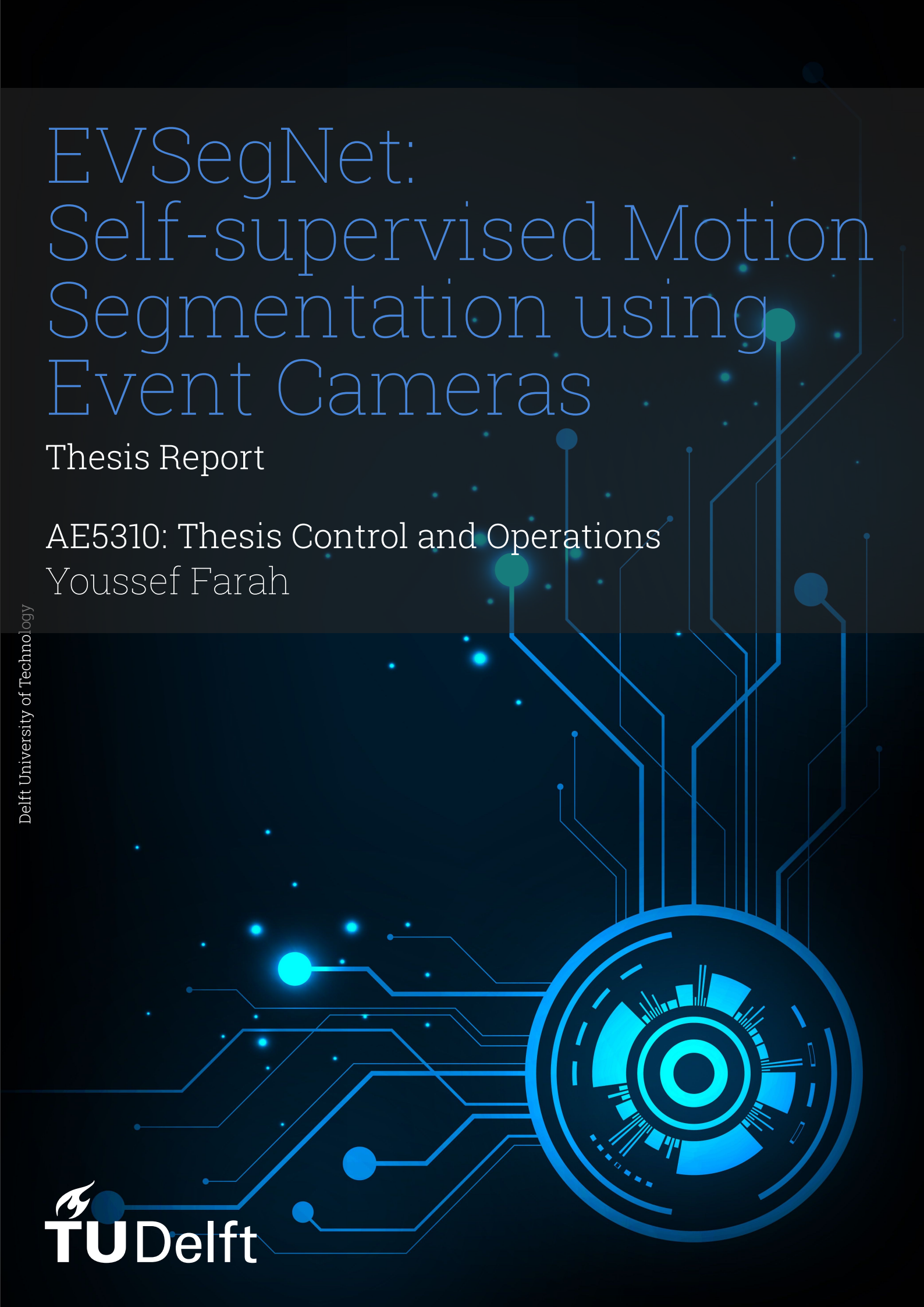
EVSegNet: Self-supervised Motion Segmentation using Event Cameras

Thesis Report

AE5310: Thesis Control and Operations

Youssef Farah

Delft University of Technology



TU Delft

EVSegNet: Self-supervised Motion Segmentation using Event Cameras

Thesis Report

by

Youssef Farah

to obtain the degree of Master of Science
at Delft University of Technology,
to be defended publicly on Thursday July 13, 2023 at 14:00.

Student number:	4651510	
Project Duration:	May, 2022 - July, 2023	
Thesis committee:	Prof. dr. ir. G.C.H.E. de Croon	TU Delft, supervisor
	Dr. Ir. E. Mooij	TU Delft
	Dr. Ir. J. Ellerbroek	TU Delft
	Ir. F. Paredes-Vallés	Sony

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

Alhamdulillah.

Youssef Farah
Delft, July 2023

Contents

List of Figures

Nomenclature

Abbreviations

Abbreviation	Definition
ALIF	Adaptive Leaky-Integrate-and-Fire
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
DVS	Dynamic Vision Sensor
EM	Expectation Maximisation
G	Generator
HDR	High Dynamic Range
I	Inpainter
IMIP	IMage InPainter module
IRR	Information Reduction Rate
IWE	Image of Warped Events
LDIS	Layered Differentiable Image Synthesis
LIF	Leaky-Integrate-and-Fire
MAP	Maximum A-posteriori Probability
MASNet	Multimotion and Appearance Self-supervised Network
MOD	Moving Object Dataset
MFE	Multibranch Flow Encoding module
OF	Optical Flow
PF	Particle Filter
PLIF	Presynaptic Leaky-Integrate-and-Fire
SLAM	Simulatenous Localization And Mapping
SNN	Spiking Neural Networks
SRM	Spike Response Model
VOS	Video Object Segmentation
XLIF	Crossover Leaky-Integrate-and-Fire

Symbols

Symbol	Definition	Unit
α	membrane decay	
α	alpha map	
δ	Dirac function	
ε_d	spike response kernel	
μ_H	mean of warped events	
η	threshold decay	
ϕ_{w1}	inpainter network	
χ_{w2}	mask generator network	
τ_r	refractory time constant	
τ_r	spike response time constant	
Φ_{bind}	iterative binding module	
Φ_{dec}	CNN decoder	
Φ_{enc}	CNN encoder	

Symbol	Definition	Unit
Ω	spatial grid	
Ω	image domain foreground	
Ω^c	image domain background	
$\mathcal{E}$	event packet	
c	scaling factor	
I	photocurrent	
I	synaptic current	
I	Image	
F	Feature Map	
$\mathbb{H}$	Shannon Entropy	
L	log photocurrent	
L	RGB intensity map	
$\mathcal{L}$	Loss	
$\mathcal{L}_{\text{contrast}}$	temporal loss	
$\mathcal{L}_{\text{cons}}$	consistency loss	
$\mathcal{L}_{\text{flow}}$	flow loss	
$\mathcal{L}_{\text{recon}}$	reconstruction loss	
$\mathcal{L}_{\text{smooth}}$	Charbonnier smoothness loss	
$\mathcal{L}_{\text{total}}$	total loss	
ΔL	intensity increment	
∇L	brightness gradient	
θ	synaptic threshold	
e	event	
t	time	
t_{ref}	reference time	
$t_{\text{ref}}^{\text{bw}}$	reference time backward	
$t_{\text{ref}}^{\text{fw}}$	reference time forward	
Δt	time increase	
C	contrast threshold	
C^+	upper contrast threshold	
C^-	lower contrast threshold	
d	delay	
D	image domain	
F	Flow map	
F^{m}	masked flow map	
F^{r}	not masked flow map	
M	segmentation mask	
p	polarity	
p	pixel index	
$\mathbf{v}$	velocity vector	
x	x-coordinate pixel location in x-y coordinate system	
$\mathbf{x}$	pixel location	
y	y-coordinate pixel location in x-y coordinate system	
N_e	number of events	
N_l	number of neurons	
N_p	number of pixels	
θ	model parameters	
H	model parameters	
H	horizontal dimension output	
H_0	horizontal dimension input	
$\mathcal{H}(t)$	Heaviside function	
f	objective function	
f_s	thresholding function	
P	presynaptic trace	

Symbol	Definition	Unit
S	neuron spike	
$T_{p'}$	average timestamp per pixel	
u	optic flow	
u_{in}	foreground optic flow	
u_{out}	background optic flow	
U	membrane potential	
$\mathbf{v}$	velocity vector	
$\mathbf{W}$	Warping function	
W	optical flow map	
W^{ff}	feedforward weight connections	
W^{rec}	recurrent weight connections	
w	flow map	
W	vertical dimension output	
W_0	vertical dimension input	

1 Introduction

1.1. Background

Event cameras are revolutionizing the field of computer vision. In contrast to traditional cameras, these bio-inspired devices operate on a distinct principle, making them exceptional tools for capturing and processing visual information.

Unlike conventional frame cameras that capture static snapshots, event cameras are designed to detect and record changes in brightness within a scene. Rather than registering pixel color values, these remarkable devices focus solely on recording the occurrence of brightness variations. This dynamic approach sets them apart from their traditional counterparts.

Another noteworthy distinction lies in the asynchronous nature of event camera recordings. Instead of capturing frames at a fixed rate, event cameras only record and relay information when a significant brightness change takes place. Consequently, these cameras excel in scenarios where the scene exhibits substantial changes, effectively eliminating redundant data capture in light-invariant situations.

The recorded brightness changes in event cameras are represented as a series of events, providing precise information about when and where these alterations occurred within the visual field. This unique data structure opens up to a large number of applications, particularly in the domains of computer vision and robotics.

One compelling application of event cameras is motion segmentation, a process aimed at identifying and delineating the shapes of moving objects within a scene observed by a mobile camera. While extensive research exists on motion segmentation, the existing methods are not-compatible for event cameras due to the fundamental differences in data structure.

To fully harness the potential of event cameras in motion-related computer vision tasks, new algorithms tailored specifically to their data characteristics must be developed. Moreover, contemporary frameworks heavily rely on training with large annotated datasets, which are expensive to create and limit the scalability of such methods. Thus, the development of event-based motion segmentation methods that can operate effectively without the need for annotated training data would unlock the full potential of event cameras in various computer vision applications.

1.2. Problem Statement

Existing algorithms for event-based motion segmentation combine statistical methods and neural networks. However, the learning-based methods rely on ground truth, which poses several limitations.

First, events alone carry very little information, as each one of them just represents a change in brightness in a specific pixel at a specific time. Therefore, creating ground truth for each single event is still extremely difficult.

Second, the only way to generate ground truth for event data is by using additional sensors and cameras.

And the problem with generating ground truth with other sensors is that any other sensor does not have the same high temporal resolution as the event cameras. This inevitably brings error measurements in the generation of ground truth.

Finally, methods that learn using ground truth are limited by the ground truth itself, meaning that they work well only in scenes similar to the ones included in the training set.

The main question at this point is whether it is possible to perform event-based motion segmentation without the need of ground truth, and it is possible. Current unsupervised methods rely on statistics to cluster together the events generated by the same moving object. Nevertheless, the computational time to generate the clusters is too long to be deployed in real-word applications.

Hence the next question would be whether it possible to perform event-based motion segmentation without using ground truth, and such that it could be implemented in real-word applications. In this regard, self-supervised learning allows deep learning architectures to learn from events without the use of ground truth. However, there is no existence of a self-supervised method to perform motion segmentation using events, which brings us to the main problem statement of this thesis project:

Research Question: How can we perform motion segmentation using event cameras without using ground truth during training, given the nature of event data and the state-of-the-art methods in motion segmentation?

1.3. Report Structure

To address the research question, the report is structured as follows.

Part I contains the main contribution of this thesis report, which is the scientific article. This chapter can be read as a stand-alone document, and introduces the first self-supervised method for motion segmentation using events. It provides an overall overview of the state-of-the-art in motion segmentation, and introduces a new motion segmentation method in the event domain by getting inspiration from the most recent advancements in frame-based computer vision. The article also introduces a new dataset that is used to train and test the newly develop architecture. At the end of the article, the final conclusions are drawn and directions for future work are provided.

Part II introduces the literature study results. The literature study was carried at the beginning of the project, and served to gather all the necessary knowledge in motion segmentation (both event-based and image-based), and self-supervised learning using events. First, a general explanation of the working principle behind event cameras is given in chapter 2, along with advantages and challenges in using this type of sensors. Second, in chapter 3 the topic of optical flow estimation is introduced, where the most relevant advancements in self-supervised learning have been made. Finally, motion segmentation is reviewed in chapter 4 by giving a general overview of the methods, and then diving deep into the state-of-the-art in both event domain and image domain. The literature study was concluded with a preliminary outline of the subsequent steps in chapter 5.

Part I

Scientific Paper

EV-LayerSegNet: Self-supervised Motion Segmentation using Event Cameras

Youssef Farah *
Micro Air Vehicle Laboratory
Delft University of Technology
Delft, The Netherlands
Y.Farah@student.tudelft.nl

Federico Paredes-Vallés †
Micro Air Vehicle Laboratory
Delft University of Technology
Delft, The Netherlands
F.ParedesValles@tudelft.nl

Guido C.H.E. de Croon †
Micro Air Vehicle Laboratory
Delft University of Technology
Delft, The Netherlands
G.C.H.E.deCroon@tudelft.nl

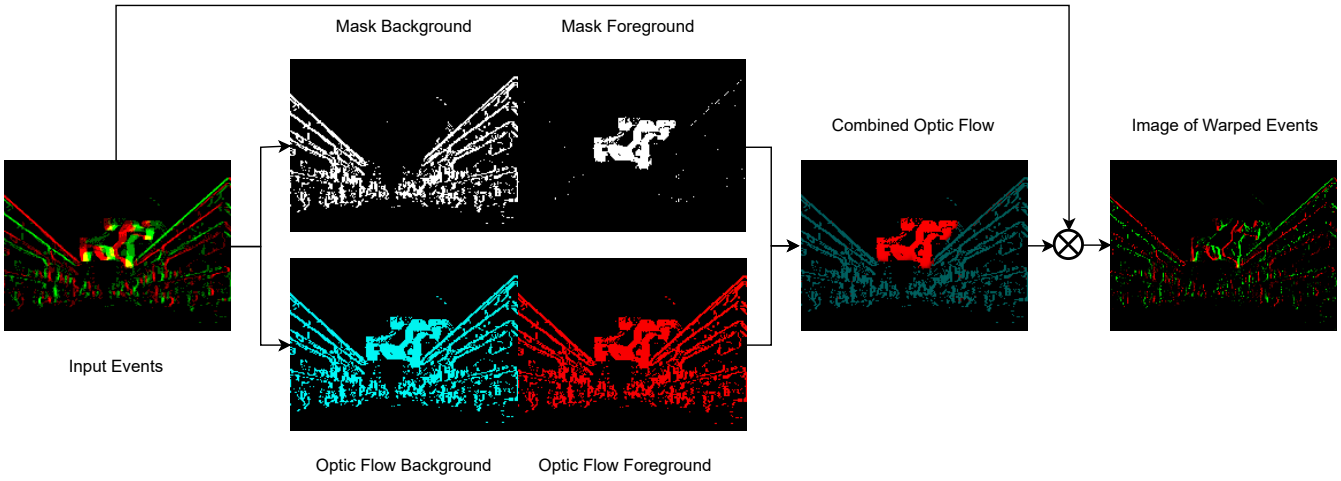


Fig. 1. Our method takes as input an event volume that results in a blurry scene. It then attempts to generate two different masks to differentiate the background from the foreground. Next, it estimates the affine optical flow for both models, and combines the flow together using the masks. Finally, it warps the events according to the combined flow. A successful motion deblur leads to accurate segmentation.

***Abstract*—Event cameras are novel bio-inspired sensors that capture motion dynamics with much higher temporal resolution than traditional cameras, since pixels react asynchronously to brightness changes. They are therefore better suited for tasks involving motion such as motion segmentation. However, training event-based networks still represents a difficult challenge, as obtaining ground truth is very expensive and error-prone. In this article, we introduce EV-LayerSegNet, the first self-supervised CNN for event-based motion segmentation. Inspired by a layered representation of the scene dynamics, we show that it is possible to learn affine optical flow and segmentation masks separately, and use them to deblur the input events. The deblurring quality is then measured and used as self-supervised learning loss.**

***Index Terms*—event-based vision, self-supervised learning, deep learning, motion segmentation, affine layered motion model**

[1]*Student, †Supervisor
[2]Our code: https://github.com/Fulmen67/event_segmentation.git

I. Introduction

Event cameras, such as the Dynamic Vision Sensor (DVS), are novel bio-inspired sensors that perceive motion by detecting brightness changes instead of capturing intensity changes in images within a time interval [4]. In other words, if at time t a brightness change larger than the threshold C is detected by a pixel with x and y coordinates, an event is generated by recording the brightness increase or decrease, the pixel location and the time the event was generated. In contrast, traditional cameras record color intensities in all pixels at a specified frame rate.

This fundamental shift in the acquisition of visual information brings several advantages. High temporal resolution and low latency (in the order of μs) make event cameras well suited for recording very fast moving scenes without suffering from motion blur as frame-based cameras. The High Dynamic Range (HDR) allows the cameras to capture details in very

dark or very bright scenes, making them appropriate to be used in light challenging environments such as the underwater ocean, night driving or fireworks display. The asynchronous recording of brightness leads to very low power consumption, thus event cameras can be deployed on platforms such as very small drones, where weight and power consumption are major constraints.

The combination of these advantages unlocks the possibility of performing tasks inaccessible until now for frame-based cameras such as low-latency optical flow estimation, high-speed control and tracking, and Simultaneous Localization and Mapping (SLAM) [4]. Nevertheless, a new generation of event-based vision algorithms needs to be developed to enable the event cameras to express their full potential, as the data structure of events is not compatible with the conventional algorithms based on images.

Furthermore, an additional challenge is posed by the lack of ground truth from real-word datasets, at microsecond resolution and with HDR. In image-based datasets (COCO [14], BD100K [33]), the ground truth is often obtained by humans manually annotating each frame. This process is already expensive and laborious for image-based datasets, and becomes unfeasible for event data, which are sparse and have low latency. To avoid manual annotation, additional sensors and cameras were used to generate the ground truth in traditional event-based datasets such as DSEC [6] and MVSEC [38]. Yet, sensors and cameras are limited by their natural Field Of View (FOV), spatial and temporal resolution. Therefore, it is of paramount importance to develop algorithms that are able to perform their tasks by relying on the nature of events, without the use of ground truth that are expensive to collect and that carry error measurements from additional sensors.

In the context of motion estimation and its use in other tasks such as surveillance, tracking or obstacle avoidance, segmenting the scene into independently moving objects is fundamental and often referred as *motion segmentation* [25]. Even though great progress has been made using frame-based cameras, the latter are not ideally suited for tasks involving motion as they suffer from motion blur and keep track of all pixel values even when no motion occurs [25]. On the other hand, assuming constant illumination, event cameras are well suited for this task, since events are sampled at exactly the same rate of the scene dynamics and the information acquired is only related to the motion of the camera and the objects in the scene .

Event-based motion segmentation can be broadly categorized in model-based methods ([25], [36], [18]) and learning-based methods ([23], [17], [19]). Model-based methods exploit the nature of event data to cluster events generated by Independently Moving Objects (IMO) in an iterative fashion, thus they are unsupervised and do not use ground truth. Yet, the iterative approach requires long computational time and limits their real-word applicability. On the other hand, learning-based methods apply deep learning techniques while also relying on the event characteristics. In some instances, they perform motion segmentation in combination with other tasks such as tracking and optical flow, but they all require ground truth during training. To the best of our knowledge, there is no existence of a learning-based method that is self-supervised, thus learning directly from the input events instead of relying on ground truth.

Looking outside of the event domain, significant progress has been made in self-supervised image segmentation. These methods segment the scene into the dominant moving object as foreground and the rest as background, and learning is driven by image or optic flow reconstruction. In most instances, they frequently make use of layered representation [27]. The pixels are separated into layers based on motion similarity, and they are moved according to the associated motion and synthesized together to reconstruct the next image in the sequence. Recently, *Shrestha et al.* [24] integrated this process into a end-to-end differentiable CNN pipeline which segments the dominant moving object in a video using two consecutive frames.

Inspired by their work, we propose the first self-supervised and learning-based method for motion segmentation using events. Specifically, we transfer the CNN segmentation approach from [24] to the event domain, and combine it with the encoder-decoder structure from [39]. We show that, under the assumption of affine motion and constant brightness, it is possible to segment independently moving objects using contrast maximization loss.

We identify our main contributions as follows:

1) We propose a novel self-supervised network for learning event-based motion segmentation in an end-to-end CNN with raw events as input.
2) We introduce a novel optical flow module that enables self-supervised learning of affine optical flow and a segmentation module that learns separately the masks corresponding to the independently moving objects.
3) We show that our network is able to learn motion segmentation without the use of any sort of ground truth or other inputs apart from the event data. We contribute a new event-based dataset of background and several objects moving according to affine motion.

II. Related Work

In computer vision, motion segmentation is defined as the task of retrieving the shape of moving objects [35]. Image difference such as in [34] attempt to detect moving objects by finding intensity differences between pixels across video frames. Layer-based techniques ([5],[9]) divide frames into layers, based on the number of uniform motions. Statistical methods such as Expectation Maximization (EM) are among the most common methods used. Recent developments in deep learning and optical flow estimation allowed a sharp improvement in this motion task, often combined with statistics and layer-based techniques. However, the main limitation of the state-of-the-art methods in motion segmentation is the reliance on ground truth labels, which limit the applicability of such methods scenes outside the annotated datasets. In this regard, attempts to develop self-supervised methods have resulted in

frameworks developed in the frame domain, while methods for event cameras were yet to be found.

A. Frame-based Motion Segmentation

In literature, many methods are considered unsupervised or self-supervised, but only during inference and test. In reality, parts of their architecture (i.e. networks, masks) are pretrained on ground truth. For example, COSNet [15] uses co-attention to capture rich correlations between frames of a video, but masks are pretrained on ground truth. Similarly, *Ye et al.* [32] perform motion segmentation using global sprites, but it also requires precomputed masks. *Li et al.* [12] propose the use of instance embeddings to find the moving object based on motion saliency and objectness. However, the dense embeddings are obtained from an instance segmentation that is performed by a network pretrained on static images. In contrast, we consider as self-supervised methods only the ones which do not need any type of ground truth during training and inference.

Early attempts of self-supervised or unsupervised learning of motion segmentation start from ConvNet [21]. The network learns high-level features in single frames as follows. A set of single frames with optical flow maps is passed to the network. It then generates the segmentation masks by associating optical flow vectors with similar direction and rate. Using these masks as pseudo-ground truth, the network tries to reproduce the masks by looking only at the static frames .

Yang et al. [28] propose a network that performs foreground/background motion segmentation by using a precomputed optic flow as input. Slot attention is then used to group together pixels with visual homogeneity, but the result are very sensitive to the accuracy of the optical flow estimation.

While in [24] and [28] the models try to segment the moving object by using the optical flow of the object itself, *Yang et al.* [30] propose to find the moving object by using a Generator (G) and an Inpainter (I) network. The generator applies a mask to the input optic flow, with the aim of hiding the optic flow of the foreground. Next, the generator passes the masked optic flow to the inpainter, which has the task of reconstructing the optic flow that has been hidden by the generator. Accurate motion segmentation is achieved when the optic flow reconstruction works poorly. It obtains good result in public datasets, however the model fails to capture complete objects or differentiate regions in the background. This is because it operates with one single scale temporal information and it also introduces the bias from the camera motion. To solve this issue, *Yang et al.* [29] extend this work with MASNet, where they expand the network pipeline with more generator and inpainter modules to deal with multiple input optic flow maps.

Finally, *Shrestha et al.* [24] propose LayerSegNet, a network that combines a layered representation of the scene and affine optical flow estimation. The network takes two images as input and aims to separately estimate the motion models (foreground and background) and the masks. Subsequently, they apply the masks to the first image and warp the masks according to the associated motion models. They then combine the two masks together to generate the second image, and they use the difference between the generated second image and actual second image as learning parameter.

B. Event-based Motion Segmentation

Event-based motion segmentation is very recent and attempts to make use of the outstanding properties of event cameras.

Stoffregen et al. [25] use Expectation Maximization and propose the idea of segmenting moving objects by simultaneously grouping the events into different clusters and estimating the motion model associated to each cluster. The events are then warped according to the associated motion model to produce an Image of Warped Events (IWE). An iteration process is then used to find the right motion models and clusters such that the objective function is maximized. The model is proven not to be sensitive to the number of clusters specified in the iteration process and works well in real-world dataset,but it is not able to estimate the number of moving objects in the scene.

Zhou et al. [36] propose to solve this issue by introducing two spatial regularizers, which minimize the number of clusters. The underlying concept remains the same as in [25], however the input events are first initialised in a space-time event graph cut and the clusters are smoothly sharped via the addition of energy terms in the objective function.

Always based on event clustering, *Chethan et al.* [18] splits the scene into multiple motions and merges them, allowing also for feature tracking.

Despite these methods produce promising results, they are not appropriate to be deployed in real-world applications as the iteration processes takes considerable amount of time and computational effort, which inhibit the potential applications on mobile platforms such as drones. Therefore, a learning-based approach needs to be developed.

Sanket et al. [23] proposes EVDodgeNet, an learning-based pipeline that solves simultaneously motion segmentation, optical flow and 3-D motion, but relies on ground truth masks. Instead, *Mitrokhin et al.* [17] uses Graph Convolutional Networks to learn motion segmentation from a 3-D representation of events. *Chethan et al.* [19] also propose SpikeMS, the first Spiking Neural Network (SNN) for event-based motion segmentation.

Despite these efforts, the aforementioned methods continue to depend on ground truth, leaving a gap in the availability of self-supervised approaches.

Inspired by LayerSegNet [24], we propose an end-to-end CNN architecture that learns motion segmentation by jointly estimating affine optical flow and segmentation masks, using contrast maximization as learning loss.

III. Method

In this section, the overall methodology is explained.

The approach to the segmentation task is inspired by [24]. However, given the unique nature of event data, a distinct

approach to the input is needed. To this end, we use the input event representation and from [39]. We then jointly estimate the segmentation masks and the affine optical flow maps. Next, we apply the segmentation masks to the associated optical flow, and we combine these maps to obtain a single optical flow map. We then use this map to warp the events forward and backward in time, and we apply the self-supervised loss as in [7]. The more the image of warped events is deblurred with respect to the input events, the more accurate are the segmentation masks.

A. Input Event Representation

Selecting the right representation of events for a given task is still a challenging problem. Event-based methods can process single events $e_k \doteq (x_k, t_k, p_k)$, however events alone carry very little information and are subject to noise. It is often preferred to process a group of events $\mathcal{E} \doteq \{e_k\}_{k=1}^{N_e}$ that yields a sufficient signal-to-noise ratio, which also carries more information for the given task. The most common methods either discretize the group of events in different frames of event counts ([11],[10],[20],[26],[39]) or the per-pixel average/most recent event timestamps ([13],[31],[37]).

Since we develop a non-recurrent CNN pipeline, temporal information need to be encoded in the input, thus we adopt the event representation from [39]. Given N input events and B bins, we first scale the event timestamps in the range $[0, B-1]$ and generate the event volume:

$$t_i^* = (B-1)\,(t_i - t_0)\,/\,(t_N - t_1) \tag{1}$$

$$V(x,y,t) = \sum_i p_i k_b\,(x - x_i)\,k_b\,(y - y_i)\,k_b\,(t - t_i^*) \tag{2}$$

$$k_b(a) = \max(0, 1 - |a|) \tag{3}$$

In order to perform 2D convolutions, the time domain is treated as a channel in a traditional 2D image.

B. Motion Model

The authors in [24] propose the segmentation of the dominant moving object in the scene using a two-layer representation. The foreground layer captures the dominant object, while the background layer represents the rest of the scene.

In our case, assuming constant illumination, events are generated by the relative motion between the camera and the scene. Since events are triggered by brightness changes, these are produced by moving edges. Hence, if the camera is moving and the scene is static, the events are generated by the motion of the camera. When objects are also moving within the scene, the generation of events occurs due to both the edges of moving objects and the edges of static objects caused by the egomotion of the camera.

Let us assume a scene where an object and background are moving distinctively. We also assume that the motion is affine, hence the scene undergoes may undergo translation, rotation, shearing and scaling. The two motions can then be described by two motion models A_1 and A_2:

$$W_i(x,y) = \mathbf{A}_i \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} a_i^1 & a_i^2 & a_i^3 \\ a_i^4 & a_i^5 & a_i^6 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}, i \in \{1,2\} \tag{4}$$

where W_i represent the dense flow maps corresponding to affine motion matrix A_i.

Corresponding to the two affine motions, we separately generate two alpha masks, which represent the moving objects. Ideally, the segmentation masks should be a one-hot vector for each pixel, which assigns the pixel to the corresponding object. However, such operation would be non differentiable, preventing the backpropagation of the gradients during training. We overcome this issue by assigning two arbitrary numbers for each pixel, where each number corresponds to one layer. We then apply softmax to ensure that the values are bounded in [0,1] and the values sum up to 1. For each pixel, we then retain the maximum value in the two layers, and set the other one to zero. This operation, called maxout operation, makes the classification differentiable. It also makes sure that one pixel can not belong to two objects at the same time.

Next, we compute the combined optical flow by element-wise multiplication of the affine flow maps with the corresponding alpha maps:

$$W_{\text{comb}} = \alpha_1 \odot W_1 + \alpha_2 \odot W_2 \tag{5}$$

The maxout operation ensures that each pixel will have flow components from only one motion model, scaled with the respective value of the alpha map to which the pixel is associated. During training, we expect that the network will adjust itself to the scaling effect of the flow computation.

C. Self Supervision by Contrast Maximization

Self-supervised learning is achieved by applying contrast maximization [3]. As described in [7], events which are generated by the same moving edge encode exact optical flow. Since these events are misaglined in the input partition (motion blur), the events can be propagated to a reference time t_{ref} using per-pixel optical flow $\boldsymbol{u}(\boldsymbol{x}) = (u(\boldsymbol{x}), v(\boldsymbol{x}))^T$ to realign them and show the initial edge that has generated the events:

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + (t_{\text{ref}} - t_i)\,\boldsymbol{u}\,(\boldsymbol{x}_i) \tag{6}$$

The metric used to measure the deblurring quality is the the per-pixel and per-polarity average timestamp of the image of warped events ([16],[39]). The lower the loss, the better the deblurring, which consequently means that the estimated optical flow and alpha maps are more accurate.

We initially generate an image of the average timestamp at each pixel for each polarity using bilinear interpolation, as in [7]:

$$\begin{aligned} T_{p'}\,(\boldsymbol{x}; \boldsymbol{u} \mid t_{\text{ref}}) &= \frac{\sum_j \kappa\big(x - x_j'\big)\kappa\big(y - y_j'\big)t_j}{\sum_j \kappa\big(x - x_j'\big)\kappa\big(y - y_j'\big) + \epsilon} \\ j = \{i \mid p_i = p'\}, &\quad p' \in \{+,-\}, \quad \epsilon \approx 0 \end{aligned} \tag{7}$$

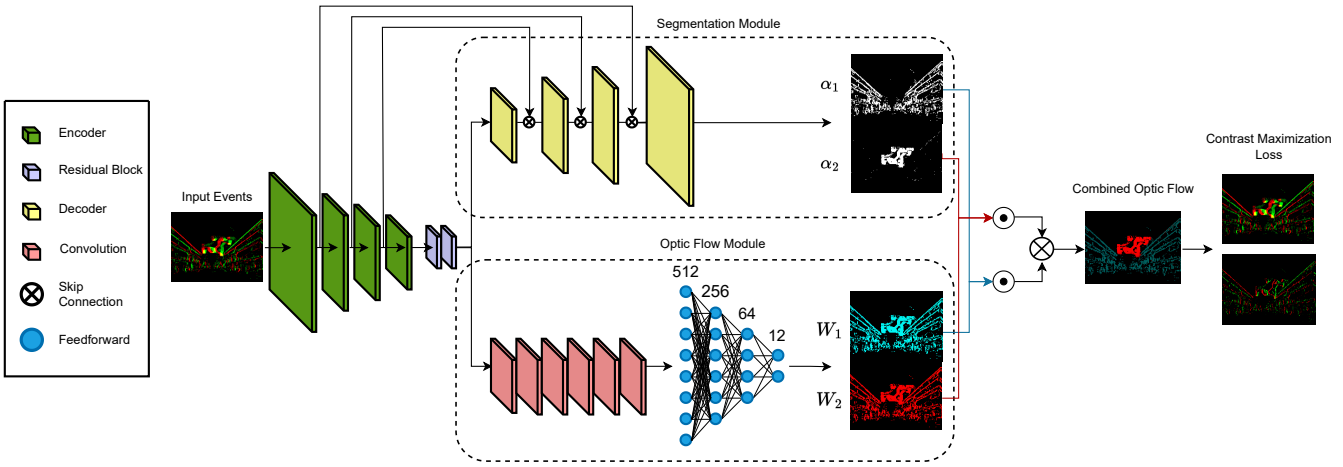


Fig. 2. EV-LayerSegNet architecture. Events are passed downsampled by 4 encoding layers before being passed to 2 residual blocks. The output of the residual blocks is then passed to the segmentation and optical flow module. The segmentation module upsamples with 4 decoder layers connected with the encoders by skip connections. The optical flow module is composed of 6 convolutions and a feedforward network of 4 layers.

The loss is then the sum of the squared temporal images. This is also scaled by the sum of pixels with at least one event, in order to prevent the network from keeping events with large timestamps out of the image space, such that they would not contribute to the loss function.

$$\mathcal{L}_{\text{contrast}}\ (t_{\text{ref}}) = \frac{\sum_{\boldsymbol{x}} T_{\pm}\left(\boldsymbol{x};\boldsymbol{u} \mid t_{\text{ref}}\right)^2}{\sum_{\boldsymbol{x}}\left[n\left(\boldsymbol{x}'\right)>0\right]+\epsilon} \tag{8}$$

The warping process is performed both forward (t_{ref}^{fw}) and backward (t_{ref}^{bw}) to prevent temporal scaling issues during back-propagation. The total loss is then the sum of the backward and forward warping loss, and λ is a scalar balancing the Charbonnier smoothness prior $\mathcal{L}_{\text{smooth}}$ [2].

$$\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{contrast}}\left(t_{\text{ref}}^{\text{fw}}\right) + \mathcal{L}_{\text{contrast}}\left(t_{\text{ref}}^{\text{bw}}\right) \tag{9}$$

$$\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{contrast}} + \lambda \mathcal{L}_{\text{smooth}} \tag{10}$$

Notice that the self-supervised loss becomes a strong supervisory signal only when there is enough blur in the input event partition. It is of paramount importance to check that the input partition contains enough events for linear blur.

IV. Network Implementation

Our network EV-LayerSegNet is similar to encoder-decoder architectures and it is inspired by [24] and [39].

As in [39], events are downsampled by 4 encoder layers and passed to 2 residual blocks. We then stack the output of the residual blocks and pass it to the segmentation module and optical flow module.

A. Optical Flow Module

The optical flow module contains 6 convolutional layers, each followed by leaky ReLU activation. We then flatten the output of the last convolution layer and pass it to a feedforward network consisting of 4 layers (512, 256, 64 and 12 output units), followed by tanh activation except the last layer. We then use the output of the feedforward network and split it to two sets of 6 affine motion parameters, and we compute the two flow maps W_1 and W_2.

B. Segmentation Module

In the segmentation part, the output of the residual blocks is bilinearly upsampled by 4 decoding layers. Each decoding layer is connected to the respective encoding layer by skip connection. We apply a leaky double-rectified ReLU activation (*leaky DoReLU*, [24]) which promises an increase in segmentation performance:

$$y(x) = \begin{cases} 1 + \frac{x-1}{\gamma} & , x > 1 \\ x & , \text{ if } 0 \leq x \leq 1 \\ \frac{x}{\gamma} & , x < 0 \end{cases} \tag{11}$$

In this module, each decoding layer is followed by leaky DoReLU activation with γ = 10. At the last layer, softmax is applied instead to ensure that the channel values are bounded [0,1] and sum up to 1.

V. Experiments

A. Training Details

We implement our pipeline in Pytorch. We use Adam optimizer [8] and a learning rate of $1 \cdot 10^{-5}$. At training, we used a batch size of 8 and train for 5000 epochs. The input consists of N = 200,000 events, which is a sufficient number of events to obtain blur in the scene. The Charbonnier loss is balanced with weight λ = 0.001 .

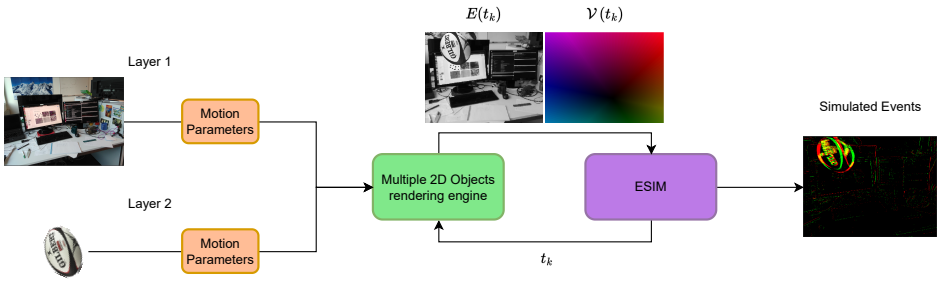


Fig. 3. Working principle of ESIM, as explained in [22]. The images are passed to the rendering engine along with the associated motion model. The rendering engine generates an irradiance map E and motion field map $\mathcal{V}$ at time t_k. ESIM then computes the brightness change, which is used to generate the simulated events and choose the next rendering time t_{k+1}.

B. Dataset

We train EV-LayerSegNet on our dataset named AffineObjects. The dataset is generated using the Event Camera Simulator (ESIM) from [22]. Specifically, we make use of the Multiple 2D Objects rendering engine to generate the simulated events. The working principle of the simulator using the rendering engine is shown in Figure 3. We explicitly model the motion of the objects in the scene using the affine transformation parameters in Table I. While the affine motion parameters in Equation 4 are coefficients for the linear transformation, the parameters in Table I represents the start and end points for each motion in the image plane. The layers then undergo affine motion from $t_0 = 0$ to $t = t_{\text{max}}$ from start to end. In this work, we generate recordings that span 5 seconds.

TABLE I
AFFINE MOTION PARAMETERS IN ESIM

Parameter	Symbol
Rotation	θ_0, θ_1
Translation	x_0, x_1, y_0, y_1
Scaling	$S_{x,0}$, $S_{x,1}$, $S_{y,0}$, $S_{y,1}$

The training dataset was generated using the images in Figure 4. We use similar settings as in a DVS camera, with resolution 640x480 pixels and threshold C = 0.5.

Fig. 4. Images used to generate AffineObjects training set. Top row: images used as background. Bottom row: images used as foreground.

After training, we test the network in the test set. In order to test the segmentation module, we generate the test set using the images in Figure 5. In each layer, we specify different motion parameters compared to the training set such that also the optical flow module is tested.

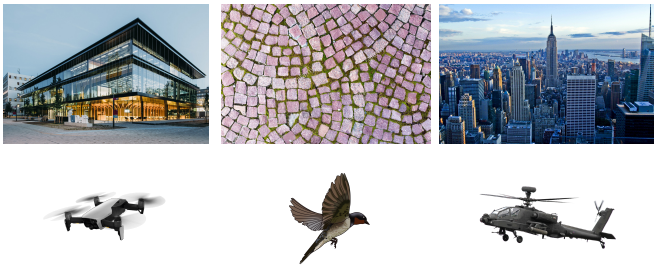


Fig. 5. Images used to generate AffineObjects test set. Top row: images used as background. Bottom row: images used as foreground.

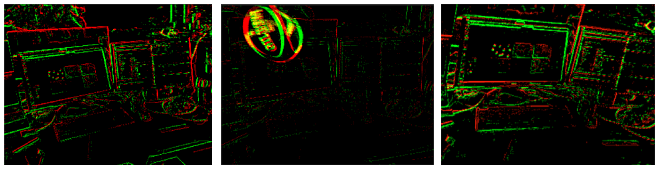
During the generation of the recordings, we notice that it is important to choose images that have similar levels of brightness. For example, when generating the recording shown in Figure 3, we notice that the rugby image plays a shading effect on the background, resulting in fewer events generated by the background when the rugby ball enters the field of view of the camera. This differs from the reality, where the brightness level would be the same. As the difference between the brightness levels increases, the shading effect becomes larger.

VI. RESULTS

A. Optical Flow

Before training EV-LayerSegNet on motion segmentation, it is of paramount importance that the optical flow module is able to estimate the flow map correctly and that the network is able to deblur the input events. Thus we first omit the segmentation module and train the pipeline on optical flow.

To generate the training set for learning optical flow, we use the backgrounds shown in Figure 4. For each scene, we make 2 recordings for each distinct affine motion: translation (right and left), rotation (clockwise and counterclockwise), and



(a) (b) (c)

Fig. 6. Shading effect. In (a), only the background moves. In (b), the rugby image enters the camera's field of view, shading the background and thus generating fewer events. When the ball exits the scene, the background regains its own level of brightness (c).

scaling (zoom in and zoom out). Single-direction shearing is not considered since rotation already includes coupled shearing in both x-y directions. Finally, we make 2 additional recordings including a combination of all motion types. This resulted in 24 recordings.

In testing, we use the building image in Figure 5 and test the network in unseen translations, rotations, scalings and complex motions. The test results are shown in Figure 8. Successful deblurring demonstrates that the optical flow pipeline inspired from [24] is able to learn event-based optical flow using contrast maximization.

B. Segmentation - Train Results on AffineObjects

With the optical flow module working, it is now possible to train the entire EV-LayerSegNet pipeline.

To ensure a progressive increase in difficulty, we initially start with training the network to learn segmentation in the simplest possible scenes. In this case, the background and foreground undergo horizontal translations in opposite directions. For this, we take the rugby ball and translate it horizontally left and right, while the background translates in the other direction. This results into 6 recordings.

We next focus on scenes where the background and foreground move in the same direction. Clearly, if the foreground and background move at the same speed, distinguishing between them becomes a challenging task even for human observers. However, different speeds provide crucial information for discerning the two in such cases. We then generate other 6 recordings where the rugby ball translates much faster than the background, but in the same direction.

Finally, we challenge the network to learn segmentation in scenes with intricate dynamics. We extend the dataset with additional 8 recordings, where the rugby ball and the chair take in turn complex motions with the office and the carpet as background. The results on the training set are shown in Figure 7.

The segmentation of the rugby ball from the background is highly accurate when it undergoes opposite translation.

Furthermore, the network demonstrates its ability to segment the ball when it moves in the same direction as the background at a higher speed. However, in these cases, the borders of the ball are mislabeled as background. As a result, it becomes evident that the borders are the only regions in the scene that remain blurry in the image of warped events.

When it comes to more complex motions, the network successfully captures the majority of the foreground, although it does miss some parts. It is important to note that in these instances, the rugby ball moves at a faster pace compared to the previous examples, resulting in increased blurriness. Similarly to the previous scenario, we believe that the network faces challenges in accurately performing segmentation as the blurriness of the scene intensifies.

C. Segmentation - Test Results on AffineObjects

To assess the generalization of EV-LayerSegNet, we extend AffineObjects to include test scenes with completely unseen scenarios and motions, and test the network trained on the training set.

The test results on AffineObjects dataset are shown in Figure 9.

As in the training set, the network works remarkably well when the background and foreground undergo opposite translations. It also shows that is able to segment the foreground moving in the same direction as background. In contrast to the training case where the network struggled to deblur certain parts of the rugby ball, in this particular instance, the bird exhibits less blurriness compared to the rugby ball. Consequently, the segmentation is more accurate. Finally, the network also showcases good segmentation results with more complex motions.

Based on these findings, we can conclude that the network performance does not solely rely on memorization of the scenes in the dataset but rather showcases the ability to generalize effectively.

In almost all scenarios, including both the train and test results, we observe that certain pixels are misclassified as foreground. This occurrence may be attributed to the fact that the events contained in these pixels, although generated by the background motion, contribute significantly to the loss function when they are warped with the foreground.

As last note, it is important to realize that the segmentation is performed on the input events, which leads to "blurry" segmentation masks.

D. Motion Segmentation with only Background Motion

As the network shows good performance in segmenting both the foreground and the background, we investigate its behavior when confronted with scenes where only the background is in motion.

Theoretically, we would expect the network to cluster all events into a single mask, corresponding to a unified motion model. The second mask would remain empty as there is no foreground present in the scene.

To test this hypothesis, we evaluate the network on a scene featuring only horizontal motion, as illustrated in Figure 8. The resulting segmentation is presented in Figure 10.

We observe that the network struggles to accurately segment the scene in the first row. Upon closer examination of the

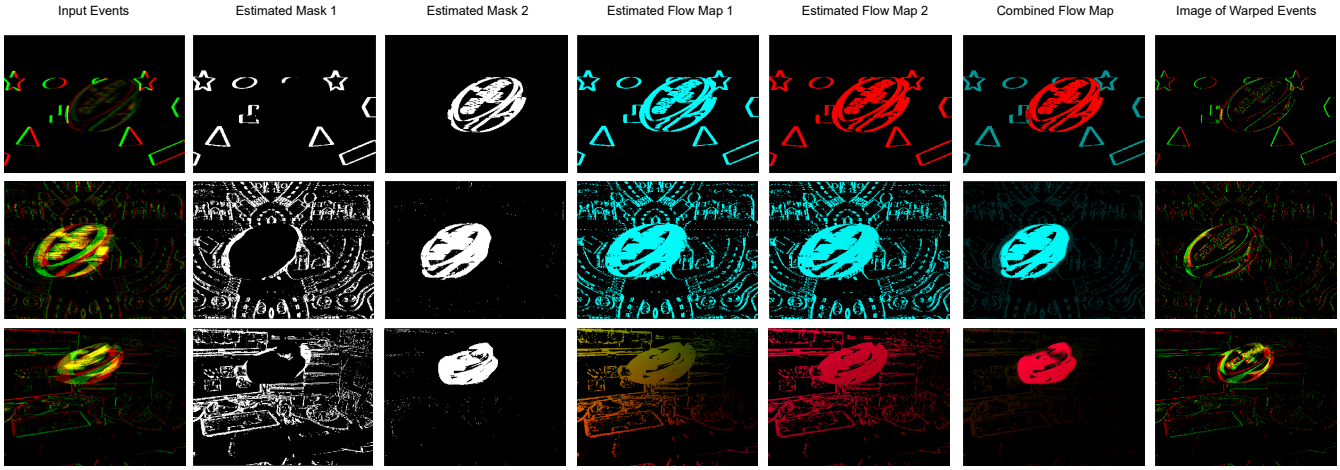


Fig. 7. Train results on AffineObjects dataset. In the first row, the rugby ball and the poster translate in opposite directions, leading to a simple segmentation. In the second row, the network is then challenged to segment the ball when it translates in the same direction of the background but with different speed. In the last row, both background and the ball have complex motion, resulting in a more difficult segmentation. The color scheme can be found in Appendix.

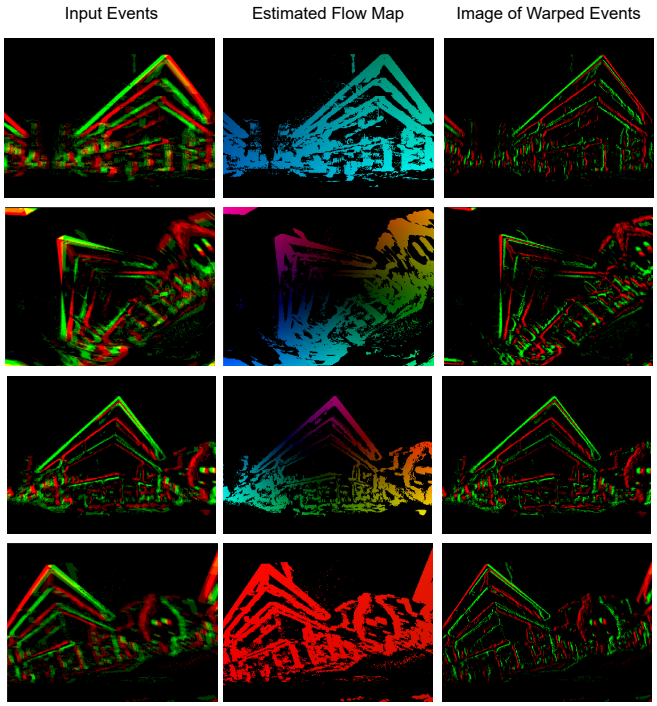


Fig. 8. Motion deblur. From top to bottom: translation, rotation, scaling, and combinations of all three. From left to right: input events, estimated optic flow map, and deblurred events. The color encoding scheme can be found in Appendix.

input events and the two masks, we notice that the region segmented in the second mask coincides with the portion of the input events that exhibits greater blurriness compared to the rest of the scene. It appears that the network creates a distinct class for this region and associates it with a motion model specifically designed to deblur it more effectively than the rest of the scene.

To verify this new hypothesis, we reduce the number of input events from 200,000 to 100,000 events, resulting in reduced blurriness in that particular region and enough in the rest of the scene. The corresponding results, shown in the second row of Figure 10, demonstrate that because the region now possesses a similar level of blurriness to the rest of the scene, it is appropriately grouped into a single class, with only a few sparse pixels assigned to the second class.

E. Motion Segmentation with more Objects than Classes

Having explored the network behavior when only the background is in motion, we now investigate its performance in scenarios where two objects are in motion. Since the network considers background and foreground as two distinct clusters, the additional object in the scene will be incorporated in one of these clusters.

Our hypothesis is rooted in the learning process being driven by the objective of improving deblurring quality. We expect that the network aims at maximizing the alignment between events, taking into account the clusters that contribute most significantly to the loss function. Given that the background typically generates events throughout the entire scene, it often constitutes the primary contributor to the loss function. As for the remaining two objects, we believe the network considers object that generates the highest number of events, whether due to its faster motion or larger size. Consequently, the network assigns a motion model capable of deblurring the dominant object after the background to the second cluster. The remaining object is allocated to either the background or the other object cluster, depending on its contribution to the learning loss.

We validate our hypothesis in Figure 11, where a rugby ball and chair are set in motion against the background.

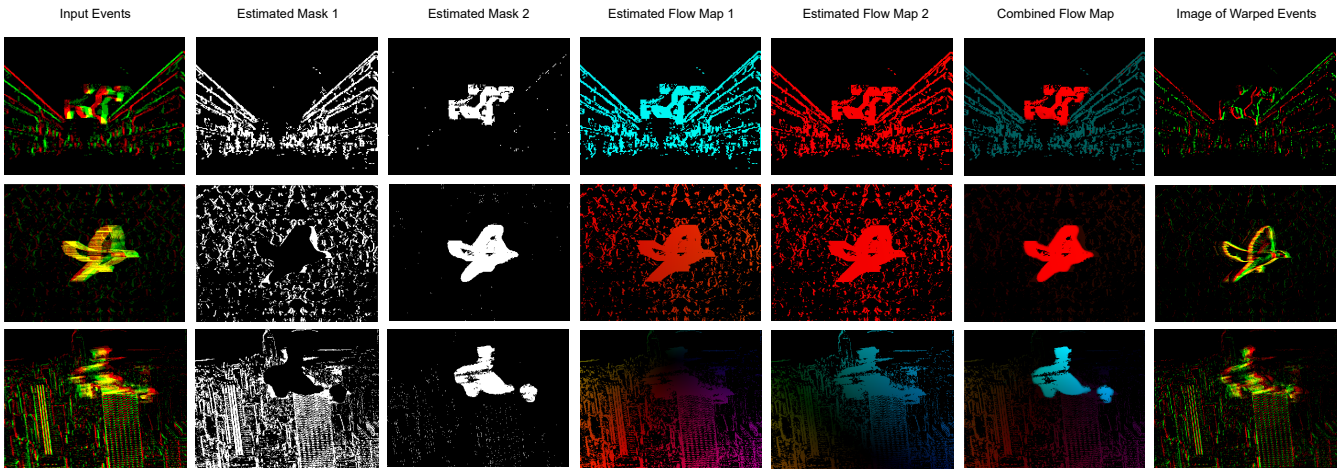


Fig. 9. Test results on AffineObjects dataset. In the first row, the drone translates in opposite direction with respect to the building. In the second row, the network is then challenged to segment a bird when it translates in the same direction of the bricks but with different speed. In the third row, the complex motion of the helicopter and city background lead to difficult segmentation. The color scheme can be found in Appendix.

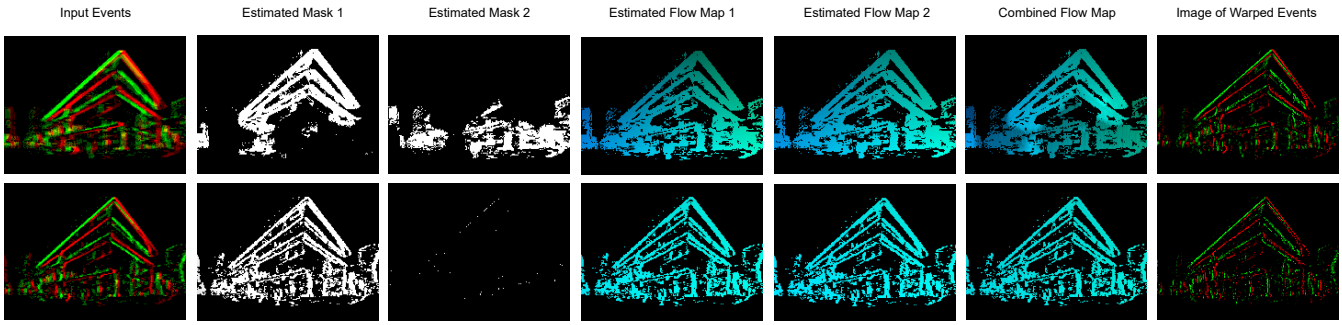


Fig. 10. EV-LayerSegNet test results on scene with background-only motion. In the first row, the network struggles in accurately segmenting the scene, with the second mask erroneously capturing the region of higher blurriness. By reducing the number of input events from 200,000 to 100,000 in the second row, the blurriness in that specific region is reduced, resulting in improved segmentation with the majority of pixels assigned to a single class and sparse pixels to the second class.. The color scheme can be found in Appendix.

The test results confirm the validity of our hypothesis. The background and the rugby ball, being the primary event generators, exhibit improved deblurring. However, since the chair contributes to a lesser extent in terms of generated events, it is assigned to the rugby ball cluster. Consequently, the chair experiences increased blurriness as the associated motion model is tailored to deblur the rugby ball rather than the chair.

F. Increasing Number of Classes in EV-LayerSegNet

The proposed method treats the event streamline as foreground/background segmentation. Nevertheless, the architecture can be adapted to include more masks. For example, it is possible to adjust the final layer of the feedforward network in the optical flow module to include 6 more outputs for a third mask. In the segmentation part, a third channel can be added in the last decoder layer to have a similar number of masks.

We then extend the network to generate 3 motion models and 3 masks, hence enabling the model to segment the background and 2 independently moving objects.

We generate a second training set in AffineObjects. As background, we use the office and the carpet images from Figure 4. For foreground, we use the chair and rugby ball from Figure 4 and the drone from Figure 5.

The training results are depicted in Figure 12. Surprisingly, the network accurately defines the background, but it fails to distinguish the other two moving objects, in this case the rugby ball and the drone. However, by examining the estimated flow maps, it becomes visible that the motions of these, which are opposite translations, can be effectively represented by one single motion model. Thus the network considers the two opposite translations as one single counterclockwise rotation.

In conclusion, the observed behavior of the network indicates a tendency to group as many events as possible into a single class. To achieve accurate motion segmentation, it is necessary to design strategies that appropriately reward pixels in close proximity while penalizing cases where distinct

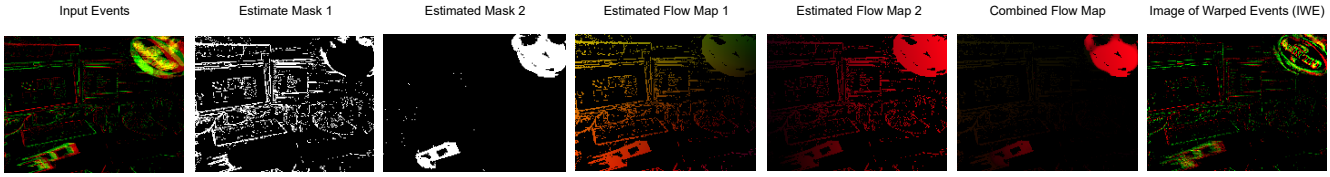


Fig. 11. EV-LayerSegNet test results on AffineObjects scene with 2 moving objects on background. The network assigns the dominant object, the rugby ball, and the background to separate clusters, resulting in improved deblurring. However, the chair, contributing fewer events, is assigned to the rugby ball cluster, leading to increased blurriness due to the tailored motion model. The color scheme can be found in Appendix.

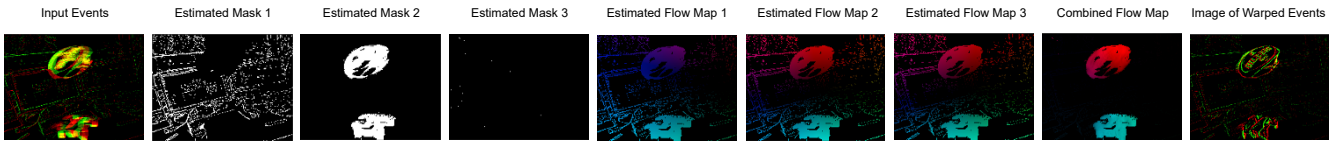


Fig. 12. Train Results on AffineObjects dataset, with EV-LayerSegNet adapted to segment up to 3 masks. The network correctly identifies the background but tries to cluster the events generated by the drone and the ball in one single class. In this case, the drone and the ball translate in opposite directions, and the network tries to retrieve a motion model that captures both translations in one single rotation. The color scheme can be found in Appendix.

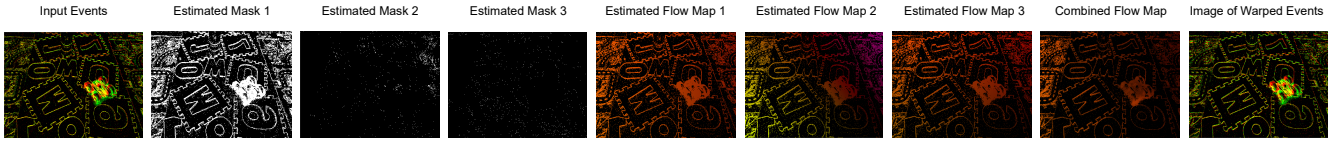


Fig. 13. Training results on EVIMO-2 dataset. While the network demonstrates efficient deblurring of the background, it faces challenges in accurately segmenting the independently moving objects. The color scheme can be found in Appendix.

groups of pixels are erroneously assigned to the same class.

G. Evaluation on EVIMO-2 Dataset

After examining the behavior of our network with different numbers of classes and moving objects in the scene, we aim to evaluate its performance on a public dataset. We choose to test our method on the EVIMO-2 dataset, the most recent event-based dataset available [1]. The dataset consists of events recorded using two Prophesee Gen3 cameras (left and right) and one Samsung Gen3 camera (central). For our specific analysis, we focus on the motion segmentation subset of the dataset.

Initially, we train our network using the daylight recordings of EVIMO-2 since the night conditions pose significant challenges due to high levels of noise. Additionally, since all scenes in the dataset feature background motion and at most two independently moving objects, we adopt a three-class configuration in our EV-LayerSegNet, while being aware of the tendency of the network to cluster together as many events as possible.

Subsequently, we train our network on the EVIMO-2 dataset and present the training results in Figure 13. In most scenes of the training set, the network effectively deblurs the background. However, it encounters difficulties in accurately segmenting the independently moving objects, in this case a toy car (IMO). We believe this challenge arises due to the high level of noise, which poses a significant obstacle for our current pipeline. While reducing the noise by passing less events in the voxel grid is a possible solution, it introduces the risk of insufficient blurring of the background, which would then hinder accurate background labeling. We therefore suggest further research on noise filtering techniques and their integration into our pipeline.

VII. Ablation Study

A. Event Voxel vs Event Count

In our proposed method the temporal information is encoded in the input event representation. As explained in [7], temporal information need to be encoded in the input event representation. This is vital for accurate optical flow estimation using non-recurrent neural networks such as EVFlowNet [31] and ours.

To test this, we change the input event representation to be per-pixel and per-polarity event count as in [7]. By implementing this adjustment, we ensure that only spatial information are passed to the network. We then train and test our network on AffineObjects.

The training and test results are shown in Figure 14.

While the network demonstrates to learn accurate flow maps and segmentation masks during training, we observe a

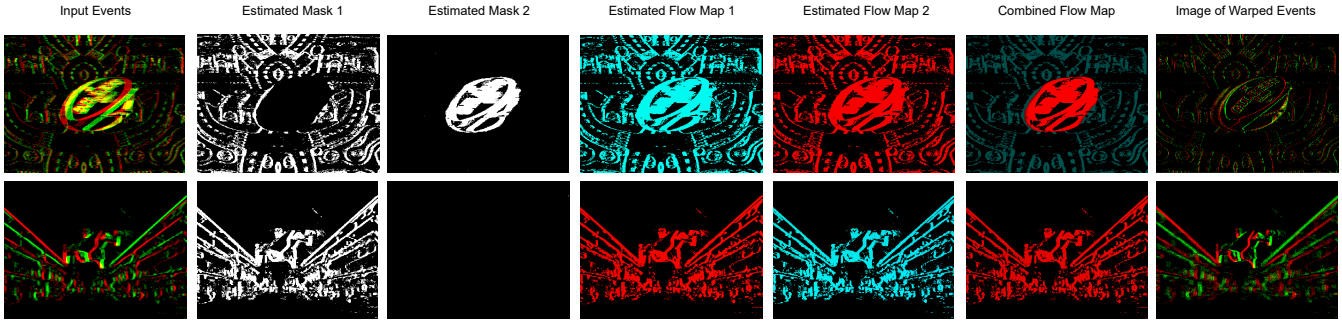


Fig. 14. Comparison of training and test results using per-pixel and per-polarity event count. The first row shows the training results, while the second row displays the test results. Despite the network's ability to learn accurate flow maps and segmentation masks during training, it struggles to generalize effectively during testing. This highlights the limitations of using event count representation alone. The color scheme can be found in Appendix.

significant performance drop during testing. This discrepancy can be attributed to a clear case of overfitting during training. As a result, our findings emphasize the vital role of encoding temporal information within the input representation for mitigating overfitting and improving the generalization capabilities of our proposed method.

B. Leaky ReLU vs Leaky DoReLU

In [24] the authors discover that using their leaky DoReLU activation function instead of leaky ReLU improves segmentation. The key distinction between these two activation functions lies in the restriction of the slope for input values above 1 in leaky DoReLU. The authors argue that the reason for the segmentation improvement with leaky DoReLU is the enhanced stability of gradients, considering their alpha maps are limited to the range [0,1].

Inspired by their segmentation approach, we chose to implement leaky DoReLU instead of leaky ReLU. However, it is crucial to investigate the significance of this modified activation function in our specific case.

For this reason, we create an alternative EV-LayerSegNet architecture where leaky ReLU activations are used instead of leaky DoReLU. We train it on the AffineObjects dataset and subsequently test on a scene featuring horizontal motion, as depicted in Figure 9.

The results in Figure 17 show that the network encounters difficulties in accurately segmenting the entire drone from the background, managing only to segment the outer borders of the drone correctly. This observation strongly suggests that the leaky DoReLU activation function plays a vital role in our network's ability to achieve successful segmentation.

C. Decreasing the slope in Leaky DoReLU

By establishing the significance of the leaky DoReLU activation function in our segmentation module, we now turn our attention to exploring the impact of modifying the slope coefficient γ. In our investigation, we seek to understand how adjusting γ affects the segmentation performance of our network.

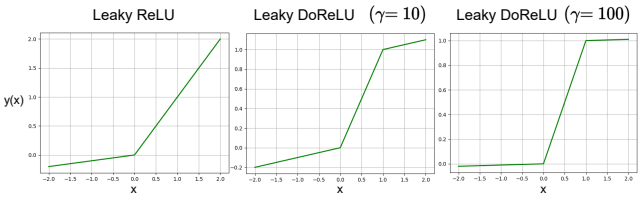


Fig. 15. Visualization of leaky ReLU (left) and leaky DoReLU with slope coefficient $\gamma = 10$ (center) and $\gamma = 10(right)$. The x-axis is the input x and the y-axis is the output $y(x)$.

First, let us examine the characteristics of the activation functions involved, as depicted in Figure 15. Notably, we observe that leaky DoReLU exhibits a smaller slope compared to leaky ReLU. This reduced slope plays a vital role in stabilizing the flow of gradients beyond the range of [0,1]. Thus we embark on exploring the consequences of further decreasing the slope in our specific context, effectively increasing the slope coefficient.

Empirical observations reveal that augmenting γ leads to a near-flattening of the activation function slope beyond the [0,1] region, as depicted in Figure 15. A diminished slope implies that the gradients during training will be correspondingly smaller. While employing γ = 10 already provides improved gradient stability compared to leaky ReLU in the segmentation task, a nearly flat activation function may result in less substantial updates to the network parameters. Consequently, this can lead to slower convergence or the network becoming trapped in suboptimal solutions, a phenomenon commonly referred to as the vanishing gradient problem.

To validate our hypothesis, we proceed to increase the value of γ from 10 to 100 and train the network on the AffineObjects dataset. The corresponding test results are presented in Figure 16. Although certain regions of the drone are correctly segmented as foreground, the network does not mislabel any significant portion of the background. This outcome suggests that while striving to deblur the scene, the network may have

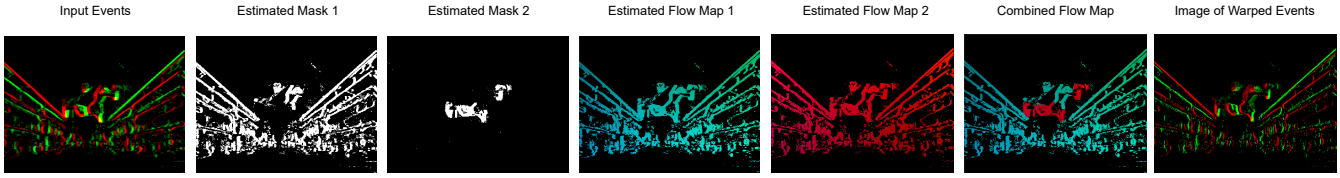


Fig. 16. test results obtained by increasing γ to 100 in the leaky DoReLU activation function. The segmentation output shows partial correctness in identifying the foreground regions of the drone, while accurately preserving the background. This outcome suggests that the network, in its pursuit of deblurring the scene, may have reached a suboptimal solution for the segmentation task. The color scheme can be found in Appendix.

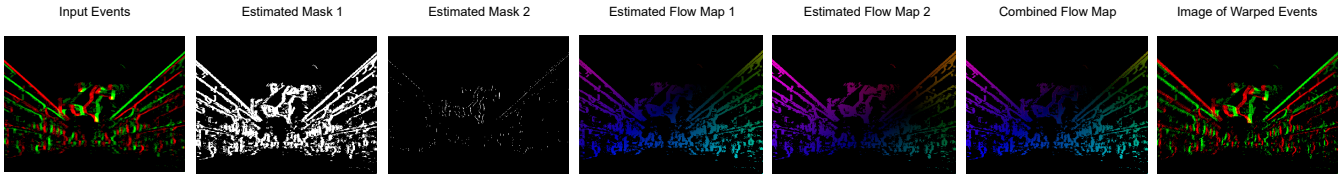


Fig. 17. Segmentation test results of the leaky ReLU variant of EV-LayerSegNet. Poor segmentation indicates the importance of the leaky DoReLU activation function in achieving accurate motion segmentation. The color scheme can be found in Appendix.

arrived at a suboptimal solution for the segmentation problem at hand.

By investigating the effects of both the leaky DoReLU activation function and its slope, we gain valuable insights into the interplay between these factors and their influence on the segmentation performance of our network. These findings contribute to a deeper understanding of the optimization challenges inherent in the design of our segmentation module.

VIII. Conclusion

We present the first end-to-end CNN network that performs self-supervised motion segmentation using event cameras. We take inspiration from the state-of-the-art methods in unsupervised motion segmentation, and use the latest development in event-based optical flow estimation using artificial neural networks. We show that it is possible to learn motion models and segmentation masks separately from a blurry scene, and combine them together to deblur the input events. To train and test our method, we generate a new dataset with background and objects moving affinely. We show that the network is able to correctly segment the background and the independently moving objects in the event stream. Next, we show that our method is not strictly dependent on the number of moving objects in the scene, but can be extended to include more objects. On the other hand, our method shows to be sensitive to motion complexity, significant blurriness and noise, for which we recommend the integration of noise filtering techniques in the pipeline. Furthermore, we discover the tendency of the network to fit as many events as possible in one single class, by estimating the motion model that fits most of the events. For this reason, we recommend the inclusion of spatial regularizers for future work. As last recommendation for future work, we suggest to develop the method using Spiking Neural Networks, therefore allowing for fast and low-power consuming computation on board of weight-constrained vehicles such as drones. We hope that our work draws attention to the potential that self-supervised learning can unlock for event-based cameras, allowing for learning-based methods to be trained and used without expensive ground-truth annotations.

References

[1] L. Burner et al. “EVIMO2: An Event Camera Dataset for Motion Segmentation, Optical Flow, Structure from Motion, and Visual Inertial Odometry in Indoor Scenes with Monocular or Stereo Algorithms”. In: May 2022. URL: https://arxiv.org/abs/2205.03467.

[2] P. Charbonnier et al. “Two deterministic half-quadratic regularization algorithms for computed imaging”. In: *Proceedings of 1st International Conference on Image Processing*. Vol. 2. 1994, 168–172 vol.2. DOI: 10.1109/ICIP.1994.413553.

[3] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. “Focus Is All You Need: Loss Functions for Event-Based Vision”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12272–12281. DOI: 10.1109/CVPR.2019.01256.

[4] Guillermo Gallego et al. “Event-based Vision: A Survey”. In: (Apr. 2019).

[5] Liyue Ge et al. “Optical Flow Estimation from Layered Nearest Neighbor Flow Fields”. In: Oct. 2018, pp. 1–6. DOI: 10.1109/CISP-BMEI.2018.8633120.

[6] Mathias Gehrig et al. “DSEC: A Stereo Event Camera Dataset for Driving Scenarios”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4947–4954. DOI: 10.1109/LRA.2021.3068942.

[7] Jesse Hagenaars, Federico Paredes-Valles, and Guido de Croon. "Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 7167–7179. URL: https://proceedings.neurips.cc/paper/2021/file/39d4b545fb02556829aab1db805021c3-Paper.pdf.

[8] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014).

[9] M. Kumar, Philip Torr, and A. Zisserman. "Learning Layered Motion Segmentations of Video". In: *International Journal of Computer Vision* 76 (Mar. 2008), pp. 301–319. DOI: 10.1007/s11263-007-0064-x.

[10] Chankyu Lee, Adarsh Kumar Kosta, and Kaushik Roy. "Fusion-FlowNet: Energy-Efficient Optical Flow Estimation using Sensor Fusion and Deep Fused Spiking-Analog Network Architectures". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 6504–6510. DOI: 10.1109/ICRA46639.2022.9811821.

[11] Chankyu Lee et al. "Spike-FlowNet: Event-based Optical Flow Estimation with Energy-Efficient Hybrid Neural Networks". In: *European Conference on Computer Vision*. 2020.

[12] Siyang Li et al. "Instance Embedding Transfer to Unsupervised Video Object Segmentation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6526–6535. DOI: 10.1109/CVPR.2018.00683.

[13] Zhuoyan Li, Jiawei Shen, and Ruitao Liu. "A Lightweight Network to Learn Optical Flow from Event Data". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 1–7. DOI: 10.1109/ICPR48806.2021.9413238.

[14] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[15] Xiankai Lu et al. "See More, Know More: Unsupervised Video Object Segmentation With Co-Attention Siamese Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3618–3627. DOI: 10.1109/CVPR.2019.00374.

[16] Anton Mitrokhin et al. "Event-Based Moving Object Detection and Tracking". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8593805.

[17] Anton Mitrokhin et al. "Learning Visual Motion Segmentation Using Event Surfaces". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 14402–14411. DOI: 10.1109/CVPR42600.2020.01442.

[18] Chethan M. Parameshwara et al. "0-MMS: Zero-Shot Multi-Motion Segmentation With A Monocular Event Camera". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9594–9600. DOI: 10.1109/ICRA48506.2021.9561755.

[19] Chethan M. Parameshwara et al. "SpikeMS: Deep Spiking Neural Network for Motion Segmentation". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 3414–3420. DOI: 10.1109/IROS51168.2021.9636506.

[20] Federico Paredes-Vallés and Guido C. H. E. de Croon. "Back to Event Basics: Self-Supervised Learning of Image Reconstruction for Event Cameras via Photometric Constancy". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3445–3454. DOI: 10.1109/CVPR46437.2021.00345.

[21] Deepak Pathak et al. "Learning Features by Watching Objects Move". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6024–6033. DOI: 10.1109/CVPR.2017.638.

[22] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conf. on Robotics Learning (CoRL)* (Oct. 2018).

[23] Nitin J. Sanket et al. "EVDodgeNet: Deep Dynamic Obstacle Dodging with Event Cameras". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10651–10657. DOI: 10.1109/ICRA40945.2020.9196877.

[24] Sahir Shrestha et al. "Learning To Segment Dominant Object Motion From Watching Videos". In: *2021 Digital Image Computing: Techniques and Applications (DICTA)*. 2021, pp. 01–08. DOI: 10.1109/DICTA52665.2021.9647227.

[25] Timo Stoffregen et al. "Event-Based Motion Segmentation by Motion Compensation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 7243–7252. DOI: 10.1109/ICCV.2019.00734.

[26] Timo Stoffregen et al. "Reducing the Sim-to-Real Gap for Event Cameras". In: *European Conference on Computer Vision*. 2020.

[27] J.Y.A. Wang and E.H. Adelson. "Representing moving images with layers". In: *IEEE Transactions on Image Processing* 3.5 (1994), pp. 625–638. DOI: 10.1109/83.334981.

[28] Charig Yang et al. "Self-supervised Video Object Segmentation by Motion Grouping". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 7157–7168. DOI: 10.1109/ICCV48922.2021.00709.

[29] Fan Yang et al. "Multi-motion and Appearance Self-Supervised Moving Object Detection". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2101–2110. DOI: 10.1109/WACV51458.2022.00216.

[30] Yanchao Yang et al. "Unsupervised Moving Object Detection via Contextual Information Separation". In: *2019 IEEE/CVF Conference on Computer Vision and*

Pattern Recognition (CVPR). 2019, pp. 879–888. DOI: 10.1109/CVPR.2019.00097.

[31] Chengxi Ye et al. "Unsupervised Learning of Dense Optical Flow, Depth and Egomotion with Event-Based Sensors". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5831–5838. DOI: 10.1109/IROS45743.2020.9341224.

[32] Vickie Ye et al. "Deformable Sprites for Unsupervised Video Decomposition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 2657–2666.

[33] Fisher Yu et al. "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling". In: *ArXiv* abs/1805.04687 (2018).

[34] Luca Zappella et al. "Enhanced Local Subspace Affinity for feature-based motion segmentation". In: *Pattern Recognition* 44 (Feb. 2011), pp. 454–470. DOI: 10.1016/j.patcog.2010.08.015.

[35] Yun Zhang, Bin Luo, and Liangpei Zhang. "Permutation Preference Based Alternate Sampling and Clustering for Motion Segmentation". In: *IEEE Signal Processing Letters* 25.3 (2018), pp. 432–436. DOI: 10.1109/LSP.2017.2777997.

[36] Yi Zhou et al. "Event-Based Motion Segmentation With Spatio-Temporal Graph Cuts". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021), pp. 1–13. DOI: 10.1109/TNNLS.2021.3124580.

[37] Alex Zihao Zhu et al. "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras". In: *ArXiv* abs/1802.06898 (2018).

[38] Alex Zihao Zhu et al. "The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2032–2039. DOI: 10.1109/LRA.2018.2800793.

[39] Alex Zihao Zhu et al. "Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 989–997. DOI: 10.1109/CVPR.2019.00108.

APPENDIX

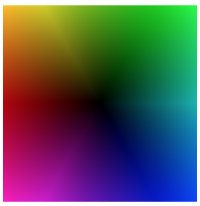


Fig. 18. Optical flow color coding scheme. Direction is encoded in color hue, and speed in color brightness.

Part II

Literature Study

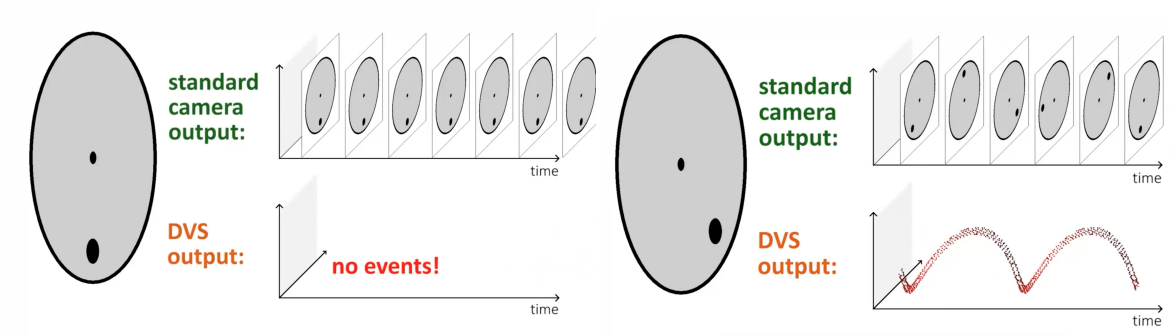
2

Event Cameras

Event cameras represent a new form of acquisition of visual information. As the aim of this research project is to develop a computer vision method for event cameras, it is of paramount importance to first understand the working principle of these types of cameras, along with the advantages and challenges that their data output pose. In section 2.1, the working principle of the event cameras is explained, whereas in section 2.2 the advantages and challenges are shown. Finally, a brief explanation of the representation and processing of the event data is given in section 2.3.

2.1. Principle of Operation of Event Cameras

Event cameras are bio-inspired sensors that pose a paradigm shift in the way visual information is acquired. In contrast to conventional cameras, which acquire full images at a fixed rate, event cameras such as the Dynamic Vision Sensor (DVS) respond to changes of brightness in the scene asynchronously and independently for every pixel [20]. To understand the difference, a comparison between traditional camera and event camera is given in Figure 2.1. In Figure 2.1a the black dot is not moving, hence the event camera is not recording any data, whereas the standard camera generates an output at fixed rate. In Figure 2.1b, the black dot starts moving and the event camera records the motion at a higher time resolution.



(a) No motion **(b)** Counterclockwise motion

Figure 2.1: Black dot moving in a circular motion on a grey disk. The frame camera register all pixels values, whereas the event camera only records the motion of the ball, hence the event camera will only record the black dot when it is moving

Events cameras work as follows. An event camera has independent pixels that respond to changes in brightness, hence to changes in their log photocurrent I, as shown in Equation 2.1 [20]:

$$L \doteq \log(I) \tag{2.1}$$

Assuming a noise-free scenario, an event e_k is triggered at pixel $\mathbf{x}_k$ (with x_k and y_k coordinates) at time t_k as soon as the brightness change since the last event at the pixel (Equation 2.2) reaches a temporal contrast threshold $\pm C$ as in Equation 2.3, where $C > 0$, Δt_k is the time elapsed the since the last event at the same pixel, and the sign of the brightness change is described by the polarity $p_k \in \{-1, +1\}$ [20].

$$\Delta L(\mathbf{x}_k, t_k) \doteq L(\mathbf{x}_k, t_k) - L(\mathbf{x}_k, t_k - \Delta t_k) \tag{2.2}$$

$$\Delta L(\mathbf{x}_k, t_k) = p_k C \tag{2.3}$$

Contrast Sensitivity

Essentially positive events ("ON", brightness increase) and negative events ("OFF", brightness decrease) are triggered when the upper threshold C^+ or lower threshold C^- are exceeded. This contrast is generally determined by the pixel bias currents [29][30], and typical DVS cameras set the thresholds between 10% and 50% [20][40]. Setting the lower threshold too low results in a storm of noise events [7].

Moving Edges

Assuming constant illumination, for a small Δt the intensity increment ΔL can be approximated as in Equation 2.4:

$$\Delta L \approx -\nabla L \cdot \mathbf{v} \Delta t \tag{2.4}$$

This means that the intensity increment is caused by a brightness gradient $\nabla L(\mathbf{x}_k, t_k) = (\partial_x L, \partial_y L)^T$ with velocity $\mathbf{v}(\mathbf{x}_k, t_k)$ on the image plane, over a displacement $\Delta \mathbf{x} = \mathbf{v} \Delta t$. If the motion is parallel to the edge, there is no event generated since $v \dot{\nabla} L = 0$. However, if the motion is perpendicular to the edge ($\mathbf{v} \mid\mid \nabla L$), events are generated at the highest rate, since minimal time is required to achieve brightness change.

2.2. Advantages and Challenges of Event Cameras

Event cameras offer numerous potential advantages over traditional frame-based cameras. First, event cameras have high temporal resolution. In analog circuitry, monitoring of brightness changes is fast and the read-out of events is digital with 1 MHz of frequency [7]. This means that events are detected and stamped with microsecond resolution, hence this type of camera is able to capture fast motions without suffering from motion blur, which is recurrent in frame-based cameras. Second, event cameras have low latency. Since every pixel works independently, there is no need to wait that every pixel detects a brightness change to generate an event, hence the latency is minimal. The latency ranges from 10 μs on the lab bench to sub-millisecond in the real world [7]. Third, the power consumption is low because only brightness changes are detected and transmitted, thus removing redundant data. Among the range of different event cameras, the power consumption is usually less than 100 mW [7]. Last, the cameras have a very High Dynamic Range (HDR), hence they can adapt to very dark as well as very bright stimuli. This is because the pixels operate on a logarithmic scale and independently.

Nevertheless, event cameras represent a complete different approach in the acquisition of visual information, and this poses the challenge of designing novel methods to unlock their potential. First, the main challenge arises from the different space-time output of the cameras. While frames are synchronous and dense, events are asynchronous and spatially sparse. This different space-time distribution makes the current frame-based vision algorithms not suitable for event data. Second, the photo-metric sensing is different. While standard cameras provide grayscale information, event contain only contain brightness changes, which are represented by binary values for increase and decrease. Additionally, changes in brightness do not depend only on the scene but also on the relative motion between scene and camera. Last, the noise from photoreceptors to transistor circuits is substantial and quite complex to model and filter out. Therefore new methods are needed to rethink about how to model the noise in event data so as to make them more meaningful for post-processing.

2.3. Event Representations and Processing

The shift towards event data poses questions regarding the methods to acquire meaningful information from events for a given task. As mentioned before, event cameras acquire information in an asynchronous

and sparse way, with low latency and high temporal resolution. This means that the temporal aspect of events plays a central role when processing the events. Depending on how many events are processed at the same time, algorithms can be divided into methods that operate on a *event-by-event basis* or process *groups of events*, which introduce some latency.

Individual Events

Methods that operate using single events $e_k \doteq (x_k, t_k, p_k)$ are usually Spiking Neural Networks (SNN) and filters. Deterministic filters (e.g. space-time convolution filters and activity filters) have been used for noise reduction, feature extraction [2], image reconstruction [34][38] and brightness filtering [37]. Also probabilistic filters such as Kalman and particle filters have been used for pose tracking in SLAM (Simultaneous Localization And Mapping) systems [8][13][14][42]. The advantage of processing one event a time is that the state of the system (estimated unknowns) can change when another event arrives, therefore the latency is minimal. However, the main disadvantage is that events alone carry very little information for state estimation. Furthermore, they can be very sensitive to noise. Filters and SNNs use additional information to leverage from this drawback. This information is usually built up from past events or given by external knowledge (i.e. sensors data fused with events).

Event Packet

Since each single event carries little information and is subject to noise, events are grouped to yield a sufficient signal-to-noise ratio event packets $\mathcal{E} \doteq \{e_k\}_{k=1}^{N_e}$ are usually a collection of event that form a spatio-temporal neighbourhood and are processed together to produce a meaningful output. The number of events processed simultaneously N_e is critical for the assumptions of the algorithm used for the given task (i.e. constant motion speed). Many processing methods just perform pre-processing to enable the usage of frame-based computer vision tools with events by generating event frames. Event frames have been used for example in traditional stereo methods [15][16] and optical flow computation [21]. With this respect, event frames offer many advantages. First, event frames have an intuitive interpretation for the human eye (i.e. edge map). Second, they inform not only about the presence of events but also about their absence. Last, the data structure is compatible with traditional computer vision. However, using event frames is not ideal since they disregard event sparsity and are very sensitive to the number of events used to generate the frame. Other methods, such as in [27], use a 3D representation of the events, where the third dimension becomes a geometric one i.e. $(x_k, y_k, t_k) \in R^3$

3

Optical Flow Estimation with Event Cameras

As already mentioned in chapter 2, events represent a paradigm shift in the way visual information acquired. In particular, the fast-changing nature of events represent a substantial advancement in computer vision tasks such as optic flow. As the objective of this research project lies in elaborating a self-supervised framework for motion segmentation, in a similar way self-supervised optical flow estimation is analyzed. For this task, establishing a correspondence between events is essential. In section 3.1, such establishment is sought with motion compensation, whereas in section 3.2 the current state-of-the-art frameworks on self-supervised optical flow are analyzed in depth.

3.1. Motion Compensation (Contrast Maximization)

Assuming constant illumination in a scene, event cameras detect moving edges. In absence of other appearance information, for most computer vision tasks such as depth and optical flow estimation the problem of establishing correspondence between the events becomes of primary importance. In this regard, *Gallego et al.* [6] elaborated a framework aimed at maximising the alignment between the events. This is known as *motion compensation* or *contrast maximization*. An example is shown in Figure 3.1. The events shown in Figure 3.1.a do not have any data correlation between them. By maximising an objective function (the contrast) the events are best aligned to reveal the edges that have caused them first Figure 3.1.b.

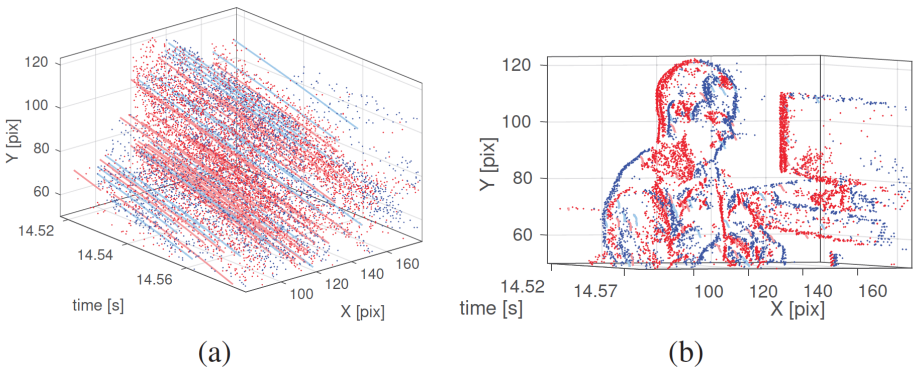


Figure 3.1: a) Events caused by moving edges (blue is brightness increase, red is brightness decrease). (b) events visualized according to the point trajectories, revealing the edges that have caused them [6]

More specifically, given a set of N_e events, continuous in time, the purpose is to find the point trajectories $\mathbf{x}(t)$ that best fit the event data. The events are generated on an image plane where the geometric model

of the problem addressed (i.e. optical flow) is known. Then the objective becomes estimating the parameters θ of such model from the events, assuming that there are fewer unknown parameters than events, they are shared among the events and that they are observable. Then the process can be broken down as follows:

1. Warp the events and create an image of warped events H, according to the point trajectories defined by the candidate parameters
2. Compute the objective function f based on H
3. Optimize f with respect to θ

As an example, the problem of optical flow estimation is considered. Recall that each event $e_k \doteq (x_k, y_k, t_k, p_k)$ consist of space-time coordinates and brightness change defined by the polarity $p_k \in \{-1, +1\}$. In optical flow, the ideal is that the measurement is taken over a very small time interval, such that the trajectories followed by the points are locally straight. Then the geometric model can be approximated as the translation shown in Equation 3.1, where $\mathbf{x} \doteq (x, y)^T$ is the point and $\mathbf{v}$ its velocity (optical flow). Corresponding events should lie in such trajectories.

$$\mathbf{x}(t) = \mathbf{x}(0) + \mathbf{v}t \tag{3.1}$$

Next, given a candidate optical flow $\mathbf{v}$ (which from now represents the motion parameters θ), the objective is to sum the polarities lying on the same point trajectory. For this, events need to be translated to a reference time t_{ref}. This is known as warping and it is shown in Equation 3.2, where $\mathbf{W}$ stands for warping function.

$$\mathbf{x}'_k \doteq \mathbf{W}(\mathbf{x}_k, t_k; \theta) = \mathbf{x}_k - (t_k - t_{\text{ref}})\,\theta \tag{3.2}$$

Then, an Image of Warped Events (IWE or H) is built as in Equation 3.4, where b_k can either be 1 or have the value of the sum of polarities at the specific pixel. As mentioned in the supplementary material of [6], the main difference lies in the extension of the basin around the optimal value of the objective: using the polarities for b_k results in a narrower basin that is mainly due to the cancellation of polarities when thin edges are causing the events and overlap. Therefore, $b_k = 1$ is more often considered.

$$H(\mathbf{x}; \theta) \doteq \sum_{k=1}^{N_e} b_k \delta\left(\mathbf{x} - \mathbf{x}'_k\right) \tag{3.3}$$

Finally, the objective function is computed. For optical flow estimation, the variance of H, known as *contrast*, is shown in Equation 3.4, where N_p is the number of pixels in H and $\mu_H \doteq \frac{1}{N_p}\sum_{i,j} h_{ij}$ is the mean of H:

$$f(\theta) = \sigma^2(H(\mathbf{x}; \theta)) \doteq \frac{1}{N_p}\sum_{i,j}\left(h_{ij} - \mu_H\right)^2 \tag{3.4}$$

By maximizing the variance, the accumulation of events in the image plane is favoured. Since the number of events is constant, the accumulation of events in certain regions of the image plane will directly result in dispersion of events in other regions. Hence higher variance will result in a higher contrast (hence *contrast maximization*) as shown clearly in Figure 3.2.

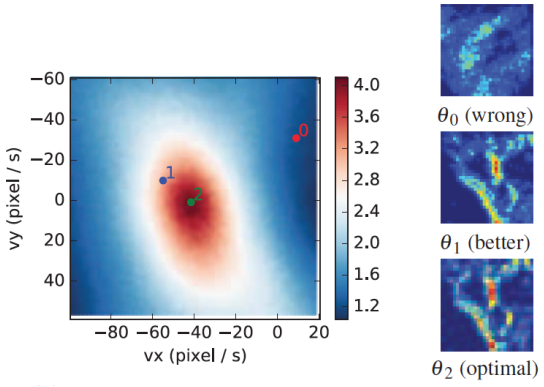


Figure 3.2: (a) variance of H as a heat map. (b) Images of Warped Events (IWE)

Furthermore, *Gallego et al.* [5] extend the work in [6] by analysing 20 more loss functions from traditional image processing techniques, with the conclusion that the variance, the gradient and the Laplacian magnitudes are among the best loss functions in terms of accuracy and computational time

3.2. Self-supervised learning of optical flow via contrast maximization

Early attempts on using contrast maximization for learning optical flow in a self supervised way have been made by *Zhu et al.* [53]. The pipeline is shown in Figure 3.3. Events are treated as 2D image, hence they are grouped in patches of different timestamps and passed to a convolutional encoder-decoder network, which outputs the optical flow in the style of a traditional 2D image. Then the self-supervision is done via motion compensation, hence the motion is reversed to a reference time and a temporal loss inspired from *Mitrokhin et al.* [26] is applied along with a local smooth regularization loss. The main limits of this method is that it disregards the neuronal dynamics of the event data, and rather treats them as images for traditional applications with Artificial Neural Networks (ANN). A natural fit to events would be the use of SNN, where time is not encoded in the input but directly fed in the spiking network.

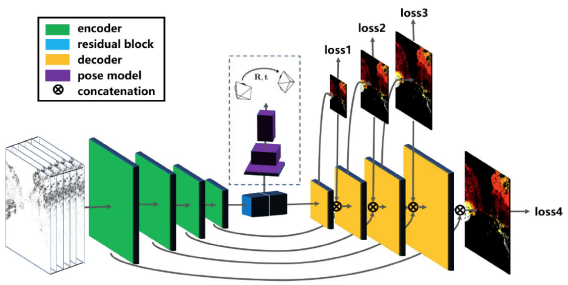


Figure 3.3: Self-supervised optical flow estimation using event volumes and Artificial Neural Networks (ANN) [53]

In this regard, *Paredes et al.* [11] propose the first SNN for self-supervised optical flow estimation. The pipeline is shown Figure 3.4. The method can be seen as a conversion from ANN to SNN of the method in [53]. In addition to this, the input is treated differently. While in [53] the event stream is split into different time slots and treated the grouped events as a frame, in [11] the events are split into groups of equal count and the temporal information is not encoded in the input but directly extracted by the network.

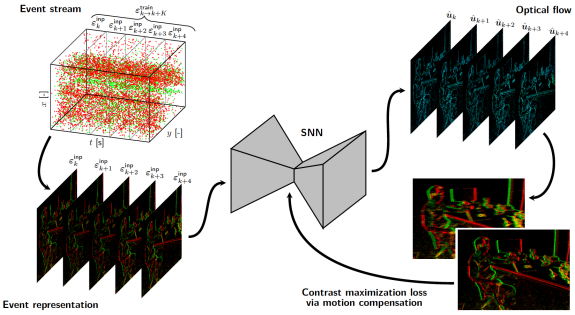


Figure 3.4: Pipeline for self-supervised optical flow estimation using events and SNN [11]

The self-learning is also based on contrast maximization as in [53], hence the deblurring quality is used as measure, namely the per-pixel and per-polarity average timestamp of the IWE. As this metric becomes smaller, the deblurring quality increases hence the optical flow estimation is more accurate. However, in [11] the temporal loss is reformulated as follows. Similarly to [53], an image of the average timestamp per pixel for each polarity p' is generated as in Equation 3.5

$$\begin{aligned} T_{p'}\left(\boldsymbol{x};\boldsymbol{u} \mid t_{\text{ref}}\right) &= \frac{\sum_j \kappa\left(x-x'_j\right)\kappa\left(y-y'_j\right)t_j}{\sum_j \kappa\left(x-x'_j\right)\kappa\left(y-y'_j\right)+\epsilon} \\ j = \{i \mid p_i = p'\}, \quad p' \in \{+,-\}, \quad \epsilon &\approx 0 \end{aligned} \tag{3.5}$$

Successively, the temporal loss in [53] is improved by scaling the sum of squared temporal images with the number of pixels with at least one warped event. This is done in order to make the loss convex, which is shown in Equation 3.6, where $n(\boldsymbol{x}')$ is the per-pixel event count of IWE.

$$\mathcal{L}_{\text{contrast}}\left(t_{\text{ref}}\right) = \frac{\sum_{\boldsymbol{x}} T_+\left(\boldsymbol{x};\boldsymbol{u} \mid t_{\text{ref}}\right)^2 + T_-\left(\boldsymbol{x};\boldsymbol{u} \mid t_{\text{ref}}\right)^2}{\sum_{\boldsymbol{x}}\left[n\left(\boldsymbol{x}'\right) > 0\right] + \epsilon} \tag{3.6}$$

To prevent temporal scaling issues during backpropagation, the warping process is done both forward ($t^{\text{fw}}_{\text{ref}}$) and backward $t^{\text{bw}}_{\text{ref}}$. The total final loss is shown in Equation 3.7, where λ is a scalar balancing the two losses and $\mathcal{L}_{\text{smooth}}$ is a Charbonnier smoothness prior [4].

$$\begin{aligned} \mathcal{L}_{\text{contrast}} &= \mathcal{L}_{\text{contrast}}\left(t^{\text{fw}}_{\text{ref}}\right) + \mathcal{L}_{\text{contrast}}\left(t^{\text{bw}}_{\text{ref}}\right) \\ \mathcal{L}_{\text{flow}} &= \mathcal{L}_{\text{contrast}} + \lambda\mathcal{L}_{\text{smooth}} \end{aligned} \tag{3.7}$$

Spiking Neuron Model

The spiking neural network used in [11] is based on the leaky-integrate-and-fire (LIF) neuron, where the membrane potential U, the synaptic current I and time t_k are modelled as in Equation 3.8 and Equation 3.9. Here j and r refer to presynaptic neurons, i is for postsynaptic, α is the membrane decay or leak (which can be fixed or learned), $S \in \{0,1\}$ a neuron spike, and W^{ff} and W^{rec} feedforward and recurrent weight connections:

$$U_i^k = \left(1 - S_i^{k-1}\right)\alpha U_i^{k-1} + (1-\alpha)I_i^k \tag{3.8}$$

$$I_i^k = \sum_j W_{ij}^{\text{ff}} S_j^k + \sum_r W_{ir}^{\text{rec}} S_r^{k-1} \tag{3.9}$$

A neuron fires a spike S whenever the membrane potential exceeds a threshold θ. The threshold can be fixed, learned or adaptive, making the LIF model ALIF (adaptive). In Equation 3.10 and Equation 3.11, the threshold is made adaptive by using a second state variable T. It acts as a low-pass filter over the

output spikes, hence adapting the threshold based on the neuron activity. $\beta_{\{0,1\}}$ are learnable constants, whereas η is the learnable threshold decay.

$$\theta_i^k = \beta_0 + \beta_1 T_i^k \tag{3.10}$$

$$T_i^k = \eta T_i^{k-1} + (1-\eta) S_i^{k-1} \tag{3.11}$$

To regularize the neuron firing, presynaptic activity (PLIF model) is used by subtracting a presynaptic trace P to the input current, as shown in Equation 3.12 and Equation 3.13, where $\rho_{\{0,1\}}$ are learnable addition and decay constants, and R_i is the set of receptive fields of neuron i over all channels:

$$I_i^k = \sum_j W_{ij}^{\text{ff}} S_j^k + \sum_r W_{ir}^{\text{rec}} S_r^{k-1} - \rho_0 P_i^k \tag{3.12}$$

$$P_i^k = \rho_1 P_i^{k-1} + \frac{1-\rho_1}{|R_i|} \sum_{j \in R_i} S_j^{k-1} \tag{3.13}$$

This is useful since the adaptation delay is minimized, a crucial feature for the fast-changing nature of event data. Due to this, [11] propose a crossover between adaptive and presynaptic models (XLIF), where the threshold is adapted based on presynaptic activity, as shown in Equation 3.14

$$\theta_i^k = \beta_0 + \beta_1 P_i^k \tag{3.14}$$

4

Motion Segmentation

As mentioned in chapter 1, the aim of this research project is to develop a self-supervised motion segmentation method for event cameras. The literature on motion segmentation (event-based and frame-based) is very large, and a good understanding of the state-of-the-art methods in both kinds of visual information acquisition is needed in order to combine the necessary knowledge and elaborate a successful method which will be the first of its kind in event-based vision. In this regard, first a general review on motion segmentation, aimed at identifying the most popular categories, is done in section 4.1. Finally, the literature on self-supervised methods for traditional frame-based cameras is looked in depth, and the relevant findings are shown in section 4.2.

4.1. Review on Motion Segmentation

Motion segmentation is the computer vision task that attempts to retrieve moving objects [52]. Despite the methods being very different between each other, they present similar attributes. Here the main ones are presented:

- Methods can be *Feature-based* or *Dense-Based*. In feature-based, the moving objects are represents by few points or salient characteristics like corners or edges, whereas dense-based methods compute pixel-wise motion [17].
- *Occlusions*: some methods are able to handle objects that are partially or fully hidden for a certain period of time
- *Multiple objects*: methods can be able to distinguish multiple moving objects in a scene or simply retrieve the main moving part (foreground) which can be one object or more (without any distinction among them)
- *Temporary stopping*: the ability of handling the temporary stop of moving objects
- *Prior knowledge*: some methods need prior knowledge of the scene to train on or to be explicitly included inside the model
- *Training*: some methods require training as they may use neural networks

First, motion segmentation can be feature-based or dense-based. In feature-based methods, the moving objects is represented by a limited number of features like edges or salient points, whereas in dense-based methods the algorithm computes pixel-wise motion to find the moving object [17]. Second, motion segmentation presents the challenge of dealing with occlusions, which is the partial (or total) disappearance of a moving object for a certain amount of time (for example, a car moving behind a tree). Some methods fail when occlusions happen, and others are able to resolve occlusion to a certain extent. Third, a large number of methods are able to recognize one moving object in the scene, however other algorithms are able to deal with multiple objects moving at the same time. Fourth, some methods are able to deal with temporary stopping of moving objects. Fifth, algorithms need to be able to deal with missing data in the scene. Last, some methods require some inputs from the camera model in use for motion segmentation.

The literature on motion segmentation is extensive and many approaches are specific to certain applications. Despite the extensiveness, there are distinct working principles which allow the methods to be classified into different categories. Here the main ones are reported [49][25]. In subsection 4.1.1, image difference is explained, , whereas in subsection 4.1.4 and subsection 4.1.3 the statistical methods and optic flow are shown respectively. Then, techniques based on layers and deep learning are elaborated in subsection 4.1.2 and subsection 4.1.5. To end, a brief evaluation of all the methods is given in subsection 4.1.6.

4.1.1. Image Difference

Image difference is the simplest method, which consists in detecting moving objects by simply finding the intensity difference between pixels across video frames [49]. An example of image difference is shown in Figure 4.1. Methods based on Image Difference are able to detect multiple independently moving objects and can also handle occlusions, however these algorithms are very sensitive to noise, brightness changes, temporary stopping, and can not discern the motion of the camera from the scene [50]. The method in [18] uses the pixel variance and covariance to learn a statistical model of the background, and uses average frame difference to achieve the segmentation.

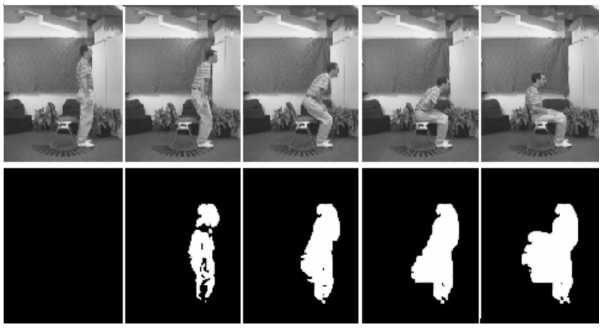


Figure 4.1: Sequence of video frames along with their image difference results [1]

4.1.2. Layers

Layers based techniques aim at finding motion by dividing the frames into different layers based on the number of uniform motions [9][17]. An example of this class of methods is shown in Figure 4.2. Each layer contains information associated to the depth, the motion parameters and the parameters regarding motion visibility (i.e. occlusions) This is the most natural solution to solve occlusions in the scenes, however the complexity of the algorithms gets very high with increasing number of parameters, leading to very long execution times. Moreover, some kind of priori knowledge is needed to provide an initial estimate [49].

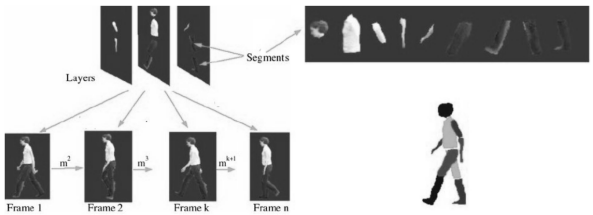


Figure 4.2: An example of layer-based method [17]

4.1.3. Optical Flow

Optical Flow (OF) methods rely on optic flow, which is the distribution of apparent velocities describing the apparent motion pattern of brightness in the pixels [49]. It is one of the earliest methods used to analyze motion in video sequences, but it fails to segment motion whenever there are occlusions or temporary stoppings [43]. Since it is also sensitive to noise and changes in lightning conditions, optic flow needs to be processed to reduce these effects and typical methods include the use of Convolutional Neural Networks (CNN) [25].

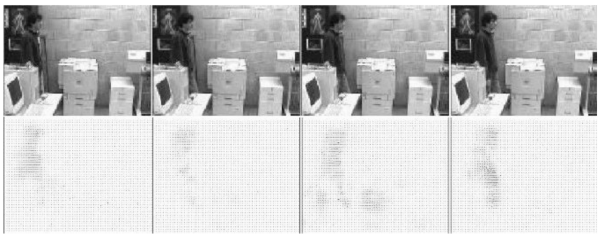


Figure 4.3: Optic flow field: darker zones are the regions where velocity vectors are greater than zero [49]

4.1.4. Statistical Theory

Statistical theory is among the most common principles used in motion segmentation [49]. In essence, motion segmentation can be seen as a classification problem, where the pixels either belong to the background (the region where little to no motion is present) or the foreground, where the dominant moving object of the scene is located. The most common frameworks are Maximum A Posteriori probability (MAP), Particle Filter (PF) and Expectation Maximisation (EM).

Maximum A Posteriori Probability

MAP is based on the Bayes Rule as in Equation 4.1:

$$P\left(w_j \mid x\right) = \frac{p\left(x \mid w_j\right) P\left(w_j\right)}{\sum_{i=1}^{c} p\left(x \mid w_i\right) P\left(w_i\right)} \tag{4.1}$$

where $P(w_j|x)$ is the 'a posteriori probability' x is the pixel to be classified, $w_1 ... w_c$ are the c classes (often only foreground and background), $p(x|w_j)$ the conditional density, $P(w_j)$ the 'a priori probability' and $\sum_{i=1}^{c} p\left(x \mid w_i\right) P\left(w_i\right)$ the density function. Essentially, MAP tries to classify pixel x to one of the classes w in order to maximise the a posteriori probability. therefore giving for every pixel the probability of belonging to each class (background or foreground) and the objective is to maximise the posteriori probability [7][12].

Particle Filter

In Particle Filter, the objective is to track the evolution of a variable over time by constructing a sample-based representation of the probability density function. Often this variable is a geometric active contour of the moving object is tracked with the help of a probability density function [25]. The method is an iterative algorithm, where each iteration is composed of prediction and update. A series of actions are taken, and each action modifies the state of the variable according to some model (prediction). Then each copy of the variable state (particles) is memorized along with a weight that represents the quality of that specific particle. The estimation of the future state of the variable is done by summing all previous copies. After the new state is observed, the weights are re-evaluated (update) and the particles with small weights are eliminated [35].

Expectation Maximization

Expectation Maximization is a powerful iterative algorithm that aims at estimating the model parameters that best fist the observed data. This method is particularly convenient in scenarios where data are hidden or missing. In the E-step, conditional expectation is applied to estimate the missing data. In the M-step, the likelihood function is maximised by clustering the data into spatio-temporal patches

In general, all statistical methods require a prior (the statistical model) and show good performance in scenes that are very well represented by the statistical model. In scenes that are different from the models, the methods often fail, making the motion segmentation unsuccessful.

4.1.5. Deep Learning

Last, deep learning techniques became very popular in recent times. These kind of approaches use machine learning techniques, such as artificial and convolutional neural networks, to solve the segmentation problem. Most of the deep learning based methods need to be trained on annotated

data to achieve good performance. The training stage highly influences the performance of these methods. However recent works which do not require annotated data, such as the saliency-based method developed by Lu et al [23], have shown to be able to achieve comparable performance when using network architectures that can capture intrinsic properties in among video frames.

4.1.6. Brief Evaluation

To conclude, a global distinction between motion segmentation methods is the prior knowledge. While it is true that many successful motion segmentation methods require prior knowledge for supervised training, this limits their real world application, since they need very large datasets with manually annotated data. The annotation is a very expensive and time-consuming task, which is considered the main limitation of motion segmentation methods. Hence self-supervised motion segmentation techniques have been recently developed, which do not require data to be labelled for training. This advantage would make motion segmentation widely applicable in many real world data. Current successful self-supervised motion segmentation methods show comparable performance to the supervised ones, however the same performance is not guaranteed when dealing with very different scenes or the challenges mentioned above. As currently there is no general solution to solve motion segmentation for all types of motion including challenges such as occlusions, ongoing researches are tailored towards how to best capture the scene dynamics in an self-supervised framework.

4.2. Self-Supervised Frame-based Motion Segmentation

As mentioned in section 4.1, the literature on motion segmentation is extensive. Furthermore, the literature becomes even larger when Video Object Segmentation (VOS) is considered. The main difference between these two tasks is found in the principle: while motion segmentation tries to separate *independently moving objects* from the background, video object segmentation tries to separate the foreground from the background. In the special case where the foreground is made up of one moving object, the two computer vision tasks perform the same. In case two or more moving objects are present in the foreground, motion segmentation is then required to make distinction between the two objects, whilst VOS is only required to separate them from the background but not necessarily distinguish between the two. Since the majority of state-of-the-art motion segmentation methods still rely on one moving object per scene, VOS methods are investigated.

Since the aim of this research project is to develop a self-supervised framework for motion segmentation, the current literature on self-supervised frame-based segmentation (motion and video) is explored. Often self-supervised methods are labelled as unsupervised, since ground truth is not used. However, the definition of unsupervised is ill-posed. In literature, many methods are unsupervised only during inference and test, whilst including some parts of their architecture (i.e. networks, masks) that are pretrained on an external dataset with ground truth. For example, COSNet [24] uses co-attention to capture rich correlations between frames of a video, but masks are pretrained on ground truth. Similarly, *Ye et al.* [48] perform motion segmentation using global sprites, but it also requires precomputed masks. *Li et al.* [19] propose the use of instance embeddings to find the moving object based on motion saliency and objectness. However, the dense embeddings are obtained from an instance segmentation that is performed by a network pretrained on static images.

In general, frame-based methods that are self-supervised learn by either reconstructing the image and/or reconstructing the optical flow. In subsection 4.2.1, methods relying on image reconstruction are presented, whereas in subsection 4.2.2 methods that reconstruct optical flow are shown.

4.2.1. Image Reconstruction

Currently the most promising self-supervised method based on image reconstruction is the Layered Differentiable Image Synthesis (LDIS) proposed by *Shrestha et al.* [39]. A visual representation of the newtork (LayerSegNet) is shown in Figure 4.4. Similarly to the method described in subsection 4.1.2, the core concept is that, given two consecutive RGB frames I_1 and I_2, it is possible to segment the foreground object from the background by splitting I_1 into disjoint layers, warping and combining them together to reconstruct I_2. The reconstruction loss is used to learn and train the algorithm.

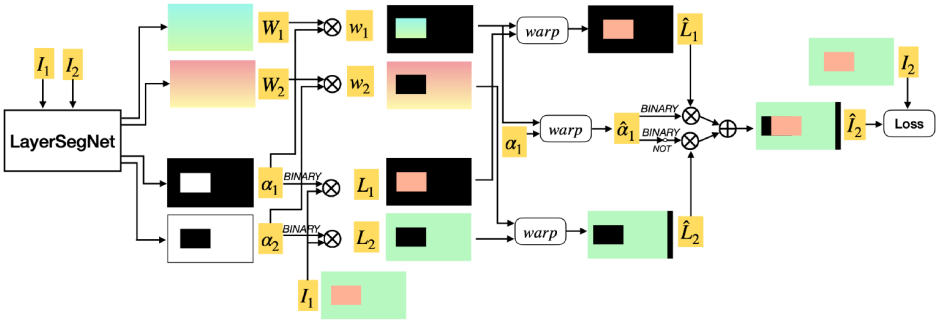


Figure 4.4: Visual example of the Layered Differentiable Image Synthesis (LDIS). Image I_1 is separated into layers that are individually warped and then combined together to obtain a prediction of the second image $\hat{I}_2$

The method works as follows. Using coordinate notation $\mathbf{x} = (x, y)^T$ to describe pixel coordinates in the image domains $I_1(\omega_1)$ and $I_2(\omega_2)$. The objective is to separate I_1 into disjoint layers, move them individually and then combine them together to form a reconstruction of the second image $\hat{I}_2$. Each layer consist of:

- RGB intensity map $L_i : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}^3$
- Alpha map $\alpha_i : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}^1$
- Flow map $w_i : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}^2$

The motion in I_1 and I_2 are described by the transformation matrices A_1 and A_2, which are used to calculate the optical flow maps $W_i : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}^2$ for all $\mathbf{x}$ in Equation 4.2:

$$W_i(x, y) = \mathbf{A}_i \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} a_i^1 & a_i^2 & a_i^3 \\ a_i^4 & a_i^5 & a_i^6 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}, i \in \{1, 2\} \tag{4.2}$$

Successively I_1 is split into layers using alpha maps. These maps define the spatial support regions. Ideally, an one-hot vector would correspond to each pixel to indicate the layer membership, since each pixel can either belong to the foreground or background. On the other hand, a discrete representation of the scene would make the pipeline non differentiable. Instead, softmax binning [47] is used to approximate hard binning (Equation 4.3).

$$\alpha^i(\mathbf{x}) \in [0, 1], i \in \{1, 2\} \tag{4.3}$$

Using a modified maxout operation [51], the maximum value of the two alpha maps is retained, while the other is set to 0. With these alpha maps, the optical flow field associated to each layer w_i using Equation 4.4, where $\odot$ represents element-wise product with broadcasting.

$$w_i = \alpha^i \odot W_i \tag{4.4}$$

Finally, the RGB map for each layer L_i by associating pixel intensities to the alpha maps (Equation 4.5).

$$L_i = \alpha^{i-\text{binary}} \odot I_1 \tag{4.5}$$

To prevent the pixel intensities being scaled with the continuous alpha maps, Equation 4.6 performs binarisation on the maps. This operation not only stops the learning process from performing backpropagation through this operation, but also acts as a constraint such that the network can only learn through Equation 4.4, hence learning to associate alpha maps to flow values in W_i without affecting the pixel intensities in I_1.

$$\alpha^{i-\text{binary}} = \left(\alpha^{i} > 0.5\right) \tag{4.6}$$

Next, forward warping [28] in Equation 4.7 is used to obtain the warped intensity maps $\hat{L}_{i\in\{1,2\}}$ and alpha maps $\alpha_{i\in\{1,2\}}$ using their flows $w_{i\in\{1,2\}}$.

$$\hat{L}_i\left(\mathbf{x}+w_i(\mathbf{x})\right) \leftarrow L_i(\mathbf{x}); \quad \hat{\alpha}^i\left(\mathbf{x}+w_i(\mathbf{x})\right) \leftarrow \alpha^i(\mathbf{x}); \quad \forall \mathbf{x} \in \mathbf{m}_i, i \in \{1,2\} \tag{4.7}$$

It is now possible to synthesise a reconstruction of I_2 using the warped layers. Due to the warp, the layers would contend for the pixels where they overlap. To solve this, depth ordering via the alpha map is used, thus the layer with the closer alpha map to the camera occludes the layer behind . This is to mimic the results of occlusions. A simple rule to determine the depth order is that the smaller i represents a layer closer to the camera. Hence $\hat{I}_2$ will be synthesised as in Equation 4.8 using $\hat{L}_1$ on top of $\hat{L}_2$, and only $\hat{\alpha}_1$ is needed for reconstruction since it is the first layer for support:

$$\hat{I}_2 = \hat{\alpha}_{1-\text{binary}} \cdot \hat{L}_1 + \left(1 - \hat{\alpha}_{1-\text{binary}}\right) \cdot \hat{L}_2 \tag{4.8}$$

The learning process is done by pixel-wise comparison between the output $\hat{I}_2$ with the original consecutive frame I_2. However, the overlap of the layers will eventually lead to some regions of the image where there is no pixel-value contribution, resulting in black regions. Equation 4.9 is used to find these pixels, which are masked out during the loss calculation.

$$\mathbf{D}(\mathbf{x}) = \begin{cases} 1 & , \text{ if } \hat{I}_2(\mathbf{x}) = 0 \\ 0 & , \text{ otherwise} \end{cases} \tag{4.9}$$

Finally the learning is driven by a photometric loss (Equation 4.10), which uses a robust generalised Charbonnier penalty function [3]:

$$\text{Loss} = \sum_{\mathbf{x}\in\Omega_2} (1 - \mathbf{D}(\mathbf{x}))\rho\left(I_2(\mathbf{x}) - \hat{I}_2(\mathbf{x})\right) \quad \rho(\mathbf{x}) = \sqrt{\mathbf{x}^2 + 0.001^2} \tag{4.10}$$

Successful results on MovingCars dataset are shown in Figure 4.5.

(a) Frame1 (b) GT (c) Prediction

Figure 4.5: Visuals results of LDIS on two scenes of Moving Cars dataset a) RGB frames b) ground truth (GT)

4.2.2. Optical Flow Reconstruction

Currently the vast majority of self-supervised methods for motion segmentation rely on the reconstruction of optical flow. In most of the cases, the methods are a crossover between layer-based techniques where the learning is implemented in a convolutional neural network. Following are the current state-of-the-art in frame-based motion segmentation.

Unsupervised Feature Learning with ConvNet

Early attempts of motion segmentation with optical flow reconstruction are seen in the ConvNet by *Pathak et al.* [33]. The overall approach is shown in Figure 4.6. The aim is to train a network to segment objects in single frames without any supervision. The core intuition behind the method is that the network can learn high-level feature representation without any supervision. First, the single frames with optical flow map are passed to the network. Moving objects are identified by pixels that have similar optical flow (direction and rate). The resulting mask is used as pseudo-ground truth for the network to learn to predict the same mask using the single frame only.

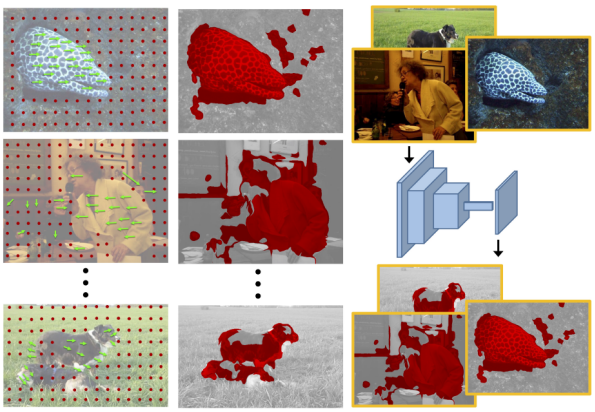


Figure 4.6: Overview of our approach. We use motion cues to segment objects in videos without any supervision. We then train a ConvNet to predict these segmentations from static frames, i.e. without any motion cues. We then transfer the learned representation to other recognition tasks

Slot Attention

Yang et al. [44] propose slot attention to solve the motion segmentation by solely using optic flow. The underlying assumption is the common fate principle, which states that elements tend to be perceived as a group if they move at the same rate in the same direction (similar optic flow). The pipeline is shown in Figure 4.7. The model takes optical flow $I_{t\to t+n_1}$ (from time t to $t+n_1$) as input, and aims at splitting the foreground and background in two layers. Each layer consists of the respective optical flow and alpha map.

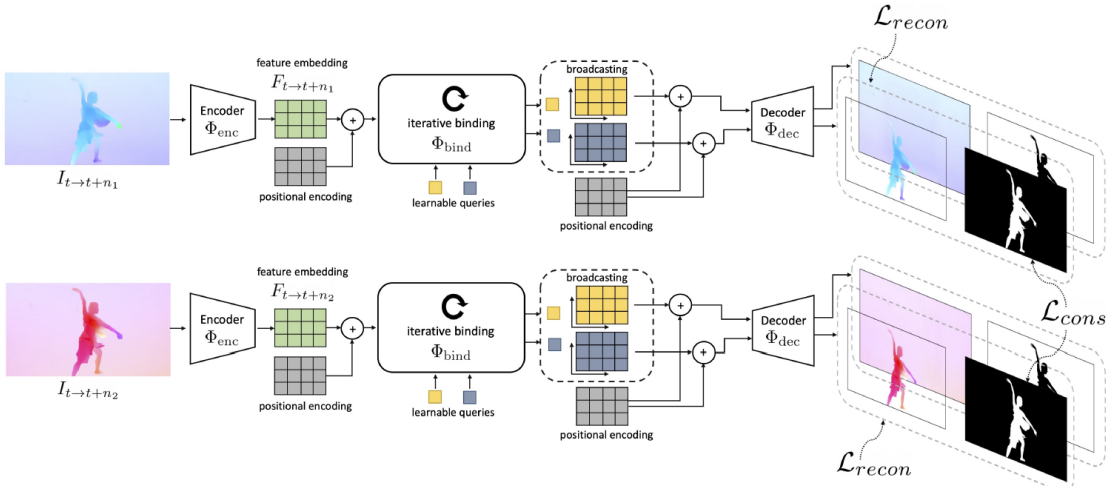


Figure 4.7: Pipeline [44]

The network is composed of three main elements: feature encoding, iterative binding and decoding

to layers. In feature encoding, he precomputed optical flow between two frames $I_{t\to t+n} \in \mathcal{R}^{3\times H_0\times W_0}$ is passed to the CNN encoder Φ_{enc}, and the output is a lower-resolution feature map, as shown in Equation 4.11, where H_0, W_0 and H, W are the spatial dimensions of input and ouptut feature maps.

$$F_{t\to t+n} = \Phi_{\text{enc}}\left(I_{t\to t+n}\right) \in \mathcal{R}^{D\times H\times W} \tag{4.11}$$

The iterative binding module Φ_{bind} aims at grouping together pixels that have similar optical flow i.e. similar velocity and direction. For this, slot attention is used [22]. More details can be found in [44]. Finally, the decoding is done by a CNN decoder Φ_{dec} decodes each of the slots from Φ_{bind} and outputs to original resolution. The outputs are the reconstruct flow fields for foreground and background, as well as their alpha maps. The learning is done via reconstruction loss of the input optic flow as shown in Equation 4.12, where p is the pixel index and Ω is the entire spatial grid.

$$\mathcal{L}_{recon} = \frac{1}{\Omega}\sum_{n\in\cap}\left|I_{t\to t+n}(p) - \hat{I}_{t\to t+n}(p)\right|^2 \tag{4.12}$$

Furthermore, pixel-wise entropy regularization loss is introduced in Equation 4.13 to force the mask to be binary. The loss is zero when the alpha channels are one-hot, and maximum when they are of equal probability.

$$\begin{aligned}\mathcal{L}_{entr} = \frac{1}{\Omega}\sum_{p\in\Omega}\Big(&-\alpha^1_{t\to t+n}(p)\log\alpha^1_{t\to t+n}(p)\\ &-\alpha^0_{t\to t+n}(p)\log\alpha^0_{t\to t+n}(p)\Big)\end{aligned} \tag{4.13}$$

A third loss (consistency loss) is introduced in Equation 4.14 to address the issue of temporary stopping of moving objects. In this case, a second set of frames is used to make another prediction of the moving object, and the network is encouraged to predict the same foreground/background segmentation. However, this loss is used only in training and not during inference.

$$\begin{aligned}\mathcal{L}_{\text{cons}} = \tfrac{1}{\Omega}\min\Big(&\textstyle\sum_{p\in\Omega}\left|\alpha^1_{t\to t+n_1}(p) - \alpha^1_{t\to t+n_2}(p)\right|^2\\ &\textstyle\sum_{p\in\Omega}\left|\alpha^1_{t\to t+n_1}(p) - \alpha^0_{t\to t+n_2}(p)\right|^2\Big)\end{aligned} \tag{4.14}$$

Finally, the total loss is shown in Equation 4.15. The model is robust when the following values for the hyperparameters are used: $\gamma_r = 10^2$, $\gamma_c = 10^{-2}$ and $\gamma_e = 10^{-2}$.

$$\mathcal{L}_{\text{total}} = \gamma_r\mathcal{L}_{\text{recon}} + \gamma_c\mathcal{L}_{\text{cons}} + \gamma_e\mathcal{L}_{\text{entr}} \tag{4.15}$$

Contextual Information Separation (CIS)

While in [39] and [44] the models try to segment the moving object by using the optical flow of the object itself, *Yang et al.* [46] propose to find the moving object by using only the information contained in the background. The pipeline is shown in Figure 4.8. The idea is that the motion of the moving object is independent of the background, therefore the motion of the background is uninformative of the motion of the foreground and vice-versa.

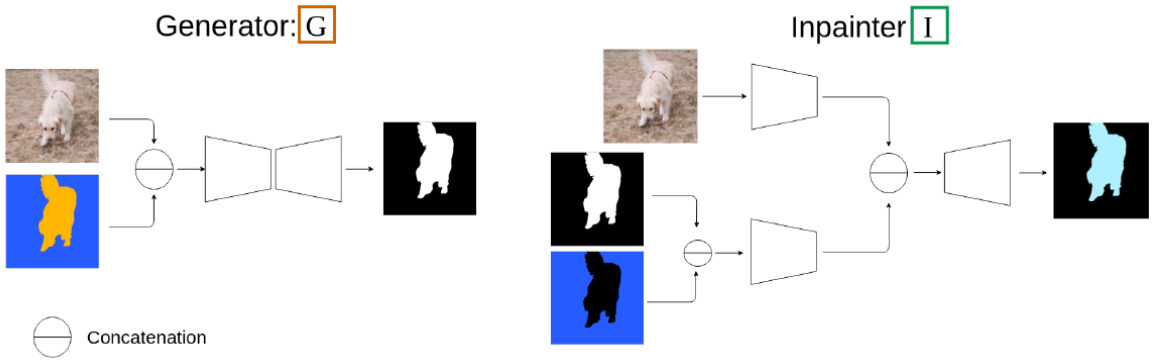


Figure 4.8: Pipeline of the method described in [46]. The generator tries to hide the optic flow of the moving object with a mask, while the inpainter tries to compute the hidden optic flow by using the optical flow available from the generator

In this regard, the model is composed of two modules, a Generator (G) and Inpainter (I). The generator applies a mask to the input optic flow, with the aim of hiding the optic flow of the foreground. Next, the generator passes the masked optic flow to the inpainter. The inpainter has the task of reconstructing the optic flow that has been hidden by the generator. An example is shown in Figure 4.9. The two diagrams show the learning process of the Generator after the inpainter has learned how to inpaint a masked flow. In the upper diagram, a poorly trained generator is shown. Due to the impecise masking, a part of the foreground optical flow is visible to the inpainter, hence the inpainter can perform an accurate reconstruction. The opposite is shown in the lower diagram. Here the generator is able to accurately mask the foreground optic flow, therefore the inpainter has no information of the foreground optic flow and performs a bad reconstruction. In essence, accurate motion segmentation is achieved when the optic flow reconstruction works poorly.

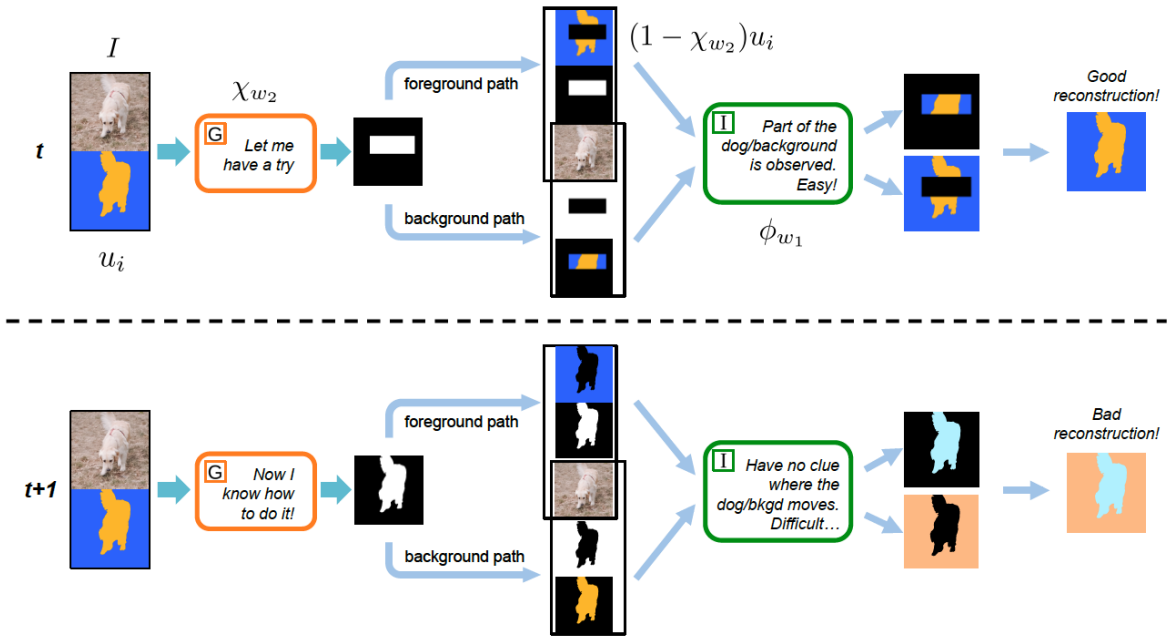


Figure 4.9: Diagrams showing the performance of the generator (G): upper diagram shows an example of a poorly trained generator, whereas the lower diagram shows a well-trained generator [46]

The method is formulated as follows. Given an image I and its optical flow u, the foreground is any region of the image whose motion is unexplainable from the context, that is, the joint probability distribution of background and foreground optic flow equals the product of the marginals:

$$P(u_1, u_2) = P(u_1)P(u_2) \tag{4.16}$$

If optic flow in two locations i and j is considered, foreground can be formalized as a region Ω that is uninformative by the background:

$$\begin{cases} \mathbb{I}\left(u_i, u_j \mid I\right) > 0, i, j \in \Omega \\ \mathbb{I}\left(u_i, u_j \mid I\right) = 0, i \in \Omega, j \in D \backslash \Omega \end{cases} \tag{4.17}$$

The information separatation is implemented with the Information Reduction Rate (IRR) γ which takes $\boldsymbol{x}, \boldsymbol{y} \subset D$ and returns a non-negative scalar. $\mathbb{H}$ denotes Shannon entropy, and it is zero when two variables are independent.

$$\gamma(\boldsymbol{x} \mid \boldsymbol{y}; I) = \frac{\mathbb{I}\left(u_{\boldsymbol{x}}, u_{\boldsymbol{y}} \mid I\right)}{\mathbb{H}\left(u_{\boldsymbol{x}} \mid I\right)} = 1 - \frac{\mathbb{H}\left(u_{\boldsymbol{x}} \mid u_{\boldsymbol{y}}, I\right)}{\mathbb{H}\left(u_{\boldsymbol{x}} \mid I\right)} \tag{4.18}$$

The region of the image that belong to the foreground Ω are the ones that minimize the loss function in Equation 4.19:

$$\mathcal{L}(\Omega; I) = \gamma\left(\Omega \mid \Omega^c; I\right) + \gamma\left(\Omega^c \mid \Omega; I\right) \tag{4.19}$$

However, this loss is intractable. Assumptions need to be taken on the underlying probability model so that the loss in Equation 4.19 can be minimized. Given the optic flow of the foreground $u^{\text{in}} = \{u_i, i \in \Omega\}$ and the background $u^{\text{out}} = \{u_i, i \in \Omega^c\}$, in the IRR the important term is $\mathbb{H}\left(u^{\text{in}} \mid u^{\text{out}}, I\right) / \mathbb{H}\left(u^{\text{in}} \mid I\right)$, which measures the information transfer from foreground to background. The term is more elaborated in Equation 4.20:

$$\frac{\int \log P\left(u^{\text{in}} \mid u^{\text{out}}, I\right) dP\left(u^{\text{in}} \mid u^{\text{out}}, I\right)}{\int \log P\left(u^{\text{in}} \mid I\right) dP\left(u^{\text{in}} \mid I\right)} \tag{4.20}$$

The assumptions on the probability models are as in Equation 4.21, where $\phi(\Omega, y, I) = \int u^{\text{in}} dP\left(u^{\text{in}} \mid u^{\text{out}}, I\right)$ is the conditional mean given the image and complementary observation.

$$\begin{aligned} P\left(u^{\text{in}} = x \mid I\right) &\propto \exp\left(-\frac{\|x\|^2}{\sigma^2}\right) \\ P\left(u^{\text{in}} = x \mid u^{\text{out}} = y, I\right) &\propto \exp\left(-\frac{\|x - \phi(\Omega, y, I)\|^2}{\sigma^2}\right) \end{aligned} \tag{4.21}$$

Another assumption is that given a single image the best guess of the optical flow is zero ($\phi(\Omega, \varnothing, I) = 0$). This leads to the approximation of Equation 4.20 in Equation 4.22:

$$\frac{\int \left\|u^{\text{in}} - \phi\left(\Omega, u^{\text{out}}, I\right)\right\|^2 dP\left(u^{\text{in}} \mid u^{\text{out}}, I\right)}{\int \|u^{\text{in}}\|^2 dP\left(u^{\text{in}} \mid I\right)} \approx \frac{\sum_{i=1}^N \left\|u_i^{\text{in}} - \phi\left(\Omega, u_i^{\text{out}}, I\right)\right\|^2}{\sum_{i=1}^N \left\|u_i^{\text{in}}\right\|^2} \tag{4.22}$$

Hence the loss can be minimized with the approximation given in Equation 4.23, where N is the number of samples available. The only remaining terms to be determined are the region Ω and the function ϕ.

$$\mathcal{L}(\Omega; I) = 1 - \frac{\sum_{i=1}^N \left\|u_i^{\text{in}} - \phi\left(\Omega, u_i^{\text{out}}, I\right)\right\|^2}{\sum_{i=1}^N \left\|u_i^{\text{in}}\right\|^2 + \epsilon} + 1 - \frac{\sum_{i=1}^N \left\|u_i^{\text{out}} - \phi\left(\Omega^c, u_i^{\text{in}}, I\right)\right\|^2}{\sum_{i=1}^N \left\|u_i^{\text{out}}\right\|^2 + \epsilon} \tag{4.23}$$

The region Ω that minimizes Equation 4.23 is part of the power set of the image domain D, which can be rewritten with the indicator function shown in Equation 4.24 such that Ω can be rewritten as $u_i^{\text{in}} = \chi u_i$ and the background as $u_i^{\text{out}} = (1-\chi)u_i$:

$$\begin{aligned} \chi : D &\rightarrow \{0,1\} \\ i &\mapsto 1 \text{ if } i \in \Omega; 0 \text{ otherwise} \end{aligned} \tag{4.24}$$

The function ϕ is non-linear,non-local and highly dimensional, since it has to predict the flow in a region based on the flow information outside of that region. Since the generator and inpainter follow an encoder-decoder architecture style, both ϕ and χ are parameterized with parameters w, such that the corresponding functions are ϕ_{w1} and χ_{w2}. Hence the negative terms of the loss (negative loss) in Equation 4.23 can be rewritten as in Equation 4.25:

$$\mathcal{L}(w_1, w_2; I) = \frac{\sum_i \left\| \chi_{w_2}\left(u_i - \phi_{w_1}\left(\chi_{w_2}, u_i^{\text{out}}, I\right)\right)\right\|^2}{\sum_i \left\| u_i^{\text{in}} \right\|^2} + \frac{\sum_i \left\| \left(1-\chi_{w_2}\right)\left(u_i - \phi_{w_1}\left(1-\chi_{w_2}, u_i^{\text{in}}, I\right)\right)\right\|^2}{\sum_i \left\| u_i^{\text{out}} \right\|^2} \tag{4.25}$$

ϕ_{w1} is the *inpainter network*, and must be chosen to minimize the loss in Equation 4.25. Conversely the foreground, represented by χ_{w2}, is called *mask generator network*, and it should be chosen so that u^{out} is as uninformative as possible of u^{in} . Therefore the loss in Equation 4.25 needs to be maximised. This leads to the minimax problem shown in Equation 4.26:

$$\hat{w} = \arg \min_{w_1} \max_{w_2} \mathcal{L}(w_1, w_2; I) \tag{4.26}$$

Multi-motion and Appearance Self-supervised Network (MASNet)

The work in [46] obtains good result in public datasets. However, the model fails to capture complete objects or differentiate regions in the background. The reasons are two-folded.

First, the model works only with one single scale temporal information. Since moving objects are composed of many regions, one single scale of temporal information may encode only a fraction of the moving object. An example is shown in Figure 4.10. In A the flow from t to $t+1$ detects only the left leg of the dancer, whereas the flow from t to $t+2$ captures the motion of his right leg and body, but not the left leg. Therefore, single scale temporal information are insufficient to fully detect an articulated moving object.

Second, the moving camera introduces a bias in the scene. The joint movement of camera and object violates the motion independence hypothesis. For example, in Figure 4.10 B the object and background motion are similar. Thus, it is impossible to differentiate object from background in the current framework. In addition, moving camera can yield locally independent moving regions in background. Such regions mislead the model to generate false detection in background (the red box of Figure 1B).

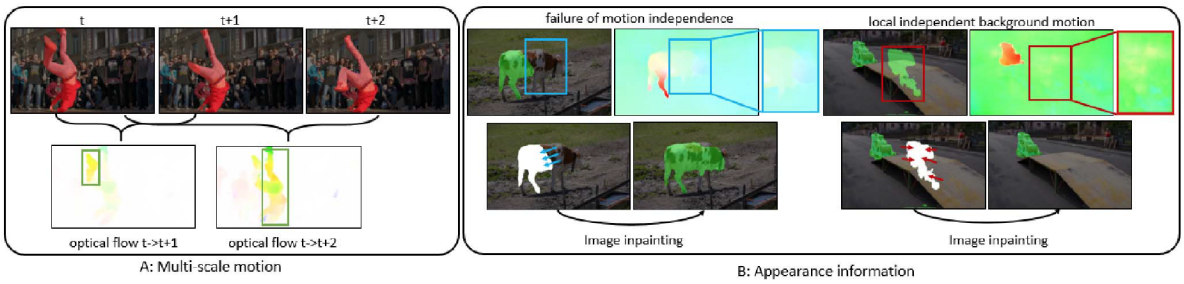


Figure 4.10: Multi-scale motion and appearance information. In A, the green boxes represent different moving regions at different temporal scales. In B, the blue boxes indicate regions of the background that should be incorporated in the foreground since they belong to the same moving object, whereas the red boxes shows regions of the background that are wrongly labelled as foreground. Image inpainting can solve these issues using contextual information [45]

In this regard, *Yang et al.* [45] developed a Multi-motion and Appearance Self-supervised Network (MASNet) to solve the issues mentioned above. Specifically, MASNet is composed of two modules: a Multibranch Flow Encoding module (MFE) and an IMage InPainter module (IMIP). The modules are shown in Figure 4.11. The MFE module takes as input multiple optic flow maps to encode multi-scale motion information over the different temporal scales to form a final detection. The IMIP module is designed to tackle the problems caused by camera movement. To remove false detection in the background, the IMIP exploits spatial appearance information. This is based on the hypothesis that object appearance is different from background. Similarly, for missing detection in moving object, IMIP can infer the masked region by the appearance in surrounding regions.

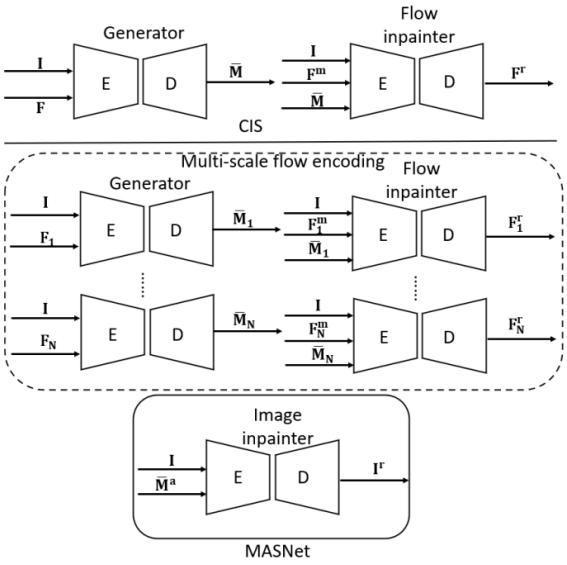


Figure 4.11: Comparison between CIS [46] and MASNet [45]. I is the image, F the optical flow map and M the predicted mask. As indexes, N is the frame index, r recovered, a average, and m masked

The architecture is shown in Figure 4.12. The MFE is composed of generator and inpainters. The generators takes image I and optical flow maps corresponding to frames $F_1, ..., F_N$ at time $t_1, ..., t_N$. This is to generate masks $\bar{M}_1, \ldots, \bar{M}_N$. Then the image I, segmentation mask $\bar{M}$ and masked flow F^m are passed to the inpainter which attempts to recover the masked flow map F^r

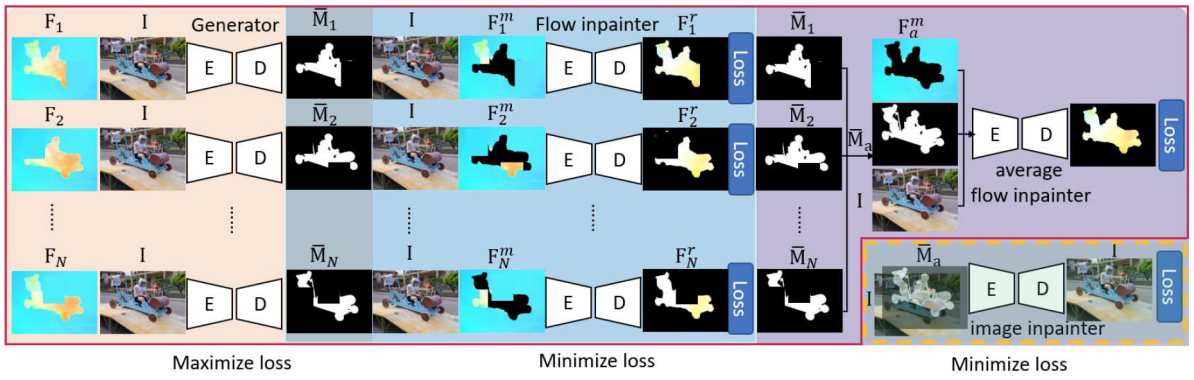


Figure 4.12: MASNet architecture [45]

The generator can be expressed as $\bar{M} = G(F, I)$ where $G(\dot{})$ is the generator function. The inpainter can be expressed as $F^r = \mathcal{I}\left(\bar{M}, F^m, I\right)$, where I is the inpainter function, $F^m = F \times (1 - \bar{M})$ is the masked flow map, and F^r the flow map. The loss in the recovered flow map is described as in Equation 4.27, where F^m_{in} is the flow in the mask:

$$\frac{\left\|\bar{M} \times (F - F^r)\right\|_2^2}{\left\|F_{in}^m\right\|_2^2} \tag{4.27}$$

Whereas the loss in the region outside the mask is calculated as in Equation 4.28, where F_{out}^m is the flow outside the mask:

$$\frac{\left\|(1 - \bar{M}) \times (F - F^r)\right\|_2^2}{\left\|F_{\text{out}}^m\right\|_2^2} \tag{4.28}$$

Therefore the final loss is shown in Equation 4.29. As in [46], the generator tries to hide the foreground as much as possible whereas the inpainter tries to recover it the best way possible, and this gives rise to a minimax problem as shown in Equation 4.30.

$$\begin{aligned}\mathcal{L}(\mathcal{G}, \mathcal{I}; I) = &\frac{\left\|\mathcal{G}(F, I) \times \left(F - \mathcal{I}\left(\mathcal{G}(F, I), F_{\text{out}}^m, I\right)\right)\right\|_2^2}{\left\|F_{in}^m\right\|_2^2} + \\ &\frac{\left\|(1 - \mathcal{G}(F, I)) \times \left(F - \mathcal{I}\left(1 - \mathcal{G}(F, I), F_{in}^m, I\right)\right)\right\|_2^2}{\left\|F_{\text{out}}^m\right\|_2^2}\end{aligned} \tag{4.29}$$

$$\hat{F} = \arg\min_{\mathcal{I}} \max_{\mathcal{G}} \mathcal{L}(\mathcal{G}, \mathcal{I}; I) \tag{4.30}$$

On the other hand, MASNet uses multi-scale information, hence there are multiple losses according to each time-scale. Taking a look at the red box in Figure 4.12, given image I at time t and optical flow maps $F_1, ..., F_N$ at time $t_1, ..., t_N$, it is possible to identify the loss $\mathcal{L}^n$ as in Equation 4.31. The loss function of the whole scheme including all generators and inpainters is summarized in Equation 4.32.

$$\begin{aligned}\mathcal{L}^n\left(\mathcal{G}^n, \mathcal{I}^n; I_t\right) = &\frac{\left\|\mathcal{G}^n \times \left(F^n - \mathcal{I}^n\left(\mathcal{G}^n, F_{\text{out}}^m, I_t\right)\right)\right\|_2^2}{\left\|F_{in}^n\right\|_2^2} + \\ &\frac{\left\|(1 - \mathcal{G}^n) \times \left(F^n - \mathcal{I}^n\left(1 - \mathcal{G}^n, F_{in}^n, I_t\right)\right)\right\|_2^2}{\left\|F_{\text{out}}\right\|_2^2}\end{aligned} \tag{4.31}$$

$$\mathcal{L}_{ms}\left(\mathcal{G}, \mathcal{I}; I_t\right) = \sum_{n=1}^{N} \mathcal{L}\left(\mathcal{G}^n, \mathcal{I}^n; I\right) \tag{4.32}$$

Additional information can be retrieved when the segmentation masks produced by all generators are combined together as an input for an extra inpainter to retrieve an average optical flow map. This adds additional supervision to the generators during the training process. The function for the average loss is shown in Equation 4.33, where a indicates average over N frames. The average inpainter will then guide the training of generators in each branch.

$$\begin{aligned}\mathcal{L}_{avg}\left(\mathcal{G}^a, \mathcal{I}^a; I\right) = &\frac{\left\|\mathcal{G}^a \times \left(F^a - \mathcal{I}^a\left(\mathcal{G}^a, F_{\text{out}}^a, I\right)\right)\right\|_2^2}{\left\|F_{\text{in}}^a\right\|_2^2} \\ &+ \frac{\left\|(1 - \mathcal{G}^a) \times \left(F^a - \mathcal{I}^a\left(1 - \mathcal{G}^a, F_{in}^a, I\right)\right)\right\|_2^2}{\left\|F_{\text{out}}\right\|_2^2}\end{aligned} \tag{4.33}$$

Finally, for the image inpainter IMIP the loss is shown in Equation 4.34, where I_m is the masked image

$$\mathcal{L}_{im}\left(\mathcal{G}^a, \mathcal{I}^{im}; I\right) = \left\|\mathcal{G}^a \times \left(I - \mathcal{I}\left(I^m_{\text{out}}\right)\right)\right\|_2^2 + \left\|(1-\mathcal{G}^a) \times \left(I - \mathcal{I}\left(I^{1-m}_{\text{in}}\right)\right)\right\|_2^2 \tag{4.34}$$

4.3. Event-based Motion Segmentation

Currently there is no existence of event-based motion segmentation methods that are self-supervised. Nevertheless, the current literature on methods for motion segmentation with event cameras are investigated. In subsection 4.3.1 the state-of-the-art application of motion compensation to motion segmentation is shown, while in subsection 4.3.2 supervised and semi-supervised methods are explained.

4.3.1. Event-based Motion Segmentation by Motion Compensation

In the current literature, there is no method that performs motion segmentation with events in a self-supervised way. Despite not having any learning framework, the only current method that performs motion segmentation without any use of annotated data (hence unsupervised) is based on motion compensation. *Stoffregen et al.* [41] propose the idea of segmenting moving objects with motion compensation. The pipeline is shown in Figure 4.13. Similar to layer-based approaches, the idea is to classify a set of events in a space-time window into different layers ("clusters") that each represent a coherent motion, including background. This is done by estimating, at the same time, the motion parameters that describe the moving object(s) and the likelihood of events belonging to clusters based on motion compensation.

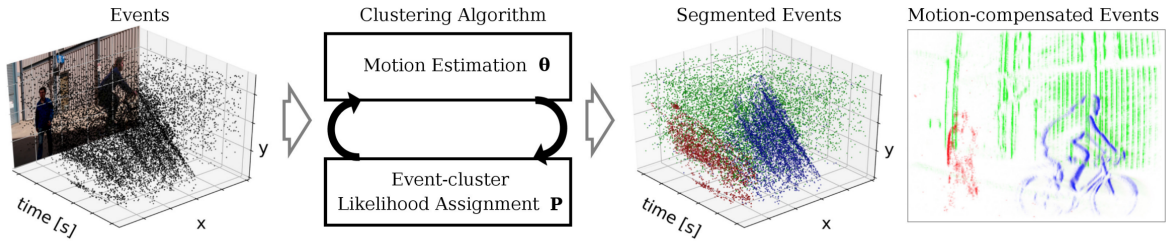
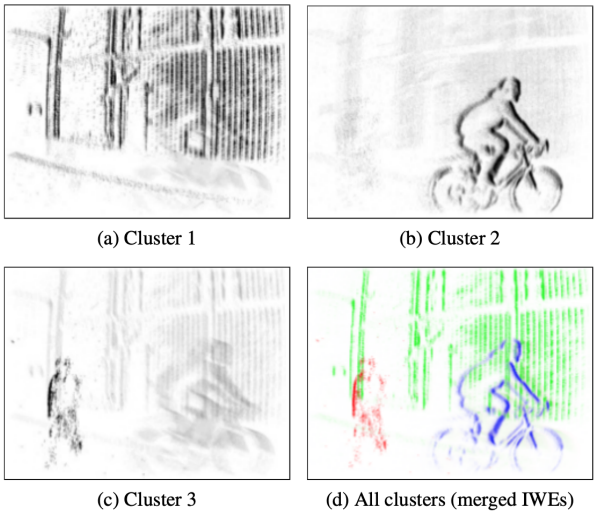


Figure 4.13: Pipeline of the method in [41]

Recalling the working principles of motion compensation from section 3.1, motion segmentation can be done by just using motion compensation, in a scenario where only one object is moving. However, in a scenario where multiple objects are moving with different motions, motion compensation alone is not enough to discern the objects since the events are triggered by moving objects with no correlation. Hence, depending on the number of objects in the scene, the same number of warps and motion models are needed. Additionally, allocation of events to the correct cluster is necessary to make the warping successful. In essence, if the motion model is correct and the events are correctly grouped in the cluster, a sharp Image of Warped Events (IWE) is produced. In Figure 4.14, three motion models are generated for background, pedestrian and cyclist. After the iteration shown in Figure 4.13, three sharp IWEs are shown representing the different entities. If events are wrongly clustered, blurred IWE will be produced. Hence the sharpness of these IWEs is used to evaluate the segmentation quality.



(a) Cluster 1 (b) Cluster 2

(c) Cluster 3 (d) All clusters (merged IWEs)

Figure 4.14: Event-cluster associations (a, b, c) and final segmentation (d) **[9010]**

The method is mathematically formulated as follows. The event-cluster association is described with $p_{kj} = P\left(e_k \in \ell_j\right)$, which is the probability of the k-th event belonging to the j-th cluster. $\mathbf{P} \equiv \left(p_{kj}\right)$ is a N_e x N_l matrix that contains all event-cluster probability associations. Each row must add up to 1 and each entry must be non-negative. Events are warped to a reference time t_{ref} as in Equation 4.35:

$$e_k \doteq \left(\mathbf{x}_k, t_k, s_k\right) \mapsto e'_k \doteq \left(\mathbf{x}'_k, t_{\text{ref}}, s_k\right) \tag{4.35}$$

Hence the weighted IWE is described as in Equation 4.36,with $\mathbf{x}'_{kj} = \mathbf{W}\left(\mathbf{x}_k, t_k; \boldsymbol{\theta}_j\right)$ the warped event location and δ the Dirac delta. Equation 4.36 states that events are warped and the values $p_{kj} > 0$ are accumulated at the warped locations $\mathbf{X}'_k$.

$$I_j(\mathbf{x}) \doteq \sum_{k=1}^{N_e} p_{kj}\delta\left(\mathbf{x} - \mathbf{x}'_{kj}\right) \tag{4.36}$$

The sharpness of IWEs is estimated using the variance shown in Equation 4.37, where μ_{I_j} is the mean over the image plane Ω [6]:

$$\text{Var}\left(I_j\right) \doteq \frac{1}{|\Omega|}\int_{\Omega}\left(I_j(\mathbf{x}) - \mu_{I_j}\right)^2 d\mathbf{x} \tag{4.37}$$

The problem is then formulated as finding the probability associations $\mathbf{P}$ and model parameters θ that maximise the sum of contrast of all clusters, as shown in Equation 4.38:

$$\left(\boldsymbol{\theta}^*, \mathbf{P}^*\right) = \arg\max_{(\boldsymbol{\theta}, \mathbf{P})} \sum_{j=1}^{N_\ell} \text{Var}\left(I_j\right). \tag{4.38}$$

Unlike in traditional motion-compensation methods, the solution for the sharpest IWEs does not rely only on the motion parameters θ but also on the probability matrix $\mathbf{P}$. Therefore an iteration algorithm is used. First, $\mathbf{P}$ is fixed and the motion parameters are updated as in Equation 4.39, hence by taking a step ($\mu > 0$) in ascent direction of the objective function in Equation 4.38 with respect to the motion parameters

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu \nabla_{\boldsymbol{\theta}} \left(\sum_{j=1}^{N_\ell} \mathrm{Var}\left(I_j\right) \right) \tag{4.39}$$

Once the motion parameters are updated, $\mathbf{P}$ is updated using the closed-form probability partitioning law shown in Equation 4.40, where $c_j(\mathbf{x}) \neq 0$ is the sharpness of j-th cluster at pixel $\mathbf{x}$.

$$p_{kj} = \frac{c_j\left(\mathbf{x}'_k\left(\boldsymbol{\theta}_j\right)\right)}{\sum_{i=1}^{N_\ell} c_i\left(\mathbf{x}'_k\left(\boldsymbol{\theta}_i\right)\right)} \tag{4.40}$$

Thus, each event is softly assigned to each cluster based on how it contributes to the sharpness of all IWEs. The algorithm iterates until convergence.

Several initialization schemes for $\boldsymbol{\theta}$ and $\mathbf{P}$ can be used to provide a preliminary estimation of the initial parameters. In this case, a greedy approach is used. Equal probability associations are given to all the events, and contrast is maximised for the first cluster with respect to its motion parameters. From here it is possible to see the gradient of the local contrast of each event with respect to the motion parameters. The events that are associated with the first cluster will appear less when moving away from the optimized parameters, hence they will have a negative gradient and will be given a high association probability to that cluster and low for all other ones. The process is repeated until all motion parameters and probability associations have been filled.

The work in [41] is also used as a basis by *Parameshwara et al.* [31], where long term segmentation is made more robust by adding feature tracking. However the method relies on tracklets and ground truth data such as position and velocity of the camera.

4.3.2. Supervised Event-Based Motion Segmentation

Events represent a new paradigm for acquisition of visual information. For this reason, the majority of the motion segmentation methods are supervised at this stage. Despite being supervised, it is of primary importance to understand the underlying working principle of these methods, which can be entirely or partially used for the method develop in this research project. For relevance, the following methods on dodging obstacles (DodgeNet) and use of SNN for motion segmentation are shown.

Deep Dynamic Obstacle Dodging with Event Cameras (DodgeNet)

Sanket et al. [36] propose DodgeNet, a network with the aim of segmenting independently moving objects (IMOs) and dogding them. The pipeline is shown in Figure 4.15. The input are the event frames (E_t^F, E_{t+1}^F) and the output are the segmentations along with the predicted optical flow, which is further used to control the dogding mechanism of the drone. The architecture is mainly composed of three CNN networks which solve the task of deblurring (*EVDeblurNet*), estimating the position of the drone from the monocular event camera (homography with *EVHomographyNet*) and jointly estimate the segmentation of the IMOs along with their optic flow (*EVSegFlowNet*). The main interest lies in the segmentation part. To solve both the problem of complexity and accuracy, the *EVSegFlowNet* is trained in a semi-supervised way, hence the ground truth for the segmentation mask is provided, but no annotated data is provided for the optic flow part. The network is trained on *Moving Object Dataset* (*MOD*).

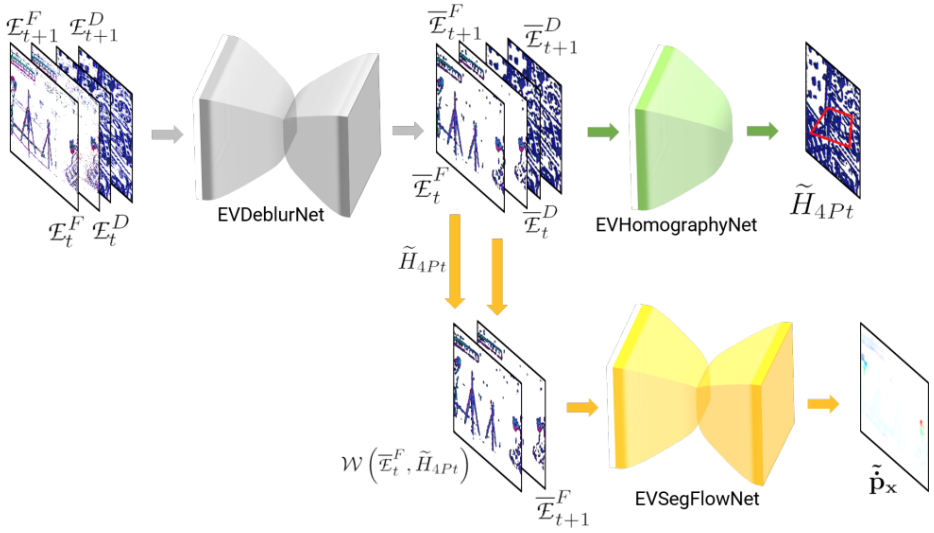


Figure 4.15: DodgeNet pipeline [36]

Deep Spiking Neural Networks (SpikeMS)

Currently the only application of spiking neural networks for event-based motion segmentation is done by *Parameshwara et al.* [32]. The pipeline is shown in Figure 4.16. The SNN has an encoder-decoder structure (*SpikeMS*), where the input event stream is passed (red is brightness increase, blue is brightness decrease) and the output are the regions containing only the moving objects.

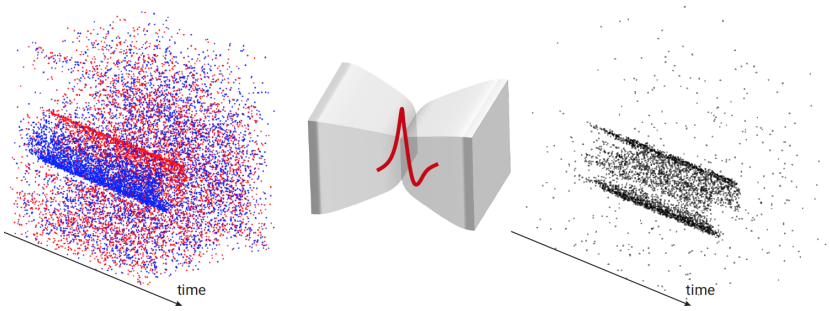


Figure 4.16: SpikeMS pipeline [32]

The main advantage of using SNN is that this type of network also encodes temporal information. In contrast with artificial neural networks, where all the neurons are used, in SNN only the neurons triggered by the corresponding pixels are used, reducing the computational effort required. Therefore SNN are very suitable for exploiting the nature of the events data structure. The working principle of a spiking neuron is shown in Figure 4.17. The neuron receives input either from data or other neurons from lower layers (colored arrows) which generate bump in the membrane voltage $u(t)$. If the voltage exceeds a threshold θ (dotted line), the neuron outputs a spike, and then enters a refractory phase where most likely it will not output other spikes for a short period of time. In [32], the Spike Response Model (SRM) is used, where the refractory phase and the influence of the incoming pulses on the voltages are governed by ε and ν.

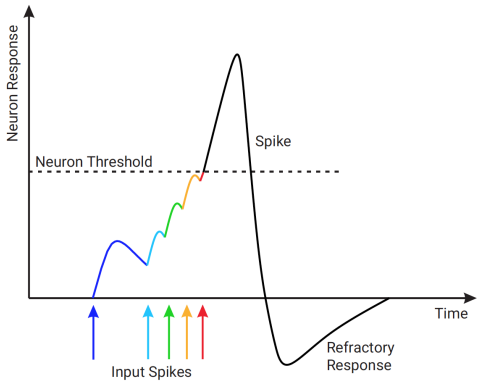


Figure 4.17: Neuron model

For a given neuron i at timestep t, the incoming voltage $u(t)$ is expressed as in Equation 4.41, where Equation 4.41 takes into account all weight connections from pre-synaptic neurons $1, .., j$, $s_i(t)$ is an input spike train in a neuron and $*$ denotes the convolution operator.

$$\begin{aligned} u_i(t) &= \left(\sum_j w_j \left(\varepsilon * s_j \right) \right) + (\nu * s) \\ &= \mathbf{w}^\top \mathbf{a} + (\nu * s) \end{aligned} \tag{4.41}$$

SRM is chosen since the refractory behaviour is modeled in such a way that there is no need of running multiple differential equations. The filters are modelled according to [10] in Equation 4.42 and Equation 4.43, where $\mathcal{H}(t)$ is the Heaviside function, τ_s is the time constant for the spike response and τ_r is the refractory time constant.

$$\varepsilon(t) = \frac{t}{\tau_s} e^{1-\frac{t}{\tau_s}} \mathcal{H}(t) \tag{4.42}$$

$$\nu(t) = -2\theta e^{1-\frac{t}{\tau_r}} \mathcal{H}(t) \tag{4.43}$$

SNN neurons are structured in layers, similarly as ANN. The feed-forward weight matrix $\mathbf{W}^{(l)} = [\mathbf{w}_1, ..., \mathbf{w}_{N_l+1}]$ resulting from a 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 with thresholding function f_s. Thus for all layers the activity if forward-propagated as follows, with ε_d the spike response kernel with delay d. The event data is represented with $s^{(0)}$.

$$a^{(l)}(t) = \left(\varepsilon_d * s^{(l)} \right)(t) \tag{4.44}$$

$$u^{(l+1)}(t) = \mathbf{W}^{(l)} a^{(l)}(t) + \left(\nu * s^{(l+1)}(t) \right) \tag{4.45}$$

$$s^{(l+1)}(t) = f_s \left(u^{(l+1)}(t) \right) \tag{4.46}$$

5

Conclusion

The purpose of this report was to show the findings of the literature study, which was done on event cameras and the current state-of-the-art motion segmentation methods, both for traditional frame-based cameras and events. The literature started with a preliminary investigation on the working principles of event cameras and techniques for optical flow estimation, followed by a wider study on the motion segmentation methods for event cameras and then for traditional cameras. For the latter, a particular focus was given to the self-supervised methods, as the research scope of this project is to develop a self-supervised method.

In the very wide literature on motion segmentation, it was found that the most popular methods can be categorized in image difference, layer-based methods, methods that use statistical knowledge, optical flow and neural networks. Most of these methods can not be categorized in only one of these categories, but are most of the time combining them. For self-supervised frame-based methods, the self-supervision relies heavily on the reconstruction of the input optic flow, suggesting that using layer representation of the optic flow and neural networks is currently the most promising path towards self-supervised algorithms. As the optic flow plays a central part in the acquisition of motion information from event cameras, it is concluded that motion compensation has a fundamental role in finding correlations between events, since it used for other applications such as depth and ego-motion estimation. As motion compensation does not require any ground truth, self-supervised optical flow estimation with motion compensation is found to be a crucial element to include in the framework for motion segmentation using event cameras. Finally, in event-based motion segmentation, it was found that methods typically use motion compensation with a likelihood-cluster assignment framework or other inputs such as ego-motion. Other methods demonstrate the feasibility of using spiking neural networks for segmenting moving objects with event cameras. Nevertheless, currently there is no existence of a method that operates with self-supervision, leaving a research gap and high potential for this project.

As a recommendation to the continuation of this research project, efforts should be tailored towards the development of a framework which uses self-supervised estimates of optical flow combined with either a likelihood-cluster assignment framework or an active Generator-Inpainter network. In the case a neural network is to be used, spiking neural networks are the recommended choice, since they naturally fit with the neuronal data structure of events. At this stage it is not yet possible which of the two methods might bring the most accurate results, nevertheless motion compensation will play a central role in both methods as it is currently the most promising self-supervision method.

References

[1] A. Bobick and J. Davis. "An appearance-based representation of action". In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 1. 1996, 307–312 vol.1. DOI: 10.1109/ICPR.1996.546039.

[2] Tobias Brosch, Stephan Tschechne, and Heiko Neumann. "On event-based optical flow detection". In: *Frontiers in neuroscience* 9 (Apr. 2015), p. 137. DOI: 10.3389/fnins.2015.00137.

[3] Andres Bruhn, Joachim Weickert, and Christoph Schnörr. "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods". In: *International Journal of Computer Vision* 61 (Feb. 2005), pp. 211–231. DOI: 10.1023/B:VISI.0000045324.43199.43.

[4] P. Charbonnier et al. "Two deterministic half-quadratic regularization algorithms for computed imaging". In: *Proceedings of 1st International Conference on Image Processing*. Vol. 2. 1994, 168–172 vol.2. DOI: 10.1109/ICIP.1994.413553.

[5] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. "Focus Is All You Need: Loss Functions for Event-Based Vision". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12272–12281. DOI: 10.1109/CVPR.2019.01256.

[6] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3867–3876. DOI: 10.1109/CVPR.2018.00407.

[7] Guillermo Gallego et al. "Event-based Vision: A Survey". In: (Apr. 2019).

[8] Guillermo Gallego et al. "Event-Based, 6-DOF Camera Tracking from Photometric Depth Maps". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.10 (2018), pp. 2402–2412. DOI: 10.1109/TPAMI.2017.2769655.

[9] Liyue Ge et al. "Optical Flow Estimation from Layered Nearest Neighbor Flow Fields". In: Oct. 2018, pp. 1–6. DOI: 10.1109/CISP-BMEI.2018.8633120.

[10] Mathias Gehrig et al. "Event-Based Angular Velocity Regression with Spiking Networks". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 4195–4202. DOI: 10.1109/ICRA40945.2020.9197133.

[11] Jesse Hagenaars, Federico Paredes-Valles, and Guido de Croon. "Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 7167–7179. URL: https://proceedings.neurips.cc/paper/2021/file/39d4b545fb02556829aab1db805021c3-Paper.pdf.

[12] S. Khan and M. Shah. "Object based segmentation of video using color, motion and spatial information". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 2. 2001, pp. II–II. DOI: 10.1109/CVPR.2001.991039.

[13] Hanme Kim, Stefan Leutenegger, and Andrew Davison. "Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera". In: vol. 9910. Oct. 2016, pp. 349–364. ISBN: 978-3-319-46465-7. DOI: 10.1007/978-3-319-46466-4_21.

[14] Hanme Kim et al. "Simultaneous Mosaicing and Tracking with an Event Camera". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014. DOI: http://dx.doi.org/10.5244/C.28.26.

[15] Jürgen Kogler, Christoph Sulzbachner, and Wilfried Kubinger. "Bio-inspired Stereo Vision System with Silicon Retina Imagers". In: Oct. 2009, pp. 174–183. ISBN: 978-3-642-04666-7. DOI: 10.1007/978-3-642-04667-4_18.

[16] Jürgen Kogler et al. “Address-Event Based Stereo Vision with Bio-Inspired Silicon Retina Imagers”. In: *Advances in Theory and Applications of Stereo Vision*. Ed. by Asim Bhatti. Rijeka: IntechOpen, 2011. Chap. 9. DOI: 10.5772/12941. URL: https://doi.org/10.5772/12941.

[17] M. Kumar, Philip Torr, and A. Zisserman. “Learning Layered Motion Segmentations of Video”. In: *International Journal of Computer Vision* 76 (Mar. 2008), pp. 301–319. DOI: 10.1007/s11263-007-0064-x.

[18] A. K. S. Kushwaha et al. “Adaptive real-time motion segmentation technique based on statistical background model”. In: *The Imaging Science Journal* 62.5 (2014), pp. 285–302. DOI: 10.1179/1743131X13Y.0000000056. eprint: https://doi.org/10.1179/1743131X13Y.0000000056. URL: https://doi.org/10.1179/1743131X13Y.0000000056.

[19] Siyang Li et al. “Instance Embedding Transfer to Unsupervised Video Object Segmentation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6526–6535. DOI: 10.1109/CVPR.2018.00683.

[20] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. “A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor”. In: *IEEE Journal of Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[21] Min Liu and Tobi Delbruck. “Adaptive Time-Slice Block-Matching Optical Flow Algorithm for Dynamic Vision Sensors”. In: Sept. 2018.

[22] Francesco Locatello et al. “Object-Centric Learning with Slot Attention”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 11525–11538. URL: https://proceedings.neurips.cc/paper/2020/file/8511df98c02ab60aea1b2356c013bc0f-Paper.pdf.

[23] Xiankai Lu et al. “Learning Video Object Segmentation From Unlabeled Videos”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 8957–8967. DOI: 10.1109/CVPR42600.2020.00898.

[24] Xiankai Lu et al. “See More, Know More: Unsupervised Video Object Segmentation With Co-Attention Siamese Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3618–3627. DOI: 10.1109/CVPR.2019.00374.

[25] Jana Mattheus, Hans Grobler, and Adnan M. Abu-Mahfouz. “A Review of Motion Segmentation: Approaches and Major Challenges”. In: *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*. 2020, pp. 1–8. DOI: 10.1109/IMITEC50163.2020.9334076.

[26] Anton Mitrokhin et al. “Event-Based Moving Object Detection and Tracking”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8593805.

[27] Anton Mitrokhin et al. “Learning Visual Motion Segmentation Using Event Surfaces”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 14402–14411. DOI: 10.1109/CVPR42600.2020.01442.

[28] Simon Niklaus and Feng Liu. “Softmax Splatting for Video Frame Interpolation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5436–5445. DOI: 10.1109/CVPR42600.2020.00548.

[29] Yuji Nozaki and Tobi Delbruck. “Authors’ Reply to Comment on “Temperature and Parasitic Photocurrent Effects in Dynamic Vision Sensors””. In: *IEEE Transactions on Electron Devices* 65.7 (2018), pp. 3083–3083. DOI: 10.1109/TED.2018.2841205.

[30] Yuji Nozaki and Tobi Delbruck. “Temperature and Parasitic Photocurrent Effects in Dynamic Vision Sensors”. In: *IEEE Transactions on Electron Devices* 64.8 (2017), pp. 3239–3245. DOI: 10.1109/TED.2017.2717848.

[31] Chethan M. Parameshwara et al. “0-MMS: Zero-Shot Multi-Motion Segmentation With A Monocular Event Camera”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9594–9600. DOI: 10.1109/ICRA48506.2021.9561755.

[32] Chethan M. Parameshwara et al. “SpikeMS: Deep Spiking Neural Network for Motion Segmentation”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 3414–3420. DOI: 10.1109/IROS51168.2021.9636506.

[33] Deepak Pathak et al. “Learning Features by Watching Objects Move”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6024–6033. DOI: 10.1109/CVPR.2017.638.

[34] Christian Reinbacher, Gottfried Munda, and Thomas Pock. “Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation”. In: *International Journal of Computer Vision* 126 (Dec. 2018). DOI: 10.1007/s11263-018-1106-2.

[35] Ioannis M. Rekleitis. “Cooperative Localization and Multi-Robot Exploration”. http://www.cim.mcgill.ca/~yiannis/Publications/thesis.pdf. PhD thesis. Montreal, Quebec, Canada: School of Computer Science, McGill University, Feb. 2003.

[36] Nitin J. Sanket et al. “EVDodgeNet: Deep Dynamic Obstacle Dodging with Event Cameras”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10651–10657. DOI: 10.1109/ICRA40945.2020.9196877.

[37] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. “Asynchronous Spatial Image Convolutions for Event Cameras”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 816–822. DOI: 10.1109/LRA.2019.2893427.

[38] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. “Continuous-time Intensity Estimation Using Event Cameras”. In: *Asian Conf. Comput. Vis. (ACCV)*. Dec. 2018, pp. 308–324. DOI: 10.1007/978-3-030-20873-8_20.

[39] Sahir Shrestha et al. “Learning To Segment Dominant Object Motion From Watching Videos”. In: *2021 Digital Image Computing: Techniques and Applications (DICTA)*. 2021, pp. 01–08. DOI: 10.1109/DICTA52665.2021.9647227.

[40] Bongki Son et al. “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)*. 2017, pp. 66–67. DOI: 10.1109/ISSCC.2017.7870263.

[41] Timo Stoffregen et al. “Event-Based Motion Segmentation by Motion Compensation”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 7243–7252. DOI: 10.1109/ICCV.2019.00734.

[42] David Weikersdorfer and Jörg Conradt. “Event-based particle filtering for robot self-localization”. In: *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2012, pp. 866–870. DOI: 10.1109/ROBIO.2012.6491077.

[43] Li Xu, Jianing Chen, and Jiaya Jia. “A Segmentation Based Variational Model for Accurate Optical Flow Estimation”. In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–684. ISBN: 978-3-540-88682-2.

[44] Charig Yang et al. “Self-supervised Video Object Segmentation by Motion Grouping”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 7157–7168. DOI: 10.1109/ICCV48922.2021.00709.

[45] Fan Yang et al. “Multi-motion and Appearance Self-Supervised Moving Object Detection”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2101–2110. DOI: 10.1109/WACV51458.2022.00216.

[46] Yanchao Yang et al. “Unsupervised Moving Object Detection via Contextual Information Separation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 879–888. DOI: 10.1109/CVPR.2019.00097.

[47] Yongxin Yang, Irene Morillo, and Timothy Hospedales. *Deep Neural Decision Trees*. June 2018.

[48] Vickie Ye et al. “Deformable Sprites for Unsupervised Video Decomposition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 2657–2666.

[49] Luca Zappella, Xavier Llado, and Joaquim Salvi. “Motion Segmentation: a Review”. In: vol. 184. Jan. 2008, pp. 398–407. DOI: 10.3233/978-1-58603-925-7-398.

[50] Luca Zappella et al. “Enhanced Local Subspace Affinity for feature-based motion segmentation”. In: *Pattern Recognition* 44 (Feb. 2011), pp. 454–470. DOI: 10.1016/j.patcog.2010.08.015.

[51] Xi Zhang et al. “Layered Optical Flow Estimation Using a Deep Neural Network with a Soft Mask”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI’18. Stockholm, Sweden: AAAI Press, 2018, pp. 1170–1176. ISBN: 9780999241127.

[52] Yun Zhang, Bin Luo, and Liangpei Zhang. “Permutation Preference Based Alternate Sampling and Clustering for Motion Segmentation”. In: *IEEE Signal Processing Letters* 25.3 (2018), pp. 432–436. DOI: 10.1109/LSP.2017.2777997.

[53] Alex Zihao Zhu et al. “Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 989–997. DOI: 10.1109/CVPR.2019.00108.