robust feature tracking in dvs event stream using bézier mapping. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1658–1667, 2020.

[136] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Eklt: Asynchronous photometric feature tracking using events and frames. *International Journal of Computer Vision*, 128(3):601–618, 2020.

[137] Laurent Dardelet, Sio-Hoi Ieng, and Ryad Benosman. Event-based features selection and tracking from intertwined estimation of velocity and generative contours. *arXiv preprint arXiv:1811.07839*, 2018.

[138] Ignacio Alzugaray and Margarita Chli. Asynchronous multi-hypothesis tracking of features with event cameras. In *2019 International Conference on 3D Vision (3DV)*, pages 269–278. IEEE, 2019.

[139] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Deep Learning for Visual SLAM Workshop*, 2018.

[140] Qi Wang, Mulin Chen, Feiping Nie, and Xuelong Li. Detecting coherent groups in crowd scenes by multiview clustering. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):46–58, 2018.

[141] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pages 282–295. Springer, 2010.

[142] Nitin J Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. PRGFlow: Benchmarking SWAP-Aware Unified Deep Visual Inertial Odometry. *arXiv preprint arXiv:2006.06753*, 2020.

[143] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[144] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in Neuroscience*, 12:774, 2018.

[145] Mathias Gehrig, Sumit Bam Shrestha, Daniel Mouritzen, and Davide Scaramuzza. Event-based angular velocity regression with spiking networks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4195–4202. IEEE, 2020.

[146] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[147] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.

[148] 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.

[149] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, 2010.

[150] 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):500–544, 1952.

[151] Terrence J Sejnowski and Gerald Tesauro. The Hebb rule for synaptic plasticity: algorithms and implementations. In *Neural Models of Plasticity*, pages 94–103. Elsevier, 1989.

[152] Bernhard Nessler, Michael Pfeiffer, and Wolfgang Maass. Stdp enables spiking neurons to detect hidden causes of their inputs. *Advances in neural information processing systems*, 22:1357–1365, 2009.

[153] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[154] Matthew Evanusa, Cornelia Fermüller, and Yiannis Aloimonos. A deep 2-dimensional dynamical spiking neuronal network for temporal encoding trained with STDP. *arXiv preprint arXiv:2009.00581*, 2020.

[155] Chankyu Lee, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning. *Frontiers in Neuroscience*, 12:435, 2018.

[156] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[157] 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 *European Conference on Computer Vision*, pages 366–382. Springer, 2020.

[158] W. Gerstner. Spike-response model. *Scholarpedia*, 3(12):1343, 2008. revision #91800.

[159] Mark van Rossum. A novel spike distance. *Neural Computation*, 13:751–763, 04 2001.

[160] T. Kreuz. Measures of spike train synchrony. *Scholarpedia*, 6(10):11934, 2011. revision #190333.

[161] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[162] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[163] Alina Kloss, Georg Martius, and Jeannette Bohg. How to train your differentiable filter. *Autonomous Robots*, 45(4):561–578, 2021.

[164] Jason Fischer, John G Mikhael, Joshua B Tenenbaum, and Nancy Kanwisher. Functional neuroanatomy of intuitive physical inference. *Proceedings of the national academy of sciences*, 113(34):E5072–E5081, 2016.

[165] Xun Xu, Loong-Fah Cheong, and Zhuwen Li. 3d rigid motion segmentation with mixed and unknown number of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):1–16, 2019.

[166] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5391–5399, 2017.

[167] B. Son, Y. Suh, S. Kim, H. Jung, J. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu. 4.1 a 640×480 dynamic vision sensor with a 9µm pixel and 300meps address-event representation. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66–67, 2017.

[168] Prophesee s.a. and sony corporation today announced they have jointly developed a stacked event-based vision sensor with the industry's smallest 4.86um pixel size and the industry's highest 124db (or more) hdr performance, Feb 2020.

[169] 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.

[170] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. *Advances in Neural Information Processing Systems*, 33:16639–16652, 2020.

[171] Srutarshi Banerjee, Zihao W Wang, Henry H Chopp, Oliver Cossairt, and Aggelos K Katsaggelos. Lossy event compression based on image-derived quad trees and poisson disk sampling. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2154–2158. IEEE, 2021.

[172] Denis Kleyko, Mike Davies, E Paxon Frady, Pentti Kanerva, Spencer J Kent, Bruno A Olshausen, Evgeny Osipov, Jan M Rabaey, Dmitri A Rachkovskij, Abbas Rahimi, et al. Vector symbolic architectures as a computing framework for nanoscale hardware. *arXiv preprint arXiv:2106.05268*, 2021.

[173] Zhuowen Zou, Haleh Alimohamadi, Farhad Imani, Yeseong Kim, and Mohsen Imani. Spiking hyperdimensional network: Neuromorphic models integrated with memory-inspired framework. *arXiv preprint arXiv:2110.00214*, 2021.

[174] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *British Machine Vision Conference (BMVC)*, 2018.

[175] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. ESIM: an open event camera simulator. *Conf. on Robotics Learning (CoRL)*, October 2018.

[176] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, July 2018.

[177] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014.

[178] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[179] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, July 2018.