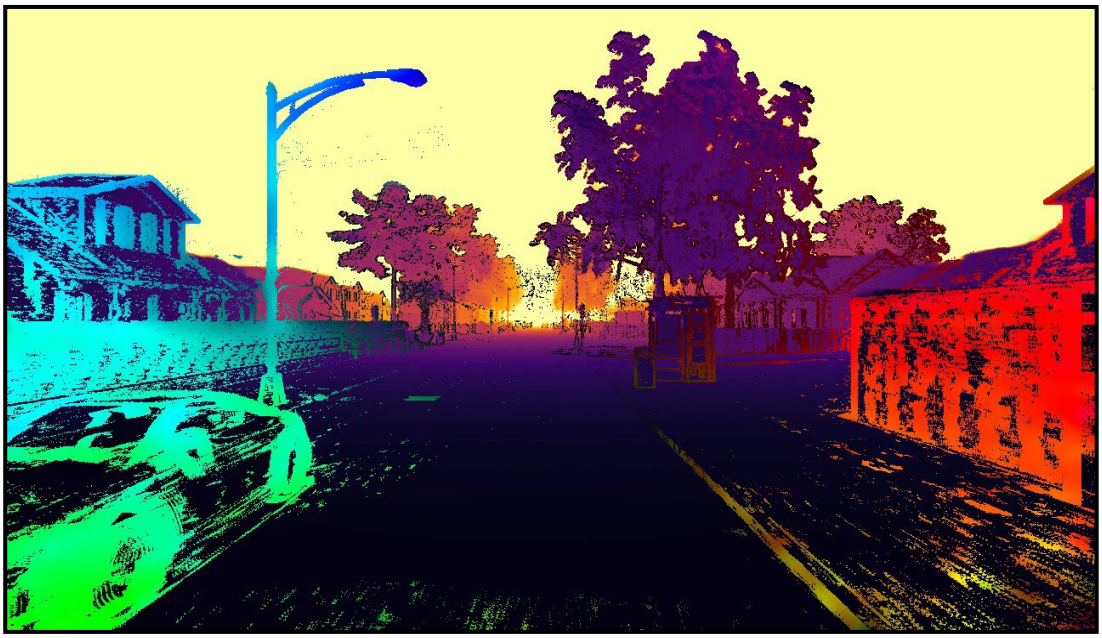
Multimodal estimation of movement and depth based on events for scene analysis

Vincent Brebion

Par **Vincent BREBION**

Multimodal estimation of movement and depth based on events for scene analysis

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 11 janvier 2024
Spécialité : Sciences et Technologies de l'Information et des Systèmes : Unité de recherche Heudyasic (UMR-7253)
D2795

THESE de DOCTORAT

MULTIMODAL ESTIMATION OF MOVEMENT AND DEPTH BASED ON EVENTS FOR SCENE ANALYSIS

Par **Vincent BREBION**

Spécialité :
Sciences et Technologies de l'Information et des Systèmes

Université de Technologie de Compiègne
Heudiasyc Lab, UMR CNRS 7253

Soutenue le 11 janvier 2024

Jury :

REVIEWERS	Guillermo Gallego	TU Berlin
	Vincent Lepetit	École des Ponts ParisTech
EXAMINERS	Rémi Boutteau	Univ. de Rouen Normandie
	Véronique Cherfaoui	Univ. de Technologie de Compiègne
	Jean Martinet	Univ. Nice Sophia Antipolis
SUPERVISORS	Franck Davoine	CNRS
	Julien Moreau	Univ. de Technologie de Compiègne

The *Smart Thesis* template used in this document was written by Jan Philip Göpfert and Andreas Stöckel, and was inspired by the *Classic Thesis* template developed by André Miede.

The source code of this document is available at `https://vbrebion.github.io/PHD`.

This work was supported in part by the Hauts-de-France Region and in part by the SIVALab Joint Laboratory (Renault Group - UTC - CNRS).

The night is long
And the path is dark
Look to the sky
For one day soon
The dawn will come

— EXCERPT FROM "THE DAWN WILL COME"
BY TREVOR MORRIS

ABSTRACT

With their asynchrony and independence to illumination conditions, event cameras open new perception capabilities. They allow for the analysis of highly dynamic scenes with complex lighting, a situation in which traditional frame-based cameras show their limits. In the context of this thesis, two low-level perception tasks were examined in particular, as they constitute the foundation of many higher level tasks required for scene analysis: (1) optical flow and (2) depth estimation.

In the case of optical flow, an optimization-based approach was developed, allowing for the estimation of optical flow in real-time with a single high-resolution event camera. Short temporal windows of events are converted into frame-like representations, with denoising and with a novel negated exponential densification applied. A state-of-the-art low-latency frame-based optical flow method is then used to compute the final optical flow. This heuristic approach provides accurate results, and is still to this day the only event-based optical flow method working in real-time with high-resolution event cameras.

As for the depth estimation, a learning-based data-fusion method between a LiDAR and an event camera was proposed for estimating dense depth maps. For that purpose, a convolutional neural network was proposed, named ALED. It is composed of two separate asynchronous encoding branches for the LiDAR point clouds and for the events, central memory units where the asynchronous fusions are applied, and a final decoding branch. A novel notion of "two depths per event" was also proposed, with a theoretical analysis as to why this notion is fundamental given the change-based nature of events. A simulated dataset was finally proposed, containing high-resolution LiDAR and event data, as well as perfect depth maps used as ground truth. Compared to the state of the art, an error reduction of up to 61% was achieved, demonstrating the quality of the network and the benefits brought by the use of our novel dataset.

An extension to this depth estimation work was also proposed, this time using an attention-based network for a better modeling of the spatial and temporal relations between the LiDAR and the event data. Initial experiments were conducted on a fully sparse network, able to directly output the two depths for each event without relying on a dense representation, but both theoretical and technical limitations were met. A subsequent rework of the method, this time on dense inputs and outputs, allowed us to overcome these limitations. The proposed network, DELTA, is both recurrent and attention-based. It is composed of two encoding branches for the LiDAR point clouds and the events, a propagation mechanism for inferring LiDAR data at a higher rate, a central memory unit where the fusion between the two modalities is applied, and a final decoding branch. Compared to ALED, DELTA is able to improve results across all metrics, and especially for short ranges (which are the most critical for robotic applications), where the average error is reduced up to four times.

RÉSUMÉ

Grâce à leurs propriétés uniques en termes d'asynchronisme et d'indépendance aux conditions de luminosités, les caméras à évènements ouvrent aujourd'hui de nouvelles portes dans le monde de la perception. Elles rendent possible l'analyse de scènes hautement dynamiques et avec un éclairage complexe, des situations pour lesquelles les caméras traditionnelles reposant sur des images montrent leurs limites. Dans le cadre de cette thèse, deux tâches de perception bas niveau ont été considérées en particulier, car constituant la fondation de nombreuses tâches de plus haut niveau requises pour l'analyse de scènes : (1) le flot optique et (2) l'estimation de profondeur.

En ce qui concerne le flot optique, une approche basée optimisation a été développée, permettant l'estimation de flot optique en temps réel avec une unique caméra à évènements haute définition. Pour cela, de courtes fenêtres temporelles d'évènements sont converties vers une représentation dense basée image, après application d'une étape de débruitage, et d'une densification inversement exponentielle proposée dans le cadre de ce travail. Une méthode de flot optique de l'état de l'art basée images est ensuite appliquée afin de calculer le flot optique final avec une latence basse. Cette approche heuristique permet de fournir des résultats justes, et est à ce jour la seule méthode de flot optique basée évènements capable d'opérer en temps réel avec des caméras à évènements haute définition.

Pour ce qui est de l'estimation de profondeur, une méthode basée apprentissage pour de la fusion de données a été proposée, permettant de combiner les informations provenant d'un LiDAR et d'une caméra à évènements afin d'estimer des cartes de profondeur denses. Dans le cadre de ce travail, un réseau de neurones à convolution, appelé ALED, a été proposé. Il est composé de deux branches d'encodage asynchrones pour les nuages de points LiDAR et les évènements, de mémoires centrales où la fusion asynchrone des deux types de données est réalisée, et d'une branche de décodage. En particulier, une nouvelle notion de "deux profondeurs par évènement" a également été proposée, accompagnée d'une analyse théorique sur l'importance fondamentale de cette notion à cause du fait que les évènements soient indicatifs d'un changement. Enfin, un jeu de données enregistré en simulation a également été proposé, contenant des données LiDAR et évènements haute définition, ainsi que des cartes de profondeur servant de vérité terrain. En comparaison avec l'état de l'art, une réduction jusqu'à 61% de l'erreur moyenne a pu être atteinte, démontrant la qualité de notre réseau et des bénéfices apportés par l'utilisation de notre nouveau jeu de données.

Une extension de ce travail sur l'estimation de profondeur a également été proposée, utilisant cette fois-ci un réseau de neurones basé attention pour une meilleure modélisation des relations spatiales et temporelles entre les données LiDAR et évènements. Des expérimentations ont été menées dans un premier temps dans l'objectif de proposer un réseau entièrement épars, capable d'associer directement à chaque évènement ses deux profondeurs, sans avoir besoin de passer par des représentations denses. À cause de limitations à la fois théoriques et techniques, une refonte de cette méthode a été proposée, cette fois-ci sur des entrées et sorties denses, afin de pouvoir s'affranchir de

ces limitations. Le réseau final proposé dans le cadre de ce travail, DELTA, combine à la fois un aspect récurrent et une approche basée attention. Il est composé de deux branches d'encodage pour les nuages de points LiDAR et les évènements, d'un mécanisme de propagation afin d'être capable d'inférer les données LiDAR a une plus haute fréquence que celle d'entrée, d'une unique mémoire centrale pour la fusion des modalités, et d'une branche de décodage. En comparaison avec ALED, DELTA améliore les résultats pour l'ensemble des métriques considérées. Cette amélioration est particulièrement prononcée pour les distances courtes (qui constituent les distances les plus critiques pour des applications robotiques), avec une erreur moyenne jusqu'à quatre fois moins importante.

ACKNOWLEDGEMENTS

Before diving into the main content of this thesis, I would like to thank all the people who have made this work flourish and become what it is today.

First and foremost, I would like to thank Franck and Julien for supervising this thesis, and for guiding me as they have done over the past three and a half years. Our theoretical and technical discussions, as well as our numerous drawings on whiteboards during our weekly meetings have been the main contributing factors of this thesis.

Secondly, I would like to thank Guillermo Gallego and Vincent Lepetit for reviewing this thesis, and for sending me their precious comments to improve it. I would also like to thank Rémi Boutteau, Véronique Cherfaoui, and Jean Martinet for examining this work, as well as Pascal Vasseur and Philippe Xu for providing an external view on my work across these three years.

Thirdly, I would like to thank all the fellow PhD students at Heudiasyc I have had the privilege of working with during all my time at the lab: Maxime & Maxime, Ali, Antoine, Matthieu, Rémy, Loïc, and Michaël. I would also like to thank Thierry and Stéphane, for their precious technical assistance and availability. More generally, I would like to thank all the members and personnel of Heudiasyc, SIVALab, and UTC, whom it has always been a pleasure to work with.

Finally, I would like to thank from the bottom of my heart my parents and my brother Benoît, for constantly giving me precious advices, strength, and emotional support over all these years.

CONTENTS

Abstract **v**

Résumé **vii**

Acknowledgements **ix**

Contents **xi**

1 General Introduction **1**
1.1 Context 1
1.2 Objectives and Overview of the Thesis 4
1.3 Thesis Overview 5

2 Event Cameras **7**
2.1 Introduction 7
2.2 Core Principle 7
2.3 Advantages and Challenges 10
2.4 Application to Intelligent Robotics 12
2.5 Review of the State of the Art 12
2.6 Popular Event Camera Models 16
2.7 Datasets 17

3 Real-Time Event-Based Optical Flow **21**
3.1 Introduction 21
3.2 Problem Formulation 22
3.3 Related Work 22
3.4 Our Orientation: Densifying Events 24
3.5 Method 25
3.6 Evaluation 33
3.7 Conclusion and Discussions 47

4 Event- and LiDAR-Based Depth Estimation using a Convolutional Network **49**
4.1 Introduction 49
4.2 Related Work 50
4.3 Depth Change Map: Two Depths per Event 51
4.4 Method 53
4.5 The SLED Dataset 57
4.6 Evaluation 59
4.7 Conclusion and Discussions 66

5 Event- and LiDAR-Based Depth Estimation using an Attention-Based Network **69**

5.1 Introduction 69
5.2 An Introduction to the Transformer and Attention 70
5.3 Related Work 73
5.4 Predicting Sparse Depths with Transformers 74
5.5 Dense DELTA Method 80
5.6 Evaluation 83
5.7 Conclusion and Discussions 93

6 General Conclusion **95**
6.1 Conclusion 95
6.2 Contributions 96
6.3 Discussions and Perspectives 98

A Additional Experiments **101**
A.1 Acquisition of Real-World Data 101
A.2 Extensions to the SLED Dataset 105

B Additional Results for Chapter 4 **109**
B.1 Detailed Results on our SLED Dataset 109
B.2 Additional Dense Depths Results on our SLED Dataset 109
B.3 Additional Dense Depths Results on the MVSEC Dataset 109
B.4 Additional Depth Change Maps Results on our SLED Dataset 109

C Additional Results for Chapter 5 **119**
C.1 Detailed Results on our SLED Dataset 119
C.2 Additional Visual Results on the SLED Dataset 119
C.3 Additional Visual Results on the MVSEC Dataset 119
C.4 Additional Visual Results on the M3ED Dataset 119

Glossary **129**

Acronyms **131**

Bibliography **133**

1 GENERAL INTRODUCTION

1.1 CONTEXT

1.1.1 *General Context*

From the simplest robotic vacuum cleaner to the Perseverance rover on Mars, scene analysis is one of the most fundamental requirements in robotics. In the traditional robotic pipeline illustrated in Fig. 1.1, perception constitutes the very first module, dedicated to sensing, analyzing, and understanding the world for navigating safely.

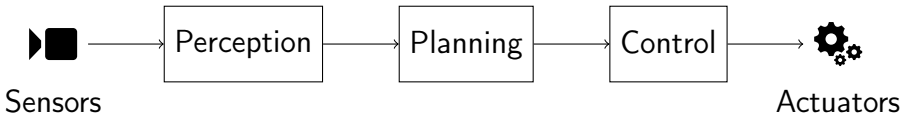


Figure 1.1 – The traditional robotic pipeline, from sensors to actuators.

In that sense, motion understanding is critical for perceiving how every element of the scene evolves. This motion understanding actually includes two components: **(1)** the ego-motion, whose extraction is critical for asserting the ego-position in applications like visual odometry and SLAM, and **(2)** the motion of every other mobile object in the scene, which in most cases are potential obstacles, that should therefore be avoided; knowing their motion allows for the anticipation of their actions and future state, and therefore for the avoidance of future collisions by taking preventive actions.

However, for evolving in dynamic scenes, motion is of limited use without any knowledge of depth. Being able to determine that a given object is moving every second by a certain amount of pixels in a given direction is interesting information, but without a three-dimensional metric equivalence, taking adequate preventive actions is complex. A simple example of this phenomenon is shown in Fig. 1.2: while in the top row, both cubes appear to be moving at the same speed towards the center of the image, the 3D view in the bottom row actually shows that the green cube is actually positioned closer to the camera, and that it should therefore be given priority for the following "planning" and "control" modules of the pipeline.

In the context of this thesis, we are therefore interested in the estimation of both motion and depth. For that purpose, the event camera constitutes an interesting starting point, as it naturally encodes changes in the scene it observes, changes which are mostly caused by motion in dynamic scenes. Yet, while a single event camera might suffice for this estimation of motion, the addition of a second modality is a requirement for lifting any ambiguity related to the depth as shown in Fig. 1.2. Therefore, as part of this thesis, we will also consider the LiDAR sensor, for providing sparse but accurate measurements of the depth of the scene. By using both the

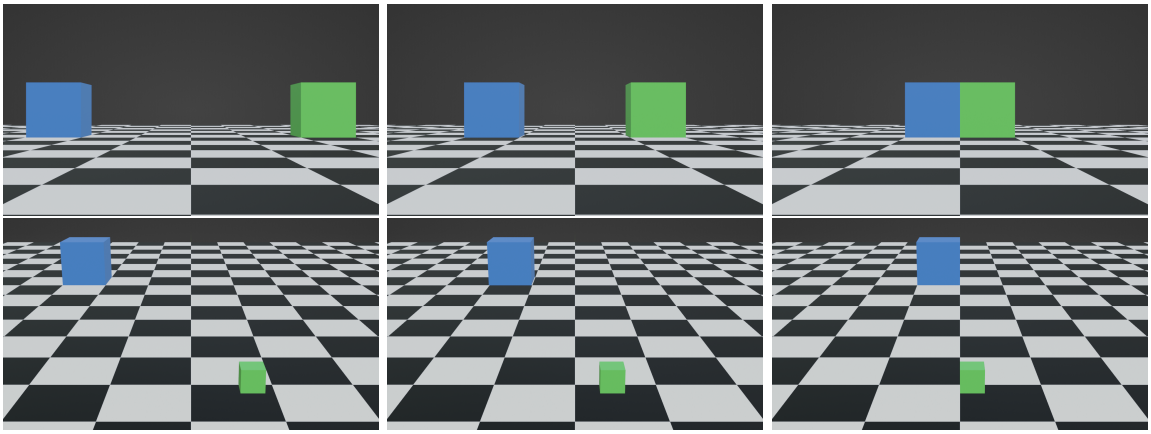


Figure 1.2 – Illustration of the importance of depth for scene analysis. Top row: two cubes moving towards the center of the image, with the same apparent velocity. Bottom row: the very same scene, this time viewed from a higher point of view; both cubes are in reality far from each other, with the green cube being closer, much smaller, floating above ground, and moving towards the center at a slower pace than the blue cube.

event camera and the LiDAR, we aim to exploit and combine the unique capabilities of both sensors, and cancel out their weaknesses: in case of the event camera, a very-high-speed low-latency sensing of the scene, but with a vision of only textures and edges of objects with relative motion, and in the case of the LiDAR, very precise depth measurements, but at a low-rate and in a sparse way with few points at the edges of objects.

1.1.2 *The Event Camera*

Initially proposed in 1991 by Mahowald and Mead [1, 2], the event camera has since grown and is slowly becoming more recognized in the field of computer vision. Compared to a traditional camera which accumulates light during short periods of time to create images, event camera only detect instantaneous changes of light for each pixel asynchronously, and transmits them as spikes of information with a minimal latency. This novel sensing principle constitutes a fundamental change of paradigm, as it allows for very-high-frequency observations with a high dynamic range and a low-latency, and therefore opens new perception opportunities that were never thought of due to the limitations of more traditional cameras. This new paradigm, however, comes with a fundamental drawback: the need to rethink the whole bibliography, for adapting it to this novel sensor.

As of today, event cameras have found their way in numerous and diverse domains. At the lowest, micrometer scale, event cameras have been applied for controlling micro-robots to manipulate cells [3]: the event camera gives the manipulator a finer control of the robot, and allows for a better measurement of interaction forces that can be fed back to the operator. Event cameras are also expected to make a breakthrough in the smartphone domain: their low energy consumption, their high dynamic range, and their high speed makes them perfect candidates for handheld photography and video recording. At the human scale, event cameras are mainly

applied for industrial automation (fault detection [4], fast object counting [5]), autonomous driving [6], and more specific applications like the measurement of fluid flows [7]. Finally, at the highest scale, event cameras are also used in the space domain, for either tracking stars from the ground [8], or for detecting debris in Earth orbit thanks to an event camera launched in space in 2021 [9].

Due to their central position in this thesis, Chapter 2 is dedicated to an in-depth explanation of the working principles and specificities of the event cameras.

1.1.3 *Application to the Automotive Domain*

One of the field of application of our methods that will be of special interest to us in this thesis is the automotive domain. As transportation remains at the heart of our daily lives, and despite progress on legislation, road safety, and vehicle standards, approximately 1.3 million people still die every year due to a road traffic crash [10]. Intelligent road vehicles aim to reduce this number of crashes by reducing or completely eliminating the human factor.

However, autonomous driving in open environments calls for deep understanding abilities from the intelligent vehicle to make it able to navigate safely. While recent progress has been made on that topic, most of the results from the literature were achieved in favorable conditions (adequate lighting and weather, clearly visible objects), which only represent a fraction of the real-life situations a driver is confronted with. Recent studies have particularly shown the limits of these approaches in more complex conditions (at dawn/dusk, when the object to detect is partially occluded or very close to the ego-vehicle, . . .), raising multiple safety questions [11, 12].

In that context, the use of a multimodal sensing system appears to be a critical component for proposing a safe autonomous navigation in all environmental conditions. As described in Section 1.1.2, the event camera constitutes an interesting candidate for supplementing traditional RGB cameras, but even more so in the automotive context. Being able to output similarly-looking information in broad daylight and during night, and not being dazzled by sudden changes of lighting (such as at the entrance or exit of a tunnel), are both critical components for ensuring the operability of the perception module of the vehicle in a wider range of conditions. Their low-latency also constitutes an appealing argument in their favor, as being able to detect and identify obstacles more quickly gives the "planning" and "control" blocks of the robotic pipeline more time to adjust the trajectory of the vehicle accordingly. Our use of the LiDAR is also linked to this automotive context: while an RGB-D camera for instance would provide denser depth measurements, its use would be restricted to indoor situations. In comparison, a LiDAR can operate independently both indoor and outdoor and can sense the world with a long range, and constitutes therefore a common sensor used in intelligent vehicles.

1.2 OBJECTIVES AND OVERVIEW OF THE THESIS

As noted in Section 1.1, the subject of this thesis is composed of a dual objective: the estimation of motion and the estimation of depth. For proposing answers to these issues, two specific problems are of particular interest to us: **(1)** optical flow and **(2)** dense depth maps prediction.

Optical flow constitutes a widely explored subject in computer vision. While mostly solved for frame-based vision, it remains a challenging topic in event-based vision. When this thesis began in 2020, event-based optical flow accuracy was still limited, with learning-based methods only starting to become the state of the art. One limitation that interested us in particular was the lack of any real-time[1] event-based optical flow method for high-resolution cameras. These cameras, like the Prophesee Gen4 [13], were slowly becoming available, but all the state-of-the-art optical flow methods we tested were painstakingly slow and were producing mostly inaccurate results due to the changes in resolution and camera specificities. Therefore, the first year of the thesis was spent on proposing a novel method for treating events, in order to be able to estimate an accurate event-based optical flow in real-time and with a low latency.

As an alternative to 2D optical flow, 3D scene flow could have been treated for motion estimation instead. As a main advantage, scene flow would have solved the ambiguities a more simple 2D optical flow can present, as illustrated before in Fig. 1.2. But this approach would have already required more than a single event camera, and would have been a complex subject to undertake especially given the lack of event-based datasets on that subject.

As for dense depth maps prediction, being able to obtain a dense view of the depth of all objects in a scene is precious when evolving in said scene. This problem has already been explored with frame-based and event-based cameras (either in a monocular or stereo manner), and with LiDARs (with a LiDAR alone or with frames acting as a densification guide). When our thesis work started on this depth estimation problem in mid-2021, only a single work [14] had explored the fusion of LiDAR point clouds and events, with several limitations (only sparse depths were estimated, limited to the vertical range of the LiDAR sensor, and evaluation was only conducted on a private dataset). The idea of estimating dense depth maps from LiDAR and event data had not been treated until that point, and presented a valuable research opportunity. Therefore, the second and third year of the thesis were spent on proposing two novel methods for fusing LiDAR and event data, in order to be able to estimate dense depth maps at a high rate and with a high accuracy.

It should also be noted that, as part of this thesis, a collaboration between the Heudiasyc laboratory and Prophesee has been put in place, for deeper exchanges both on the theoretical and technical points of view.

1 Throughout this document, we will use the term "real-time" for any event-based method that verifies the two following criterions: **(1)** being able to process all incoming events from a live camera even under a high throughput without discarding any of them, and **(2)** if events are accumulated over a time window Δt, being able to process them in a time lower than Δt.

1.3 THESIS OVERVIEW

This thesis is split in four main chapters, which cover the work conducted over the three years of this thesis.

In Chapter 2, we begin by giving an overview of the event camera. Due to its central position in this thesis, and due to some concepts needing a formal introduction as they are fundamentally different from those of frame-based cameras, we give in this chapter a detailed description of its working principle, its advantages and its challenges, and offer a quick review of the state of the art, the camera models, and the datasets.

In Chapter 3, we describe our first work, on the "motion" aspect of the thesis, by proposing a fully event-based and model-based optical flow method. We describe a pipeline architecture, able in real-time and with a low latency to process events, transform them into a dense frame-based intermediate representation, and compute optical flow using a state-of-the-art frame-based method. We show in particular that our intermediate representation is critical for improving the accuracy of the optical flow results for both low-, mid-, and high-resolution event-based sensors, thanks to the use of a proposed negated exponential distance transform formulation.

In Chapter 4, we describe our second work, this time on the "depth" aspect of the thesis. We investigate in this fourth chapter the fusion of high-rate events with low-rate LiDAR point clouds, in order to estimate high-rate dense depth maps of the observed scene. We propose here a convolutional-based neural network, the Asynchronous LiDAR and Events Depths densification network (ALED), able to conduct this task with high accuracy. Compared to the event-based state of the art, we show that ALED is the best performing method, offering significant improvements, especially on complex automotive scenarios.

In Chapter 5, we describe our extensions to this work on depth estimation. We investigate especially how attention-based networks can better represent the relations between the LiDAR- and event-based data, and how they can improve the results obtained with ALED. An in-depth analysis on fusing fully sparse LiDAR points and events for estimating sparse depths is first conducted, with theoretical and technical limitations of attention-based networks being highlighted. A dense attention-based network, DELTA, is then proposed. We show that DELTA further improves the results of ALED, especially for close ranges.

Finally, in Chapter 6, we draw some general conclusions to this thesis, and discuss potential extensions to our work.

2 EVENT CAMERAS

2.1 INTRODUCTION

Event cameras — also known as neuromorphic cameras, dynamic vision sensors, or silicon retinas — are novel, biologically-inspired sensors. Compared to traditional cameras, the specific architecture of the event cameras allows them to perceive highly dynamic scenes with low latency and with high dynamic range. Thanks to these properties, they represent a sensor of interest to answer the limitations encountered by the use of more traditional sensors in complex conditions.

Even though the first concept of an event camera was proposed over 30 years ago [1, 2], this sensor has gained traction within the scientific community over the past decade. This rise in popularity is especially due to their easier access, with the arrival of commercial sensors (especially all-in-one frame- and event-based sensors like the DAVIS [15]), and with a growing number of software and development kits.

Due to their central position, this first chapter of the thesis is dedicated to an in-depth review of the event camera. We explain here its core concepts, describe its advantages and challenges, and quickly review the state of the art.

2.2 CORE PRINCIPLE

In a traditional camera, an image is captured by accumulating light during a short period of time (the *exposure time*). Each pixel is then assigned an intensity value based on the amount of light received. All pixels are synchronously controlled by a single global clock, and images are outputted at a predefined frame rate (e.g., 30FPS).

In opposition, in an event camera, all pixels are independent units. Each pixel responds to changes in the log-irradiance (or "brightness") it receives. More specifically, if we note the brightness $L \doteq \log(E)$ (where E is the irradiance), then an event $e \doteq (\mathbf{x}, t, p)$ is triggered at a pixel $\mathbf{x} \doteq (x, y)$ and at time t if the brightness difference since the last event at that pixel, i.e., if

$$\Delta L(\mathbf{x}, t) \doteq L(\mathbf{x}, t) - L(\mathbf{x}, t - \Delta t) \tag{2.1}$$

crosses a threshold δ, i.e.,

$$|\Delta L(\mathbf{x}, t)| \geq \delta \tag{2.2}$$

where $\delta > 0$, and where Δt is the amount of time since the last event at that pixel. The polarity p of the event can then be defined as the sign of the brightness difference, i.e.,

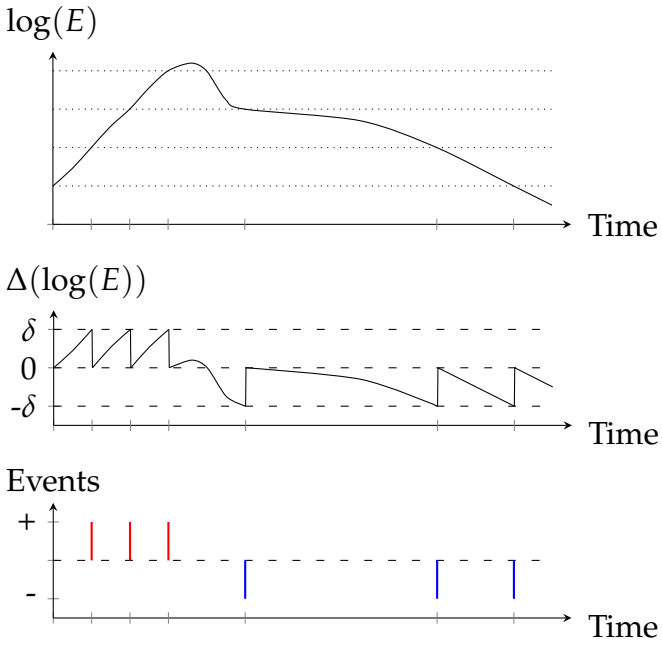


Figure 2.1 – Internal working of a single pixel of an event camera, for sample brightness values. From top to bottom: brightness received by the pixel along time; brightness difference (reset every time the threshold is hit); corresponding events fired by the pixel (positive events are in red, negative ones in blue). Figure inspired by [16].

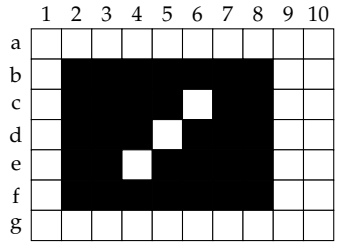
$$p \doteq \text{sgn}(\Delta L(\mathbf{x}, t)) \tag{2.3}$$

For an easier understanding, an illustration of these equations is given in Fig. 2.1.

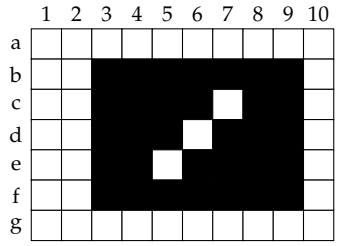
Due to these formulations, events can be triggered by two highly different reasons: **(1)** lighting changes in the observed scene or **(2)** relative motion between the camera and the objects in the scene. If both the event camera and the observed scene are fully static, then events can only be triggered by lighting changes **(1)**, generated for instance by blinking lights. On the contrary, in the more interesting case where the event camera moves or where the observed scene is dynamic (or both), then events can still be generated by lighting changes **(1)**, but also and mostly by relative motion between the camera and objects in the scene **(2)**. In case **(2)**, as shown in Fig. 2.2, only the edges and the textures of the objects will trigger events, as they are the places where the brightness values change. In contrast, untextured areas have nearly constant brightness values across them, and thus do not trigger events.

Furthermore, the output rate of a pixel (and therefore, of the whole event camera) is highly variable. In case of no or very few changes in the scene, the brightness values will remain almost constant, and thus little to no events will be fired. On the contrary, under highly varying light conditions or for highly dynamic scenes, the brightness values will vary greatly, leading to many events being fired.

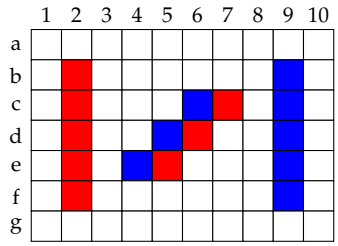
A comparison between frames and events for a same scene is given in Fig. 2.3, demonstrating all these principles for a practical example.



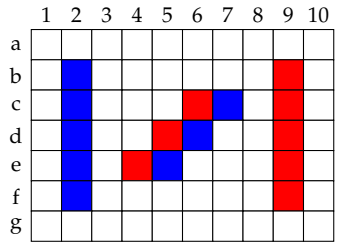
(a) Black rectangle on the left side



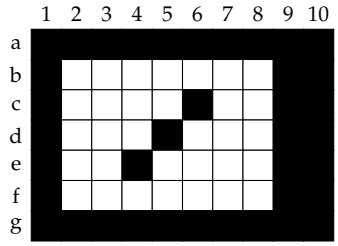
(b) Black rectangle on the right side



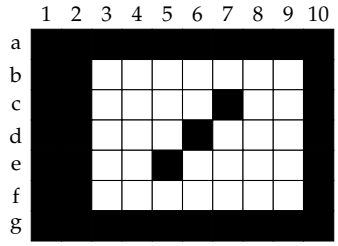
(c) Events from (a) to (b)



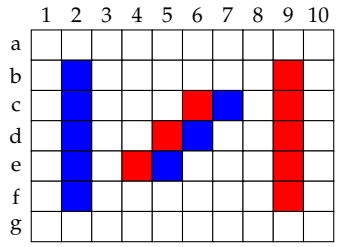
(d) Events from (b) to (a)



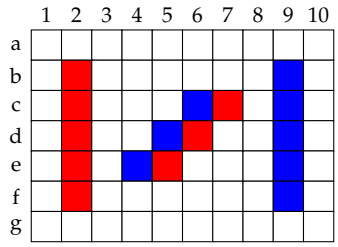
(e) White rectangle on the left side



(f) White rectangle on the right side



(g) Events from (e) to (f)



(h) Events from (f) to (e)

Figure 2.2 – Simple "moving rectangle" example. Each cell in the four grids represents a pixel. (a) A simple black rectangle on a white background, with an oblique line in its center. (b) The same rectangle, one pixel to the right. (c) The events generated due to internal motion and/or camera motion, from the (a) to the (b) situation (positive events are in red, negative ones in blue); notice how only the central oblique line and the left and right edges trigger events, as they are the only places where brightness values changed. (d) Like (c), but for the inverse motion; notice how the polarities have been inverted. (e) to (h): like (a) to (d), but for the inverse colors; notice how the the polarities have been inverted once again.

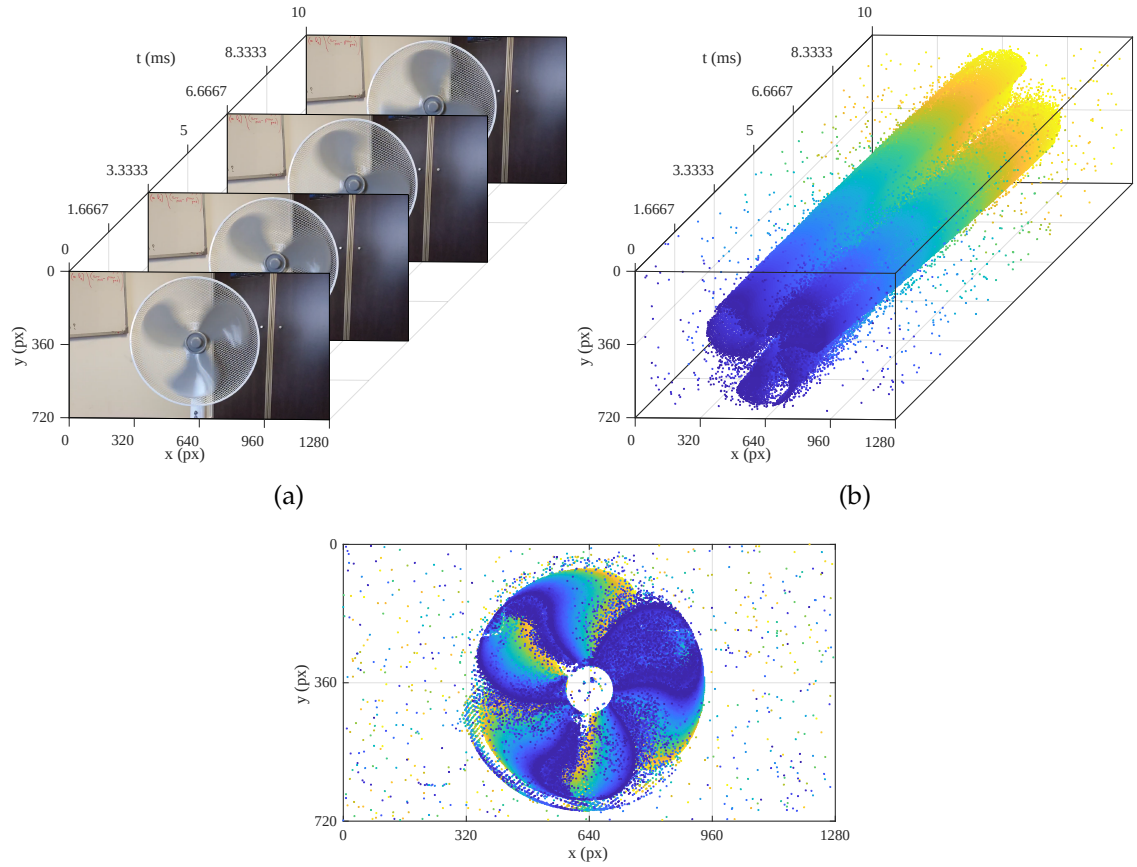


Figure 2.3 – Comparison between frames and events for a simple "Rotating fan" sequence. (a) Even with a 300FPS camera, only four frames are captured over the course of 10ms, all showcasing motion blur and motion discontinuity for the blades, as well as redundant background information. (b) In comparison, with an event camera, the full motion of the blades can be acquired: each dot is an event, color-coded according to its timestamp for a better visualization. (c) A front view of (b) is given to better showcase the motion continuity property of the event camera in a frame-like visualization.

2.3 ADVANTAGES AND CHALLENGES

Due to their unique working principle, event cameras have many advantages when compared to more traditional sensors, but they also come with some notable drawbacks.

2.3.1 *Advantages*

Asynchrony The main advantage of event cameras compared to their frame-based counterpart is that they do not rely on a synchronous sampling of the scene. Instead, brightness changes are detected and transmitted as soon as they happen. This way, the information of what is happening in the scene can be known with a very low latency. In addition, the problems of motion blur and of under-/over-exposure are

eliminated, since they are both consequences of the light accumulation process in frame-based cameras.

High Dynamic Range (HDR) Whereas most frame-based cameras are restricted to a dynamic range of 40 to 60dB, event cameras can reach a dynamic range of more than 120dB [13]. This is due to the two intrinsic properties of the event camera: working in the logarithmic framework, and not being limited by exposure issues since they do not rely on an accumulation process. This high dynamic range makes them able to see under very dark or very bright conditions, e.g., during nighttime or in broad daylight.

Low Energy Consumption As they only capture and transmit brightness changes, i.e., non-redundant information, most event cameras have a power consumption of under 100mW (20 times less than comparable frame-based cameras), making them particularly suited for low-power embedded systems.

2.3.2 *Challenges*

Change of Paradigm The main challenge with event cameras is that they propose a completely novel paradigm. As such, most of the frame-based literature and algorithms proposed over the past decades have to be entirely revamped. In addition, computers and GPUs are better suited to process synchronous dense matrix-based data like frames, rather than sparse and asynchronous event-based data.

Noise Since event cameras do not rely on a temporal accumulation process, they are much more sensitive to noise than frame-based cameras. This will result in events being triggered without a corresponding brightness change, or on the contrary, events being missed. This noise comes from different sources: shot noise, internal circuits noise, hot pixels, etc.

Output Rates As noted earlier, the output rate of an event camera is highly variable. While the output rate of a frame-based camera is predetermined (up to 300FPS for high-speed cameras), event cameras can reach output rates of up to millions of events per sec for HD sensors. Designing methods able to handle such rates in real-time is one of the most critical issues with event cameras.

Lack of Absolute Brightness Values As shown in Eqs. (2.1) and (2.2), events only provide information about relative brightness changes. No reference absolute brightness value is available, meaning that some problems like frame reconstruction from events alone are particularly difficult. To circumvent this issue, some camera models offer both the frame and event modalities in a single sensor [15, 17]. However, adding this frame modality is currently limited to low- or mid-resolution cameras, and it results in a more noisy event stream due to electrical interferences between both circuits.

2.4 APPLICATION TO INTELLIGENT ROBOTICS

As noted in the introduction to this thesis (Section 1.1), intelligent robotic systems operating in open environments are in a critical need of novel sensors and methods able to work even in the most complex conditions. Intelligent vehicles are particularly concerned, as for instance they should be able to quickly detect a pedestrian at night even on non-illuminated roads. As such, event cameras may represent a sensor of choice, as they provide answers to the most predominant issues encountered with frame-based cameras.

Independence to Lighting Conditions The high dynamic range of the event cameras allows them to observe scenes even with low or with very high illumination. In the case of intelligent vehicles, these conditions are typically met when driving during night, or when the sun is facing the windshield at dawn and dusk. A mixture of both low and high illumination can also be encountered, for instance at the exit of a tunnel, and frame-based cameras are notorious for producing bad results under these conditions.

Low Reaction Times Their asynchronous response to brightness changes and their low latency allows for the design of novel methods with very low reaction times. Such reaction times are particularly critical when operating in open environments, as external vulnerable users may always appear unexpectedly, requiring quick evasive maneuvering.

Absence of Motion Blur The lack of any motion blur is also critical, as the ego-motion of the robot or the motion of the objects in the scene does not introduce additional noise, and therefore makes their detection and identification easier.

2.5 REVIEW OF THE STATE OF THE ART

We give in this section a quick review of the current state of the art. The objective is not to be exhaustive, but rather to give the reader an overview of the work achieved so far using event cameras. More detailed reviews of the state of the art will be given in each of the following chapters of this thesis for the corresponding issues. Yet, if the reader is interested, a complete review of the domain published in 2020 is available in [16], and a more recent deep-learning-centered survey on event-based vision is available in [18].

2.5.1 *Representations*

One of the main questions in the literature is about how to represent events to treat them efficiently. We list here some of the most common representations.

Raw Stream Some authors (for instance, [19, 20, 21]) use directly the raw stream of events, and treat each of them individually, allowing for a fully event-based processing philosophy. This approach is the most conservative, as no information is lost by converting to another representation. However, it is also highly inefficient, due to the sparsity of the event data.

Time Surface The Time Surface [22, 23] is a simple representation of events as a 2D image. Constructed over a short temporal window of events, each pixel of the Time Surface is given the timestamp of the most recent event. This formulation allows for a simple conversion of event-based problems into a frame-based equivalent. However, the Time Surface is a lossy representation, as only the most recent event for each pixel is kept.

Event Volume The Event Volume [24, 25] keeps both the spatial and temporal information of events by storing them as a 3D tensor, especially adapted for learning-based approaches. While the full spatial resolution of the events is kept, timestamps are interpolated to place them in the adequate channels of the tensor, at the expense of some precision loss. In the original version of Zhu *et al.* [24], events of negative and positive polarities are added together, leading to some additional information loss. The formulation of Perot *et al.* [25] splits them in separate channels to avoid this issue, but results in a heavier representation.

Learned Representation More recently, some authors [26, 27] have started advocating for the learning of the event-based representation itself. The intrinsic idea is that, even though the Time Surface or the Event Volume are easily understandable by humans, they might not be the most adapted for neural networks. Optimizing automatically the event-based formulation allows for task-specific representations to appear, which only keep the required data. While these representations are harder to interpret, they have been proven to improve state-of-the-art performance in several tasks.

Reconstructed Images Finally, some authors [28, 29, 30] reconstruct full images from the event stream. While this process is highly inefficient, it allows for the reuse of proven methods from the frame-based state of the art, yielding accurate results in numerous tasks.

2.5.2 *Processing Strategies*

Three main strategies are used across the literature to process events: filter-based, geometry-based, and learning-based methods.

Filter-Based Methods In the spirit of keeping a fully event-based processing methodology, some authors use individual events to update a central state using a filter-based approach. This strategy allows for a fully asynchronous processing of the event stream, and is particularly well-suited to the low amount of information

each event can bring. In the case of [28], a filter-based approach is used for reconstructing images at a high rate: images from a DAVIS [15] camera are used as an optional state initialization, and events from the same camera are used to update this state to recreate subsequent images.

Model-Based Methods Model-based approaches aim at finding an optimal solution to a theoretical model of the considered problem which could be explained by the input data. In the case of event data, model-based methods are often applied on 2D images of events, using classical image processing algorithms, or directly on spatio-temporal 3D point clouds of events, using 3D-geometry-based or graph-based algorithms. For instance, Benosman *et al.* [31] used a 3D plane-fitting algorithm to estimate the motion of events with respect to time.

Learning-Based Methods As in numerous other domains, learning-based methods have revolutionized event-based computer vision. Their natural capacity of learning from data directly allows them to provide accurate results even for the most complex tasks, and allows them to circumvent limitations with more traditional model-based methods (noise, missing or ambiguous data, ...). Network architectures originally developed for frame-based computing have been proven to be also efficient for event-based tasks [32, 33], at the expense of requiring event data to be accumulated in a frame- or tensor-like representation. Spiking Neural Networks (SNNs) are an alternative approach [34, 35], which allow for a sparse processing of the event data, but which require specific hardware to be trained efficiently. More recently, Graph Neural Networks (GNNs) have started to be used with events [36, 37], as they represent explicitly relations between events while keeping their sparsity, and as they can be trained on standard GPUs.

2.5.3 *Applications*

More than 30 years after their inception [1, 2], event cameras have been applied to virtually every domain in computer vision. We list here some of their most popular applications.

Optical Flow Information encoded by an event camera is intrinsically linked to motion, as only the parts of the scenes which change along time due to motion of objects in the scene or of the camera itself create events. Therefore, optical flow constitutes one of the most explored issues in the event-based literature. While initial works were focused on sparse, per-event optical flow on limited motions [31, 38], most recent works are able to compute dense optical flow maps in all types of scenes [39, 40].

Disparity / Depth Estimation As with frame-based cameras, disparity / depth estimation is a very popular topic with event cameras. While many works use a stereo pair of event cameras [41, 42, 43], other works have tried to estimate depth

from the fusion of events with a different secondary modality (RGB [33], RGB-D [44], LiDAR [14, 45]), or from a single monocular event camera [24, 46, 47].

High Framerate HDR Video Since event cameras are able to provide relative brightness change information at a very rate and with a high dynamic range, it is technically possible to "revert" the event generation process to recreate images at a very high framerate and in HDR. While early methods [28] still required images as initial absolute brightness references, more recent methods [48, 49, 50] have been able to circumvent this requirement by using learning-based approaches.

Object Detection and Tracking Thanks to their high frequency and low latency, event cameras are a sensor of choice for detecting and tracking objects in dynamic scenes. While initial works were focused on the recognition of simple shapes [51, 52], more recent works are able to harness the power of learning-based methods to detect complex objects like vehicles or pedestrians [25, 53, 54].

2.5.4 *State-of-the-Art Datasets*

Datasets are an important requirement, both for popularizing the event camera and making recordings available to the greatest number, and for providing benchmarks for evaluating event-based methods. In this era of dominance of neural networks, large datasets are also a critical need for training these networks efficiently. We list here the three main categories of event-based datasets: real-life, simulated, and converted frame-based datasets.

Converted Frame-Based Datasets Early works [55, 56] with event cameras tried to exploit the large amount of frame-based datasets available in the literature. To do so, an event camera was placed in front of a screen, where the videos or images from these datasets were displayed. In case of static images, the event camera was manually shaken to still display events. More recent works [57, 58, 59] have tried to convert directly frame-based videos into their event-based equivalent, by extracting brightness changes along time. This method however requires a temporal upsampling of the video for more realistic timestamps, but results in the introduction of artifacts and errors compared to a real event camera.

Real-Life Datasets Following the wider availability of event cameras, numerous real-life datasets were recorded and published during the past decade. While early works were focused on simple and specific tasks (identification of playing cards [60], recognition of gestures [61], ...), more recent works have started to build long, complex, multitask datasets. Automotive datasets [25, 62, 63, 64] are especially popular, as they can showcase highly varying environmental conditions (lighting, weather, traffic, ...), and offer a wide range of problems to solve (sensor fusion, pose estimation, optical flow, object detection and recognition, semantic segmentation, ...).

Simulated Datasets While real-life datasets are particularly useful, they often have limitations: partial and/or approximate ground truths, difficulties to synchronize and calibrate the sensors, lack of variety, few edge cases, etc. In comparison, fully simulated datasets can have a perfect ground truth, synchronization, and calibration, and can place the event camera in a wider range of situations. Several simulators [65, 66, 67] were proposed along the years, relying on the frame-to-event methods, but generating traditional frames at a high rate so that no temporal upsampling of the frame-based data is required. A model of the event camera was especially included in the automotive CARLA simulator [68], based on the work of Rebecq *et al.* [67].

2.6 POPULAR EVENT CAMERA MODELS

In this thesis, we will consider mainly three event camera models, which were used as part of the datasets of the thesis: the DAVIS346, Prophesee Gen3.1, and Prophesee Gen4 cameras.

2.6.1 *DAVIS346*

The DAVIS [15] camera was initially released in 2014. Designed by iniVation[1], this camera is one of the first event-based sensor prototype which was made available to the public (after the DVS128 [69] in 2008). In its DAVIS346 version, this sensor only has a resolution of 346px $\times$ 260px, but is able of a high dynamic range of 120dB, a temporal resolution of 1µs, a typical latency under 1ms, and it is able to produce at most 12MEPS.

In addition to the event-based modality, the DAVIS346 camera also embarks a frame-based sensor, sharing the same pixels than the event-based sensor. Despite only being able to produce grayscale frames at the same low resolution of 346px $\times$ 260px and with only a maximum frame rate of 40FPS, this secondary modality was an achievement at the time. It allowed for the first comparisons of frame-based and event-based methods, for a better comprehension of the strengths and weaknesses of the event-based modality, and for the apparition of novel methods based on the fusion of frames and events.

2.6.2 *Prophesee Gen3.1*

The Prophesee Gen3.1 sensor was released in early 2020. Designed by Prophesee[2], it has a resolution of 640px $\times$ 480px (VGA), a high dynamic range of above 120dB, a typical latency of 200µs, and can output at most 50MEPS.

While it does not embark a frame-based modality, the Prophesee Gen3.1 camera constituted an interesting upgrade in terms of resolution from the DAVIS cameras. As such, it was included in multiple datasets, like DSEC [63] and EVIMO2 [70].

1 https://inivation.com
2 https://www.prophesee.ai

2.6.3 *Prophesee Gen4*

The Prophesee Gen4 [13] camera was released in mid-2020. Also designed by Prophesee, this camera is one of the first high definition event sensor, with a resolution of 1280px $\times$ 720px. Similarly to the DAVIS346 and Prophesee Gen3.1 cameras, the Prophesee Gen4 has a high dynamic range (above 124dB), a high bandwidth (up to 1066MEPS), and a low latency (between 20 and 150µs).

As will be shown in the following sections of this thesis, the availability of high definition event sensor was becoming a critical need. Events from low- and mid-resolution sensors lack detail, leading to low performances for applications which rely on precise object edges and/or textures (e.g., object recognition), but remain popular for low-power embedded systems (like the Prophesee GenX320 sensor [71] for AR/VR). However, the higher level of details of high definition event cameras brings new issues, in particular a much higher event throughput which becomes harder to treat in real time.

2.7 DATASETS

As noted in Section 2.5.4, datasets including the event-based modality have been an increasing need over the past decade. This need has been further reinforced with the arrival of learning-based methods, which require large amounts of data to be trained. In this section, we give a more detailed focus on three datasets of the state of the art, which were used throughout the thesis: the MVSEC [62], DSEC [63], and M3ED [64] datasets. A summary of the content of these datasets is given in Table 2.1.

	MVSEC	DSEC	M3ED
Year	2018	2021	2023
Event data	2× DAVIS346 (346×260)	2× Prophesee Gen3.1 (640×480)	2× Prophesee Gen4 (1280×720)
Images	2× DAVIS346 (346×260) 2× VI-Sensor (752×480, 20Hz)	2× FLIR Blackfly S (1440×1080, 20Hz)	OVC 3b (3 cameras) (1280×800)
LiDAR data	Velodyne VLP-16 (16 channels, 100m range, 20Hz, 30° vertical FOV)	Velodyne VLP-16 (16 channels, 100m range, 10Hz, 30° vertical FOV)	Ouster OS1-64U (64 channels, 120m range, 10Hz, 45° vertical FOV)
Types of scene	Drone, handheld, car, motorcycle	Car	Drone, car, legged robot
Day and night	Yes	Yes	Yes
Ground truth data	Pose, semi-dense depth, semi-dense optical flow	Pose, semi-dense disparity, semi-dense optical flow, dense semantic maps, 2D bounding boxes	Pose, sparse depth, dense semantic maps, 3D bounding boxes
Total duration	1 hour	53 hours	200 hours
Train/val/test sets?	No	Yes	Yes

Table 2.1 – Comparison of the state-of-the-art datasets used throughout the thesis.

2.7.1 *MVSEC*

The Multivehicle Stereo Event Camera (MVSEC) dataset [62] was recorded in 2018. It contains data from multiple sensors: low-resolution (346×260) events and images from two DAVIS346 cameras, higher-resolution (752×480) images from two VI-Sensor cameras, IMU measurements from both the DAVIS346 and VI-sensor cameras, point clouds from a 16-channel Velodyne LiDAR, and GPS data from a UBlox receiver. In total, the MVSEC dataset is composed of four types of sequences, for more than 1 hour of data: 6 short flying sequences, 2 handheld sequences, 5 long outdoor driving sequences, and a single long motorcycle riding sequence.

This dataset was originally intended for depth and pose estimation, as its authors offer semi-dense ground truth depth maps and groud truth reference poses. Yet, following subsequent work [32], they were also able to compute semi-dense ground truth optical flow maps, by using the depth maps and the poses.

However, as highlighted by its authors [62], this dataset suffers from multiple flaws: the data is loosely synchronized, the biases of the event cameras were not adjusted, the calibration is approximate, and moving objects in the scene create errors in the ground truth data. For this last point in particular, the authors of MVSEC accumulate consecutive point clouds to construct a dense view of the scene, which is then reprojected in 2D to construct the ground truth depth maps. However, if an element moves in the scene during the accumulation period, then it will appear "blurry" in the accumulated view, leading to errors in both the ground truth depth maps and optical flow maps. This issue is particularly critical, as the driving sequences contain many moving vehicles and pedestrians. Yet, MVSEC remains the most popular dataset for both depth estimation and optical flow, and as such constitutes an interesting benchmark for the thesis work.

2.7.2 *DSEC*

The Stereo Event Camera Dataset for Driving Scenarios (DSEC) [63] was recorded in 2021. It contains data from mid-resolution (640×480) events from two Prophesee Gen3.1 cameras, high-definition (1440×1080) images from two FLIR Backfly S cameras, point clouds from a 16-channel Velodyne LiDAR, and GPS data from a UBlox receiver. In total, the DSEC dataset is composed of 53 sequences of day and night driving, for more than 53 hours of data.

This dataset was originally intended for disparity and depth estimation, as its authors offer semi-dense ground truth disparity maps. However, the DSEC dataset received several extensions, and now offers ground truth semi-dense optical flow maps [39], dense semantic segmentation maps [72], and 2D bounding boxes for pedestrians and vehicles [73].

2.7.3 *M3ED*

The Multi-Robot, Multi-Sensor, Multi-Environment Event Dataset (M3ED) [64] was recorded in 2023, and acts as an informal successor to the MVSEC dataset. It contains data from high-definition (1280×720) events from two Prophesee Gen4 cameras, high-definition (1280×800) images from three cameras mounted on an OVC 3b platform, point clouds from a 64-channel Ouster LiDAR, and GPS data from a UBlox receiver. In total, the M3ED dataset is composed of three types of sequences, for more than 200 hours of data: 40 indoor and outdoor sequences filmed with an UAV and with a legged robot, and 17 outdoor driving sequences.

Like the MVSEC dataset, M3ED is mainly intended for depth and pose estimation, as its authors offer sparse ground truth depth maps and ground truth reference poses. However, following the extensions of the DSEC dataset, they also offer dense 2D semantic maps, as well as 3D bounding boxes for pedestrians, buildings, cars, and trees.

One of the key elements of the M3ED dataset is that its authors have fixed the issue of the incorrect depths for the moving objects of the MVSEC dataset, by identifying them and discarding erroneous values automatically. Yet, their conservative approach also tends to discard correct data, resulting in very sparse ground truth depth maps.

3 REAL-TIME EVENT-BASED OPTICAL FLOW

Following the focus on event cameras, we describe in this second chapter of the thesis how we proposed to solve the "motion" part of the subject of the thesis. Beyond allowing for a comprehension of how the ego-platform and external objects are moving in a scene, motion is an essential component for describing the current status of the observed scene, as well as for anticipating its future states. As such, we propose here a real-time optical flow method, which only relies on events from a single camera.

The presented method and the associated results of this chapter were initially published as a journal article in IEEE Transactions on Intelligent Transportation Systems in December 2021 [74], and were then slightly refined and presented as part of a French national conference (RFIAP) in July 2022 [75]. A project page is also available at `https://vbrebion.github.io/RTEF/`, and contains the links to the original article, the source code, the dataset, and videos associated to this work.

3.1 INTRODUCTION

By estimating the displacement of each pixel in a camera along time, optical flow describes in 2D the motion field of the observed scene. Optical flow is a key enabler for many major applications, such as object detection and tracking [76], motion estimation [77], visual odometry [78], and image segmentation [79]. In the context of intelligent robotics, optical flow constitutes one of the fundamental building bricks of the perception pipeline, especially implied in the detection of moving objects like vulnerable external users.

Optical flow literature is highly abundant for frame-based cameras. However, as discussed in Section 2.3.2, there is no direct translation for frame-based optical flow algorithms to event cameras. The sparse and asynchronous nature of their output constitutes a major paradigm shift. In this chapter, we describe the work conducted over the first year of the thesis, dedicated to this problem. In particular, we describe here our proposition: a novel optimized framework for computing real-time event-based optical flow (EBOF, in short) for both low- and high-resolution event cameras, using a *frame-based* approach. We call our method RTEF, for "Real-Time Event-based Flow".

We first give a formal description of the problem in Section 3.2. Then, in Sections 3.3 and 3.4, we review the state of the art of both the frame-based and event-based optical flow methods, and discuss our choice of using a frame-based approach for computing EBOF. Finally, we present our pipeline-based method and evaluate it in Sections 3.5 and 3.6, before drawing some conclusions in Section 3.7.

3.2 PROBLEM FORMULATION

The objective of optical flow is to estimate the two-dimensional displacement of intensity patterns [80]. More specifically, in its frame-based formulation, optical flow describes the apparent motion field of pixels between consecutive images, caused by the relative motion between the camera and the elements in the observed scene.

If we note $I(x, y, t)$ the intensity of a pixel at location (x, y) and at time t, if this pixel moves by Δx and Δy pixels over a period Δt (Δt being the time between two consecutive frames), and if we consider that brightness remains constant over that period, then we have the following equality:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) \tag{3.1}$$

By linearizing this equation using Taylor expansion, it can then be shown [81] that:

$$I_x u + I_y v + I_t = 0 \tag{3.2}$$

where I_x, I_y, and I_t are respectively the spatial and temporal derivatives (gradients) of the image at coordinates (x, y) and at time t, and where u and v are the x and y components of the optical flow.

The objective of the optical flow problem is therefore to determine these u and v components for every pixel of each pair of consecutive images, in the form of dense motion fields.

3.3 RELATED WORK

3.3.1 *Frame-Based Optical Flow*

In Eq. (3.2), the two components u and v of the optical flow are unknown, but since only one equation is available, the system is underdetermined (this is the so-called *aperture problem*). To solve this issue, researchers have proposed to add supplementary constraints. In 1981, Horn and Schunck [81] proposed a global *smooth flow* constraint, where flow differences should be minimal for adjacent pixels in the whole image. The same year, Lucas and Kanade [82] proposed a local *constant flow* constraint, where optical flow components should be the same for every pixel of a local neighborhood.

More recent approaches have proposed extensions to these methods. Since the Horn-Schunck and the Lucas-Kanade methods are restricted to small motions, Meinhardt *et al.* [83] and Bouguet [84] proposed pyramidal-based approaches to compute optical flow at multiple scales. To also prevent the oversmoothing of the Horn-Schunck method, Black and Anandan [85] and Sun *et al.* [86] proposed to add regularization terms, making it more robust.

In the past few years however, a paradigm shift has started appearing, with the transition from geometry-based approaches to learning-based approaches. Neural networks are especially notorious for their capability of learning from data to generalize even in presence of noise and inconsistencies, which is a clear advantage when trying to estimate optical flow on real scenes. Networks like FlowNet [87] or UnFlow [88] stood as state of the art over the past few years. However, as proposed by Teed and Deng with RAFT [89], the reintroduction of geometry-based concepts in networks has proven to be an efficient approach.

More recently, unified models able to solve multiple tasks at once have gained popularity, as interactions between these tasks lead to a better understanding of the scene and a better generalization of learned models. The GMFlow network of Xu *et al.* [90] is a perfect example of this trend, as it can compute optical flow, disparity, and depth maps, and as it defines the current state of the art on optical flow.

3.3.2 *Event-Based Optical Flow (EBOF)*

As noted in Chapter 2, previously listed frame-based methods can not be directly applied to solve the optical flow problem for event cameras. We describe here the various approaches proposed so far in the state of the art, which either try to reformulate the optical flow problem for event-based cameras, or try instead to leverage the frame-based state of the art by converting the event-based optical flow problem into a frame-based problem.

Pure Events In order to compute event-based optical flow, some authors use the events and all their properties without accumulation into frame-based representations. Such a frameless approach was proposed by Benosman *et al.* [31], by using a plane-fitting method on short temporal windows of events to determine their motion in the visual scene. Other works [38, 91, 92, 93, 94] proposed contrast maximization schemes as proxies for computing optical flow, by evaluating the sharpness of motion-compensated images of accumulated events. More recently, Paredes-Vallés *et al.* [34, 95] and Cuadrado *et al.* [96] proposed to exploit Spiking Neural Networks (SNNs) for a fully biologically-inspired EBOF estimation.

Reconstructed Frames Due to the great advances on optical flow estimation with traditional cameras, other authors have proposed to build image-like representations from the events, in order to use them as input for these state-of-the-art methods. Almatrafi *et al.* [97] proposed to accumulate events in short temporal windows, to construct a binary image from them, and then to use the distance transform to create stable dense images designed to be used with any frame-based optical flow method. Zhu *et al.* [98] also proposed to create images of accumulated events, but they instead compute a sparse optical flow by extracting visual features through the use of the Harris corner detector [99], and they track them using an expectation maximization algorithm. An alternative approach was proposed by Nagata *et al.* [100], as they use a surface matching approach on short time-shifted images of accumulated events (*time surfaces*) to evaluate their displacement.

Learning-Based Recent works have also adapted proven neural network architectures for a use with images of events; [32, 101] proposed FlowNet-inspired [87] networks for inferring EBOF, while Gehrig *et al.* [39] proposed a RAFT-inspired [89] one. Some authors also proposed novel types of networks, designed specifically for the characteristics of event cameras. Paredes-Vallés *et al.* [48] proposed a light and real-time network based on event warping, called FireFlowNet. More recently, Liu *et al.* [40] designed a network able to incorporate the notion of temporal continuity of event data, to compute temporally fine-grained EBOF.

Additional Modalities Some methods also exploit the capacity of certain neuromorphic sensors to produce more than events, such as frames or inertial measurements. Pan *et al.* [102] used the flow of events as a deblurring tool for the frames of the DAVIS camera in highly dynamic scenes, allowing for a better optical flow estimation. Rueckauer *et al.* [103] used the IMU integrated in the DAVIS240C camera to determine an exact optical flow estimation for pure rotational movements. Finally, Lee *et al.* [104] extended their FlowNet-inspired network [101] by adding the grayscale images from the DAVIS camera as a secondary input.

3.4 OUR ORIENTATION: DENSIFYING EVENTS

None of the methods presented in Section 3.3.2 has considered the issue of computing optical flow with a high-definition event camera. Furthermore, very few of them ([48, 92, 103, 105]) have been able to achieve real-time compatibility even for low-resolution inputs. In this section, we analyze why both these constraints are difficult to hold, and how we propose to solve them.

First of all, approaches relying on pure events are by definition more complex to develop, as they can not reuse the frame-based state of the art. While they might make the most sense in terms of exploiting the novel event-based paradigm to its full potential, these approaches are especially heavy, slow, and hard to optimize. Furthermore, they have mostly been applied so far to simple scenes with limited motions, and have difficulties scaling to scenes of higher complexity.

Learning-based approaches appear as more promising, as they currently hold state-of-the-art results. However, neural networks always come with a trade-off between the quality of the results and the time it takes for their inference; reaching our objective of real-time performances with a high-resolution input would require sacrificing the quality greatly. Learning-based methods also require a high amount of training data, which is still quite limited to this day for high-resolution sensors. Finally, the quality of the results on never-seen-before scenes is hard to predict, as generalization of the network is not guaranteed, and as it highly relies on the training data, the training procedure, the camera settings, ...

As such, frame-based approaches are of particular interest to us[1]. Sure, accumulating

1 We insist here that we do not aim at fully reconstructing "real" images of the scenes like the ones that would be captured by a traditional camera. Instead, we aim at creating pseudo-images, i.e., representations of the event stream as frames which should then be optimized for a use with traditional frame-based optical flow methods.

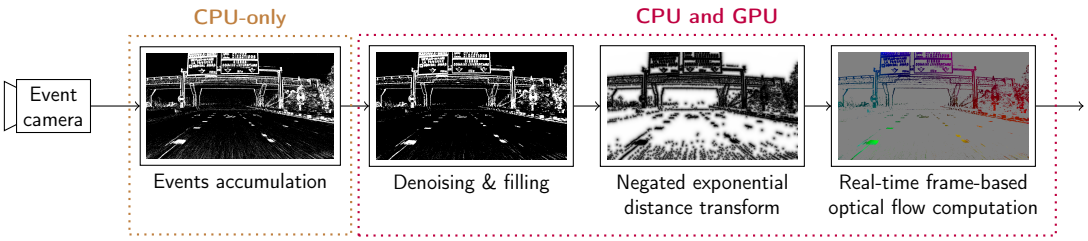


Figure 3.1 – Our event-based optical flow (EBOF) computation architecture, RTEF, able to run in real-time with low- and high-resolution event cameras. Due to the pipeline architecture, all four blocks are independent parallel processes. Each block depicts the result it produces, for a sample driving sequence.

events to construct pseudo-frames does lead to the loss of the asynchrony property of the event camera, it can lead to losing some precision in the information, and it can reintroduce frame-related problems like motion blur. However, in return, this pseudo-frame construction **(1)** allows for the reuse of frame-based methods which have been developed and optimized over the past decades, **(2)** allows for the exploitation of computer architectures to their full potential, especially GPUs, **(3)** can be exploited with traditional geometry-based optical flow approaches, which do not rely on specific training dataset and training procedures, making the quality of results in theory independent of the type of scene observed. Furthermore, we underline here that not all properties of the event cameras are lost, and that they are still more interesting than frame-based cameras: the HDR capacity is still present (so the optical flow should compute similarly during daytime or under low-light), and the event accumulation process can be adjusted on the fly, allowing for a fine control over the level of information that should be kept in a pseudo-frame and the overall responsiveness of the system.

3.5 METHOD

Following the problem formulation and the reason for the use of a dense representation to reach real-time performances, we detail in this section the novel framework (RTEF) we developed for computing real-time EBOF. In order to optimize computational time and reach real-time performances, we propose to parallelize tasks through the use of a pipeline architecture [106]. An illustration of this framework with example results for each step is available in Fig. 3.1. The following subsections will detail how each block contributes towards obtaining the real-time EBOF.

3.5.1 *Accumulation for Edge Images*

As discussed in Section 3.4, the very first step needed in our case to ultimately compute optical flow is to accumulate events in the form of a pseudo-image. The first component of our architecture is therefore responsible for receiving and accumulating the events from the camera in short temporal windows, to form "edge images". These binary matrices indicate whether each pixel produced at least one

event during the accumulation time Δt. More formally, if we note $E(\mathbf{x}, t)$ the content of the pixel at coordinates $\mathbf{x}$ in the edge image E produced at time t, then:

$$\begin{cases} E(\mathbf{x}, t) = 1 & \text{if } \exists e = (\mathbf{x}_e, t_e, p_e), \mathbf{x}_e = \mathbf{x}, t \leq t_e < t + \Delta t \\ E(\mathbf{x}, t) = 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

By doing so, each edge image depicts a binary representation of the main edges of the objects with relative motion in the visual scene, which can be used as a first stable medium for computing optical flow. For notation simplicity, when the notion of time is not important, we will simply denote the content of a pixel of an edge image as $E(\mathbf{x})$ in the rest of this chapter.

As can be noted through Eq. (3.3), these edge images do not take into account the polarity of the events. As argued by Almatrafi *et al.* [97], and as we have experimented, both positive and negative events represent similarly the edges of the objects in the visual scene. In addition, and as was shown through Fig. 2.2, polarities are dependent on the orientation of the motion and on their relative color to the background: introducing polarities into the edge images would mean that appearance of objects would change during motion. We want to avoid this phenomenon at all cost if we want to reuse traditional frame-based optical flow methods, which require images to be as stable as possible.

The choice of Δt is also important and linked to the application: taking a Δt too short will lead to edge images with too few events, resulting in an unstable appearance, while taking a Δt too long will fail to capture clearly the movement of the objects by introducing blur.

Compared to other dense formulations from the literature (time surface [22, 23, 100], motion-compensated images [91], reconstructed images [29]), our formulation has the benefit of keeping only the information required for frame-based optical flow estimation. Computationally speaking, this makes this solution extremely efficient, as each packet of events received from the camera only needs to be appended to a buffer. In parallel, a second thread, triggered when the time window has expired, is responsible for collecting all the events from the buffer and creating the edge image, which is then sent for further processing. As such, due to its simplicity, this component runs solely on the CPU, as it would not benefit from the parallelization capabilities of the GPU.

3.5.2 *Denoising and Filling*

Event cameras generate a significant amount of noise, impacting the quality and stability of the edge images, which in return would affect the final optical flow computation if left untreated.

A solution could be to use one of the state-of-the-art denoising solutions of the literature [22, 107] during the accumulation step, that is, before creating the edge image. However, doing so would be computationally expensive, as the sparse and asynchronous nature of the events at that step makes it hard to look for neighbor

pixels states. Furthermore, many of these solutions were designed for low-resolution sensors, and translate difficultly to higher definition ones.

To circumvent this issue, we propose in this work a novel, fast yet efficient, method for discarding incorrect events. Our approach relies on applying denoising after the edge image creation. Proposed process is computed in two steps — denoising and filling — and is similar to applying morphological erosion and dilatation (morphological opening) on the edge image. In the denoising step, described in Algorithm 1, erroneous edge pixels are sought to be eliminated, by removing isolated events. On the contrary, the filling step described in Algorithm 2 aims at filling locations where an edge pixel is missing, but should have been produced by the camera, in order to help to stabilize the edge images. An illustration of both these steps on a real scene is given in Fig. 3.2.

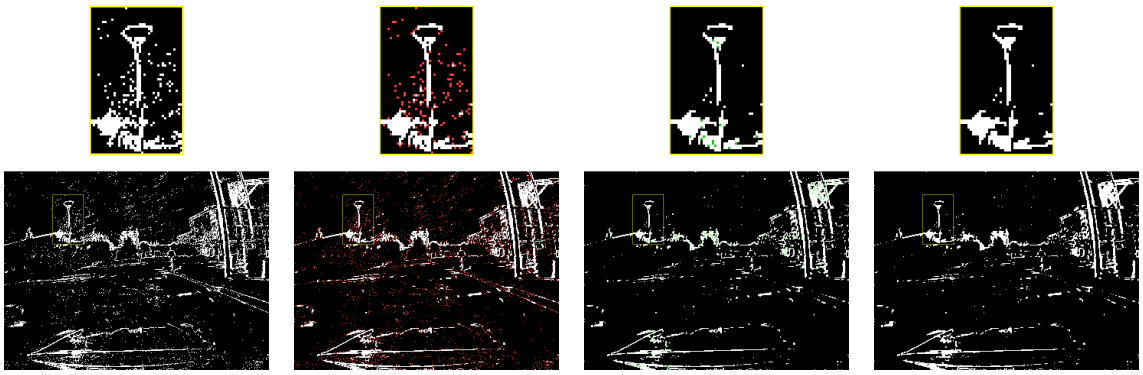
Algorithm 1: Denoising

Inputs: An edge image E
The denoising threshold N_d
Output: The denoised edge image E_d
$E_d \leftarrow E$;
foreach *pixel index* $\mathbf{x} \in E$ **do**
  if $E(\mathbf{x}) = 1$ **then** // $E(\mathbf{x})$ is an edge pixel
    $n_d \leftarrow$ count of edge pixels among the 4 direct neighbor pixels of $\mathbf{x}$ in E;
    if $n_d < N_d$ **then**
      $E_d(\mathbf{x}) \leftarrow 0$; // $E_d(\mathbf{x})$ is not an edge pixel anymore

Algorithm 2: Filling

Inputs: A denoised edge image E_d
The filling threshold N_f
Output: The denoised and filled edge image E_{df}
$E_{df} \leftarrow E_d$;
foreach *pixel index* $\mathbf{x} \in E_d$ **do**
  if $E_d(\mathbf{x}) = 0$ **then** // $E_d(\mathbf{x})$ is not an edge pixel
    $n_f \leftarrow$ count of edge pixels among the 4 direct neighbor pixels of $\mathbf{x}$ in E_d;
    if $n_f \geq N_f$ **then**
      $E_{df}(\mathbf{x}) \leftarrow 1$; // $E_{df}(\mathbf{x})$ becomes an edge pixel

We underline here the importance of computing denoising and filling separately in this order, to avoid creating inconsistencies. Indeed, if the filling step was processed simultaneously with the denoising, then pixels that would later be discarded as noise could contribute to creating incorrect filling pixels, thus introducing new noise.



(a) (b) (c) (d)

Figure 3.2 – Steps of the denoising and filling process, for a noisy edge image. A zoomed view of the streetlamp (squared in yellow) is also provided for better visibility. (a) The original noisy edge image. (b) The same image with edge pixels identified as noise in red ($N_d = 2$). (c) The denoised edge image with the newly added pixels for the filling in green ($N_f = 3$). (d) The final denoised and filled edge image.

Denoising and filling thresholds N_d and N_f depend on the event camera configuration, as each camera may give a different noise profile. The aim of the denoising is to discard isolated pixels, that is, pixels with very few neighbors: $N_d = 1$ or 2 appear therefore as the best options. As can be seen in Algorithm 1, setting $N_d = 0$ disables the denoising. Then, the goal of the filling is to slightly stabilize the appearance of the edge images, by adding edge pixels in locations where there are enough neighboring edge pixels to be confident that an edge pixel should have been produced: values of $N_f = 4$ or 3 are therefore the best compromise to add such pixels. As can be seen in Algorithm 2, setting $N_f = 5$ disables the filling. A general advice is to select $N_d < N_f$. A sensitivity analysis on N_d and N_f is conducted in Section 3.6.10.

Finally, while this formulation tends to remove small details from the edge images by considering them as noise (as can be seen for instance for the buildings on the right side of the edge images of Fig. 3.2), it actually helps to obtain more stable images, by extracting the main edges from the scene, and discarding superfluous textures.

Another advantage of this formulation lies in its simple and parallelizable formulation, as the computation for each pixel is independent of the one of its neighbors. Therefore, we implemented it on the GPU, to exploit its capabilities, and to relieve the CPU so that it can undertake other complex tasks.

3.5.3 *The Negated Exponential Distance Surface*

Even after denoising and filling, the edge images are only binary matrices. As such, they can hardly be used for computing optical flow with traditional frame-based algorithms: as shown in Eq. (3.2), these methods require image gradients, which would be discontinuous for our binary edge images. In order to make them viable

for frame-based optical flow computation, densifying them through the use of the distance transform, as proposed by Almatrafi *et al.* [97], is an interesting baseline for introducing smooth variations in the images, and therefore having smooth gradients.

In their approach, they consider a set of edge pixels Φ, which is defined in our case after denoising and filling as:

$$\Phi \doteq \{\mathbf{x}_i \mid E_{df}(\mathbf{x}_i) = 1\} \tag{3.4}$$

Then, they compute a distance surface D (which is of the same size as the edge image), by giving to each pixel $\mathbf{x}$ of D its distance to the closest edge pixel in Φ. Formally, it can be written as follows:

$$D(\mathbf{x}) \doteq \min_{\mathbf{x}_i \in \Phi} d(\mathbf{x}, \mathbf{x}_i) \tag{3.5}$$

where d is the L^2 distance function:

$$d(\mathbf{x}_1, \mathbf{x}_2) \doteq \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)(\mathbf{x}_1 - \mathbf{x}_2)^T} \tag{3.6}$$

In order to obtain an actual 8-bit image, a final normalization is applied to contain the values of the distance surface as integers between 0 and 255:

$$D_{\text{normalized}}(\mathbf{x}) \doteq \left\lfloor 255 \times \frac{D(\mathbf{x})}{\max(D)} \right\rceil \tag{3.7}$$

However, this approach has the main drawback of needing a near-perfect denoising, as a single noisy event can disrupt the appearance of the whole distance surface, as shown in the second row of Fig. 3.3. The computed distances do not have an upper bound, meaning that the area of influence of each edge pixel can be infinite, and depends on the presence of other close neighbors. In addition, as shown in Eq. (3.7), the image representation of the distance surface depends on its maximum value, and the appearance of objects can therefore vary greatly based on their position. An example of this behavior is shown in the third row of Fig. 3.3: the inside part of the square appears darker when it is in the bottom right of the image, due to the maximum distance increasing. An answer to these problems could be to limit the computed distances to a maximal value, restricting the influence of an edge pixel to a fixed neighborhood. This solution, however, would introduce a non-smooth transition in the distance transform function. It can become an issue for the gradient computation on distance surfaces, often used as part of the optical flow estimation.

Another issue of the approach of Almatrafi *et al.* also appears when distinct objects come close to each other: their edges tend to merge together in the resulting distance surface, making the individual objects indistinguishable, such as in the last row of Fig. 3.3. This phenomenon can lead to incorrect optical flow results, especially when a block-matching or image warping formulation is employed, due to the lack of texture on the produced image. Giving more emphasis to the pixels directly surrounding the edge pixels would help to create distance surfaces with more prominent object edges, limiting this merging issue. A solution could be to employ a function with a logarithmic shape.

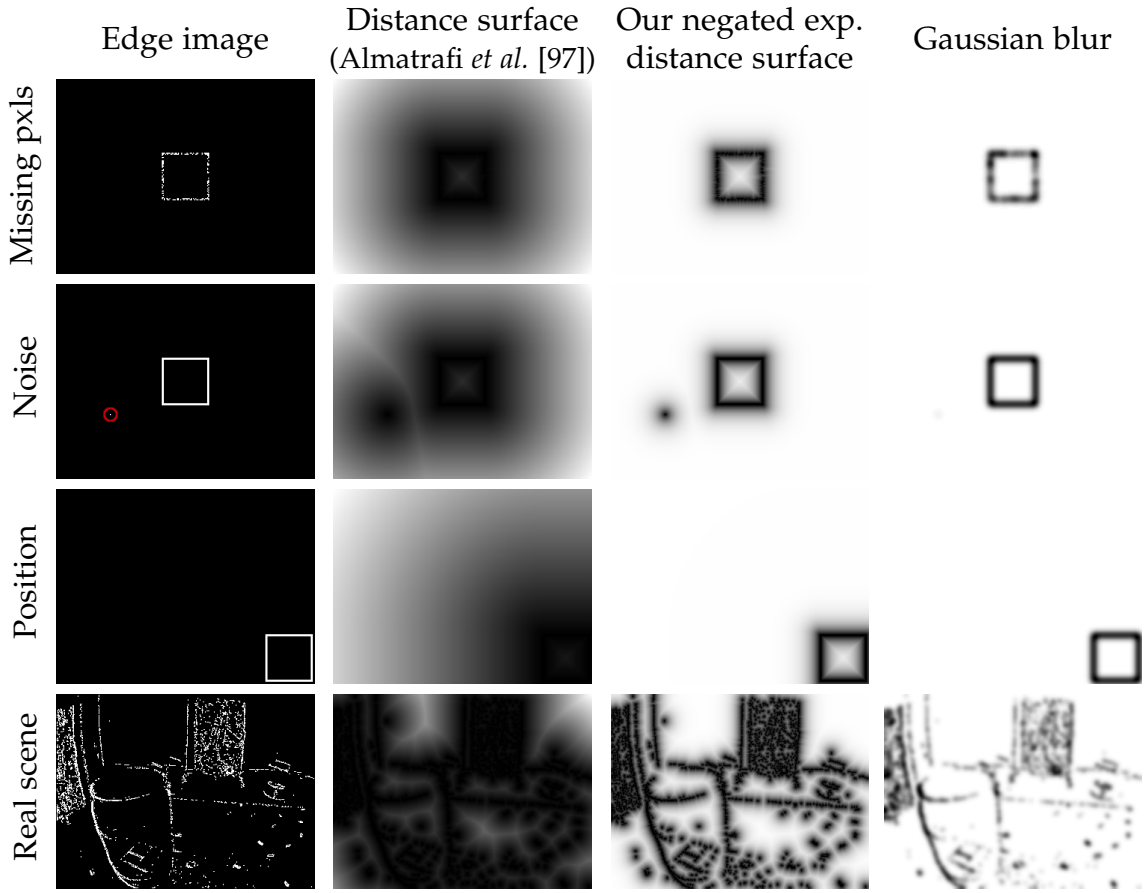


Figure 3.3 – Comparison between the original distance surface, proposed negated exponential distance one (with $\alpha = 8$ for a good visibility), and a simple Gaussian blur (with $\sigma = 3$px). From top to bottom: a simulated square with 50% of its pixels randomly removed; the same square with a single pixel of noise added (circled in red); the same square in the bottom right of the image; and an indoor flying scene from the MVSEC dataset [62].

To solve jointly both these issues, we propose in this work to replace the distance surface D with a novel negated exponential distance surface D_{exp}, formulated as follows:

$$D_{\text{exp}}(\mathbf{x}) \doteq \min_{\mathbf{x}_i \in \Phi} d_{\text{exp}}(\mathbf{x}, \mathbf{x}_i) \tag{3.8}$$

where d_{exp} is a negated exponential distance function:

$$d_{\text{exp}}(\mathbf{x}_1, \mathbf{x}_2) \doteq 1 - \exp\left(-\frac{d(\mathbf{x}_1, \mathbf{x}_2)}{\alpha}\right) \tag{3.9}$$

where α is a spreading parameter. Figure 3.4 compares the aforementioned functions over the distance to the closest edge pixel. As can be seen through the plot, the main advantage of our negated exponential formulation is that, while close to a logarithmic formulation, each edge pixel also has a restricted influence area, after which the values saturate to a value of 1. Therefore, the normalized 8-bit image version can then be re-written as:

$$D_{\text{exp\_normalized}}(\mathbf{x}) \doteq \lfloor 255 \times D_{\text{exp}}(\mathbf{x}) \rceil \tag{3.10}$$

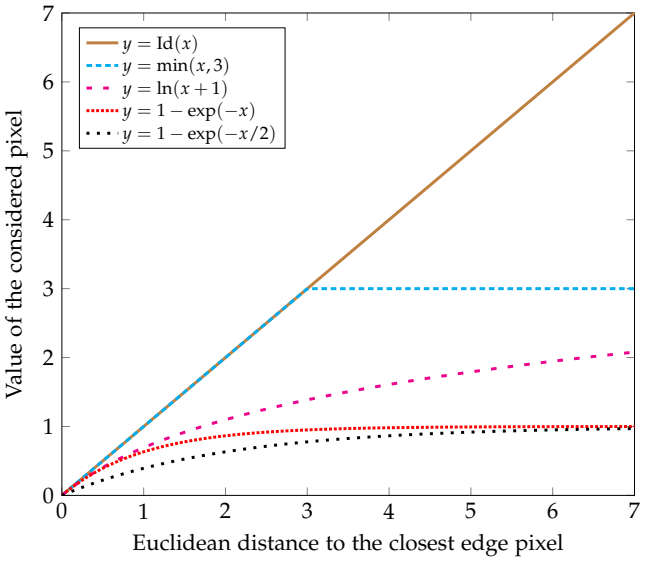


Figure 3.4 – Values of the distance transform as a function of the distance to the closest edge pixel. In the order of the legend, the curves represent the original distance transform, the same with an upper bound set to 3px, the natural logarithm version, and our negated exponential formulation, with α set to 1 and 2 respectively.

where the dependence on the maximum distance is removed.

The spreading parameter α can be used to determine the size of the neighborhood influenced by each edge pixel. This parameter conditions the appearance of the distance surface, and regulates the remaining imperfections of the edge images. A low value for α restricts the area of influence of each edge pixel to only its close neighbors. It limits the influence of the noise on its appearance, but makes the distance surface less stable and more prone to variations. On the contrary, selecting a higher value for α has the opposite effect: variations of the appearance of the various objects in the scene are well compensated, but noisy events have a more important effect.

In addition, α can be rewritten from Eq. (3.9) as a function of the wanted distance of saturation for d, which we will name here d_{sat} (we remove the function parameters for more simplicity in the notations):

$$\alpha = -\frac{d_{\text{sat}}}{\ln(\varepsilon)} \tag{3.11}$$

with

$$\varepsilon \doteq 1 - d_{\text{exp}} \tag{3.12}$$

where by definition $\varepsilon > 0$. ε is formulated so as to represent the gap between d_{exp} and the saturation value of 1. Saturation is therefore reached when this gap ε is as small as possible, i.e., $\varepsilon \to 0$. Since we work in the discrete 8-bit domain, saturation is reached when $\varepsilon < 1/255$. Integrating this value in Eq. (3.11) results in the final formulation of α as a function of d_{sat}:

$$\alpha = -\frac{d_{\text{sat}}}{\ln(1/255)} \approx \frac{d_{\text{sat}}}{5.541}. \tag{3.13}$$

The sensitivity of d_{sat} is studied in the analysis Section 3.6.10.

The interest of our negated exponential distance surface formulation is illustrated in Fig. 3.3 as a side-by-side visual comparison with the original distance surface. As expected, our negated exponential formulation compensates the missing pixels similarly to the original distance surface, while limiting the impact of noisy edge pixels. As seen in the third row, the appearance of objects also remains the same independently of their position, due to the removal of the dependence on the maximum distance in the image normalization process. Finally, this formulation displays more distinct edges and keeps objects texture, especially visible in the real complex scene represented in the last row (note how the board and the barrel keep a clear appearance using negated exponential formulation, while they are hardly distinguishable with the original distance surface).

In addition, while at first glance a simple Gaussian blur could be considered to have similar effects to our formulation for a lower computational cost, we showcase in Fig. 3.3 that this is not really the case. Missing pixels are much less compensated as seen in the first row, while areas with more edge pixels appear darker in the last row. Both these phenomenons are due to the use of a kernel for blurring the image: the more edge pixels fall into the kernel centered around a given pixel, the higher the value of this pixel will be, meaning that the value of a pixel depends on its close neighbors. This is something we want to avoid at all cost, since we want to make objects as similar-looking as possible through time for a good computation of optical flow.

Regarding the implementation of the distance transform, we employed the fast solution described by Coeurjolly *et al.* [108]. The choice of this method was especially motivated by its optimized formulation, allowing for large parallelization, and by the little amount of modifications required to incorporate our negated exponential formulation. This block of our pipeline was therefore also implemented on GPU, to exploit its parallelization capabilities.

3.5.4 *Frame-Based Optical Flow*

The final block of our architecture is the computation of the optical flow itself. Since the previous steps led to dense image-like structures from the flow of events, any state-of-the-art frame-based optical flow method could be used here.

As part of this work, we selected the "flow filter" approach of Adarve and Mahony [109]. Their method is based on an update-prediction architecture, similar to the one of Black [110], which predicts optical flow using an image warping process, and temporally propagates the optical flow estimations using an incremental framework. Multiple update-prediction loops are stacked as a pyramidal structure, enabling the capture of both large and fine displacements in the images. Their method was designed for a fast and accurate estimation of the optical flow field, and is implemented on GPU.

The choice of this method as our optical flow computation solution was mainly guided by their use of a predictive filter-based formulation, which, beyond enabling

real-time compatibility on GPU, allows for a temporal smoothing of the flow. This property brings stability and robustness to the overall optical flow thanks to its memory effect, which is beneficial given the sometimes unstable nature of events.

Finally, while this method returns a dense optical flow covering the whole surface of the image, we restrict it to the edge pixels of the denoised and filled edge image. Indeed, by nature, events encode sparse information, detailing the pixels for which a change in luminosity was observed. The densification produced by the use of the negated exponential distance transform differs from an inference of missing data: it is only intended for creating texture and smooth transitions, which are necessary to determine the optical flow.

3.6 EVALUATION

3.6.1 *Datasets*

As part of the evaluation of the proposed methods, five datasets are going to be used in the following subsections. The first one is the low-resolution MVSEC dataset [32, 62], which was the first event vision dataset with real data that included a ground truth optical flow. However, as noted in Section 2.7, the MVSEC dataset suffers from several limitations (lack of physical synchronization, approximate calibration, ...). While it remains an interesting baseline, its use starts to decline now that more recent datasets have been published. As such, we will also evaluate our method on the mid-resolution DSEC dataset [63], which constitutes a more stable benchmark. Both will serve as the basis for comparison with other state-of-the-art methods. For high-definition data, three complementary datasets are used: the 1 Megapixel Automotive Detection Dataset [25], for a deep evaluation on daily driving sequences; a 20-minute-long driving sequence recorded by Prophesee, for visual comparison with the current frame-based state of the art; and a novel high-speed high-definition event-based indoor dataset we recorded as part of this thesis, to demonstrate the accuracy of RTEF even under large motions. A summary of these datasets is given in Table 3.1.

	Resolution	Scenes	Ground truth?	Frames?	Conditions
MVSEC	346×240 (low)	Indoor & Vehicular	Semi-dense	Yes	Day, night
DSEC	640×480 (mid)	Vehicular	Semi-dense	Yes	Day, night
1 Megapixel Automotive Detection	1280×720 (HD)	Vehicular	No	No	Day, varying lighting and weather
20-minute-long driving sequence	1280×720 (HD)	Vehicular	No	Yes	Day, single long sequence
Our high-speed event dataset	1280×720 (HD)	Indoor	No	No	Very fast and erratic motions

Table 3.1 – Comparison of the event-based datasets used for the evaluation of our optical flow method RTEF.

3.6.2 *Setup*

The implementation of our method was made using ROS Noetic, in C++11 and CUDA 12.2, combined with the use of the OpenCV 4.2 library. Both the implementation

and the evaluation phases were conducted on an HP ZBook 17 G6 laptop, with an Intel i9-9880H CPU, an NVIDIA Quadro RTX 5000 GPU, 64 GB of RAM, and using Ubuntu 20.04.

Regarding the parameters, three configurations were used, respectively for low-, mid-, and high-resolution input data.

For the low-resolution data (346×260) of the MVSEC dataset [32, 62], we were restricted to use a temporal window of size $\Delta t = 1$ frame[2] for a fair comparison with the other state-of-the-art methods using this time window [24, 32, 34, 39, 48, 101, 105, 111]. For the denoising and filling, we set $N_d = 1$ and $N_f = 4$, due to the high noise in these recordings. The negated exponential distance transform was configured with $\alpha = 1.08$ (so that $d_{\text{sat}} = 6$px, see Eq. (3.11)). Finally, the "flow filter" method of Adarve and Mahony [109] was configured with 3 pyramidal layers, with their regularization weights set respectively to 50.0, 250.0, and 500.0, and with 50, 25, and 5 smooth iterations per layer.

For the mid-resolution data (640×480) of the DSEC dataset [63], we used a temporal window of size $\Delta t = 20$ms, instead of the intended $\Delta t = 100$ms one (more discussion on that topic is given in Section 3.6.6). No denoising nor filling was used on this dataset ($N_d = 0$, $N_f = 5$), due to the low amount of noise in these recordings. Similarly to the low-resolution data, α was set to a value of 1.08 ($d_{\text{sat}} = 6$px), and the "flow filter" method was configured with 3 layers, with their regularization weights set respectively to 5.0, 150.0, and 200.0, and with 200, 150, and 7 smooth iterations per layer.

For the high-resolution data (1280×720), finally, a temporal window $\Delta t = 15$ms was used, to better capture the movements. $N_d = 2$ and $N_f = 3$ were empirically chosen, as the best compromise between removing noise and keeping the main edges. $\alpha = 1.08$ ($d_{\text{sat}} = 6$px) also proved to be the more adequate, allowing to keep the scene details, while compensating potential imperfections. The "flow filter" method was configured with 3 layers, with regularization weights all set to 500.0, and with 20 smooth iterations per layer.

Finally, in the EBOF illustrations and videos in the following subsections, the pixels where no event was received are colored in medium gray.

3.6.3 *Evaluation Metrics*

In order to evaluate the quality of our optical flow results, three metrics are used as part of this chapter.

The first two ones, the percentage of outliers and the Average Endpoint Error (AEE), are traditional optical flow metrics, used for instance in the KITTI benchmark [112]. The percentage of outliers reports the number of pixels for which the error is above

2 In MVSEC, $\Delta t = 1$ frame $\simeq$ 32ms for "Indoor flying" sequences, $\simeq$ 22ms for "Outdoor day" sequences, and $\simeq$ 97ms for "Outdoor night" sequences.

3px and 5% of the magnitude of the flow vector. The AEE is a raw error measurement on both orientation and magnitude of the flow, computed as following:

$$\text{AEE} = \frac{1}{N} \sum_{i=1}^{N} |f_i - \hat{f}_i| \tag{3.14}$$

where N is the total number of flow vectors, f_i the i^{th} estimated flow vector, and $\hat{f}_i$ its ground truth equivalent.

However, still as of the writing of this thesis, no complex high-resolution event-based dataset with a ground truth for optical flow exists. In order to still leverage high-resolution datasets, for instance Prophesee's 1 Megapixel Automotive Detection dataset [25], and to provide a quantitative evaluation of our EBOF results, we adopt the Flow Warping Loss (FWL) metric proposed by Stoffregen *et al.* [111]. The principle is to compensate the motion of each raw event (considering its polarity and timestamp) through its computed optical flow, in order to accumulate them in an image of compensated events at a reference time t. If the optical flow is accurate, compensated events superimpose in the same pixel position, producing sharp edges. The FWL then evaluates the sharpness of the produced image, compared to the one where events are not compensated:

$$\text{FWL} = \frac{\sigma^2(I_{\text{comp}})}{\sigma^2(I_{\text{uncomp}})} \tag{3.15}$$

where σ^2 is the image variance function, I_{comp} the flow-compensated image of events, and I_{uncomp} the original uncompensated image. By doing so, a final FWL value greater than 1 is sought to be obtained, as it indicates that the computed flow is better than the "zero flow" (uncompensated) reference, as the compensated image is more sharp.

While the FWL metric allows for comparison on datasets without a ground truth, it suffers from several issues. One such issue is the "event collapse" highlighted by Shiba *et al.* [93]: high FWL scores are obtained if the motion of all events is compensated such that they fall in the same pixel, resulting in images with a maximal sharpness. While such an issue was not observed in our case, results presented with this metric should therefore still be taken with a grain of salt.

3.6.4 *Ablation Studies*

To show the validity of our contributions, we also conducted evaluations with ablations or distance surface alternatives:

- RTEF_{NDF}, the full proposition without denoising and filling;
- $\text{RTEF}_{\text{DS\_L}}$, with the linear distance transform, $y = \text{Id}(x)$;
- $\text{RTEF}_{\text{DS\_LB}}$, with the upper-bound distance transform (set to 6px, equal to the used d_{sat} value with proposed negated exponential formulation), $y = \min(x, 6)$;

- $\text{RTEF}_{\text{DS\_Log}}$, with the logarithmic distance transform, $y = \log_e(x+1)$.

As a reminder, Fig. 3.4 illustrates the shape of these variants.

3.6.5 *Evaluation on the MVSEC Dataset*

We first evaluate our EBOF method on the low-resolution (346×260) MVSEC dataset proposed by Zhu *et al.* [32, 62]. Despite several shortcomings highlighted by its authors — namely, errors created by moving objects, an approximate synchronization, an approximate calibration, and the use of default biases for the event camera — this dataset remains the main reference for evaluating EBOF results on complex real-life sequences. Therefore, we present in Table 3.2 our error measurements on this dataset, compared to other reference methods from the literature (both non real-time and real-time capable). We also compare them to a "zero flow" reference, i.e., error measurements when the estimated optical flow is set to a null vector field. Note that, similarly to other authors such as [32, 111], for `outdoor` sequences, we ignored the pixels where the hood of the car is visible, as the ground truth values for these pixels are incorrect in the MVSEC dataset.

	`indoor_flying_1`		`indoor_flying_2`		`indoor_flying_3`		`outdoor_day_1`		`outdoor_day_2`		`outdoor_night_1`		`outdoor_night_2`		`outdoor_night_3`	
	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓	AEE ↓	% outl. ↓
Zero flow	1.71	8.9	3.03	40.2	2.53	29.1	1.46	5.1	1.70	13.0	5.41	63.8	6.62	73.7	7.20	77.1
Non Real-Time																
$\text{EV-FlowNet}_{\text{MVSEC}}^{*}$ [32]	0.85	0.9	1.29	7.5	1.13	5.3	0.56	0.4	-	-	**1.90**	**18.6**	**2.26**	**21.9**	**2.13**	**20.4**
Zhu *et al.* [24]	0.58	**0.0**	1.02	4.0	0.87	3.0	0.32	**0.0**	-	-	-	-	-	-	-	-
$\text{EV-FlowNet}_{\text{HQF}}$ [111]	0.56	-	0.66	-	0.59	-	0.68	-	**0.82**	-	-	-	-	-	-	-
Spike-FlowNet [101]	0.84	-	1.28	-	0.87	-	0.49	-	-	-	-	-	-	-	-	-
Spiking EV-FlowNet [34]	0.60	0.5	1.17	8.1	0.93	5.6	0.47	0.2	-	-	-	-	-	-	-	-
E-RAFT [39]	1.10	5.7	1.94	30.8	1.66	25.2	**0.24**	**0.0**	-	-	-	-	-	-	-	-
Fusion-FlowNet [104]	0.56	-	0.95	-	0.76	-	0.59	-	-	-	-	-	-	-	-	-
MultiCM [93]	**0.42**	0.1	0.60	0.6	0.50	0.3	0.30	0.1	-	-	-	-	-	-		
OF_EV_SNN [96]	0.58	-	0.72	-	0.67	-	-	-	-	-	-	-	-	-	-	-
TMA [40]	1.06	3.6	1.81	27.2	1.58	23.2	0.25	0.1	-	-	-	-	-	-	-	-
$\text{ADM-Flow}_{\text{D}}$ [113]	0.48	0.1	**0.56**	**0.4**	**0.47**	**0.0**	0.52	**0.0**	-	-	-	-	-	-		
TamingCM [94]	0.44	**0.0**	0.88	4.5	0.70	2.4	0.27	0.1	-	-	-	-	-	-		
Real-Time																
Akolkar *et al.* [105]	1.52	-	1.59	-	1.89	-	2.75	-	-	-	4.47	-	-	-	-	-
FireFlowNet [48]	0.97	2.6	1.67	15.3	1.43	11.0	1.06	6.6	-	-	-	-	-	-	-	-
RTEF	0.52	**0.1**	0.98	5.5	0.71	2.1	**0.53**	**0.2**	**0.74**	**1.2**	**2.91**	**30.6**	**3.45**	**39.1**	**3.62**	**39.8**
RTEF_{NDF}	**0.49**	**0.1**	**0.92**	**4.6**	**0.68**	**1.6**	0.54	0.4	0.75	1.3	2.99	31.8	3.56	40.4	3.70	40.9
$\text{RTEF}_{\text{DS\_L}}$	1.81	16.4	2.54	26.4	1.95	18.2	2.12	21.7	1.30	8.7	4.04	45.8	4.78	55.6	5.10	58.7
$\text{RTEF}_{\text{DS\_LB}}$	0.62	0.3	1.02	5.6	0.79	1.8	0.64	0.5	0.79	1.3	3.05	32.5	3.68	41.8	3.90	43.4
$\text{RTEF}_{\text{DS\_Log}}$	0.70	1.4	1.07	6.5	0.82	2.4	0.69	1.6	0.79	1.9	3.08	33.0	3.70	42.1	3.93	43.8

*The authors of EV-FlowNet provide an updated version of their model (`https://github.com/daniilidis-group/EV-FlowNet/`), which we recomputed their results with.

Table 3.2 – Results on the MVSEC dataset. Best results for non real-time and real-time versions are indicated separately.

From these results, we obtain AEEs in the order of one pixel, except for nighttime sequences, where the longer accumulation time of $\Delta t \simeq 97$ ms and the presence of flickering lights result in greater magnitudes of errors. Our AEE results are remarkably always close to or even better than all the non-real-time state-of-the-art approaches (MultiCM [93] and $\text{EV-FlowNet}_{\text{HQF}}$ [111] notably). We display vastly better results than FireFlowNet [48], which is our main comparison point when it comes to fast EBOF methods.

Outlier percentages are also very low, only increasing for the nighttime driving scenes. However, as noted by Ye *et al.* [114], MVSEC ground truth flow is valid only for static world; the moving objects, numerous in the nighttime scenes, could not be kept in the reference, creating errors in the ground truth.

When compared to the ablated versions of our method, it can be seen that the "No denoising" version RTEF_{NDF} performs slightly better for the indoor sequences, where the lighting of the scene is controlled, and the noise therefore less prominent. In that case, the denoising and filling step will tend to eliminate small texture details from the scene, which could in reality be kept to improve the stability of its appearance. On the outdoor sequences, on the contrary, enabling our denoising allows for slightly better results, as the noise generated by the environment becomes much more essential to discard to obtain accurate optical flow results. Regarding the distance surface alternatives, they all display worse results than the proposed negated exponential formulation, both for indoor and outdoor sequences; the original linear distance surface $\text{RTEF}_{\text{DS\_L}}$, notably, displays here the worst results, even worse than the zero flow baseline in some sequences.

Despite the presence of a ground truth in MVSEC dataset, we also computed the FWL metric, in Table 3.3. Our results consistently surpass the value of 1, indicating an optical flow estimation better than the zero flow reference. Most importantly, they surpass those of $\text{EV-FlowNet}_{\text{MVSEC}}$ [32] and $\text{EV-FlowNet}_{\text{HQF}}$ [111] in most of the sequences. While these results may sometimes slightly contrast with those presented in Table 3.2, they actually further underline the inconsistencies in the ground truth flow of the MVSEC dataset. Learning-based methods will tend to mimic these inconsistencies, resulting in better AEE values overall on the dataset, but will in reality yield more incorrect optical flow values, hence the lower FWL results.

	indoor_flying_1*	indoor_flying_2*	indoor_flying_3*	outdoor_day_1*	outdoor_day_2*	outdoor_night_1	outdoor_night_2	outdoor_night_3
	FWL ↑	FWL ↑	FWL ↑	FWL ↑	FWL ↑	FWL ↑	FWL ↑	FWL ↑
$\text{EV-FlowNet}_{\text{MVSEC}}$ [32]†	1.02	1.13	1.06	1.15	**1.21**	-	-	-
$\text{EV-FlowNet}_{\text{HQF}}$ [111]	1.14	1.36	1.23	**1.27**	1.20	-	-	-
RTEF	**1.21**	**1.41**	**1.37**	1.17	1.17	**1.35**	**1.45**	**1.42**
RTEF_{NDF}	1.16	1.34	1.29	1.15	1.14	1.28	1.37	1.34
$\text{RTEF}_{\text{DS\_L}}$	1.16	1.29	1.27	1.13	1.12	1.21	1.28	1.25
$\text{RTEF}_{\text{DS\_LB}}$	1.20	**1.41**	1.35	1.16	1.17	1.34	1.43	1.39
$\text{RTEF}_{\text{DS\_Log}}$	**1.21**	1.39	1.36	1.16	1.16	1.32	1.41	1.38

*Stoffregen *et al.* [111] selected cuts from these sequences to evaluate their method; we use the same extracts here.
†As computed by Stoffregen *et al.* [111].

Table 3.3 – FWL results on the MVSEC dataset.

Regarding the FWL comparison with the alternative methods, proposed denoised negated exponential formulation displays the best results for all sequences. The RTEF_{NDF} version consistently performs worse, as the small details in the scene, not discarded as noise here, compensate more difficultly than the main edges and lower the results yielded by the FWL metric. Once again, the alternative distance surface formulations all provide worse results than the negated exponential one. However, it should be underlined here that the "linear-bound" $\text{RTEF}_{\text{DS\_LB}}$ and the "logarithmic" $\text{RTEF}_{\text{DS\_Log}}$ methods both display results closer to the negated exponential distance surface than anticipated.

Finally, we present in Fig. 3.5 qualitative optical flow results for sequences from the MVSEC dataset. It can be seen that our EBOF is visually close to the reference. Limitations in the ground truth of the dataset can also be observed: the hood of the car is not taken into account in the ground truth of the outdoor sequences (second and third rows), and the moving vehicle at the right of the car in the last row is associated with an incorrectly smoothed ground truth flow. If the reader

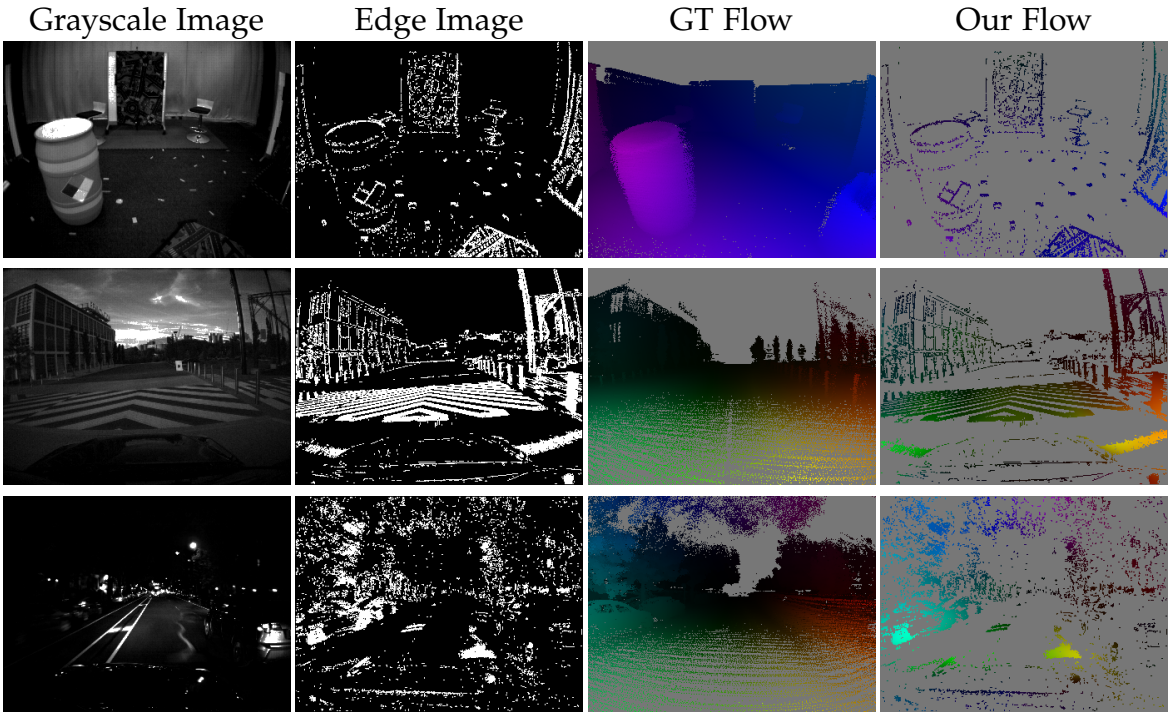


Figure 3.5 – Qualitative results on MVSEC dataset. Sequences, from top to bottom: `indoor_flying_1`, `outdoor_day_1`, and `outdoor_night_1`. Best viewed in color.

is interested, complete video results for the `indoor_flying_1`, `outdoor_day_1`, and `outdoor_night_1` sequences are available through the project page link given at the very beginning of this chapter (Page 21).

3.6.6 *Evaluation on the DSEC Dataset*

To complete the results obtained on the low-resolution data of the MVSEC dataset, we also evaluate our method on the mid-resolution (640×480) DSEC dataset of Gehrig *et al.* [63].

While this dataset does not have the same issues of approximate calibration, synchronization, and erroneous ground truth data the MVSEC dataset has, it still has one main limitation: its ground truth optical flow maps are given for temporal windows of events of $\Delta t = 100$ ms. As noted by Gehrig *et al.* [39], this means that their dataset contains ground truth optical flow magnitudes of up to 240px, and as noted by Shiba *et al.* [93], 20% of the ground truth optical flow magnitudes are over 22px.

As we observed for the night sequences of the MVSEC dataset ($\Delta t \simeq 97$ ms), high accumulation times yield bad performances for our method, as noted in Table 3.2. Therefore, when we first tested our method on the DSEC dataset with the original accumulation time of $\Delta t = 100$ ms, the results were disastrous, due to the combination of a very high amount of events and large motions, resulting in a lot of motion blur in the edge images, and therefore a very bad optical flow estimation. It

should also be noted that the "flow filter" method used for computing our optical flow tends to converge towards correct results only after several images, so the high accumulation times are once again not beneficial in our case.

Therefore, we decided to compute the optical flow over $\Delta t = 20$ ms of accumulation time instead, yielding clearer edge images, and therefore better optical flow results. To compensate for the time difference, we multiply obtained optical flow vectors by a factor of 5, to obtain an equivalent optical flow over 100ms. We acknowledge that this procedure is highly criticizable; as noted by Shiba *et al.* [93], over 100ms, the very high magnitudes of the ground truth optical flow indicate that the classical assumption of scene points flowing along linear trajectories does not hold, and that therefore optical flow over 20ms is not actually $1/5$ of the optical flow over 100ms.

An alternative solution would have been to still estimate optical flow over 20ms, but instead of multiplying the estimations by a factor of 5, to rather concatenate corresponding optical flow vectors over 5 successive estimations to result in one final estimation over a time of 100ms. This procedure would take into account the non-linear trajectories of the points in the scene, which would be more correct. However, accumulating optical flow vectors is a complex procedure, as these vectors contain floating point values: while they originate from the center of a pixel, they do not point at the center of one destination pixel, but rather in-between multiple pixels. In addition, small errors in each estimation of the optical flow would be accumulated, resulting in the end in a potentially erroneous accumulation procedure, and therefore even larger errors.

This is why the "multiplication by a factor of 5" procedure was chosen: it makes the evaluation and the analysis of errors simpler, while considering that the DSEC dataset may not actually be the most adapted for our use case (but to this day, it remains the only dataset with mid-resolution events and ground truth optical flow data available).

Another issue with this dataset should also be noted: dense optical flow results are expected during evaluation, whereas our method was initially designed for sparse results, as explained in Section 3.5.4. Therefore, we had to remove the final mask we apply on the optical flow results, and adapt subsequently the configuration of the "flow filter" method for having smoother results over the whole image (hence the high number of smoothing iterations for the two first layers, as noted in Section 3.6.2).

We report in Table 3.4 our results, compared with those from other state-of-the-art methods. While we reuse the AEE metric described for the MVSEC dataset in Eq. (3.14), as well as the percentage of outliers considering maximal acceptable errors of 1, 2, and 3px, we also give here a measure of the Average Angular Error (AAE), defined as follows:

$$\text{AAE} = \frac{1}{N} \sum_{i=1}^{N} \arccos(f_i \cdot \hat{f}_i) \tag{3.16}$$

From these results, it can be noted that the AEE and percentage of outliers are similar to those obtained on the night sequences of the MVSEC dataset for a similar

accumulation time. We display overall the worst results, but as noted before, our method was never thought nor adapted for very high magnitudes or dense flow estimation, and our evaluation protocol is very error-prone. It can also be noted that the best results on this dataset are obtained by learning-based methods, which are by nature more suited for this complex 100ms case, as they are able to mimic what they learned on the training data, and do not rely on traditional optical flow equations which are not adapted for this case.

	Learning-based?	AEE ↓	AAE ↓	% outliers (1px) ↓	% outliers (2px) ↓	% outliers (3px) ↓
TMA [40]	Yes	**0.74**	**2.68**	**10.86**	**3.97**	**2.30**
E-Flowformer [115]	Yes	0.76	**2.68**	11.23	4.10	2.45
E-RAFT [39]	Yes	0.79	2.85	12.74	4.74	2.68
OF_EV_SNN [96]	Yes	1.71	6.34	53.67	20.24	10.31
TamingCM [94]	Yes (self-sup.)	2.33	10.56	68.29	33.48	17.77
MultiCM [93]	No	3.47	13.98	76.57	48.48	30.86
RTEF	No	4.88	10.82	82.81	57.90	41.95

Table 3.4 – Overall results on the DSEC dataset.

Following the intuition of Shiba *et al.* [93], we also report in Table 3.5 the FWL results on the test sequences of the DSEC dataset. Surprisingly, despite having the worst AEE, AAE, and percentage of outliers, our method achieves much better FWL results on all test sequences by a large margin. We are unsure as to why FWL values computed by Shiba *et al.* [93] for both E-RAFT and MultiCM are so low, as uncompensated images of events accumulated over 100ms are very blurry (as shown in Fig. 3.6), and as they display good compensation results in their article (but for accumulation times smaller than 100ms).

	All			`interlaken_00_b`			`interlaken_01_a`			`thun_01_a`		
	AEE ↓	% out ↓	FWL ↑	AEE ↓	% out ↓	FWL ↑	AEE ↓	% out ↓	FWL ↑	AEE ↓	% out ↓	FWL ↑
E-RAFT [39]	**0.79**	**2.68**	1.29*	**1.39**	**6.19**	1.32*	**0.90**	**3.91**	1.42*	**0.65**	**1.87**	1.20*
MultiCM [93]	3.47	30.86	1.37	5.74	38.93	1.50	3.74	31.37	1.51	2.12	17.68	1.24
RTEF	4.88	41.95	**2.51**	8.59	59.84	**2.89**	5.94	47.33	**2.92**	3.01	29.70	**2.39**

	`thun_01_b`			`zurich_city_12_a`			`zurich_city_14_c`			`zurich_city_15_a`		
	AEE ↓	% out ↓	FWL ↑	AEE ↓	% out ↓	FWL ↑	AEE ↓	% out ↓	FWL ↑	AEE ↓	% out ↓	FWL ↑
E-RAFT [39]	**0.58**	**1.52**	1.18*	**0.61**	**1.06**	1.12*	**0.71**	**1.91**	1.47*	**0.59**	**1.30**	1.34*
MultiCM [93]	2.48	23.56	1.24	3.86	43.96	1.14	2.72	30.53	1.50	2.35	20.99	1.41
RTEF	3.91	34.69	**2.48**	3.14	34.08	**1.42**	4.00	45.67	**2.67**	3.78	37.99	**2.82**

*As computed by Shiba *et al.* [93].

Table 3.5 – FWL results on the DSEC dataset.

Despite it all, we can still highlight the fact that E-RAFT [39] displays the worst FWL results, even though it has the lowest AEE and the lowest percentage of outliers by a large margin. As noted during the evaluation on the MVSEC dataset in Section 3.6.5, this might once again indicate that the ground truth optical flow values in the DSEC dataset are imperfect, and that learning-based methods like E-RAFT will learn to replicate these errors, resulting in lower FWL values.

Finally, we show in Fig. 3.6 some qualitative results. As can be seen, in all three cases shown, the events are well compensated by the optical flow, allowing for sharp images of warped events. This is especially the case for the text on the ground in the

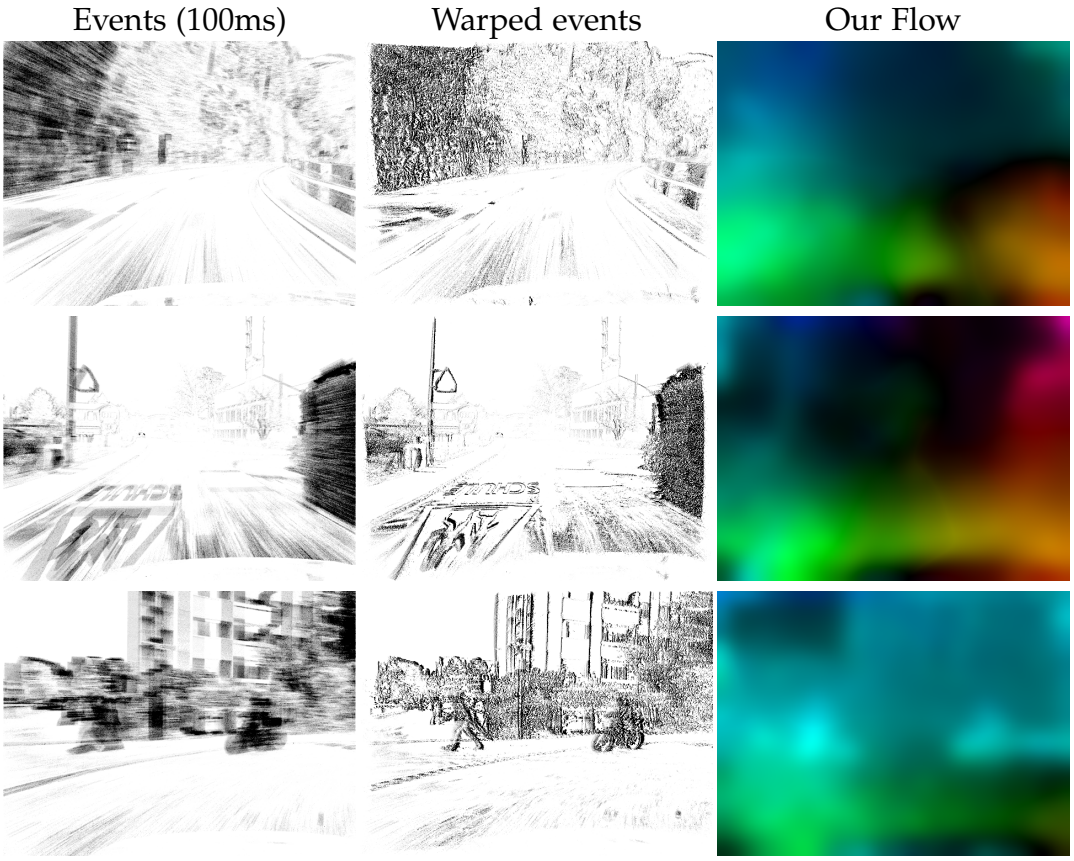


Figure 3.6 – Qualitative results on the DSEC dataset. Sequences, from top to bottom: `interlaken_00_b`, `thun_01_b`, `zurich_city_15_a`. Best viewed in color.

second row, or the pedestrian and the cyclist in the third row. As for the optical flow itself, it describes well visually the movement in the scene. However, limitations can still be highlighted for the optical flow maps: they have a very smooth appearance, lacking precision for small details, and by construction of our method, incorrect flow estimations can be seen in the parts where no input event is available. These limitations contribute to the high errors of our method described in Table 3.4.

3.6.7 *Evaluation on the 1 Megapixel Automotive Detection Dataset*

The increasing availability of high-resolution neuromorphic cameras means that a more thorough evaluation including these new sensors has to be conducted. In the context of this work, the 1 Megapixel Automotive Detection dataset [25] from Prophesee — while initially intended for automotive object recognition purposes — appears as the most complete, publicly available high-definition baseline for conducting our evaluation. Given its density (1.2 TB, 14 hours of data), we settled on the use of only its "test" sequences for the evaluation, which account for a total of 2 hours of raw data.

We initially tried to adapt the codes of the low-resolution EV-FlowNet methods proposed by Zhu *et al.* [32] and Stoffregen *et al.* [111], to provide the same compar-

isons as on the MVSEC dataset. However, the results they yielded were catastrophic, as their neural-network-based approaches were not designed nor trained for high-resolution input data. The most apparent and limiting issues are the computation times (the code was not optimized for high-resolution data), and the absence of denoising (it created large artifacts in the optical flow results).

	Jan.30	Feb.15	Feb.18	Feb.19	Feb.21	Feb.22	Apr.12	Apr.18	Jun.11	Jun.14	Jun.17	Jun.19	Jun.21	Jun.26	*Global*
RTEF	**1.63**	1.47	**1.34**	**1.64**	**1.36**	**1.51**	1.14	1.54	**1.80**	**1.35**	**1.46**	**1.38**	**1.54**	1.56	**1.46**
RTEF_{NDF}	1.43	1.37	1.25	1.52	1.28	1.41	1.08	1.43	1.63	1.27	1.35	1.31	1.46	1.46	1.37
$\text{RTEF}_{\text{DS\_L}}$	1.44	1.40	1.32	1.41	1.29	1.38	**1.21**	1.27	1.50	1.28	1.27	1.30	1.32	1.42	1.34
$\text{RTEF}_{\text{DS\_LB}}$	**1.63**	**1.48**	1.33	1.63	1.35	1.49	1.14	**1.55**	1.78	1.34	1.43	1.37	1.52	**1.57**	1.45
$\text{RTEF}_{\text{DS\_Log}}$	1.60	1.47	1.33	1.60	1.34	1.47	1.14	1.54	1.75	1.33	1.41	**1.38**	1.51	1.56	1.43

Table 3.6 – FWL results on Prophesee's 1 Megapixel Automotive Detection dataset.

We therefore present our FWL results on this dataset (since it does not contain any ground truth for optical flow) in Table 3.6, split between each recording day. Similarly to what was observed for low-resolution data, our method always yields FWL values greatly superior to 1, indicating an accurate optical flow. Compared to the ablation alternatives, it can also be observed that our final version displays the best global result, and the best results in all sequences but four (`Feb.15`, `Apr.12`, `Apr.18`, and `Jun.26`, where it is the second best alternative), showing once again the importance of the denoising and of our negated exponential distance surface. Also, similarly to what was observed on the MVSEC dataset, both the $\text{RTEF}_{\text{DS\_LB}}$ and $\text{RTEF}_{\text{DS\_Log}}$ variants offer surprisingly good FWL results.

3.6.8 *Complementary Evaluation on a 20-minute-long High-Resolution Driving Sequence*

While Prophesee's 1 Megapixel Automotive Detection dataset allows for a reproducible evaluation of EBOF methods with the FWL, it contains only event recordings, preventing the comparison with frame-based state-of-the-art methods. In order to complete our evaluation, we make use in this subsection of a twenty-minute-long driving sequence, containing both the output from high-definition frame-based and event-based cameras. This sequence presents a wide diversity of driving situations (urban/rural roads, highway, roundabouts, many other vehicles, pedestrians, ...). It has been recorded by and graciously shared with us by Prophesee.

Village (0'00 - 4'00)	Side Road (4'00 - 7'00)	Highway (7'00 - 11'00)	Suburban (11'00 - 14'30)	Urban (14'30 - 20'45)	*Full sequence* (0'00 - 20'45)
1.70	1.45	1.67	1.62	1.43	1.56

Table 3.7 – FWL results on the 20-minute-long driving sequence.

The FWL results of this evaluation are presented in Table 3.7, split between each period of the sequence. An overall FWL result on the complete sequence is also presented. It can be observed that our FWL results are greatly satisfying, by remaining largely over the value of 1.

The main advantage of this sequence, however, lies in the possibility to compare our EBOF results to frame-based ones. For this prospect, we used RAFT [89] as the

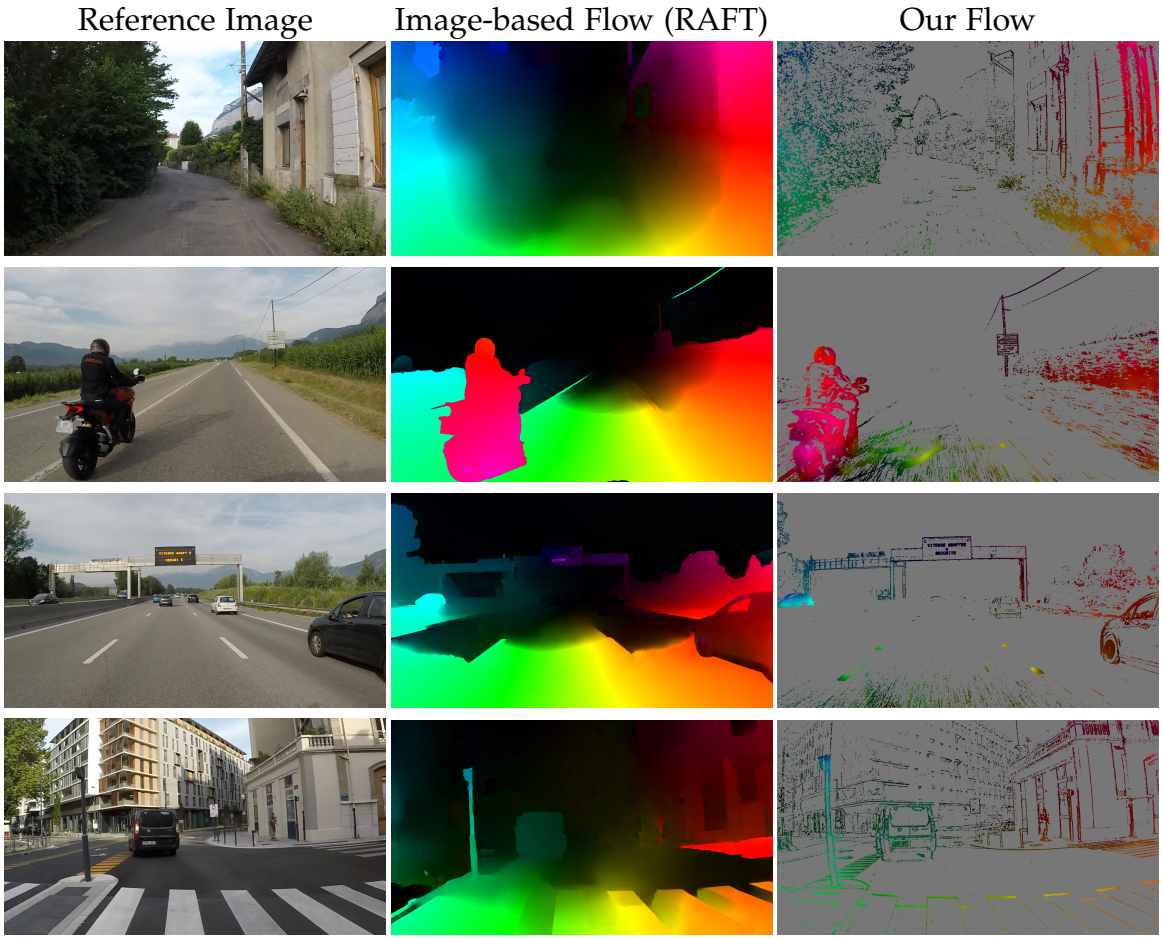


Figure 3.7 – Qualitative results on the 20-minute-long driving sequence. Extracts used, from top to bottom: a village street, a motorcycle overtaking, a highway, and an intersection. Best viewed in color.

frame-based reference, as it currently stands as one of the state-of-the-art optical flow methods. Due to the lack of calibration between the two sensors, only a qualitative evaluation can be presented. Therefore, several visual optical flow results are given in Fig. 3.7. Results on the full sequence can also be viewed in video format, through the link of the project page given at the very beginning of this chapter (Page 21). It can be seen that, similarly to when a low-resolution input is used, our optical flow remains visually very close to the reference. Also, compared to RAFT, our method does not suffer from the lack of texture of some areas, like the road for instance.

3.6.9 *Evaluation on our High-Speed High-Definition Event-Based Indoor Dataset*

For high-resolution event-based sequences, most of our optical flow results are restricted to a few pixels, due to the low accumulation time of $\Delta t = 15$ms we used throughout this work, in accordance to movements speed. In order to show how our method is able to handle movements of higher magnitudes in various situations, we recorded a high-speed high-definition event-based dataset, using a Prophesee

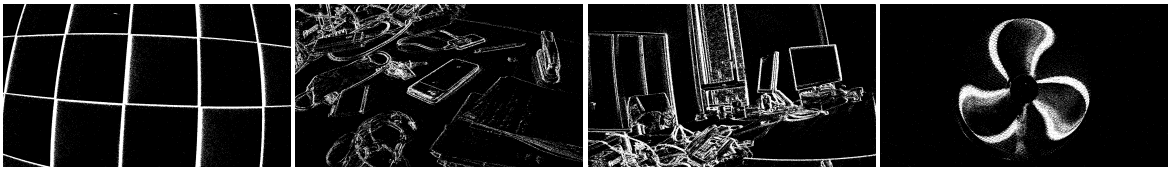


Figure 3.8 – Edge images of our high-speed high-definition event-based indoor dataset sequences. From left to right: `Checkerboard`, `Desk`, `Office`, `Fan`.

Figure 3.9 – Sample EBOF results for the `Fan` sequence of our high-speed dataset, with $\Delta t = 15$ms (left) and $\Delta t = 5$ms (right). Notice how the blades of the fan are merged together in the first case, while they appear clearly in the second one, leading to improved optical flow results.

Gen4 camera. This dataset is composed of four indoor sequences taken in an office environment (namely, `Checkerboard`, `Desk`, `Office`, and `Fan`). The first three of them were recorded by manually shaking the camera, while for the last one, the camera was fixed in front of a high-speed fan. An illustration of these sequences is given in Fig. 3.8.

	`Checkerboard`	`Desk`	`Office`	`Fan`
$\Delta t = 15$ms	1.49	1.71	1.74	1.05
$\Delta t = 5$ms	1.75	1.66	1.84	1.53

Table 3.8 – FWL results on our high-speed high-definition event-based indoor dataset.

This evaluation relies also on the FWL metric, to compare ourselves to the "zero flow" reference. The results are presented in Table 3.8. It can be seen here that we always obtain a FWL greater than 1, underlining once again the accuracy of our optical flow results, even under larger apparent motions. However, apart for the `Desk` recording, all recordings display better FWL results when a lower accumulation time of $\Delta t = 5$ms is employed (which is non real-time, see Section 3.6.11). This is due to the fact that, at such high motion speeds, the edge images become slightly too blurry to provide the best optical flow results when the accumulation time of $\Delta t = 15$ms is employed. Lowering accumulation time helps to obtain sharper edge images, and in return, more accurate optical flow and FWL results. This remark is especially true for the `Fan` sequence, where the very high speed of the blades leads them to appear too blurry for a correct optical flow to be computed when $\Delta t = 15$ms is used; an illustration is given in Fig. 3.9.

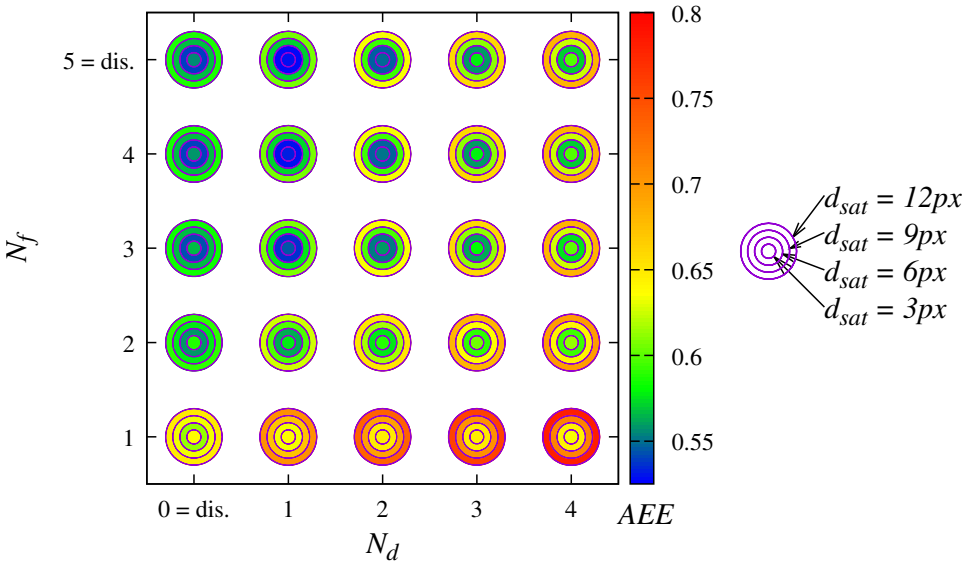


Figure 3.10 – Sensitivity analysis of our EBOF method, using the AEE on the `outdoor_day_1` sequence from the MVSEC dataset, along the N_d, N_f, and d_{sat} parameters. Lowest AEE (blue) is the best.

3.6.10 *Sensitivity Analysis*

In this subsection, we analyze the sensitivity of N_d, N_f and d_{sat} parameters on our EBOF results. As a reminder, N_d and N_f define denoising and filling thresholds (Section 3.5.2), while d_{sat} defines the saturation value for the computation of the negated exponential distance surface (Section 3.5.3). For that purpose, we use the `outdoor_day_1` sequence from the MVSEC dataset with the AEE metric. We display in Fig. 3.10 these results for $N_d \in \{0 = \text{“disabled”}, 1, 2, 3, 4\}$, $N_f \in \{1, 2, 3, 4, 5 = \text{“disabled”}\}$, and $d_{\text{sat}} \in \{3, 6, 9, 12\}$ pixels.

From this plot, it can first be noted that the denoising should not be too strong, that is, $N_d \leq 2$. In the same time, the filling threshold should also remain reasonable, i.e., $N_f \geq 3$. Overall best d_{sat} value is incontestably $d_{\text{sat}} = 6$px, and it is the most sensitive parameter. From this analysis, the best set of parameters is $\{N_d = 1, N_f = 4, d_{\text{sat}} = 6\}$, with the corresponding minimal AEE of 0.53px. The worst set of parameters is $\{N_d = 4, N_f = 1, d_{\text{sat}} = 12\}$, with a corresponding AEE of 0.78px, increasing by 47% compared to the set of the best parameters. This shows the importance of the choice of parameters to guarantee adequate optical flow results.

3.6.11 *Real-Time Compliance*

To finally demonstrate the real-time compliance of our approach for both low-, mid-, and high-resolution input data, we used respectively the `indoor_flying_1` sequence from the MVSEC dataset, the `thun_01_b` sequence from the DSEC dataset, and the `Jan.30` test sequence from Prophesee's 1 Megapixel Automotive Detection dataset. The execution time for each block was not evaluated separately, but simultaneously,

as the full pipeline should be able to run on a single computer. In order to also underline the benefits brought by the use of the GPU, we ran the evaluation on both the CPU-only and the CPU+GPU versions of our code.

	Edge image (after accumulation)*	Denoising & filling	Inv. exp. distance transform	Optical flow†	Total
Low-resolution (346 × 260)					
CPU-only	0.07±0.02	0.76±0.16	1.62±0.85	*2.81±0.08*	5.27±0.93
CPU & GPU	0.15±0.05	0.42±0.02	1.29±0.08	3.46±0.12	5.31±0.20
Mid-resolution (640 × 480)					
CPU-only	0.82±1.25	0.30±0.03	13.00±7.46	*16.41±3.27*	30.59±9.31
CPU & GPU	0.42±0.16	0.33±0.57	1.15±0.79	15.38±0.94	17.28±2.02
High-resolution (1280 × 720)					
CPU-only	0.64±0.43	5.00±0.63	22.54±3.03	*12.77±0.62*	40.96±3.27
CPU & GPU	0.47±0.04	0.66±0.06	2.69±0.45	10.41±0.85	14.21±1.22

*This module uses only the CPU.
†The optical flow library provided by Adarve *et al.* [109] does not contain a CPU-only version. The CPU-only experiments therefore use the GPU for optical flow computation, hence the similar results between the "CPU-only" and "CPU & GPU" lines for this column.

Table 3.9 – Average execution times and standard deviation of each step of our RTEF method for low-, mid-, and high-resolution inputs, in milliseconds.

These results are presented in Table 3.9. From them, considering the worst cases, it can be seen on one hand that the GPU-aided version can achieve real-time performances for both low-, mid-, and high-resolution input data, if their accumulation times are set to at least 4ms, 17ms, and 12ms respectively (the limiting factor being the optical flow module). The values for Δt used throughout this article, therefore, respect the real-time constraint. The CPU-only version, on the other hand, can achieve real-time performances for both low- and high-resolution data when the accumulation time is of at least 3ms, 21ms, and 26ms respectively (the limiting factor being here the optical flow module for low-resolution, and the distance transform module for mid- and high-resolution). As noted in Table 3.9, however, the optical flow can only be computed on GPU due to the use of the library of Adarve *et al.* [109], and transferring this computation to the CPU would likely make these execution times greatly increase.

From the results of Table 3.9 and from the previous conclusions, using our RTEF architecture, the best performances that can be reached for a low-resolution input is therefore a 250Hz flow with a latency of 10ms (for the minimal $\Delta t = 4$ms of accumulation time, including it in the computation of the latency). For a mid-resolution input, a 59Hz flow with a latency of 34ms can be achieved (including $\Delta t = 17$ms of accumulation time). For a high-resolution input, a 83Hz flow with a latency of 27ms can be achieved (including $\Delta t = 12$ms of accumulation time). An important note on the higher times for mid-resolution data is that they were computed with the configuration used on the DSEC dataset (see Sections 3.6.2 and 3.6.6), which requires more smoothing iterations in the optical flow method to achieve correct dense results, thus putting more burden on the GPU.

As a fast algorithm, FireFlowNet [48] displays a theoretical inference frequency up to 262Hz for a low-resolution (346 × 260) input, similar to ours. For a higher definition (1280 × 720), however, our approach achieves frame rates nearly 3-times better than them, as their method can only reach 29Hz. The GPU they use ranks

similarly to ours in popular benchmarks. In addition, we ran EV-FlowNet [32] on low- (346×260) and on high-resolution (1280×720) data. While EV-FlowNet inference achieved a 125Hz flow on low-definition data, it showed its limits on high-resolution input with a 12.5Hz flow output. These frequencies consider only inference process, full latency is unknown and should include events accumulation time and specific dense representation creation.

To finally compare with a state-of-the-art frame-based optical flow algorithm, we used RAFT [89], and measured a 12.5Hz output on low-resolution images, and a 2Hz one on high-definition images.

3.7 CONCLUSION AND DISCUSSIONS

Throughout this chapter, a complete pipeline for real-time computation of Event-Based Optical Flow from both low-, mid-, and high-resolution event cameras has been proposed. It includes optimized algorithmic choices as well as a novel negated exponential distance surface representation. Several evaluations have been conducted to show the relevance of our contributions. Resulting accuracies surpass or are close to the non-real-time state of the art for low-definition recordings, as well as on high-definition sequences. Results on the mid-resolution sequences of the DSEC dataset are more contrasted, but we highlight the limitations of the dataset in our case. Frame rates of respectively 250Hz, 59Hz, and 83Hz for resolutions of 346×260, 640×480, and 1280×720 were also achieved, making it, to the best of our knowledge, the most accurate EBOF method for low- and especially high-resolution event cameras that could be deployed in the wild.

To complete the observations formulated in this chapter, deep learning dominates EBOF estimation when looking for accuracy at the cost of real-time ability. The TMA [40] architecture is a perfect example of this trend, as shown in Table 3.4. On the contrary, when real-time running is needed, this is currently our proposition, which is not a neural network, that provides the overall best results. As our method is not based on machine learning, it is independent of a training process involving specific datasets and loss functions. In other words, the results are expected to be similar to the ones presented here for all types of scenes. This can be seen as an advantage, compared for example to TMA, which goes from being the state of the art on the outdoor driving sequences of MVSEC, to being one of the worst performing methods for the indoor flying scenes of the very same dataset, as shown in Table 3.2.

In hindsight, several improvements could be brought to the current method. **(1)** Regarding the event accumulation process, relying on a predetermined fixed accumulation time Δt, which highly depends on the appearance and dynamics of the visual scene, can indeed lead to instabilities in the appearance of the edge images if not chosen carefully. Introducing an adaptive method to dynamically determine the correct accumulation time to use, as proposed by Liu and Delbrück [116] for instance, could allow for obtaining clear edge images independently of the scene evolution, but would also certainly make the real-time constraint harder to achieve.

(2) Our architecture could also benefit from a more optimized implementation on specialized hardware (FPGA for instance, eliminating the need for an energy-intensive GPU), which could also allow for even higher frame rates by lowering the minimum accumulation times. **(3)** Applying our EBOF to complex automotive-related applications, such as proposed by Jung and Park [117], could also be addressed by future work, for instance for improving the pose estimation of the vehicle.

Event-based optical flow is a fast-growing area of research. At ICCV 2023 alone, 5 articles were published on that topic: [40, 94, 113, 118, 119]. While our approach of opting for real-time compatibility at the expense of some accuracy made sense for an application to the real-world, it limited us in exploiting it further as part of this thesis. In particular, several attempts were made to extract independently moving objects using the optical flow during the early months of the second year of the thesis, but each time the lack of precision of our optical flow maps limited our accuracy. We believe and hope that future works in the domain will be able to combine both the precision of the recently published method and real-time compatibility, with contrast-maximization-based approaches looking especially promising on that topic.

Nonetheless, a final topic of discussion for this work could be on the intrinsic sense of computing optical flow from events the way we did. To clarify this question, let us go back to the initial definition of events and optical flow. As shown in Eq. (2.1), events are a description of changes (i.e., the temporal derivative of the brightness information). But as shown in Eq. (3.1), optical flow is also a description of changes (i.e., the motion of pixels between two instants of time). As such, describing the optical flow of events is describing changes of changes themselves, which does not really make sense. To answer this issue, in this chapter, we introduced a temporal integration process to compensate for the "derivative-based" nature of events. We then considered that the images of integrated events are similar to an edge map (i.e., a spatial derivative) of the objects in the scene, on which computing a change along time makes more sense. Yet, our images are not really edge maps, they are merely accumulated changes; in the case of a fully static scene for instance, no event is produced and our "edge images" are empty, while a real edge map would still be able to describe the contours of the objects. Therefore, our formulation of the event-based optical flow problem is probably ill-posed, and it could be argued that this is also the case for the other works in the domain which rely on events to describe edges. In that regard, it could be argued that the approach of Pan *et al.* [102] for instance makes a better use of the event camera, as they acknowledge that events are only fine-grained temporal changes: they use them only for deblurring frames, and they compute optical flow on these deblurred frames. Nonetheless, this first work on optical flow was fundamental for giving us a better understanding of the unique problems posed by event cameras, and despite not being exploited further, it laid the groundwork for the following works of the thesis.

4 EVENT- AND LIDAR-BASED DEPTH ESTIMATION USING A CONVOLUTIONAL NETWORK

As noted in the conclusion of Chapter 3, despite our good optical flow results, our experiments showed that their slight lack of precision was a critical issue if we wanted to exploit them further. In an objective to solve the "depth" part of the subject of the thesis, we started investigating the fusion of events with LiDAR data, in order to be able to estimate dense depth maps. Our approach also fundamentally changed: while the real-time, geometry-based approach for computing optical flow was an interesting problem to solve, we realized that it limited us in terms of accuracy, and that our results were starting to be difficult to defend compared to the most recent learning-based methods. The characteristics of the problem we aim to solve (detailed in Section 4.3) also require a good understanding of the scene, which would be difficult to model with a purely geometry-based method. Therefore, we describe in this chapter an offline, learning-based approach for estimating dense depth maps from the fusion of LiDAR and event data.

The presented method and the associated results of this chapter were published as part of the 22nd Scandinavian Conference on Image Analysis in April 2023 [120]. A project page is also available at `https://vbrebion.github.io/ALED/`, and contains the links to the original article, the source code, the dataset, and videos associated to this work.

4.1 INTRODUCTION

LiDAR sensors offer accurate but sparse 3D information of their surrounding environment. As noted in Chapter 1, they are a key component for intelligent robotic and autonomous navigation, helping to solve multiple problems, e.g., obstacle detection and tracking, SLAM, scene flow, etc. Yet, the sparsity of their point clouds often constitutes a limiting factor. While 64- or 128-channel LiDARs are starting to be commercialized, they come at a significantly high cost, and are still not as dense as cameras.

In this chapter, we focus on the fusion of LiDAR and event data, which we consider as a dual problem: **(1)** LiDAR depths densification and **(2)** events-depths association. Regarding problem **(1)**, we are interested in densifying the LiDAR data using the events as a guide. As a result, dense depth maps are obtained, which allow for a dense 3D perception of the observed scene. As for problem **(2)**, we are interested in associating a depth to each event. By doing so, each event can be projected in 3D, and then even be backprojected in 2D in another vision sensor. For a fully calibrated and synced setup, this process would allow for the superimposition of events and RGB images from two different cameras for instance.

Estimating dense depth maps from sparse LiDAR data is a well-regarded problem as it solves the sparsity drawback of the LiDAR while keeping its metric scale.

However, using events to densify depth maps (i.e., problem **(1)**) might inaccurately be seen as a task that inherently includes problem **(2)**, where corresponding depths for the events could be taken from the dense depth map. We argue in this chapter that, as each event represents a change in illumination, it might also represent a change in depth. As such, two depths should be associated to each event, and we will therefore compute two depth maps: one before the events happen, and one after they happen.

As an answer to these issues, we propose in this chapter a learning-based fusion method for estimating pairs of dense depth maps from events and sparse LiDAR data. For that purpose, we propose a novel convolutional network, the Asynchronous LiDAR and Events Depths densification network (ALED), able to fuse asynchronous events and LiDAR data, and to estimate the two dense depth maps from them, while surpassing state-of-the-art accuracy. We also build a high-definition simulated dataset, the Synthetic LiDAR Events Depths (SLED) dataset, used as part of the training of the network and its evaluation.

This chapter is structured as follows. We first give an overview of the state of the art in Section 4.2. We then examine the duality of the problem and the issue of associating a single depth to each event in Section 4.3. We give a detailed description of our ALED network and of our SLED dataset in Sections 4.4 and 4.5 respectively. We finally conduct our evaluation in Section 4.6, before drawing some conclusions in Section 4.7.

4.2 RELATED WORK

4.2.1 *LiDAR Densification*

Point clouds produced by LiDAR sensors are sparse, which is challenging for numerous applications (3D reconstruction, object detection, SLAM, etc). As a consequence, LiDAR depth completion is a subject that has been widely studied in the literature.

Some authors try to obtain dense depth maps while only relying on the sparse data from the LiDAR. These methods either use machine learning [121, 122, 123] or traditional image processing operations [124].

The most successful approaches use a secondary modality as a guide for the densification process. While most of these approaches employ an RGB camera as the secondary sensor [123, 125, 126, 127], other authors have proposed using alternative modalities, such as stereo cameras [128] or more recently event cameras [45].

4.2.2 *Fusion of Events and Other Modalities*

Due to their relative youth, the literature on the fusion of data from event cameras with other sensors is quite sparse.

Most of the investigations focused on the fusion of events and frames, thanks to sensors offering both modalities like the DAVIS camera [15]. These works include frame interpolation and deblurring [28, 129, 130], feature tracking [131, 132], object detection [133, 134, 135], or even steering prediction [136].

In the past few years, a few authors have started investigating the fusion of events and LiDAR data. Explored issues concern primarily calibration [137, 138, 139] and dataset construction [62, 63, 64]. More recently, some works have been done on point clouds enhancement with events [14], LiDAR densification [45], and human tracking in adversarial lighting conditions [140].

4.2.3 *Depth Estimation with Events*

Several approaches have been proposed in order to estimate sparse or dense depth maps by using a single event camera. Kim *et al.* [46] used probabilistic filters to simultaneously estimate the motion of the camera, reconstruct a log intensity image of the observed scene, and construct a sparse inverse depth map. Zhu *et al.* [24] used a convolutional network to jointly predict depth and ego-motion, by trying to minimize the amount of motion blur in the accumulated events. Hidalgo-Carrió *et al.* [47] were the first to estimate dense depth maps, through the use of a recurrent convolutional network.

In parallel, other authors have advocated for the use of a secondary sensor to help the depth estimation. Schraml *et al.* [41, 42] and Nam *et al.* [43] used two event cameras, and estimated depths by creating images of accumulated events for each camera and applying stereo matching. While [41, 42] used traditional model-based approaches, [43] entirely relied on learning-based networks: an attention-based network to construct detailed events representations, then a convolutional network for depth map inference. Other authors have also combined the event camera with an RGB sensor; Gehrig *et al.* [33] for instance designed a recurrent network to fuse asynchronous data and estimate dense depths from them. Finally, some authors have also used depth sensors in direct combination with event cameras. Weikersdorfer *et al.* [44] used an RGB-D camera to obtain dense depths, and used the depth-augmented events to perform SLAM. Li *et al.* [14] used a LiDAR sensor to associate a depth to each event through the use of a Voronoi diagram and a set of heuristic rules. Cui *et al.* [45] also employed a LiDAR, to derive dense depth maps by using 3D geometric information.

4.3 DEPTH CHANGE MAP: TWO DEPTHS PER EVENT

In this chapter, we are interested in the fusion of LiDAR and event data. This problem is actually made of two complementary objectives: **(1)** obtaining dense depth maps from events and sparse LiDAR data, and **(2)** assigning a depth to each event. While objective **(1)** can be interpreted as a LiDAR densification method guided by the events, we argue here for objective **(2)** that associating a single depth to an event is inadequate.

(a) Ground truth for D_{bf}

(b) Ground truth for D_{af}

(c) Thresholded depth change map, using the events as a mask

Figure 4.1 – Example of the importance of the depth change map for each event on the `Town01_00` sequence from our SLED dataset. Notice how simple thresholds on this depth difference help distinguishing the events linked to the contour of real objects from the events corresponding to the texture of the road, the halo from the streetlamp, or even the noisy events in the sky.

By definition, an event represents a significant change in illumination observed by a given pixel. Under motion, observed events can either originate from **(a)** texture changes inside an object; or from **(b)** the contour of an object. In case **(a)**, associating a single depth to these events can be coherent, as depth inside an object should be subject to little variation. However, doing so in case **(b)** is erroneous, as events happening at contours of objects are likely to denote also a depth change. This reflection is analogous to the one described in the conclusion of the chapter on optical flow (Section 3.7): we want here to take into account the "change-based" nature of events, and not reproduce the common error of merely considering them as a snapshot of the texture and edges of the objects in the scene.

Therefore, we propose here to always associate *two* depths to each event to take into account this potential change of depth: the depth of the pixel *before* the event (the change) occurred, and the depth of the pixel *after* the event occurred. Associating directly a single "depth change" value to each event could be viewed as an alternative, but we argue that absolute depth values are much more important, as they are required in numerous applications. As we are interested in solving both objective **(1)** and **(2)** simultaneously, we estimate here two dense depth maps: one before the events occur, which we will denote D_{bf} in the rest of this chapter, and one after the events occur, which we will denote D_{af}. We can then formulate the depth change map as $\Delta D \doteq D_{\text{af}} - D_{\text{bf}}$ and compare the depth change $\Delta d \doteq d_{\text{af}} - d_{\text{bf}}$ for

each pixel. Three meaningful cases can be distinguished:

1. $\Delta d \approx 0$: the pixel is located in an area where depths do not vary much, i.e., inside an object;
2. $\Delta d \gg 0$: the pixel was at the edge of an object, and is now on an object further away;
3. $\Delta d \ll 0$: the pixel was located on a far object, and is now at the edge of a closer object.

The depth difference information given by the depth change map can especially help to process events to differentiate real object edges from textures or artifacts such as shadows and even noise. An illustration of some possibilities offered by the depth change map on events is given in Fig. 4.1. Other applications could also take advantage of the pair of depth maps D_{bf} and D_{af}: ego-motion and speed estimation, objects clustering, scene flow, etc.

4.4 METHOD

4.4.1 *The ALED Network*

Inspired by the Recurrent Asynchronous Multimodal Network (RAMNet) architecture of Gehrig *et al.* [33] for RGB and events fusion, we propose here a fully convolutional recurrent network to estimate dense depth maps from asynchronous LiDAR and events data. We call it ALED, for Asynchronous LiDAR and Events Depths densification network.

ALED can be decomposed in two main parts: an encoder, tasked with fusing asynchronous events and LiDAR features at different scales, and a decoder, tasked with interpreting the fused features for estimating dense depths. Both of them are illustrated in Figs. 4.2 and 4.3.

Data Encoding In the encoder part, illustrated in Fig. 4.2, the LiDAR and events inputs are fed independently. Both of them go through an encoding head, computing a first feature map of 32 channels while keeping their original height and width. Convolutional encoders (in the form of ResNet Basic Blocks [141]) are then used to compute feature maps at scales $1/2$, $1/4$, and $1/8$, doubling the number of channels every time.

States Update Each feature map is used as the input of convolutional Gated Recurrent Unit (convGRU) blocks [142], updating the state at its corresponding scale. These states share a double purpose: **(1)** they act as memories, making the network able to produce correct depth maps even in the case where the event camera would become static and thus not produce any event anymore, and **(2)** since they are shared between the LiDAR and events encoders, both parts of the network can update them independently, allowing for an asynchronous fusion of the two modalities.

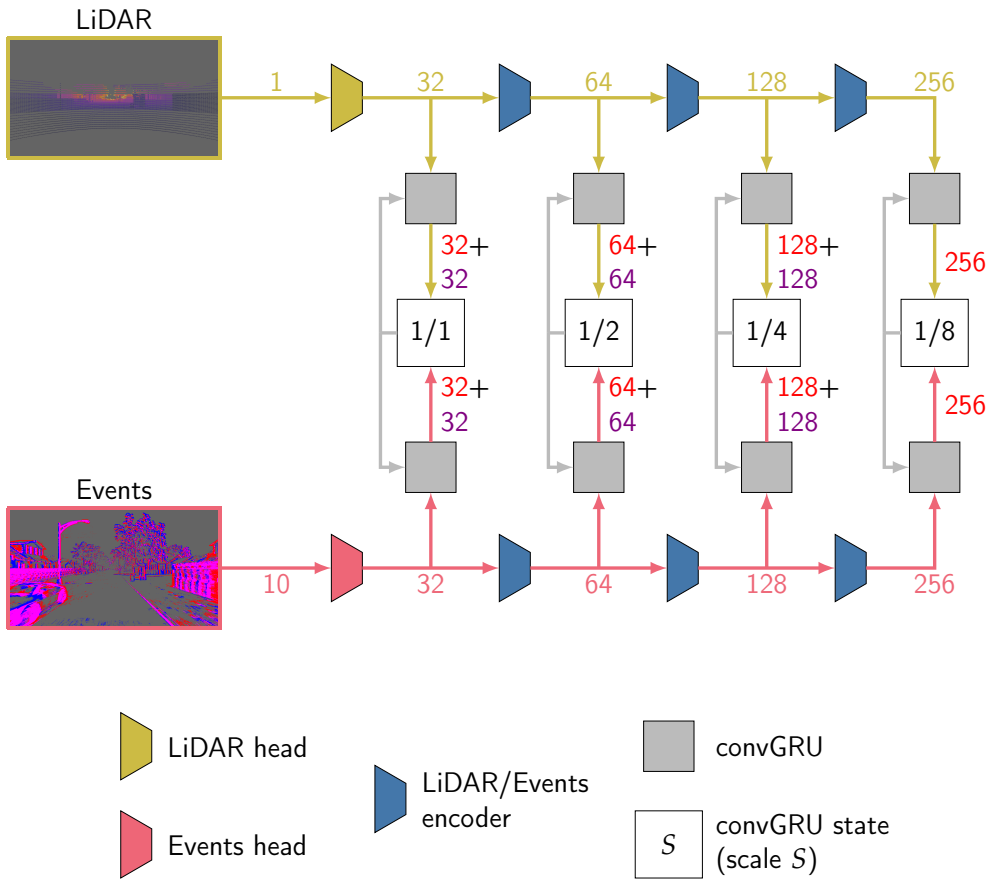


Figure 4.2 – The encoder part of the network. The numbers along the connections indicate the number of channels of the corresponding data.

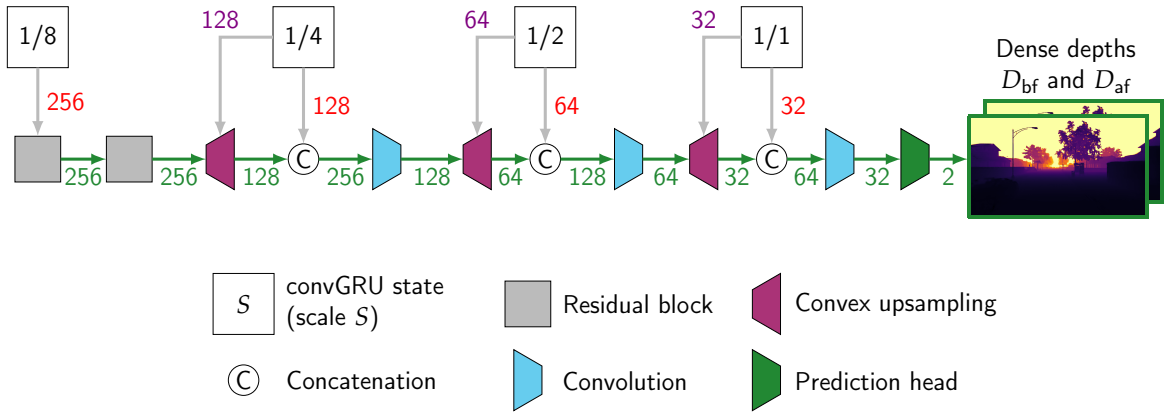


Figure 4.3 – The decoder part of the network. The numbers along the connections indicate the number of channels of the corresponding data.

Decoding In its decoder part, illustrated in Fig. 4.3, the convGRU state at the lowest scale first goes through two residual blocks. Then, for each following scale, the decoded feature map from the previous scale is upscaled by using convex upsampling [89]. While a simple bilinear upsampling was used in RAMNet [33], convex upsampling allows our network to learn how to upscale features from a lower scale, using information from a higher scale. We propose to design the convGRU such that the first half of its state (in purple in Figs. 4.2 and 4.3) guides the convex upsampling. Fusion of the upsampled decoded features and the state from the current scale is then performed by concatenating the output of the convex upsampling block with the remaining half of the convGRU state (in green in Figs. 4.2 and 4.3), and by applying a convolution to reduce the number of channels. After the last scale, a prediction head is used to obtain the two final depth maps, D_{bf} and D_{af}, in the form of a two-channel tensor, at the same full resolution as the events input.

4.4.2 *Implementation*

Regarding the implementation of ALED, both encoding heads use a kernel size of 5. LiDAR and events encoders use a kernel size of 5, with stride 2. Both the convGRU and residual blocks use a kernel size of 3. The convolutions in the convex upsampling blocks use a kernel size of 5, while the convolutions following the concatenations use a kernel size of 1. Finally, the prediction layer also uses a kernel size of 1. Convolutions are followed by a PReLU activation function [143], and instance normalization is used in the ResNet encoders as proposed by Pan *et al.* [144]. In total, the network contains 26 million trainable parameters.

4.4.3 *Data Representation*

Events We use the Event Volume [24] as the input representation for the events. We follow here the formulation of Perot *et al.* [25], where, for input events $\{e_i = (\mathbf{x}_i, p_i, t_i)\}_{i=1}^{N}$, the content of the Event Volume V for the pixel $\mathbf{x}$, the polarity p, and the bin b is defined as:

$$V(\mathbf{x}, p, b) \doteq \sum_{\{e_i | \mathbf{x}_i = \mathbf{x}, p_i = p\}} \max(0, 1 - |b - t_i^*|), \tag{4.1}$$

where t_i^* is defined as:

$$t_i^* \doteq (B-1)\frac{t_i - t_1}{t_N - t_1} \tag{4.2}$$

and where B is the number of temporal bins to split the timestamps of the events.

In our experiments, we set $B = 5$ bins, and concatenate the negative and positive polarity bins along the first dimension, resulting in a 3D tensor of shape $(10, H, W)$.

LiDAR and Depths LiDAR data is fed to the network as a 1-channel depth image. To do so, each LiDAR point cloud is projected onto the image plane of the event camera. Pixels where one or more LiDAR points fall into are given as value the

lowest depth. Pixels without any LiDAR point are given a value of 0. For an easier learning, the LiDAR projection and ground truth images are normalized between 0 and 1 based on the maximum LiDAR range (200m in the case of our synthetic SLED dataset, 100m in the case of the MVSEC dataset [62]).

4.4.4 *Loss Functions*

To train our network, we combine the use of two losses: a pixel-wise ℓ_1 loss $\mathcal{L}_{\text{pw}}$, and a multiscale gradient matching loss $\mathcal{L}_{\text{msg}}$.

The pixel-wise ℓ_1 loss operates as the main supervision loss, applied on both the "before" and the "after" depth maps, and is defined as follows:

$$\mathcal{L}_{\text{pw}} = \sum_{\mathbf{x}} \left\| D(\mathbf{x}) - \hat{D}(\mathbf{x}) \right\|_1 \tag{4.3}$$

where D and $\hat{D}$ are respectively the estimated and ground truth depth maps.

However, when supervised by the ℓ_1 loss alone, the network tends to produce blurry and non-smooth depth images. To solve this issue, we use here a multiscale gradient matching loss inspired by [145], also applied on both the "before" and the "after" depth maps, and defined as

$$\mathcal{L}_{\text{msg}} = \sum_{h \in \{1,2,4,8,16\}} \sum_{\mathbf{x}} \left\| \mathbf{g}[D](\mathbf{x}, h) - \mathbf{g}[\hat{D}](\mathbf{x}, h) \right\|_2 \tag{4.4}$$

with the discrete gradient function $\mathbf{g}$ of an image f defined as

$$\mathbf{g}[f](x, y, h) = \begin{pmatrix} f(x+h, y) - f(x, y) \\ f(x, y+h) - f(x, y) \end{pmatrix} \tag{4.5}$$

This loss helps to regulate the depth results, by making depth discontinuities more prominent, and by smoothing homogeneous regions.

Our total loss $\mathcal{L}$ for a sequence of length T is finally defined as

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{\text{bf,af}} (\mathcal{L}_{\text{pw}}^t + \alpha \mathcal{L}_{\text{msg}}^t) \tag{4.6}$$

where α is a weight parameter for the multiscale gradient matching loss.

In our experiments, we observed that giving too much importance to the multiscale gradient matching loss early in the training makes the network unable to derive correct depth estimates. Therefore, we always set $\alpha = 0.1$ during the first epoch of training, to force the network to use mainly the ℓ_1 loss and converge towards good initial depth estimates. For the remaining epochs, we set $\alpha = 1$.

4.5 THE SLED DATASET

In order to train and evaluate the proposed ALED network, we require a dataset containing both events, LiDAR point clouds, and a dense ground truth on depths. While we can use the MVSEC dataset [62] for low-resolution cameras, its ground truth is constructed by accumulating point clouds from a LiDAR sensor, a solution which introduces errors in case of moving objects, as noted in Section 2.7.1. In high-resolution, the DSEC dataset [63] would have been a perfect candidate, but its ground truth depth maps are given at the timestamps of the RGB frames and not those of the LiDAR point clouds[1], making these depth maps incompatible with the training and evaluation procedures of our network.

For these reasons, we use the CARLA simulator [68] (version 0.9.14) to generate a dataset with perfect synchronization and calibration of the sensors, and perfect ground truth depth. We call it SLED, for Synthetic LiDAR Events Depths dataset. It is composed of 160 sequences of 10 seconds each, for a total of more than 20 minutes of data. These sequences are recorded on the `Town01` to `Town07` and `Town10` maps of the simulator (20 sequences for each map), each sequence starting from a different geographic location. By doing so, a wide range of environments is represented within the dataset, as detailed in Table 4.1.

	Set	Environment	Features	Night seq.	Day seq.
`Town01`	Test	Town	Small buildings, bridges, distant forests and mountains, palm trees	4	16
`Town02`	Train	Town	Small buildings, plazas, forest road	4	16
`Town03`	Test	City	Tall and small buildings, roundabouts, tunnel, aerial railway	4	16
`Town04`	Val.	Town	Small buildings, highway, parking, lake, forests and mountains	4	16
`Town05`	Train	City	Tall buildings, parking, aerial beltway and railway	4	16
`Town06`	Train	Suburban	Small buildings, U-turns, distant hills	4	16
`Town07`	Train	Countryside	Barns, grain silos, fields, mountain road	4	16
`Town10`	Train	City	Buildings, monuments and sculptures, playgrounds, seaside	4	16

Table 4.1 – Detailed content of our SLED dataset containing 160 sequences of 10 seconds each.

Each sequence contains a 1280×720 event camera, a 40-channel LiDAR, and a 1280×720 depth camera which is perfectly aligned with the event camera. Both the event data and the depth images are recorded at 200Hz, while the LiDAR is configured to run at 10Hz. RGB images (1280×720) are also provided at a 30Hz rate, aligned with the event-based sensor. The LiDAR sensor is configured with a maximum range of 200 meters. For realism and diversity purposes, AI-controlled vehicles and pedestrians are added to the simulation. Sun altitude also varies, resulting for each map in 4 night recordings, and the other 16 recordings ranging from early morning (where the sun can be directly in front of the camera) to midday (where the sun is at its apogee). Varying cloudiness conditions are also used, adding more or less texture to the sky, and making shadows more diverse.

We also configured the event camera in CARLA to use a linear intensity scale rather than the default logarithmic one, making the events produced by the simulator more

1 For more details, see the following comment made by the authors of the DSEC dataset: `https://github.com/uzh-rpg/DSEC/issues/7#issuecomment-1416776152`

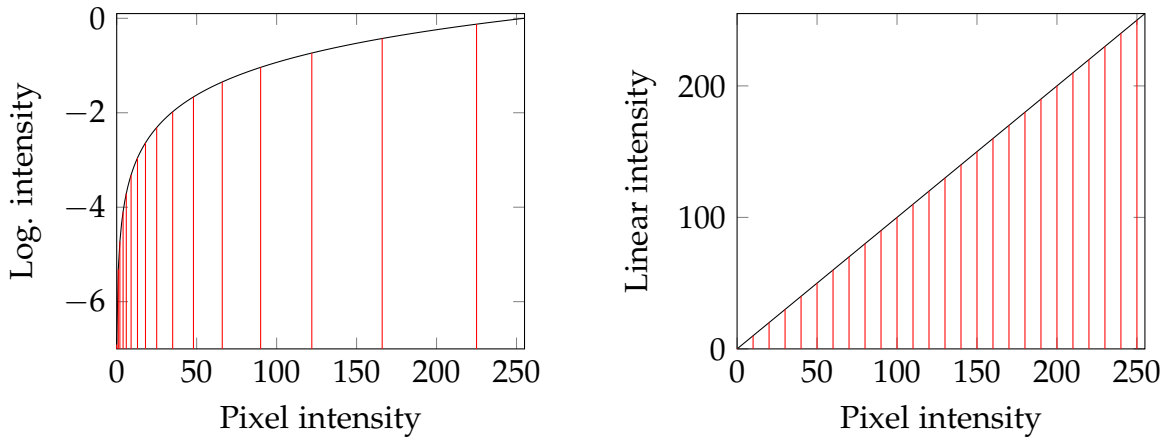
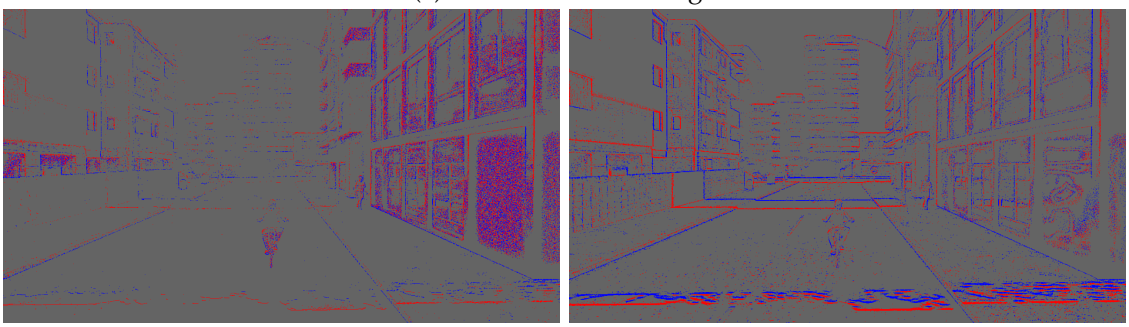


Figure 4.4 – Comparison of the triggering of events when the logarithmic and the linear scales are used. The logarithmic intensity in CARLA is computed as $\ln(I/255 + 0.001)$, where I is the pixel intensity. Each red vertical line denotes the triggering of an event, with thresholds set to 0.3 for the logarithmic scale, and 10 for the linear scale.

(a) Generated RGB image



(b) Events generated with log. scale

(c) Events generated with linear scale

Figure 4.5 – Visual comparison of the triggering of events when the logarithmic and the linear scales are used for a urban scene in CARLA. With the logarithmic scale, notice how the dark building on the right generates a high amount of events compared to the other buildings, and how details such as road markings or shadows are mostly lost. In comparison, with the linear scale, notice how the events are better distributed in the image, looking more like what an actual event camera would produce.

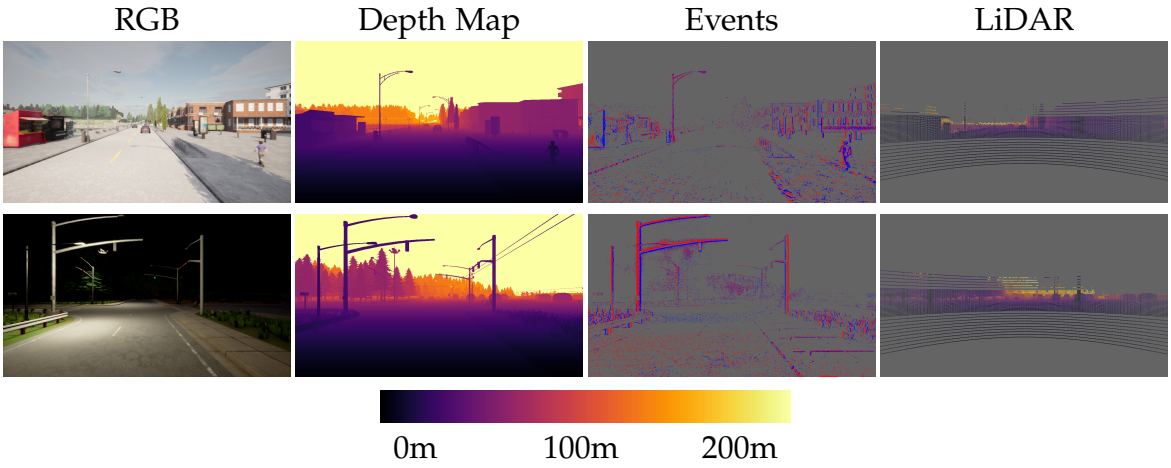


Figure 4.6 – Example data from the `Town01_04` (top) and `Town07_00` (bottom) sequences from our SLED dataset. A color scale is also provided for a better comprehension of the depth values.

realistic. We argue here that the logarithmic scale amplifies too much the creation of events in the dark areas of the image, as a very slight change in the intensity results in a large logarithmic intensity change, thus triggering an event. On the contrary, in the clearer areas, little to no events are produced, as a large intensity change is necessary to generate a logarithmic difference sufficient to trigger an event. An illustration of this phenomenon is given in Figs. 4.4 and 4.5.

LiDAR data also had to be corrected, due to incorrect values being reported by CARLA in specific cases[2]. For that purpose, the depth of all LiDAR points was extracted from the corresponding pixels in the ground truth depth maps, instead of using the values given by the sensor.

Finally, we give in Fig. 4.6 an overview of the data contained in the dataset. In particular, we display here illustrations from two very different recordings: one on `Town01` during daytime, and a second one on `Town07` during nighttime.

4.6 EVALUATION

We split our evaluation to cover the various use cases of our method: estimating dense depth maps, associating two depths to each event, and computing depth change maps.

4.6.1 *Training Details*

For training on all datasets, we use the Adam optimizer [146] with a batch size of 4. When training from scratch on the SLED and the MVSEC datasets, 50 epochs are

2 Some objects in CARLA lack the necessary collisions for them to be seen by the LiDAR sensor. For more details, see the following issue, which is still not fully solved despite having been closed: `https://github.com/carla-simulator/carla/issues/5732`

used, with a fixed learning rate of 10^{-4}. When finetuning on the MVSEC dataset, 5 epochs are used, with a learning rate of 10^{-5}.

To augment the input data for the SLED dataset, we randomly crop it to 608×608, and apply random horizontal flipping. For the MVSEC dataset, we crop it to 256×256 (due to the lower resolution), and also apply random horizontal flipping.

4.6.2 *Evaluation of the Dense Depths*

On the SLED Dataset

We begin by training ALED solely on the SLED dataset, and denote it ALED_{SL}. Numerical results of ALED_{SL} on the testing set of the SLED dataset are presented in the "Dense depths errors" column of Table 4.2. Evaluations are conducted on `Town01` and `Town03` maps, which contain challenging environments with many unique features (bridges, tunnels, ...) that are not present in the training maps. For the max range of 200 meters, ALED_{SL} estimates depth maps with an average absolute error slightly over 4.5 meters for `Town01`, and around 5 meters for `Town03`. The respective absolute relative error is around 19% for `Town01`, and around 22% for `Town03`.

	Cutoff	**Dense depths errors**				**Sparse depths errors**				**Depth change map errors**	
		On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Mean (m)	Correctly classified events (%)
		Mean (m)	AbsRel	Mean (m)	AbsRel	NN (m)	ALED_{SL} (m)	NN (m)	ALED_{SL} (m)		(with a threshold of ±1m)
`Town01`	10m	1.24	0.201	1.37	0.236	1.32	1.46	2.24	1.79	2.11	90.27
	20m	2.08	0.231	2.27	0.255	1.51	1.84	2.53	2.15	3.18	85.07
	30m	2.72	0.238	2.92	0.260	1.71	2.37	2.83	2.67	3.88	81.68
	100m	4.25	0.240	4.51	0.261	2.40	3.48	3.91	3.95	5.12	77.48
	200m	4.53	0.172	4.81	0.187	7.86	5.44	9.76	6.23	7.36	75.54
`Town03`	10m	2.00	0.289	2.09	0.301	0.47	0.56	0.67	0.66	1.14	93.70
	20m	2.85	0.299	2.97	0.311	0.64	0.75	1.12	0.87	2.54	87.16
	30m	3.33	0.291	3.45	0.302	0.92	1.11	1.61	1.26	3.23	83.71
	100m	4.60	0.274	4.77	0.284	1.88	2.55	3.17	2.88	4.47	78.50
	200m	4.86	0.215	5.03	0.223	4.43	3.60	5.93	4.10	6.20	77.23

Table 4.2 – Errors on the testing set of the SLED dataset for various cutoff depth distances. From left to right: average absolute and relative depth errors on both the "before" D_{bf} and "after" D_{af} depth maps; average absolute depth errors when associating a depth to each event; average absolute depth difference errors and percentage of correctly classified events based on this depth difference.

A first element that can explain these errors is that the LiDAR has a small vertical coverage of the image: close ground objects or the top of close buildings are not reached by the LiDAR, meaning that accurate depth estimations for these objects are complex to achieve. In opposition, sky pixels (for which ALED_{SL} produces good results) are only accounted for at the full 200m cutoff. These observations can be correlated to the larger relative errors observed for close cutoff distances than for the full 200m cutoff. It can also be observed that errors on the "after" depth maps are slightly higher. This is to be expected, as the network has to make use of the events to estimate the movement and propagate the depths accordingly. If the reader is interested, results for each sequence of the testing set are given in Appendix B.1.

Qualitative results are given in Fig. 4.7. They showcase the ability of the network to estimate accurate depths for the whole image, by using events as a guide for the

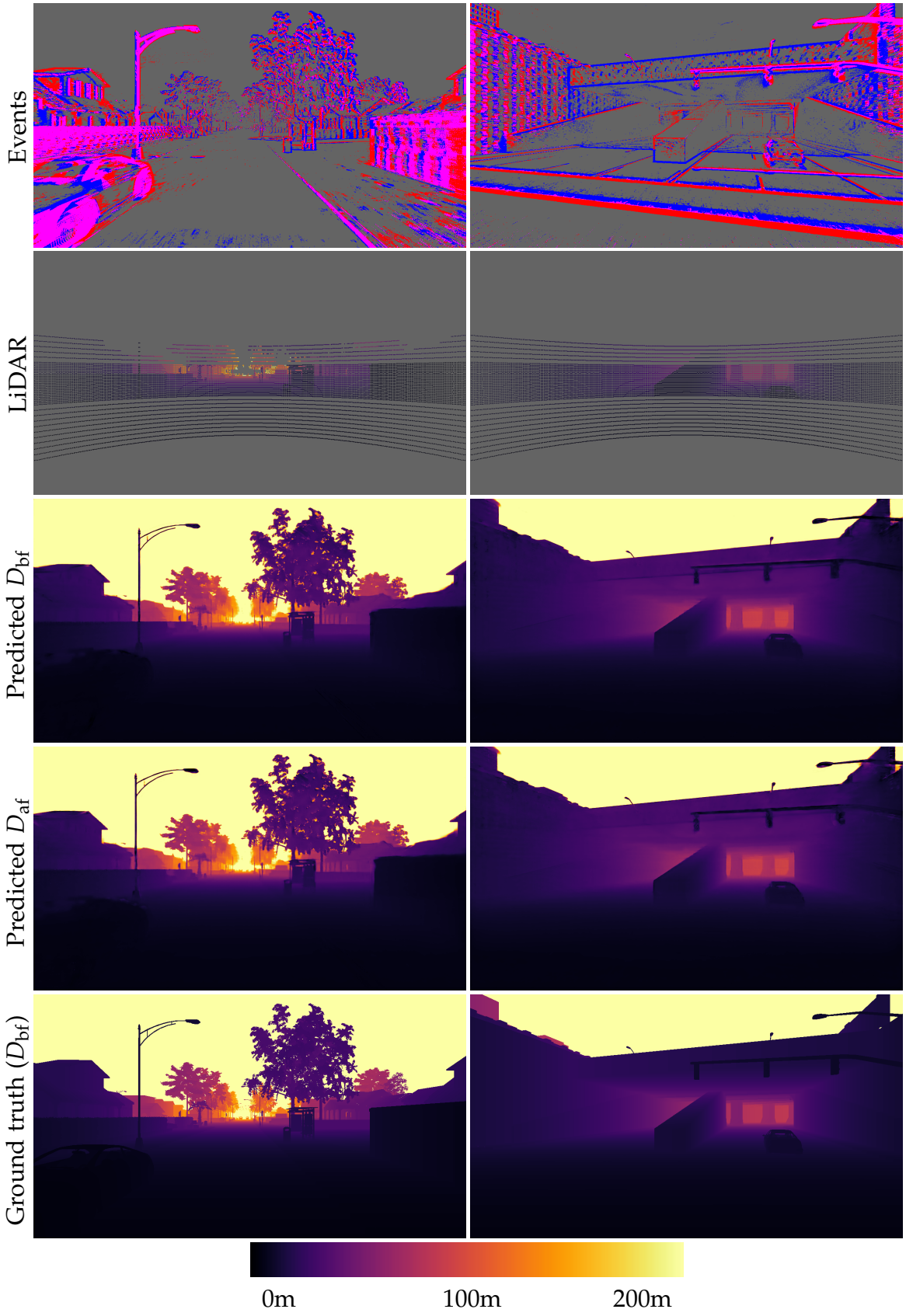


Figure 4.7 – Two qualitative results on our synthetic SLED dataset, on `Town01` (left) and `Town03` (right). From top to bottom: events, LiDAR, predicted depth maps (D_{bf} and D_{af}), ground truth (D_{bf} only, D_{af} being visually similar), color scale.

areas the LiDAR sensor can not reach. This is particularly visible for the trees and the light pole for the left column, or the ceiling of the tunnel for the right column. If the reader is interested, more visual results are given in Appendix B.2 and in the videos accessible through the link of the project page given at the very beginning of this chapter (Page 49).

On the MVSEC Dataset

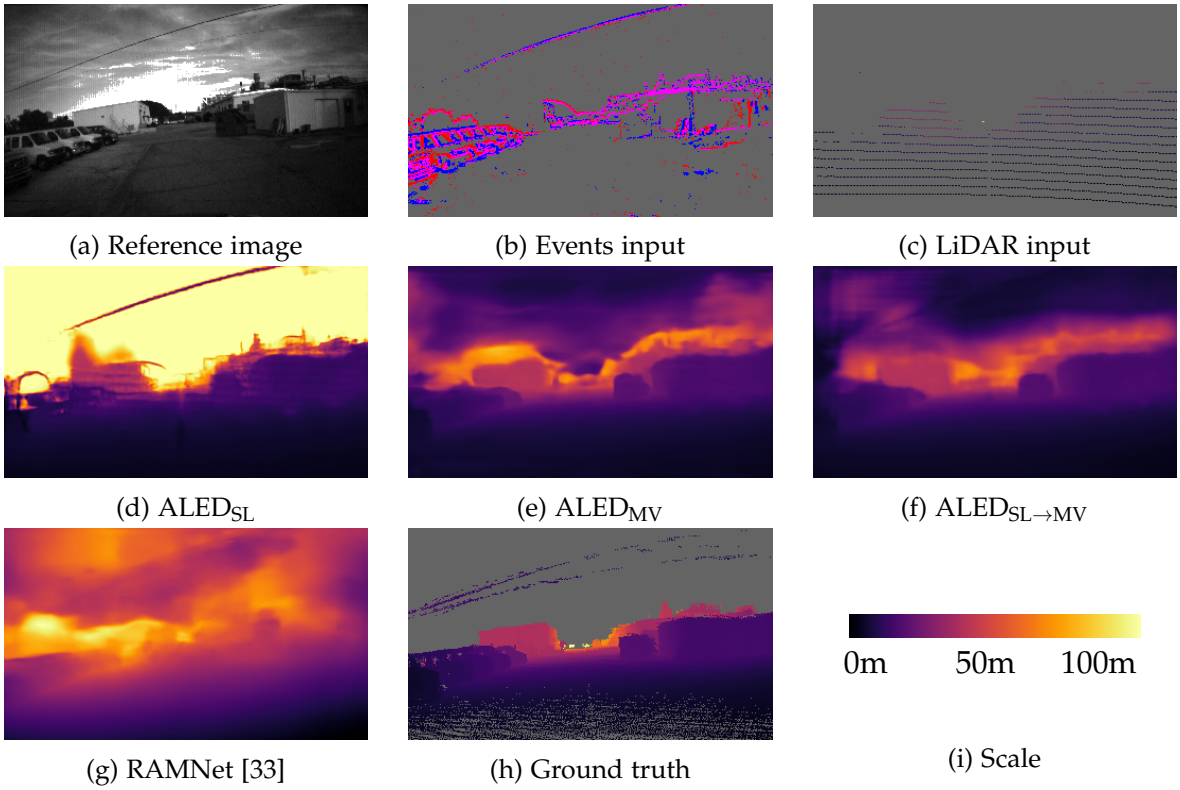
In order to be able to compare our results with the other approaches in the literature, we also train and evaluate ALED on the MVSEC dataset [62]. We conduct our evaluation under three different sets of weights from different training setups described below:

- ALED_{SL}: we reuse the weights trained on proposed SLED dataset;
- ALED_{MV}: we train a new set of weights on the MVSEC dataset;
- $\text{ALED}_{\text{SL}\to\text{MV}}$: we reuse the weights trained on SLED and finetune them on the MVSEC dataset.

	Cutoff	Events (stereo)	Events & Frames		Events & LiDAR			
		StereoSpike [35]	RAMNet [33]	EvT$^+$ [147]	Cui *et al.* [45]	**ALED$_{\text{SL}}$**	**ALED$_{\text{MV}}$**	**ALED$_{\text{SL}\to\text{MV}}$**
outdoor_day_1	10m	0.79	1.39	1.24	1.24	1.54	0.91	**0.50**
	20m	1.47	2.17	1.91	1.28	2.55	1.22	**0.80**
	30m	1.92	2.76	2.36	4.87	3.18	1.43	**1.02**
	50m	-	-	-	-	3.79	1.67	**1.31**
	100m	3.17	-	-	-	4.08	1.96	**1.60**
outdoor_night_1	10m	**1.38**	2.50	1.45	2.26	2.24	1.75	1.52
	20m	2.26	3.19	2.10	2.19	3.32	2.10	**1.81**
	30m	2.97	3.82	2.88	4.50	3.82	2.25	**1.95**
	50m	-	-	-	-	4.31	2.44	**2.20**
	100m	4.82	-	-	-	4.62	2.73	**2.54**
outdoor_night_2	10m	-	1.21	1.48	1.88	1.94	1.19	**1.09**
	20m	-	2.31	2.13	2.14	2.82	1.65	**1.49**
	30m	-	3.28	2.90	4.67	3.22	1.81	**1.64**
	50m	-	-	-	-	3.58	1.95	**1.80**
	100m	-	-	-	-	3.78	2.11	**1.97**
outdoor_night_3	10m	-	1.01	1.38	1.78	1.76	0.85	**0.81**
	20m	-	2.34	2.03	1.93	2.43	1.25	**1.16**
	30m	-	3.43	2.77	4.55	2.78	1.42	**1.33**
	50m	-	-	-	-	3.12	1.57	**1.51**
	100m	-	-	-	-	3.31	1.73	**1.66**

Table 4.3 – Average absolute depth errors (in meters) on the MVSEC dataset for various cutoff depth distances. This evaluation is performed on the "before" depth map D_{bf}, to be consistent with the methods we compare ourselves to.

Numerical results of all three variants are given in Table 4.3. Comparing them, it appears clearly that training on synthetic data before finetuning the network on the MVSEC dataset ($\text{ALED}_{\text{SL}\to\text{MV}}$) produces the best results. Training from zero on the MVSEC dataset (ALED_{MV}) is not as good as the $\text{ALED}_{\text{SL}\to\text{MV}}$ variant due to the limited data available for training. Finally, training solely on the SLED dataset (ALED_{SL}) produces the worst results, due to the large differences in terms



(a) Reference image (b) Events input (c) LiDAR input

(d) ALED_{SL} (e) ALED_{MV} (f) $\text{ALED}_{\text{SL}\to\text{MV}}$

(g) RAMNet [33] (h) Ground truth (i) Scale

Figure 4.8 – Qualitative results on the `outdoor_day_1` sequence from the MVSEC dataset.

of resolution and LiDAR models between the two datasets, and as simulation is not a perfect reproduction of real data.

Our $\text{ALED}_{\text{SL}\to\text{MV}}$ network greatly outperforms all the other approaches of the state of the art. Most impressive results are obtained with distant cutoff depths, where fewer LiDAR points are available; our network is still able to infer accurate depths, while reference methods show large errors:

- compared to the stereo events StereoSpike method of Rançon *et al.* [35], for the 20m and 30m cutoff distances respectively, we improve the error by 19.9% and 34.3% in the worst case, and by 45.6% and 46.9% in the best case;
- compared to the frames & events EvT^{+} method of Sabater *et al.* [147], this improvement is of 13.8% and 32.3% in the worst case and of 58.1% and 56.8% in the best case at maximum;
- compared to the LiDAR & events method of Cui *et al.* [45], this improvement is of 17.4% and 56.7% in the worst case and of 39.9% and 79.1% in the best case.

Qualitative results are presented in Fig. 4.8. All three ALED variants produce results which are visually close to the ground truth for ground objects. Since the MVSEC dataset lacks ground truth depth for the sky, and since the rare elements to have a ground truth for this part of the image are close buildings, close trees, or power lines, the network cannot learn to derive correct depth estimations for the corresponding

pixels, leading to the purple blobs in the upper parts of Figs. 4.8e and 4.8f. Only the ALED_{SL} variant is able to predict accurate values for sky areas (as our SLED dataset contains valid ground truth depths for all pixels), but has more difficulties for ground objects due to the lack of finetuning. Between the ALED_{MV} and $\text{ALED}_{\text{SL}\to\text{MV}}$ variants, improvement can still be seen, for instance for the edges of the objects of Figs. 4.8e and 4.8f, which are less uneven for the $\text{ALED}_{\text{SL}\to\text{MV}}$ variant. Finally, when comparing our results to RAMNet, we can clearly observe that our method provides in all cases more accurate depth maps, where object boundaries are more prominent, and where estimated depths are closer to the ground truth. This observation further demonstrates that the use of a LiDAR input — even if very sparse — is of great help for obtaining accurate dense depth maps.

If the reader is interested, qualitative results for each of the sequences of the MVSEC dataset are given in Appendix B.3. Full results are also available in video format through the link of the project page given at the very beginning of this chapter (Page 49).

4.6.3 *Evaluation of the Sparse Depths*

As stated in Sections 4.1 and 4.3, our goal is not only to estimate dense depth maps, but also to associate two depths to each event, allowing for their 3D reprojection and depth difference analysis. Evaluation of the depth association to each event on proposed SLED dataset is given in the "Sparse depths errors" column of Table 4.2.

The sparse event-LiDAR fusion literature is limited to the method of Li *et al.* [14]. However, their approach only considers one depth per event, and is intended for a Road Side Unit (RSU) application (i.e., their event camera is fixed and evaluation is conducted on a specific dataset). Therefore, we decided to compare ourselves to a more naive (and faster) baseline: the Nearest Neighbor (NN) approach, where each event is given the depth of its closest LiDAR point. As the Nearest Neighbor approach can not infer correct depths for events which are too far from a LiDAR scan, and so as to provide a fair comparison, we only consider the events between the bottom and top LiDAR scans.

As displayed in Table 4.2, in the "before" D_{bf} case, depending on the map and the cutoff distance, best results are shared between the NN approach and our ALED_{SL} network. These results can be explained by the fact that our network is more likely to commit large errors for events at the boundary of close objects, as it might estimate that they should be given the depth of the more distant background. On the contrary, the NN approach will always attribute the depth of the closest LiDAR point, and will therefore commit more frequent but smaller errors. In the "after" D_{af} case, despite this potential source of error, our network ALED_{SL} nearly always obtains the best results, as it has correctly learned the temporal propagation of the depths, a task which cannot be completed natively with the NN approach. We also remind here that these results are given for the parts of the image where LiDAR data is available: the NN method would not be able to derive correct estimations for the other parts of the image. If the reader is interested, numerical results for each sequence of the testing set are given in Appendix B.1.

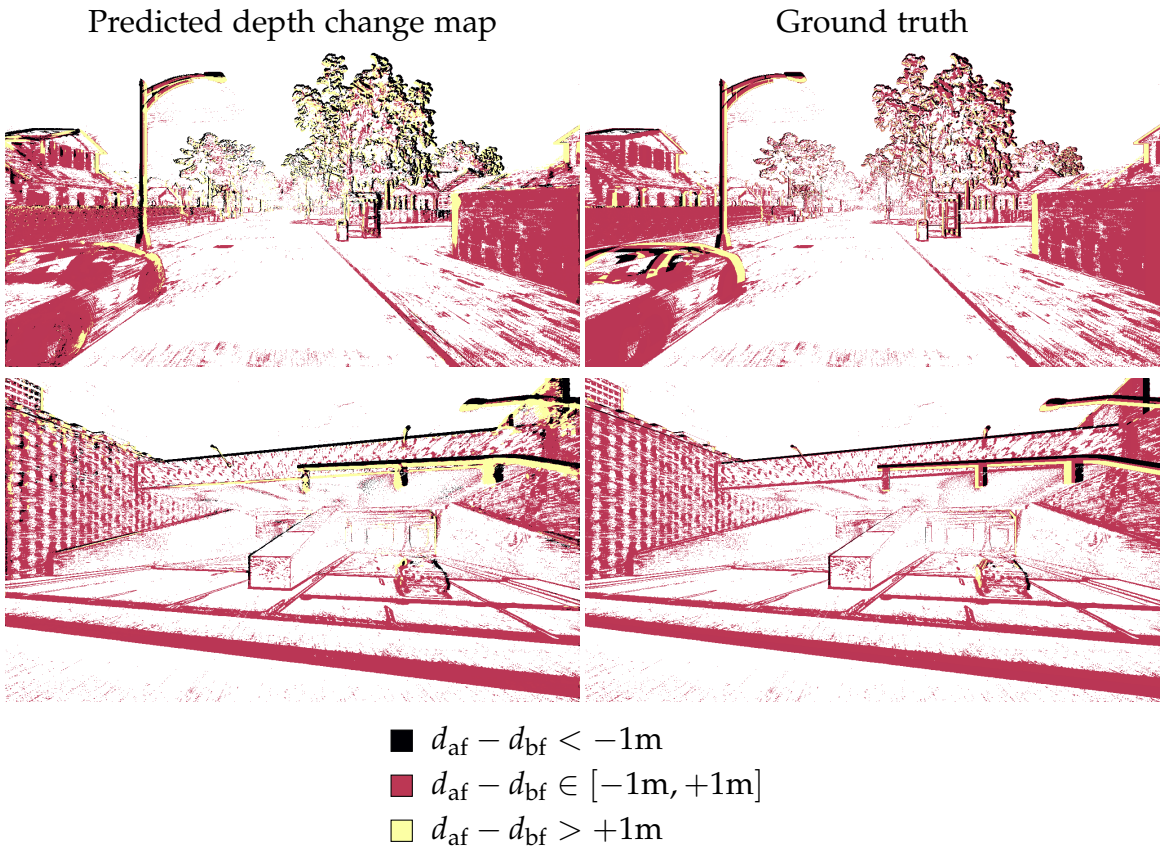


Figure 4.9 – Thresholded depth change maps from the `Town_01` and `Town_03` sequences of the SLED dataset, using the events as a mask.

4.6.4 *Evaluation of the Depth Change Maps*

We finally estimate the quality of our "two depths per event" approach through the depth change map, as presented in Section 4.3. We perform two evaluations on our SLED dataset: **(1)** the average error on the depth change map $D_{af} - D_{bf}$ compared to the true depth changes, and **(2)** the percentage of correctly classified events when using a difference threshold of 1m on the depth change map. Results of these evaluations are given in the "Depth change map errors" column of Table 4.2. If the reader is interested, numerical results for each sequence of the testing set are given in Appendix B.1.

We can observe here that, despite a significant absolute error on the depth change maps, events can still be classified correctly, with a rate of success over 75% on `Town01`, and over 77% on `Town03`. We remind here that the ALED network is not trained on these depth change map and classification tasks. As such, we believe that, while even more accurate individual depth maps could improve both the depth change map and classification errors, further improvements could be brought by designing a network specifically dedicated to these tasks.

We show in Fig. 4.9 qualitative results for the thresholded depth change maps, for the same sequences as in Fig. 4.7. These results visually corroborate the overall

accurate classification of the events observed during the numerical analysis. Some errors can still be seen, especially for the lower parts of the objects: as they are closer to the ground, the depth difference is less significant, and errors on the depth change map become therefore more critical. More qualitative results are given in Appendix B.4.

4.7 CONCLUSION AND DISCUSSIONS

Throughout this chapter, a novel learning-based approach for estimating dense depth maps from asynchronous LiDAR and event-based data has been proposed. A novel "two depths per event" notion has also been proposed, to solve the issue of events possibly representing a change of depth. A synthetic multimodal dataset has also been recorded, to train and evaluate our method. Multiple evaluations on our synthetic and a real driving datasets have been performed to show the relevance of our contributions. In particular, on the MVSEC dataset, an improvement of up to 79.1% compared to the current LiDAR-and-events state of the art has been achieved, on complex daytime and nighttime recordings.

In hindsight, further improvements could be brought to the method. **(1)** Making the network predict directly sparse depths for each event could potentially provide better results for the depth to event inference. This could be achieved by using sparse convolutional networks [148] for instance, and could be subject to future work. **(2)** The use of the Event Volume with a fixed time window as the input representation for the events in our network could also be revised, as it can become ill-suited under large motions. A solution could be to use an alternative representation, such as TORE Volumes [149], or to use adaptive accumulation times using methods such as the one proposed by Liu and Delbrück [116]. **(3)** The SLED dataset was recorded so that the ego-vehicle would not get stuck in traffic or in front of traffic lights, to avoid having fully static sequences (as a reminder, sequences last 10 seconds in SLED), because the dense depth estimation would be impossible for these sequences due to a lack of events. However, this means that the ego-vehicle almost never stops across the whole dataset, which is also an issue. An update to SLED could therefore be published in the future, for ensuring that the vehicle makes short stops in some sequences, for more diversity. **(4)** Finally, the recording of a real dataset with a high-definition event camera could also be considered, to complete the possibilities offered by the low-resolution MVSEC dataset.

One of the highlights of this work is how visually impressive the results are, especially given the sparsity of the input data to the network (as shown for instance in Fig. 4.7). In the case of our SLED dataset, the segmentation between the sky and other objects is also an important highlight, even in the case of Fig. B.2 where a suspended railway is above the vehicle. Our network also displays good results on the night sequences of the MVSEC dataset, meaning that it is able to perform well even in the presence of noisy events. Yet, looking back at Tables 4.2 and 4.3, our method seems to have more difficulties for short ranges, where the errors are relatively high. If we consider an application to intelligent robotics, this becomes an important issue: the precision of the depth of close objects should be favored

compared to the more distant ones, as we primarily want to avoid collisions with objects surrounding the robot.

Therefore, to extend this work on depth estimation, we will look in the following chapter into a more refined method, by making use of the Transformer architecture [150] and especially its concept of self- and cross-attention. Our idea is that attention-based networks provide state-of-the-art results in numerous vision-based applications, and could further improve the fusion of the event and LiDAR modalities.

5 EVENT- AND LIDAR-BASED DEPTH ESTIMATION USING AN ATTENTION-BASED NETWORK

As noted in the conclusion of Chapter 4, our ALED network has two main issues: **(1)** it shows large errors for close distances, something we want to avoid for robotic applications, and **(2)** it can only predict dense depth maps, from which we can associate depths to each event, but at the cost of some accuracy. In this chapter, we aim at exploring these two issues and seeing how they could be solved using an attention-based network, which should allow for a better understanding of interactions between event and LiDAR data, and could theoretically allow for a sparse depth estimation without needing any dense frame-based representation.

The presented method and the associated results of this chapter were submitted as part of the ECCV 2024 conference. Since ECCV is a double-blind conference, a public project page is not yet available at the time of writing of this thesis, but a private playlist of videos showcasing some results is available at `https://www.youtube.com/playlist?list=PLLL0eWAd6OXBKmvfUNCR21iq2C4Ck8-B4`.

5.1 INTRODUCTION

Attention-based networks like the Transformer [150] are particularly powerful when it comes to representing interactions between elements of a sequence, like words in a sentence. While originally intended for Natural Language Processing (NLP), they have since become the standard architecture in numerous domains, including computer vision, and have shown impressive capabilities for multimodal data fusion [151, 152].

Therefore, in this chapter, we still aim at fusing sparse LiDAR and event data available at different rates, but doing it with an attention-based network. As will be described in the following sections, while our initial goal was to accomplish this fusion directly on sparse inputs and outputs, several theoretical and technical limitations were met. Therefore, we returned to denser representations, and propose here a novel attention-based network which we call DELTA, able to combine information from low-rate projected LiDAR point clouds with higher-rate small temporal windows of events, in order to derive accurate dense depth maps.

Thanks to its attention- and recurrence-based design, we show that this network is able to extract the most relevant spatial and temporal features within and in-between the event and LiDAR data. The introduction of a propagation memory for fusion at the highest input rate and of a central memory acting as a main recurrence both allow us to outperform the state of the art, and are critical contributions as shown through an ablation study.

As for previous chapters, an extensive evaluation of DELTA is conducted on multiple automotive datasets, where LiDAR and event sensors are most commonly used

together. We show here that DELTA is able to offer a clear improvement from ALED and the rest of the state of the art, in particular for close objects (which was the main limitation of ALED), where the average error is reduced up to four times.

This chapter is structured as follows. We first give an introduction to how the Transformer model and especially its attention mechanism work in Section 5.2. We then give an overview of the state of the art in Section 5.3. We describe our attempts at estimating directly sparse depths and the issues we faced in Section 5.4, and our final dense solution in Section 5.5. We finally conduct our evaluation in Section 5.6, before drawing some conclusions in Section 5.7.

5.2 AN INTRODUCTION TO THE TRANSFORMER AND ATTENTION

Before going into the details of the work conducted in this chapter, we must take some time to explain how the Transformer architecture works, and especially its attention module.

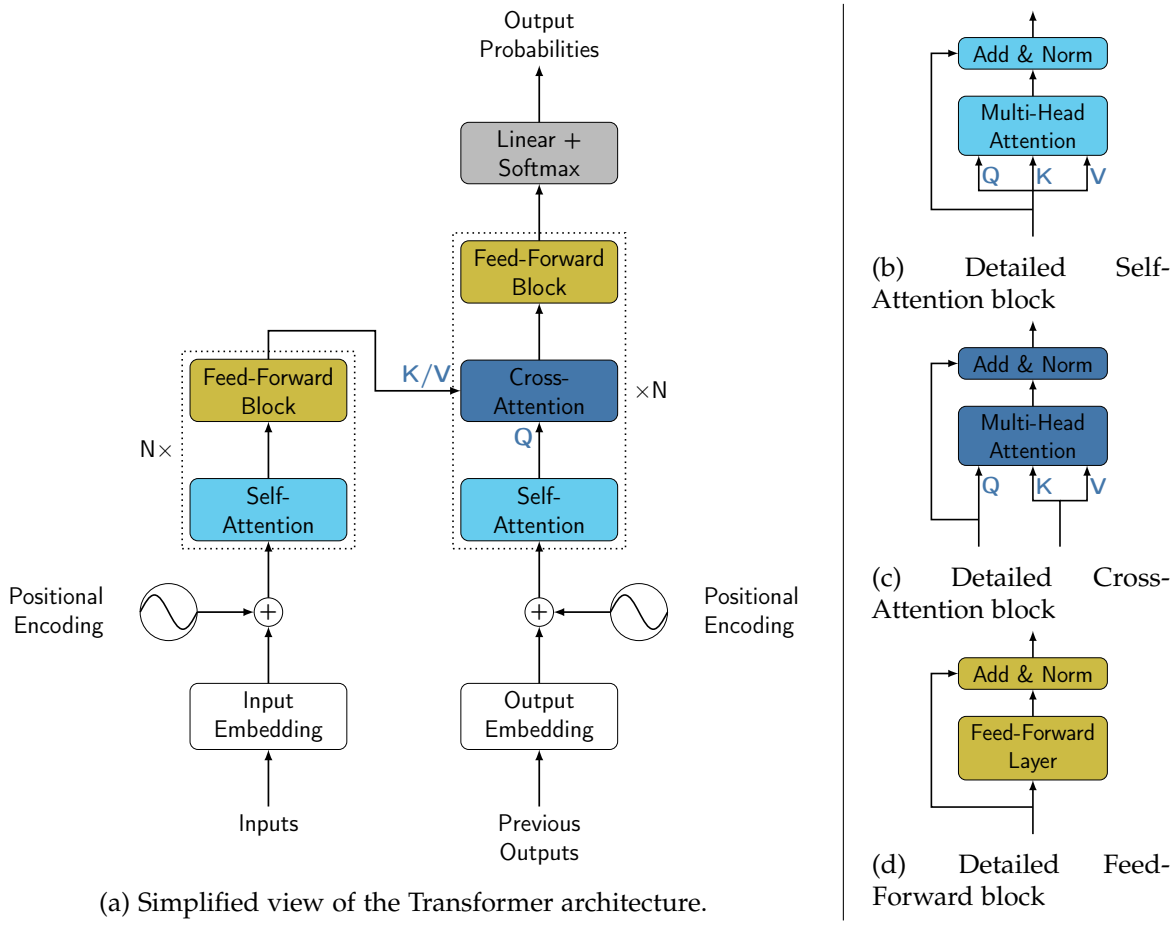
5.2.1 *Overview of the Transformer Architecture*

The Transformer [150] is a novel neural network model proposed in 2017, originally intended for Natural Language Processing (NLP). The objective of this model was to answer the limitations met by former recurrence-based networks (such as LSTMs): a tendency to forget context for long input sequences, and slow training and inference times due to their one-by-one processing of words. To solve these issues, the Transformer is able to treat the full input sequence in parallel at once, i.e., without splitting it into word-by-word inputs, and by explicitly modeling the relations between each word.

To do so, as illustrated in Fig. 5.1a, each input word is converted into a token, i.e., a fixed-size vector representation of this word. As the Transformer is an order independent architecture, a positional encoding is added to each token, representing its position in the input sequence. This list of tokens is then given to an attention-based encoder-decoder which we will describe in the next paragraphs. The decoder also takes as input the previously predicted output tokens. A final linear+softmax layer is used to give probabilities over the full dictionary of words, with the word with the best probability being added as the next word to the previously predicted words. This process is repeated until the network predicts a special End of Sentence (EOS) token.

Regarding the encoder itself, it is composed of N successive blocks, each composed of a self-attention block (illustrated in Fig. 5.1b, tasked with representing the relations within the tokens, as will be explained in Section 5.2.2) and a feed-forward block (illustrated in Fig. 5.1d, tasked with refining the output of the self-attention block).

As for the decoder, it is also composed of N successive blocks, each composed of a self-attention block, a cross-attention block (illustrated in Fig. 5.1c, tasked with



(a) Simplified view of the Transformer architecture.

(b) Detailed Self-Attention block

(c) Detailed Cross-Attention block

(d) Detailed Feed-Forward block

Figure 5.1 – Overview of the Transformer architecture. Illustration inspired by [150].

representing the relations between the previously predicted tokens and the encoded input tokens, as will also be explained in Section 5.2.2), and a feed-forward block.

In the following parts of this chapter, for simplicity of notation and of representation, we will note "*self-attention module*" the block composed of a self-attention block (■) followed by a feed-forward block (■), and "*cross-attention module*" the block composed of a cross-attention block (■) followed by a feed-forward block (■).

5.2.2 *Full Attention*

As noted before, the core element of the Transformer is its concept of attention, for representing explicitly relations within elements of a single sequence, or in-between elements of two sequences. This notion is particularly important typically (but not only) for NLP: in a simple sentence like "The cat could not eat its meal because *it* was ill.", understanding that the word "*it*" refers to the cat rather than to the meal is non-trivial for a machine; being able to represent this relation explicitly is therefore important for a good learning.

While the concept of attention is not new (we can cite here the articles of Nadaraya and of Watson [153, 154] both published in 1964, or the more recent article of

Bahdanau *et al.* [155]), the Transformer was the first to propose a fully parallelizable formulation of attention, particularly suited for learning on large sequences. In their formulation, they consider three input matrices: a matrix of queries Q of shape (N_Q, D), a matrix of keys K of shape (N_{KV}, D), and a matrix of values V of shape (N_{KV}, D), where N_Q and N_{KV} are respectively the number of vectors of queries and keys/values, and D is the dimensionality (i.e., the length) of the vectors.

They first compute a matrix of attention scores S by putting in relation the queries and the keys, i.e., by determining how the queries "match" the keys:

$$S = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) \tag{5.1}$$

where S is of shape (N_Q, N_{KV}). By using the softmax, each of the N_Q rows contains a score for each of the keys, summing to 1. This matrix of attention scores is finally applied on the matrix of values, to obtain the final matrix of values after receiving attention A:

$$A = SV \tag{5.2}$$

where A is of the same shape as the queries, i.e., (N_Q, D).

In practice, in case of self-attention, the three matrices Q, K, and V are obtained by multiplying the single input matrix I by matrices of learned weights W_Q, W_K, and W_V:

$$\begin{aligned} Q &= IW_Q \\ K &= IW_K \\ V &= IW_V \end{aligned} \tag{5.3}$$

In case of cross-attention, two input matrices are used, I_1 and I_2, and the Q, K, and V matrices are computed as:

$$\begin{aligned} Q &= I_1W_Q \\ K &= I_2W_K \\ V &= I_2W_V \end{aligned} \tag{5.4}$$

Therefore, self-attention is just a special case of cross-attention, where $I_1 = I_2$.

5.2.3 *Linearized Attention*

As shown in Eq. (5.1), the matrix of attention scores S is of shape (N_Q, N_{KV}) (due to the multiplication of Q and K^T). Therefore, and since $N_Q \gg D$ and $N_{KV} \gg D$, the space complexity of the attention process is of

$$\mathcal{O}(N_Q \times N_{KV} + N_Q \times D) \simeq \mathcal{O}(N^2) \tag{5.5}$$

i.e., it grows quadratically. In case of long input sequences, this memory requirement can quickly become a limitation. Therefore, several authors have proposed linearized

versions of the attention process. We describe here the two formulations that we tested as part of our work, that both rely on the same core idea: reducing the space complexity by carrying out the multiplication of K^T and V first.

As shown by Katharopoulos *et al.* [156], Eqs. (5.1) and (5.2) can be generalized and rewritten as

$$\left(\phi(Q)\phi(K)^T\right) V = \phi(Q)\left(\phi(K)^T V\right) \tag{5.6}$$

where $\phi(\cdot)$ is a feature map. Using the second formulation, the multiplication between K^T and V is applied first, resulting in a space complexity for the attention process of

$$\mathcal{O}(D \times D + N_Q \times D) \simeq \mathcal{O}(N) \tag{5.7}$$

i.e., a memory usage which grows linearly.

In their article, Katharopoulos *et al.* [156] use the exponential linear unit function [157] $\mathrm{elu}(\cdot)$ as their feature map:

$$\phi(x) = \mathrm{elu}(x) + 1 \tag{5.8}$$

Comparatively, Kamal *et al.* [158] use the softmax activation function as their feature map:

$$\phi(x) = \mathrm{softmax}(x) \tag{5.9}$$

However, as investigated by [159, 160], linear attention limits the ability of networks to train efficiently, reducing their maximum theoretical accuracy.

5.3 RELATED WORK

5.3.1 *Transformers for Event-Based Data*

The Transformer [150] has become the state-of-the-art architecture in numerous domains. Its attention mechanism models explicitly the relations between relevant elements in a sequence, making it able to understand structures. For computer vision, the arrival of the Vision Transformer [161] has been a notable landmark, by outperforming more traditional convolution-based networks. As such, researchers have started investigating how the Transformer architecture could be adapted to event-based cameras. Two philosophies have emerged over the years. **(1)** Some authors use directly the raw stream of events (without any pre-processing) as the input sequence to their network, and use the Transformer architecture to process it. This approach is particularly complex, as each event contains little information, making the modeling of their relations difficult for the Transformer. To contain enough context, sequences of events should also be of consequent size, whereas the Transformer was designed for smaller sequences. As such, this method has only been applied to the task of classification [158, 162], where the event data can be highly compressed by the network, as the final representation is only a small vector. **(2)** To circumvent these issues, most authors instead accumulate events in a frame-like representation, and process it using a mixture of convolutional

layers and of Transformer blocks. Investigated tasks include object detection [54], classification [147, 163, 164], depth estimation [147], optical flow [165], and video reconstruction [166].

5.3.2 *Fusion of Events and LiDAR*

As seen throughout Chapter 4, to this day, most works using both the LiDAR and event-based modalities address the problem of extrinsic calibration [137, 138, 139], or use them as part of the construction of a dataset [62, 63, 64]. Recently, authors have started investigating the issues of enhancing point clouds with event-based data [14], of estimating dense depth maps from event and LiDAR data [45], and of human tracking in adversarial lighting conditions [140].

5.3.3 *Depth Estimation using Events*

The idea of estimating sparse or dense depth maps from events has been actively explored over the past decade. Three main approaches can be distinguished. **(1)** Some authors estimate depths in a monocular fashion, using only events from a single event camera [24, 35, 46, 47, 167, 168], or using events and frames [33, 147] from a DAVIS camera [15]. These approaches are particularly challenging, as they lack any three dimensional information, and tend to result in overall lower performances. **(2)** Some authors have tried to estimate depth in a stereo fashion, by using a pair of event cameras [35, 41, 42, 43, 169, 170], with [169] and [35] achieving notably good results. **(3)** Finally, some authors prefer to use directly a depth sensor, and use the stream of event as a mean to densify and/or to temporally upsample the depth data. This depth sensor can either be an RGB-D camera [44] or a LiDAR [14, 45, 120], with our work described in the previous chapter [120] being the current state of the art on several datasets. While methods from all other authors estimate a single depth per event, we will reuse in this chapter the concept of estimating two depths per event, as explained in the previous chapter (Section 4.3).

5.4 PREDICTING SPARSE DEPTHS WITH TRANSFORMERS

Our initial objective was to be able to use a Transformer-like architecture to directly associate the sparse LiDAR points and the sparse events, and output sparse depths for each event. This idea relied on the basis that, as seen in Section 4.6.3, a simple Nearest Neighbor (NN) approach for associating LiDAR points and events works already quite well, and that the ability of the Transformer to model explicitly relations between elements would lead to even better performances (especially for the higher and lower parts of the image where no LiDAR data is available). Therefore, we tried to apply this idea with several network architectures, the two main ones being represented in Figs. 5.2 and 5.3, and described in Sections 5.4.1 and 5.4.2.

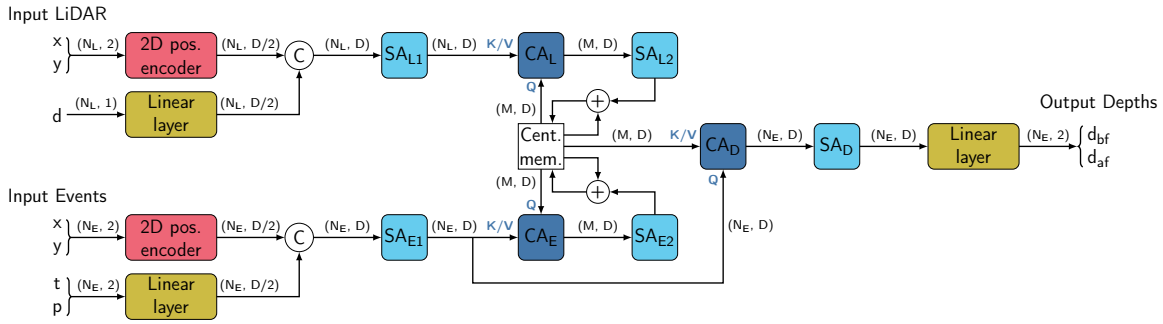


Figure 5.2 – The first version of our sparse attention-based network design. "CA" are cross-attention modules, "SA" are self-attention modules. The size of the data at each step is indicated above the arrows, where "N_L" is the number of LiDAR points, "N_E" is the number of events, "M" is the memory size, and "D" is the dimensionality.

5.4.1 *The First Version*

In its first version, shown in Fig. 5.2, we follow the design of our previous ALED network, with two independent encoding branches for the LiDAR and event data, a central memory for fusion, and a single decoding branch. As shown, no dense representation is used: the LiDAR and event data are given as a sequence (i.e., a list) of points to the network, and its output is a sequence of depths.

Encoding On the LiDAR side, the x and y positions of the N_L projected points are encoded with the use of a fixed 2-dimensional positional encoder, following the formulation of Carion *et al.* [171], while the depth of the points are encoded by a linear layer, before being concatenated back into a single sequence of shape (N_L, D). The same operation is applied on the events side: the x and y positions of the N_E events are encoded with the use of the same 2D positional encoder, and the timestamps and polarities are encoded by a linear layer, before being concatenated back into a single sequence of shape (N_E, D). A self-attention module (SA_{L1}/SA_{E1} in Fig. 5.2) is then used on both branches, to encode the internal relations between the LiDAR points and events respectively.

Memory Update Using the encoded input, a memory update is then generated. This process is the same for both the LiDAR and events side. A cross-attention module (CA_L/CA_E) is first used to generate an updated representation, where the current state of the memory (of shape (M, D)) is used as the queries (i.e., the elements composing the memory "ask" how they should be updated) and where the encoded LiDAR or event input is used as the keys/values (i.e., they provide the values for the update). At the output of the cross-attention module, an update of shape (M, D) is generated, which is refined by a self-attention module (SA_{L2}/SA_{E2}). The current state of the memory is finally updated by a summation and normalization with the refined update.

Decoding Since we want to estimate depths for each event individually, our decoding branch must be connected in some way to the input events. To do so, we use

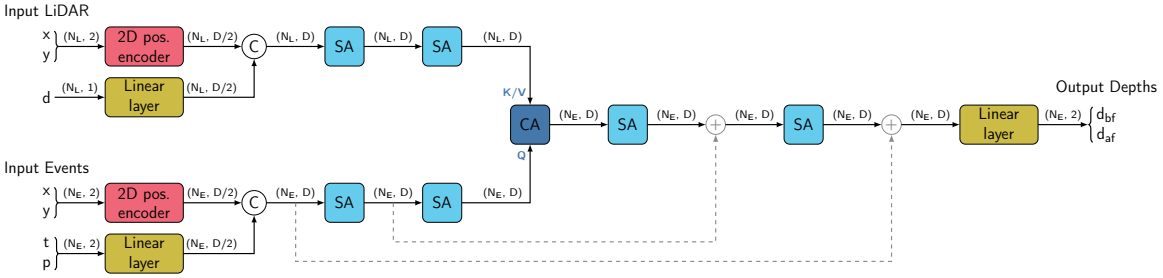


Figure 5.3 – The second version of our sparse attention-based network design, without a memory. "CA" are cross-attention modules, "SA" are self-attention modules. The size of the data at each step is indicated above the arrows, where "N_L" is the number of LiDAR points, "N_E" is the number of events, "M" is the memory size, and "D" is the dimensionality. Skip connections are in gray.

a cross-attention module (CA_D), where this time the encoded events are used the queries (i.e., they "ask" what should their depths be), and the memory is used as the keys/values (i.e., it provides the values for the current state of the fused LiDAR and event data), generating an output of shape (N_E, D). A final self-attention module (SA_D) is used to refine the decoded values, before a final linear layer is used to reduce the dimensionality of the data, to reach an output of shape $(N_E, 2)$ (i.e., a list where each element is associated to its corresponding input event, and contains just the two predicted depths, d_{bf} and d_{af}).

5.4.2 *The Second Version*

In the first version of the network, the LiDAR points and events never interact directly, they only do so indirectly through the central memory. Therefore, in its second version illustrated in Fig. 5.3, we removed this memory, and fused directly the LiDAR and event data through a single central cross-attention module.

Encoding The data encoding process remains mostly the same as the one of the first version of the network: LiDAR and event inputs are still sequences of points, and are fed in two separate branches. Two main changes can still be highlighted: **(1)** two self-attention modules are used in each encoding branch, to better represent the internal relations between the LiDAR points and events respectively, and **(2)** due to the absence of a memory, the two input branches are not independent anymore: LiDAR and event data must be fed synchronously.

Data Fusion The fusion between the LiDAR and event data is made through a single cross-attention module, where the N_E encoded events are the queries (i.e., they "ask" what should their depths be), and where the N_L encoded LiDAR points are the keys/values (i.e., they provide their depth values for the events). Since the events are the queries in the fusion process, the output is directly of shape (N_E, D).

Decoding For the decoding, to mirror the encoding process, two self-attention modules are used to refine the decoding of the data. Compared to the first version

of the network, skip connections with the input event data are also added for a better learning process. A final linear layer is used to reduce the dimensionality of the data, resulting as for the first network in an output of shape $(N_E, 2)$.

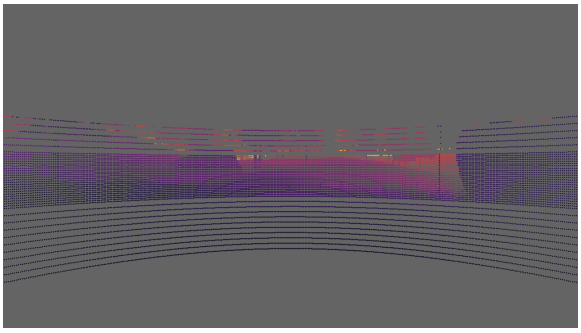
5.4.3 *Issues*

Unfortunately, both versions of the network suffered from major issues, making them ultimately unusable or producing results with low accuracy. We give in the following paragraph a rundown of all these issues.

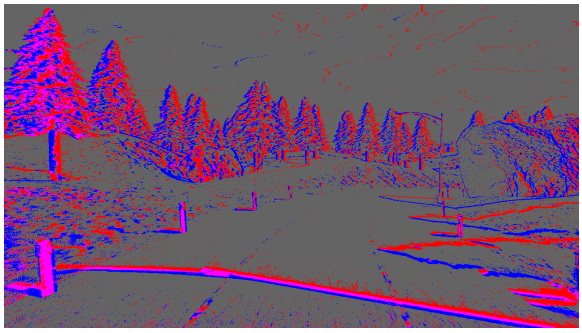
Memory Usage Both versions of the network had to be simplified as much as possible, due to initially requiring too much computer memory. This issue is mostly due to the use of the Transformer architecture: as noted in Eq. (5.5), in its base variant, the Transformer has a space complexity of $\mathcal{O}(N^2)$. In our case, we have as input thousands of LiDAR points, and up to a million of events for the most dynamic scenes, making the memory requirement explode. To counter this issue, we limited the number of input events by randomly cropping the inputs to smaller regions of size (400×400), by using the linearized attention methods of Katharopoulos *et al.* [156] and of Kamal *et al.* [158], and by reducing as much as possible the number of self- and cross-attention modules as shown in Figs. 5.2 and 5.3. However, even after all these simplifications, the first version of the network was never fully functional: because of the central memory, and as recurrence is especially heavy during the training phase, the memory requirement was still often too high.

Granularity of Data The objective of the self- and cross-attention modules in our networks is to encode the relations within and in-between the LiDAR points and the events. However, a single event or a single LiDAR point carries very little amount of data (respectively, a depth for a single pixel, and a change in illumination also for a single pixel). Therefore, representing relations between such small data points is especially complex, as they do not carry an intrinsic meaning. Compared to words in a sentence for instance, nouns, adjectives, and verbs all carry specific meaning, and their relations can be more easily interpreted and modeled by the Transformer; in our case, this would be similar to applying the Transformer on letters independently instead of on complete words.

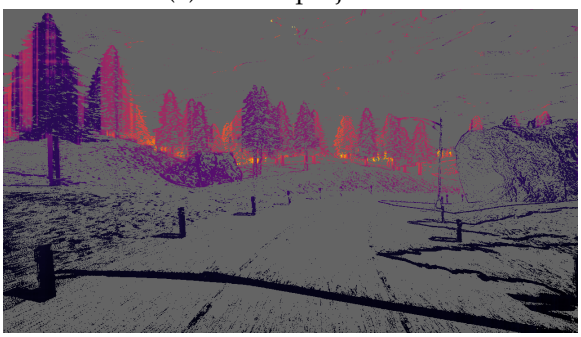
Lack of Structure When LiDAR points and events are positioned in image format, temporal and spatial relations, structures, and patterns appear immediately: buildings, vehicles, trees, ... can be identified with ease. However, in a raw sequence format, this identification becomes a much more complex issue: two LiDAR points or events might have close spatial coordinates, close depths, or close timestamps, but it does not necessarily mean that they both belong to the same object in the scene. Their order in the sequence also does not carry any specific information: the order of two LiDAR points or two events can be inverted without changing their meaning. Once again, if we compare to words in a sentence, their order matters and is crucial for determining their relations and the overall meaning of the sentence



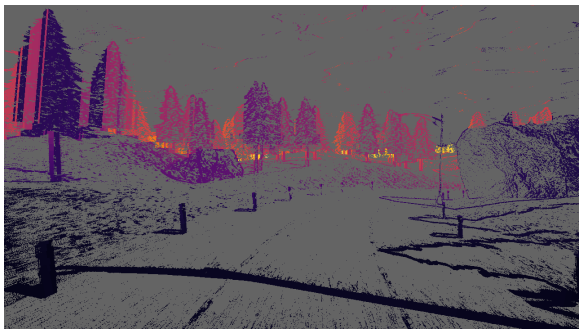
(a) LiDAR projection



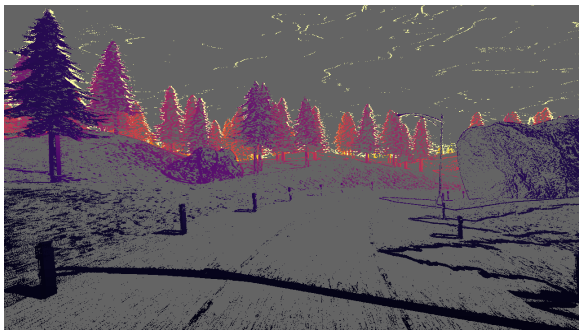
(b) Events



(c) Prediction of the network



(d) Prediction of the NN method



(e) Ground truth

Figure 5.4 – Results for the second version of our sparse network, for the previous depths d_{bf}. As a reminder, the output of the network and of the NN method is just a sequence of depths; we place them here in an image format for a better visualization.

(e.g., “I know what I like” is vastly different from “I like what I know”, despite only two words being inverted).

Nearest-Neighbor-Like Behavior We display in Fig. 5.4 example results achieved by the second version of the network after a complete training on our SLED dataset. As can be seen, while the depth association in Fig. 5.4c is not catastrophic, it remains far from the ground truth: individual trees are not well separated, weird vertical artifacts appear (especially visible for the tree on the left), and events belonging to the sky (i.e., at the maximum distance, in pale yellow in the ground truth) are never identified as such. Actually, when observing the results produced by the simple Nearest Neighbor (NN) method in Fig. 5.4d, the exact same issues appear, and the overall image looks strangely similar. This would indicate that our sparse attention-based network is only able to relate events and LiDAR points based on

their spatial position like the NN method, and that it has no real understanding of the content of the scene to correctly assign the depths.

Loss on Cross-Attention Several attempts were made to improve the results of the network. The first main one was introducing a loss on the cross-attention during the training. The idea here was that supervising how events and LiDAR points are put in relation by the network would force it to stop going for the simple Nearest-Neighbor-like behavior, and instead force it to recognize patterns. However, this additional loss did not bring any improvement, as it only converged a little before the network reached its usual Nearest-Neighbor-like behavior, at which point it stopped converging. Attempts were made to give more importance to this loss, but always resulted in the divergence of the training process.

Maximum Distance Token Another element that was tested for improving the results was the introduction of a special "maximum distance" value, to fix the issue of events belonging to the sky never being identified as such. For the implementation, for each predicted depth, a secondary binary output was added, where the network would have to say whether or not the event should be placed at the maximum distance (and if so, the predicted depth would not be considered). Two different behaviors were observed here: **(1)** in most cases, the network would mark all events as not being at the maximum distance, and would continue having the Nearest-Neighbor-like behavior as presented before; **(2)** but in some cases, the network would consider that all events above a certain y value belonged to the sky, which while more interesting, still does not show a good understanding of the content of the scene, as this y value would not change during the full duration of a sequence, and as some objects like the top of the trees would be included in the sky.

5.4.4 *Additional Experiments*

All the experiments described so far on a sparse depth estimation only constitute a fraction of all the tests that were conducted in the first part of the third year of thesis. Among all the other tests, we can list

- a purely self-attention-based version of the network, where events and LiDAR points are concatenated as a single sequence directly after the encoding heads (with a different embedding added to each modality to allow the network to recognize them);
- a U-Net version of the network, where the event and LiDAR data would be progressively compressed during the encoding, and decompressed (with skip connections) during the decoding, allowing for more attention modules while decreasing memory usage;
- the test of more complex encoding and decoding heads;

- the addition of a special "maximum range" token to the LiDAR points, with a supervision of the cross-attention to force the events at maximum range to have a high attention score when associated to this token;
- the transformation of the maximum distance of the SLED dataset from 200m to 100m, meaning that more events are at that maximum distance in the ground truth, in the hope that the network would better learn about this special case.

Yet, none of these changes brought any improvement to the results presented in Fig. 5.4.

5.5 DENSE DELTA METHOD

5.5.1 *Switching to a Patch-Based Approach*

As shown in the previous section, the fully sparse version of the attention-based depth estimation network never yielded compelling results. Therefore, an important decision was taken: switching from this sparse depth estimation approach, for which the Transformer does not seem to be able to produce correct results, to a dense depth estimation approach, as in Chapter 4, where we expect the Transformer to be more suited.

For that purpose, we will reuse the concept of patches, as initially introduced by Dosovitskiy *et al.* in their work on the Vision Transformer [161], where their goal was to apply the Transformer on images. Their idea was to split each image into N_P small (16×16) patches, linearly project each patch as a single vector of dimensionality D, add a positional encoding to differentiate each vector based on the position of the patch in the image, and use these vectors as the input sequence of shape (N_P, D) to the Transformer. Compared to our sparse approach, each patch contains much more data than a single LiDAR point or a single event, conserves the local spatial and temporal structures of the data it contains, and results in a much more compact input representation of fixed size (from up to a million of events to only a few thousands of patches for HD input), making the relations within and in-between the patches more simple for the network to learn.

While several iterations were required to achieve accurate results, we will only describe here the final version of the network for more clarity. Therefore, we propose in this chapter a novel attention-based recurrent network to estimate dense depth maps from LiDAR and event data. We call it DELTA, for Dense depths from Events and LiDAR using Transformer's Attention.

5.5.2 *Architecture*

As illustrated in Fig. 5.5, our network is based on a U-Net architecture [172], with two input branches for frame-like representations of the LiDAR and accumulated event data, a central memory state, and a decoding branch. In total, DELTA contains 180.9 million of trainable parameters.

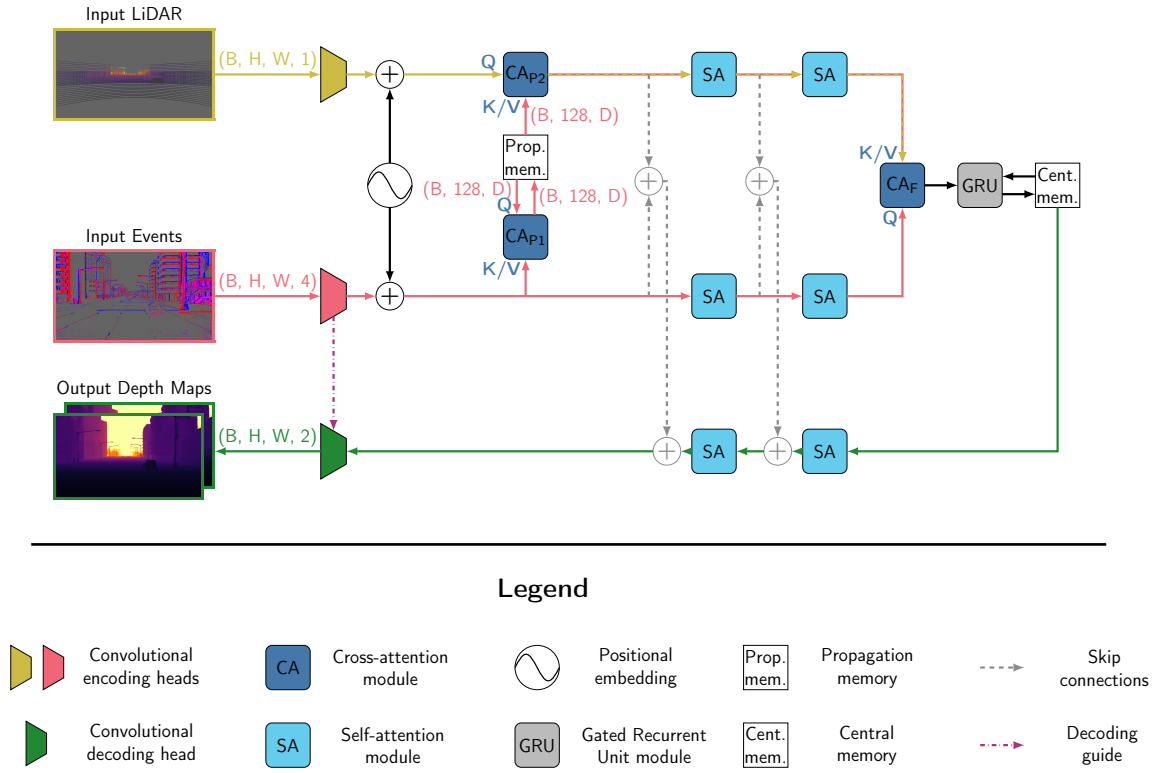


Figure 5.5 – The complete architecture of our DELTA network. Unless noted, data is of shape (B, N_P, D), where B is the batch size, N_P is the number of patches, and D is the dimensionality.

Encoding heads As explained earlier, the event and LiDAR data (both of shape (H, W, C), where C is the number of channels) are first split into small patches of size $P \times P$, as originally proposed by Dosovitskiy *et al.* [161]. This process is done through stacked 5×5 convolutional layers, and results in data of shape (N_P, D), where N_P is the number of patches, and $D = P \times P \times C$ the dimensionality of each patch. These encoded patches are then summed with a fixed 2-dimensional positional embedding, following the formulation of Carion *et al.* [171]. This way, each patch has its own unique signature, making the network able to distinguish them.

Data encoding and fusion Both the event and LiDAR patches each go through two self-attention modules, to encode the internal relations between their own patches. A cross-attention module (CA_F in Fig. 5.5) is then used to encode the cross-relations of both the event and LiDAR patches, with the encoded events being the queries and the encoded LiDAR being the keys/values (as we want in the end to obtain depth maps, similarly to the sparse network of Fig. 5.3).

LiDAR propagation Due to the use of this cross-attention module, contrary to the work proposed in Chapter 4, our LiDAR and event branches can not be totally decorrelated, as both LiDAR and event data are necessary at each time step. On the basis that event data might be more frequently available than LiDAR point clouds, unless new LiDAR data is available, we propagate at each time step the previous

LiDAR data using the incoming events. To do so, the events are used to update a small (Section 5.6.2) propagation memory via a cross-attention module (CA_{P1} in Fig. 5.5). As before, the memory is the queries (i.e., it "asks" how its values should be updated), and the events are the keys/values (i.e., the events provide the new propagation model). This propagation memory is then used in a second cross-attention module (CA_{P2}), where the previous LiDAR data is the queries, and where the propagation memory serves as the keys/values, in order to output an updated LiDAR representation.

Memory update Considering the case where the event camera and the LiDAR are placed on a dynamic platform (e.g., a road vehicle), then if that platform was to come to a halt (e.g., at an intersection or in a traffic jam), few to no event would be produced by the camera. As such, densifying the LiDAR would become a difficult task, as the events would not be able to provide the guiding information required. To solve this issue, given the sequential nature of the inputs, we add a central state, whose role is to give a memory effect to the network. This way, even if the event camera and LiDAR become static, the network can still exploit the memory of the previous information from these sensors to derive accurate predictions. Additionally, the use of this memory state has the benefit of stabilizing the output of the network. Regarding the implementation, the fused LiDAR and event data are given to a Gated Recurrent Unit (GRU) module, which updates this central memory.

Decoding To obtain the final depth maps, the data from the central memory first passes through two self-attention modules, each followed by a skip connection with the corresponding summed and normalized LiDAR and event data from the encoding branch. To be able to have an image-like output, a final decoding head is used to regroup the decoded patches and reshape the data to its original size. This decoding head is composed of stacked convex upsampling modules [89], where the upsampling is guided by the corresponding data from the events encoding head. The output is of shape $(H, W, 2)$, where the "before" depth map D_{bf} is in the first channel, and the "after" depth map D_{af} is in the second channel.

5.5.3 *Loss Functions*

To train DELTA, we use the exact same two complementary losses as in Chapter 4, Section 4.4.4: the pixel-wise ℓ_1 loss $\mathcal{L}_{\text{pw}}$, and the multiscale gradient-matching loss $\mathcal{L}_{\text{msg}}$. Both losses are applied here with the same weights, as we did not observe any improvement by giving more importance to one or the other. Our final loss $\mathcal{L}$ is therefore a summation of both these losses over a full sequence of predictions of length T:

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{\text{bf,af}} (\mathcal{L}_{\text{pw}}^{t} + \mathcal{L}_{\text{msg}}^{t}). \tag{5.10}$$

5.6 EVALUATION

5.6.1 *Datasets*

To conduct the training and evaluation of DELTA, we use in this chapter both our SLED dataset and the MVSEC dataset [62], as in the previous chapter. For a more complete evaluation, we also use the M3ED dataset [64]. However, due to its size, and since its authors do not provide any LiDAR data for the test set, we can not use it as is. To still be able to provide insightful results, we subsampled the dataset and redefined the train/validation/test sets as shown in Table 5.1.

Redefined set	Recordings	Total length
Train	`penno_small_loop_day`; `rittenhouse_day`; `penno_small_loop_night`	8m02s
Val	`horse_day`; `ucity_small_loop_night`	8m59s
Test	`city_hall_day`; `city_hall_night`	9m43s

Table 5.1 – M3ED sets used within this chapter.

5.6.2 *Implementation Details*

Data representation The event data is split in temporal windows of fixed size Δt, based on the rate of the ground truth of each dataset ($\Delta t = 50$ms for SLED and MVSEC, $\Delta t = 100$ms for M3ED). The events in each window are then accumulated into an Event Volume of shape $(H, W, 4)$, following the formulation of Zhu *et al.* [24]. Compared to ALED, where the formulation of Perot *et al.* [25] was used, the formulation of Zhu *et al.* is more compact, by not separating the polarities of the events, allowing for their data to be fully kept by our network while limiting the memory usage, as will be explained in the next paragraph. As for the LiDAR point clouds, they are represented as their projection on the event camera's image plane. Pixels without any depth value are set to 0. Both the LiDAR projections and ground truth depth maps are normalized between 0 and 1, where 1 is the maximum LiDAR range in the dataset in use.

Data size To reach a patch size P of 16 pixels, we use five convolutional layers in the encoding heads, and four convex upsampling modules in the decoding head. To be able to keep all data intact, we chose to use $D = 1024$, as each patch from the event data contains $16 \times 16 \times 4 = 1024$ elements (patch size × number of channels of an Event Volume), and as using greater values of D would raise the memory usage too high. During training, data is randomly cropped to a size of 512×512 pixels for the high-resolution SLED and M3ED datasets, and to 256×256 pixels for the low-resolution MVSEC dataset.

Memories size and initialization Since the role of the central memory is to condense past data, its shape must be the same as the rest of the data in the network,

i.e., (N, D). On the contrary, the propagation memory is only a parametric representation of how the LiDAR data should be propagated to match the current events. While its dimensionality is still constrained to D, its number of elements N can be tuned: we empirically chose a size of 128 in this work. As for their initialization, the central memory is initialized with a copy of the two-dimensional positional embedding, while the initial state of the propagation memory is learned.

Training details For training on all datasets, we use the Adam optimizer [146] with a batch size $B = 4$. When training from scratch on the SLED and the M3ED datasets, 50 epochs are used, the initial learning rate is set to 10^{-4}, and it is decayed by $0.01^{1/49}$ after each epoch (in order to reach a learning rate of 10^{-6} at the last epoch). When training from scratch on the MVSEC dataset, 20 epochs are used, and the learning rate is set to a constant value of 10^{-4}. When finetuning on MVSEC or M3ED, 3 epochs are used, and the learning rate is set to 10^{-5}.

Evaluation metrics For all datasets, we use the same metrics for dense and sparse evaluation as in Chapter 4. Following the convention on the MVSEC dataset [62], results in the following subsections will also be presented with various cutoff distances (10m, 20m, 30m, half the maximum range, and the maximum range). As for ALED, DELTA is trained on the full depth maps, i.e., at the maximum range, and these cutoffs are only applied at test time.

5.6.3 *Results on the SLED Dataset*

We begin by training the network solely on the SLED dataset, and denote this version DELTA_{SL}. Results of DELTA_{SL} on the testing set of SLED are presented in Table 5.2, compared with those of ALED_{SL} originally given in Table 4.2.

Compared to the results of ALED_{SL}, a clear improvement can be seen across all metrics. Improvement is particularly important for the close objects, as the average raw errors at the 10m cutoff is divided by 2 and by 4 for the `Town01` and `Town03` maps respectively. Improvement is less significant at longer ranges, but as noted in the conclusion of Chapter 4 (Section 4.7), close surroundings of the robot are more important when operating than distant objects.

Errors on sparse depths (i.e., depths given to each event) are also much better than those of ALED_{SL}. Despite only being trained on dense data, these errors often surpass those of the simpler Nearest Neighbor (NN) approach, which was not the case for ALED_{SL} on the "before" depth maps D_{bf}.

Errors between the D_{bf} and the D_{af} depth maps are also more consistent. Regarding the depth change maps, errors are often better than those of ALED_{SL}, and the classification of events based on them (whether or not they belong to the edge of an object) is better for all cutoff distances. All these observations indicate that our network has an overall better learning of the temporal propagation of the depths.

Visual results are also presented in Fig. 5.6. Looking at the predicted depth maps, our network is able to infer accurate results, close to the ground truth, but with

Figure 5.6 – Results on the `Town01_08` (left) and `Town03_19` (right) sequences of our SLED dataset, for the "before" depth map D_{bf}. From top to bottom: ground truth depth map; result from ALED$_{SL}$; result from DELTA$_{SL}$; error maps of both methods, where pixels with an error inferior to 0.5m are in white.

		Cutoff	Dense depths errors				Sparse depths errors				Depth change map errors	
			On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Mean (m)	Correctly classified events (%)
			Mean (m)	AbsRel	Mean (m)	AbsRel	NN (m)	$ALED_{SL}$ (m)	NN (m)	$ALED_{SL}$ (m)		(with a threshold of ±1m)
$ALED_{SL}$	Town01	10m	1.24	0.201	1.37	0.236	1.32	1.46	2.24	1.79	**2.11**	90.27
		20m	2.08	0.231	2.27	0.255	1.51	1.84	2.53	2.15	3.18	85.07
		30m	2.72	0.238	2.92	0.260	1.71	2.37	2.83	2.67	**3.88**	81.68
		100m	4.25	0.240	4.51	0.261	2.40	3.48	3.91	3.95	**5.12**	77.48
		200m	**4.53**	0.172	**4.81**	0.187	7.86	**5.44**	9.76	**6.23**	**7.36**	75.54
	Town03	10m	2.00	0.289	2.09	0.301	0.47	0.56	0.67	0.66	1.14	93.70
		20m	2.85	0.299	2.97	0.311	0.64	0.75	1.12	0.87	2.54	87.16
		30m	3.33	0.291	3.45	0.302	0.92	1.11	1.61	1.26	3.23	83.71
		100m	4.60	0.274	4.77	0.284	1.88	2.55	3.17	2.88	4.47	78.50
		200m	4.86	0.215	5.03	0.223	4.43	3.60	5.93	4.10	**6.20**	77.23

		Cutoff	Dense depths errors				Sparse depths errors				Depth change map errors	
			On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Mean (m)	Correctly classified events (%)
			Mean (m)	AbsRel	Mean (m)	AbsRel	NN (m)	$DELTA_{SL}$ (m)	NN (m)	$DELTA_{SL}$ (m)		(with a threshold of ±1m)
$DELTA_{SL}$	Town01	10m	**0.64**	**0.101**	**0.67**	**0.105**	1.32	**1.14**	2.24	**1.25**	2.19	**91.81**
		20m	**1.45**	**0.138**	**1.50**	**0.144**	1.51	**1.62**	2.53	**1.74**	**3.17**	**87.81**
		30m	**2.11**	**0.155**	**2.17**	**0.161**	1.71	**2.03**	2.83	**2.15**	**3.88**	**84.45**
		100m	**3.80**	**0.174**	**3.89**	**0.179**	2.40	**3.05**	3.91	**3.24**	5.14	**79.86**
		200m	5.37	**0.133**	5.42	**0.136**	7.86	6.04	9.76	6.24	7.94	**77.47**
	Town03	10m	**0.49**	**0.081**	**0.50**	**0.082**	0.47	**0.36**	0.56	**0.40**	**0.76**	**97.35**
		20m	**1.15**	**0.107**	**1.18**	**0.109**	0.64	**0.58**	1.12	**0.64**	**2.31**	**91.84**
		30m	**1.72**	**0.121**	**1.77**	**0.124**	0.92	**0.90**	1.61	**0.98**	**3.06**	**88.35**
		100m	**3.12**	**0.133**	**3.18**	**0.136**	1.88	**2.25**	3.17	**2.38**	**4.30**	**82.88**
		200m	**4.81**	**0.114**	**4.81**	**0.116**	4.43	**3.54**	5.93	**3.72**	6.69	**81.18**

Table 5.2 – Errors of $ALED_{SL}$ (top) and of $DELTA_{SL}$ (bottom) on our SLED dataset for various cutoff depth distances. From left to right: dense depth errors (absolute and relative) on the full D_{bf} and D_{af} depth maps; sparse depth errors (only pixels with events are considered); absolute depth change maps ($D_{af} - D_{bf}$) errors and percentage of correctly classified events based on this depth difference.

edges looking less sharp than the ones of ALED. Looking at the error maps, they confirm the observations made on the quantitative results: we commit very small errors at close ranges (especially for the ground) and larger errors at longer ranges, and ALED commits small errors over the whole depth map.

If the reader is interested, the full numerical results for each sequence of the testing set and more visual results are both given in Appendices C.1 and C.2 respectively, and results in video format are available through the link given at the very beginning of this chapter (Page 69).

5.6.4 *Results on the MVSEC Dataset*

As for ALED, to evaluate our network on the MVSEC dataset, we follow two distinct strategies:

- we only train it on MVSEC, noted as $DELTA_{MV}$, and
- we reuse the weights trained on the SLED dataset, $DELTA_{SL}$, and finetune them on the MVSEC dataset: we denote it as $DELTA_{SL \to MV}$.

The results of both versions are given in Table 5.3, in addition to the results of other methods of the state of the art.

Compared to the results of StereoSpike [35], RAMNet [33], EvT^+ [147], and Cui *et al.* [45], our method yields consistently better results, especially with pre-training on

Recording	Cutoff	Events (stereo)	Events & Frames		Events & LiDAR				
		StereoSpike [35]	RAMNet [33]	EvT+ [147]	Cui *et al.* [45]	ALED_{MV}	$\textbf{DELTA}_{\textbf{MV}}$	$\text{ALED}_{\text{SL}\to\text{MV}}$	$\textbf{DELTA}_{\textbf{SL}\to\textbf{MV}}$
outdoor_day_1	10m	0.79	1.39	1.24	1.24	0.91	0.74	**0.50**	0.59
	20m	1.47	2.17	1.91	1.28	1.22	1.10	**0.80**	0.93
	30m	1.92	2.76	2.36	4.87	1.43	1.34	**1.02**	1.17
	50m	-	-	-	-	1.67	1.65	**1.31**	1.46
	100m	3.17	-	-	-	1.96	1.98	**1.60**	1.79
outdoor_night_1	10m	**1.38**	2.50	1.45	2.26	1.75	1.52	1.52	1.54
	20m	2.26	3.19	2.10	2.19	2.10	1.93	**1.81**	2.00
	30m	2.97	3.82	2.88	4.50	2.25	2.17	**1.95**	2.25
	50m	-	-	-	-	2.44	2.44	**2.20**	2.48
	100m	4.82	-	-	-	2.73	2.81	**2.54**	2.87
outdoor_night_2	10m	-	1.21	1.48	1.88	1.19	1.17	1.09	**1.01**
	20m	-	2.31	2.13	2.14	1.65	1.60	**1.49**	1.50
	30m	-	3.28	2.90	4.67	1.81	1.79	**1.64**	1.74
	50m	-	-	-	-	1.95	1.97	**1.80**	1.91
	100m	-	-	-	-	2.11	2.17	**1.97**	2.13
outdoor_night_3	10m	-	1.01	1.38	1.78	0.85	0.94	0.81	**0.78**
	20m	-	2.34	2.03	1.93	1.25	1.37	**1.16**	1.26
	30m	-	3.43	2.77	4.55	1.42	1.60	**1.33**	1.53
	50m	-	-	-	-	1.57	1.80	**1.51**	1.74
	100m	-	-	-	-	1.73	1.99	**1.66**	1.95

Table 5.3 – Absolute depth errors (in meters) on the MVSEC dataset for various cutoff depth distances (for the "before" depth maps D_{bf}).

the SLED dataset. Compared to ALED_{MV} and $\text{ALED}_{\text{SL}\to\text{MV}}$, the results of DELTA_{MV} and $\text{DELTA}_{\text{SL}\to\text{MV}}$ remain relatively close, often achieving first or second place in the case of $\text{DELTA}_{\text{SL}\to\text{MV}}$. Why the clear improvement from ALED observed on the SLED dataset (Section 5.6.3) is not replicated here is most probably because of the large change in resolution between the SLED and MVSEC datasets, as the information contained in the 16×16 patches becomes quite different on the low-resolution data of MVSEC. The position of the patches also differs between the two datasets: a patch located at the bottom of an image from MVSEC has the position of a patch located at the middle of an image from SLED, meaning that the network must re-learn the link between the positional encoding and the information in the patches. Both these elements make the finetuning more complex for an attention-based network like DELTA than for a convolutional-based network like ALED, hence the results of Table 5.3.

Visual results are also presented in Fig. 5.7, comparing them with those of Cui *et al.* [45] and of $\text{ALED}_{\text{SL}\to\text{MV}}$. While the method of Cui *et al.* allows for a better conservation of the edges of the objects, it fails at producing smooth depth gradients (especially on the ground, where clear delimitations are visible), and it is limited to the vertical range of the LiDAR data. Comparing the results of $\text{DELTA}_{\text{SL}\to\text{MV}}$ to those of $\text{ALED}_{\text{SL}\to\text{MV}}$, the visualizations reflect the observations made during the quantitative evaluation: the depth maps have a good accuracy, but are slightly less sharp than those of $\text{ALED}_{\text{SL}\to\text{MV}}$, and lack some small details. As noted in Chapter 4 for ALED, DELTA also suffers from the lack of ground truth data for the sky, resulting in the purple blobs in the upper areas of all predictions.

If the reader is interested, more qualitative results are given in Appendix C.3, and results in video format are available through the link given at the very beginning of this chapter (Page 69).

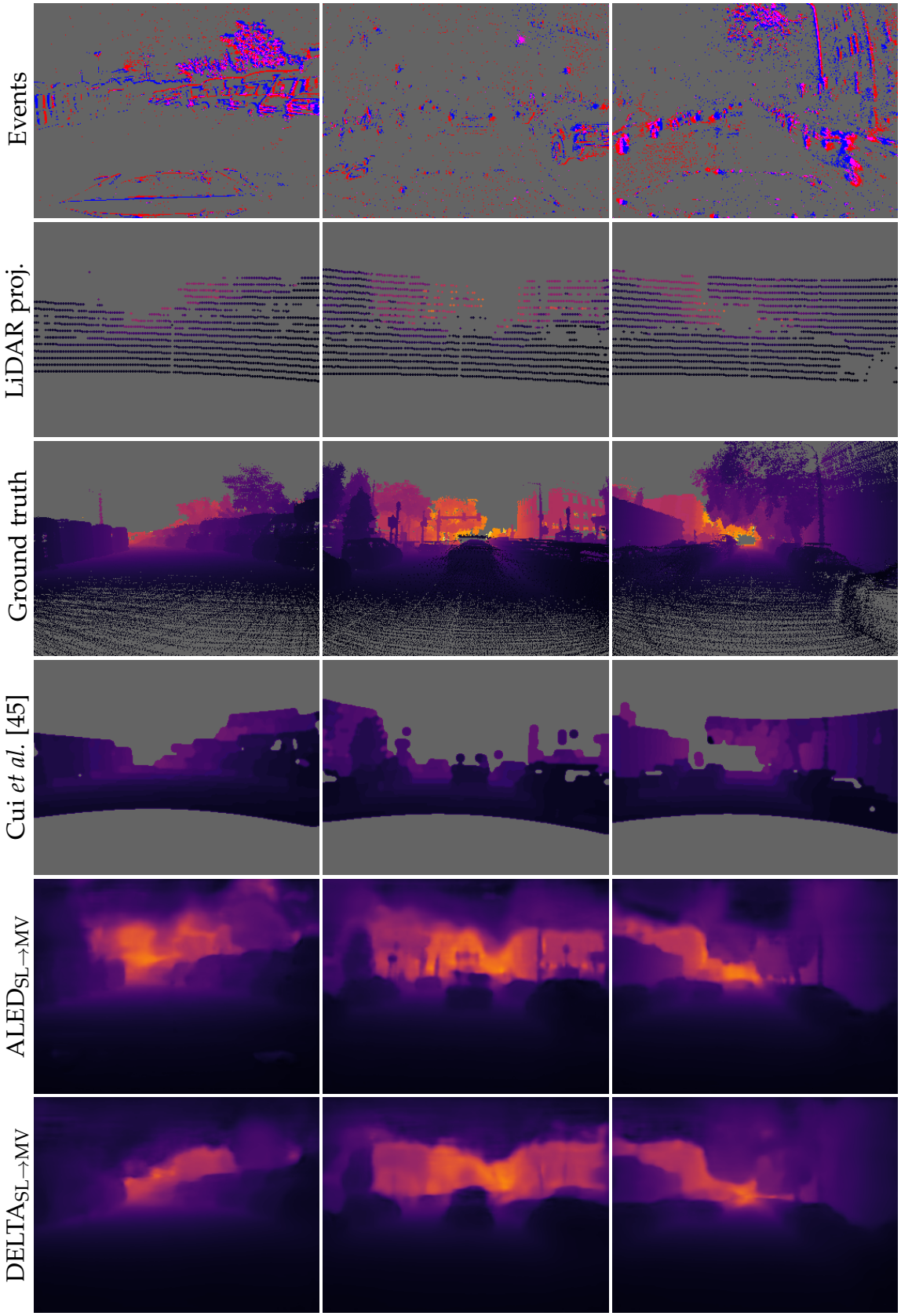


Figure 5.7 – Qualitative results on the MVSEC dataset. From top to bottom: events; LiDAR projection (with size of points increased for a better visibility); ground truth; results from Cui *et al.* [45]; results from ALED$_{SL \to MV}$; results from DELTA$_{SL \to MV}$. Sequences shown, from left to right: `outdoor_day_1`; `outdoor_night_1`; `outdoor_night_2`.

5.6.5 *Results on the M3ED Dataset*

Like the evaluation on the MVSEC dataset, we follow two distinct strategies when evaluating on the M3ED dataset:

- we only train it on M3ED, noted as DELTA_{M3}, and
- we reuse the weights trained on the SLED dataset, DELTA_{SL}, and finetune them on the M3ED dataset: we denote it $\text{DELTA}_{\text{SL}\to\text{M3}}$.

Both the results of DELTA_{M3} and of $\text{DELTA}_{\text{SL}\to\text{M3}}$ are given in Table 5.4.

Recording	Cutoff	DELTA_{M3}	$\text{DELTA}_{\text{SL}\to\text{M3}}$
`city_hall_day`	10m	**0.30**	0.58
	20m	**0.45**	0.71
	30m	**0.60**	0.85
	60m	**0.92**	1.07
	120m	**1.02**	1.19
`city_hall_night`	10m	**0.26**	0.55
	20m	**0.39**	0.67
	30m	**0.49**	0.79
	60m	**0.69**	1.02
	120m	**0.81**	1.21

Table 5.4 – Absolute depth errors (in meters) on the M3ED dataset for various cutoff depth distances (for the "before" depth maps D_{bf}).

Again, the errors remain very low on both sequences, with the error at the maximum cutoff distance being close to or under 1m. Yet, results here differ from those on the SLED and MVSEC datasets, as better errors are obtained when training directly on the M3ED dataset than when pre-training on the SLED dataset. This apparent discrepancy can be explained when observing the visual results given in Fig. 5.8. As shown, the ground truth depth maps of the M3ED dataset are very sparse, and due to the process used for building them, ground truth values are concentrated in the center of the objects, with very few values at their edges. As such, by being only trained on this dataset, DELTA_{M3} outputs good numerical errors, but poor-looking depth maps with large blobs and a scanline effect following the outline of LiDAR points projection, making the identification of objects difficult. On the contrary, the pre-training on the SLED dataset allows $\text{DELTA}_{\text{SL}\to\text{M3}}$ to output depth maps where objects appear more clearly (for instance, the lamp post in the top-left part of Fig. 5.8). However, by doing so, the output depth maps are more rough. Another element which could explain why the finetuning is difficult is that network inputs when using M3ED are intrinsically different from those of SLED, with events being accumulated over 100ms instead of 50ms (see Section 5.6.2), and with the LiDAR data covering the whole vertical range of the image.

If the reader is interested, more qualitative results are given in Appendix C.4, and results in video format are available through the link given at the very beginning of this chapter (Page 69).

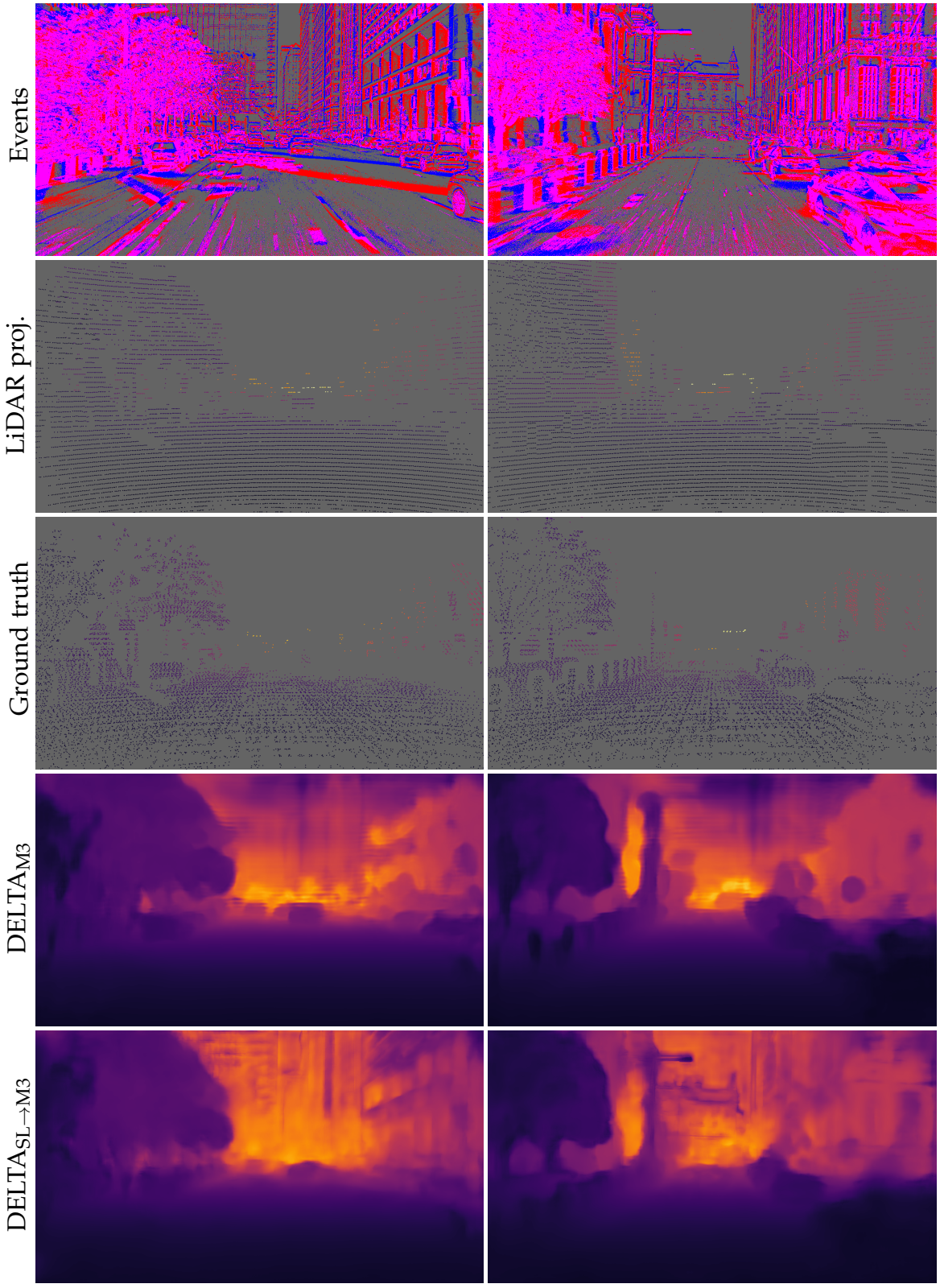


Figure 5.8 – Qualitative results for the `city_hall_day` sequence from M3ED. The size of points for both the LiDAR projection and the ground truth was increased for a better visibility. Zoom on numerical version is required to see the scanline effect on DELTA$_{M3}$.

5.6.6 *Ablation Study*

In this final subsection, we investigate the importance of the two memories in the network, as well as the role of the central cross-attention module CA_1. For that purpose, we propose three variants of DELTA:

- DELTA^{NPM}, with no propagation memory;
- DELTA^{NCM}, with no central memory;
- DELTA^{NCA}, with no central cross-attention module, and with a GRU module for each encoding branch.

We give an overview of the shape of these modified networks in Figs. 5.9 to 5.11. For the evaluation, as was done for Section 5.6.3, we trained these modified versions of DELTA on the SLED dataset, allowing for comparison with the proposed version of the network. The choice of doing this analysis on SLED was motivated by the fact that both the MVSEC and the M3ED datasets suffer from shortcomings in their respective ground truth, which could alter this ablation study.

Cutoff	DELTA$_{SL}$		DELTA$_{SL}^{NPM}$		DELTA$_{SL}^{NCM}$		DELTA$_{SL}^{NCA}$	
	D_{bf}	D_{af}	D_{bf} (rel.)	D_{af} (rel.)	D_{bf} (rel.)	D_{af} (rel.)	D_{bf} (rel.)	D_{af} (rel.)
10m	**0.57**	**0.58**	0.85 (+0.28)	0.86 (+0.28)	0.91 (+0.34)	0.91 (+0.33)	0.95 (+0.38)	0.94 (+0.36)
20m	**1.29**	**1.33**	1.64 (+0.35)	1.66 (+0.33)	1.71 (+0.42)	1.70 (+0.37)	1.80 (+0.51)	1.79 (+0.46)
30m	**1.91**	**1.96**	2.24 (+0.33)	2.28 (+0.32)	2.21 (+0.30)	2.22 (+0.26)	2.40 (+0.49)	2.41 (+0.45)
100m	**3.44**	**3.52**	3.73 (+0.29)	3.78 (+0.26)	3.53 (+0.09)	3.56 (+0.04)	3.85 (+0.41)	3.89 (+0.37)
200m	5.09	5.11	5.01 (-0.08)	5.02 (-0.09)	**4.49** (-0.60)	**4.52** (-0.59)	5.03 (-0.06)	5.10 (-0.01)

Table 5.5 – Absolute and relative depth errors (in meters) on the full testing set of the SLED dataset, for alternative versions of DELTA (No Propagation Memory, No Central Memory, No Cross-Attention).

Results are presented in Table 5.5. The base variant DELTA$_{SL}$ produces the best results, except for the maximum 200m cutoff range, where it is beaten by the three other variants. The version without a propagation memory DELTA$_{SL}^{NPM}$ has a constant additional error of around 0.3m (except at the 200m range, where the errors are similar), highlighting the importance of temporally propagating the LiDAR data with the events. The version without the central memory DELTA$_{SL}^{NCM}$ also performs worse, albeit with an additional error that reduces the greater the cutoff distance is, beating DELTA$_{SL}$ by 0.6m at the maximum cutoff range. It should be noted however that the car which the sensors are mounted on in the SLED dataset rarely stops, and if it does, it is for very short periods of time. As explained in Section 5.5.2, since the main role of the central memory is to still provide accurate results in these cases, it only serves here its secondary role of being a stabilization medium, which is still valuable at close range given the numerical results. Finally, the version without the central cross-attention module DELTA$_{SL}^{NCA}$ is the worst performing variant, as it has the largest error across all cutoff ranges, only slightly beating DELTA$_{SL}$ at the 200m cutoff. As such, the central cross-attention module CA_1 is crucial for encoding

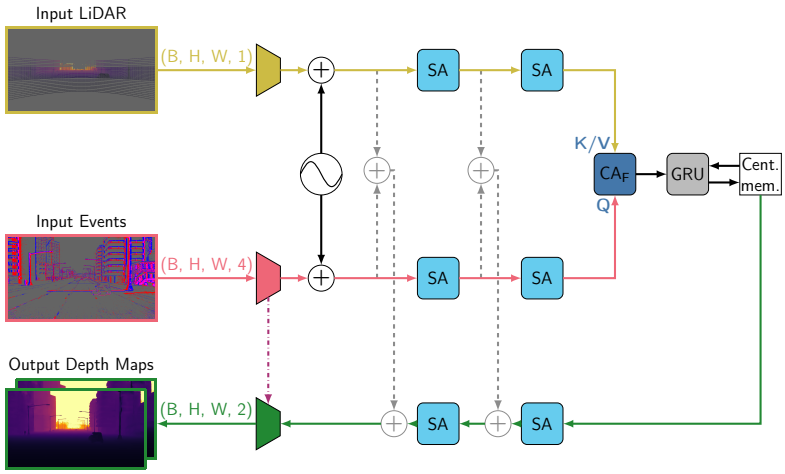


Figure 5.9 – The alternative architecture without propagation memory, DELTA^{NPM}.

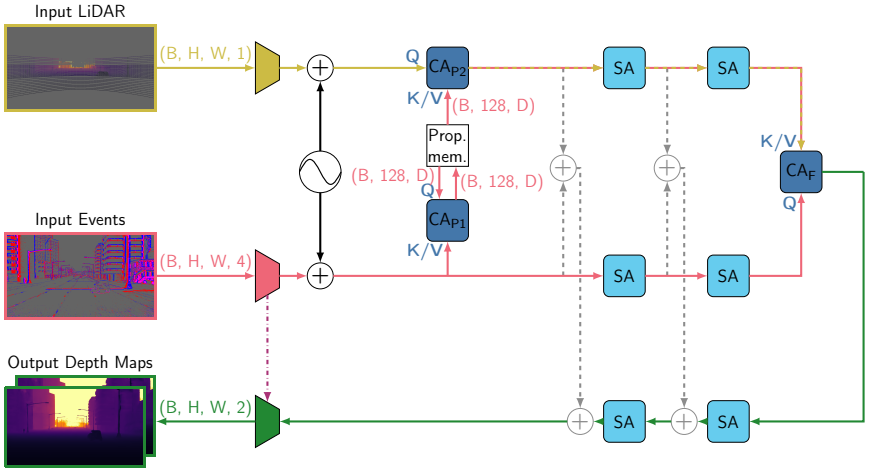


Figure 5.10 – The alternative architecture without central memory, DELTA^{NCM}.

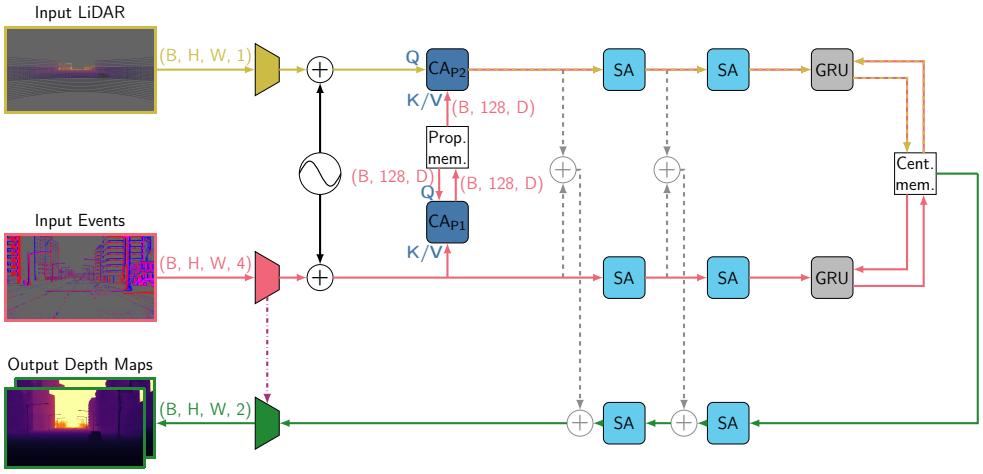


Figure 5.11 – The alternative architecture without the central cross-attention, DELTA^{NCA}.

the cross-relations between the encoded LiDAR and event data before updating the central memory.

5.7 CONCLUSION AND DISCUSSIONS

Throughout this chapter, a study on the estimation of sparse depths from sparse inputs was conducted, and a new attention-based network for fusing LiDAR and event data to construct dense depth maps was proposed, DELTA. Thanks to the introduction of a propagation memory between cross-attentions, DELTA is able to extrapolate LiDAR with events at higher rate for an optimal fusion. A GRU is also added between the encoding and decoding stages, allowing for a short-term central memory and more robust outputs. As ALED in Chapter 4, our DELTA network predicts 2 depths: before and after the events occur, allowing for depth change map computation and for richer analysis of the scene dynamics. A thorough evaluation including an ablation study was conducted on three datasets of the state of the art to demonstrate the relevance of these propositions. On our synthetic SLED dataset, a significant improvement was achieved for short ranges, with the average error being reduced up to four times when compared to ALED. On the real MVSEC and the M3ED datasets, DELTA remains competitive, with low errors across all cutoff ranges. This work is an increment for event and LiDAR processing using Transformers, still hard to apply well on these sparse modalities. We believe that the proposed architecture can serve as a basis for other applications or for fusion with other sensors.

In hindsight, further modifications could be brought to this work in order to improve its overall performance. **(1)** As shown in Section 5.6, the improvement of errors over short ranges is done at the cost of lower precision at longer ranges. Adapting the training procedure by introducing variable weights for every cutoff distance could be a solution to make sure all of them are optimized equally. **(2)** Also, as noted in Section 5.5.2, DELTA is a relatively large network, with over 180 million parameters. In comparison, ALED is composed of 26 million parameters. If required, further analysis could probably allow for a reduction in the number of parameters in our network, while retaining a similar accuracy. **(3)** As for ALED, the choice of the input representations could also be re-examined. While the Event Volume [24] is a compact and standard representation of event-based data, it can lead to information being lost under fast motion or rapidly changing lighting conditions. As proposed by Zubić *et al.* [27], automatically optimizing the input event representation could allow for a better conservation of data. As for the LiDAR data, projecting it in the frame of the event camera also leads to a loss of information. While making the LiDAR densification problem more simple by already having the event and LiDAR data in the same frame of reference, more than half of the point cloud is discarded, even though it could provide precious information about the structure of the scene. Working with a different representation of the LiDAR data, or working directly in the 3D space as advocated by Cui *et al.* [45], could both be solutions to this issue. **(4)** Finally, we regret the lack of a real-life dataset with dense and high-resolution

ground truth depth maps. Such a dataset would allow for a better training and a better baseline for comparison than the currently existing datasets.

Looking back at our initial goals described at the beginning of this chapter, improving the accuracy for close ranges has been successfully achieved, but the fully sparse aspect has remained unsuccessful. We believe that, in its current state, a fully sparse attention-based architecture is ill-suited for such a complex problem, especially due to the memory requirements even for small networks. Comparatively, a dense attention-based architecture relying on patches like DELTA seems much more adapted in the general case, at the expense of the sparsity of data. One axis of improvement we did not have time to treat as part of the thesis but that we would like to explore further on was the idea of still keeping dense patch-based inputs, but having a sparse output. This way, the spatial and temporal patterns which are not easily identifiable in sparse inputs would still be recognized, with sparsity being introduced at decoding time. In the case of DELTA, this approach would either require **(1)** the addition of a small third encoding branch (for the sparse events), of a cross-attention module for making these encoded events query the central memory, and of a second sparse decoding branch (as still predicting also dense results would probably make the training easier); or **(2)** a more in-depth redesign of the network.

6 GENERAL CONCLUSION

6.1 CONCLUSION

Throughout this thesis, the issues of motion and depth for scene analysis were studied. In particular, event cameras of both low-, mid-, and high-resolution were used for solving both problems, and the introduction of a LiDAR sensor was also critical in the case of the depth estimation.

In the case of motion estimation, a real-time optical flow method was proposed, relying on a single event camera. A pipeline-based computation was proposed, for allowing parallelism of the tasks. In particular, a 4-step method was proposed here, for temporally accumulating events as binary edge images, denoising and correcting them, converting them in a dense frame-based representation, and computing the final optical flow using a state-of-the-art frame-based method.

In the case of depth estimation, a convolutional-based network was first proposed, relying on both high-rate data from a single event camera, and on lower-rate point clouds from a single LiDAR sensor. This network is composed of two separate encoding branches for processing both inputs asynchronously, of central memory states for recurrence and for fusing the two modalities, and of a single decoding branch for estimating the final dense depth maps. A notion of "two depths per event" was also proposed, for taking into account the change-based nature of events, and a simulated dataset was recorded and released for improving the training and evaluation of our network.

Finally, an attention-based network was also proposed, for improving the fusion of the LiDAR and event data. This network is composed of two separate encoding branches for processing both inputs, of a propagation memory for temporally upsampling the LiDAR data, a central cross-attention module for fusing the two modalities, a single central memory for recurrence, and a decoding branch for estimating the final dense depth maps.

Both theoretical and practical analysis were conducted to demonstrate the relevance of the contributions. For optical flow, accuracies close to the state of the art were obtained, but for much faster computation times (250Hz and 83Hz for low- and high-resolution inputs respectively) and with a low latency (10ms and 27ms for low- and high-resolution inputs respectively). For the depth estimation, the convolutional-based network set first a new state-of-the-art reference, with an error reduction of up to 61% compared to the previous LiDAR-and-event state-of-the-art. The attention-based network then refined these results, by allowing for an error reduction for close ranges, dividing the errors of the convolutional network up to four times.

As a general conclusion, we have seen throughout this thesis that motion and depth estimation are critical components for scene analysis. By nature, the event camera constitutes a fitting sensor for these problems, due to its innate motion-related

encoding of data. The LiDAR sensor also complements surprisingly well the event camera for a multimodal depth estimation. While the robotic field was considered here, as being the one where scene analysis is the most important and where the combination of event cameras and LiDARs is the most likely to be used, we believe that our research could be extended to other fields, from biology to industrial automation.

6.2 CONTRIBUTIONS

Across this thesis work, several major contributions were made, which we summarize here.

Optical Flow (RTEF) The first and main contribution of the work on event-based optical flow was proposing a real-time method. While some works had already been done in that sense [48, 105], they lacked accuracy, and hardly scaled to a high-resolution input. On the contrary, our work was directed from the start with the multi-resolution aspect in mind, and our method is therefore able of a good accuracy for both low-, mid-, and high-resolution event cameras. Another contribution of this work was our "negated exponential distance transform", which was especially critical for improving the accuracy results by giving us the ability to use a proven frame-based optical flow method. While only used in the context of optical flow estimation here, we believe that this dense representation could also provide interesting results for other applications where real-time is required. Finally, a high-speed high-definition event-based indoor dataset was recorded and shared, for allowing the evaluation of our optical flow method on complex and fast motions, a feature which is missing from most state-of-the-art datasets.

Depth Estimation (ALED) Regarding the work on depth estimation using a convolutional network, the very first contribution was the idea itself of combining a LiDAR and an event camera for estimating dense depth maps, which had not been explored previously in the literature. As showed throughout Chapters 4 and 5, despite their apparent incompatibilities (different type of data and of sparsity, vastly different output rates), these two sensors actually complement each other perfectly in the context of this problem. Another fundamental contribution of this work was the idea of associating two depths to each event, which is the only method that takes into account the change-based nature of events while still allowing for their reprojection in the 3D world. One of the main contribution of this work also lies in the ALED network itself. While originally inspired by the RAMNet architecture of Gehrig *et al.* [33], ALED is refined in several ways (more encoding scales, the use of convex upsampling during decoding), largely improving the accuracy of the depth maps. Finally, the recording and sharing of the simulated SLED dataset is also a major contribution, for allowing the training and evaluation of depth estimation methods with dense ground truth data, a feature that is missing from the state-of-the-art real-world datasets. SLED was also made in a modular way, meaning that it could be extended for introducing further tasks (optical flow, semantic segmentation, object detection and recognition, ...).

Depth Estimation (DELTA) As for the second work on depth estimation, our first contribution is on the concept itself of using an attention-based network for fusing LiDAR and event data. Even if they were not successful, we believe that our tests on a fully sparse version of this network are valuable, as they allowed for a better understanding of the Transformer and the attention mechanism, their limitations, and as they highlight the current need for a dense, patched-based representation. Of course, our main contribution here is the DELTA network itself, which is a fully novel recurrent network architecture relying on self- and cross-attention modules. As shown in the ablation study of Chapter 5, Section 5.6.6, the most critical contributions are the propagation memory, allowing for a better temporal upsampling of the LiDAR data, the central memory and its GRU-based update, allowing for the introduction of recurrence, solving the cases where few events are available and offering more stability overall, and the central cross-attention module, for modeling the interactions between the LiDAR and event data.

Open Science A significant emphasis during these three years has also been put on ensuring that our work was as accessible and as exploitable by the community as possible. All preprints of the published articles were made available on arXiv and/or HAL. Dedicated project pages were also created, for giving a quick overview of each work:

- `https://vbrebion.github.io/RTEF/`
- `https://vbrebion.github.io/ALED/`

All the source codes of the published methods were open-sourced, published in a clean version, and properly documented:

- `https://github.com/heudiasyc/rt_of_low_high_res_event_cameras`
- `https://github.com/heudiasyc/ALED`
- `https://github.com/heudiasyc/SLED`

Finally, datasets were also made publicly available, through the platform of the laboratory:

- `https://datasets.hds.utc.fr/project/7`
- `https://datasets.hds.utc.fr/project/9`

Community Source Codes Several contributions were also made to improve other source codes and repositories of the community, and are briefly listed here:

- `https://github.com/astuff/avt_vimba_camera/pull/22`
- `https://github.com/prophesee-ai/prophesee_ros_wrapper/pull/29`

- https://github.com/uzh-rpg/rpg_dvs_ros/pull/111
- https://github.com/uzh-rpg/event-based_vision_resources/pull/150
- https://github.com/uzh-rpg/event-based_vision_resources/pull/213
- https://github.com/carla-simulator/carla/issues/5367
- https://github.com/carla-simulator/carla/issues/5732
- https://github.com/carla-simulator/carla/issues/6103
- https://github.com/carla-simulator/carla/issues/6552
- https://github.com/AlbertoSabater/EventTransformerPlus/issues/1

6.3 DISCUSSIONS AND PERSPECTIVES

As discussed throughout the chapters of this thesis, several improvements, reworks, and future works could be proposed to improve the contributions proposed as part of this thesis.

Regarding our optical flow method first, RTEF, one of the main limits to highlight is its inability to produce accurate results in case of large motions over short periods of time. This limitation mainly comes from our use of a frame-based approach, which discards the timestamps. One solution is to reduce the accumulation time for these cases, but it implies disregarding the real-time constraint. By construction, learning-based and contrast-maximization-based methods perform better on these cases, but their real-time compatibility remains an open question. Similarly, while our method had state-of-the-art performances when it was published in 2021 [74], optical flow has since become one of the most researched subjects in event-based vision, and learning-based and contrast-maximization-based methods have particularly risen as clear favorites, with impressive accuracy. Finally, a theoretical limitation that was highlighted in the concluding remarks of Chapter 3 was on the intrinsic sense of computing optical flow on events the way we did, that is, computing motion of changes themselves. This remains an open question, and we believe that an in-depth theoretical analysis on this issue may be required for making sure that event cameras are exploited correctly.

Regarding our two depth estimation methods, ALED and DELTA, both could also be improved in several ways. As discussed, the choice of the Event Volume as the input representation for the events could be revised. While it was chosen at the time as a compact representation of the event data that still kept a nearly maximal amount of information, more recent works [27] have shown that having a learned representation could also be beneficial, by only keeping the required data in an even more compact form factor. Both our convolutional and attention-based networks are also not real-time compatible. The emphasis was never set on this specific point during these works, as we observed through our work on optical flow that this constraint limited our final accuracy, and that it should not be a priority considering that we were among the firsts to examine the fusion of the

LiDAR and event modalities. However, for an implementation on a real robotic system, some revisions would have to be brought to our networks, in an effort to make them faster while retaining most of their accuracy. This could be done through quantized low-precision methods for instance, like 4-bit networks [173, 174]. Despite our best efforts, we were also unable to make the fully sparse version of the attention-based network produce adequate results. We believe that one of the biggest limitations was the small size of these networks, which limited their learning ability on such a complex task. With the popularity of the Transformer architecture, however, more optimized implementations of the attention calculation process are starting to be integrated in popular libraries, which could be of great help in the future for building bigger attention-based networks with less memory restrictions. Finally, we believe that a real-world dataset with dense ground truth depth maps would be of great help for training and evaluating more accurately the current and future event-based depth estimation methods. Several sub-works in that direction were conducted during the thesis, as described in Appendix A, but several technical issues and an overall lack of time ultimately led us to the recording of the simulated SLED dataset.

More generally, as discussed earlier, one of the main extensions of this work could be on 3D scene flow, which could be obtained in our case by combining both the 2D optical flow data and the depth estimations. Scene flow is slowly rising and becoming an important issue in computer vision, but it remains to this day a rather unexplored issue with event cameras [175, 176], and could therefore constitute an interesting subject to explore.

Further work could also be conducted on the interactions with the "planning" and "control" modules of the robotic pipeline shown in Page 1 of this thesis. The work conducted during these three years was purely centered on a perception aspect, but being able to apply the optical flow and depth estimation methods on a real robot would probably pose interesting questions as to how they could be used further down the pipeline, and as to the level of accuracy needed for accomplishing desired tasks.

As a more general conclusion to these perspectives, we believe that, while the event camera is an intriguing and fascinating sensor, large amounts of work remain to be done to level them with their traditional frame-based counterparts, and to exploit their unique properties to their best. As such, this thesis constitutes only a small contribution towards that goal, but we hope that both our theoretical and technical works will be able to guide and inspire other authors.

A ADDITIONAL EXPERIMENTS

In this first appendix, we describe some of the experiments that were conducted during the three year of thesis, but that were not included in the main part of thesis for diverse reasons.

A.1 ACQUISITION OF REAL-WORLD DATA

A.1.1 *For Optical Flow*

Before making use of the Flow Warping Loss (FWL) metric in order to evaluate the quality of our optical flow results on high-resolution data, we initially tried to find an event-based dataset with a high-resolution camera and with a ground truth for optical flow. This research being unsuccessful, we decided to record our own dataset.

To do so, as illustrated in Fig. A.1, we used a Parrot Jumping Sumo minidrone as a moving object, and placed our event cameras (Prophesee Gen4 [13] and DAVIS240C [15]) directly above it, looking downwards. In order to generate the ground truth, we also added a high-resolution RGB camera (Allied Vision Mako G-192C) between the two event cameras, also looking downwards. Since the robot was evolving on the ground, and since its height was negligible compared to the height at which the cameras were positioned, an approximation of the robot moving on the ground being equivalent to a flat shape moving on a plane could be made. With this approximation, and with adequate calibration and synchronization between the cameras, a simple homography could then be applied to superimpose data from various cameras, as shown in Fig. A.2. Finally, by using a state-of-the-art optical flow method on the frames (RAFT [89]), and by applying our optical flow method on the events, a quantitative and qualitative comparison could then be made.

Five sequences were recorded, where the robot was following various patterns under the cameras: moving in a straight line, slaloming, drawing a circle, drawing a square, and moving in a straight line with a second robot coming from the opposite direction. An example of results is shown in Fig. A.3, and a playlist of videos is available at `https://www.youtube.com/playlist?list=PLLL0eWAd6OXBdeFOWaNMw3d24qW4EhM5f`.

In the end, this dataset was not used as part of the evaluation of the final work, due to its simplicity and due to the FWL metric allowing for an evaluation on any event-based dataset. Yet, it was a useful tool for initially designing, tweaking, and verifying the performances of our method on simple scenarios.

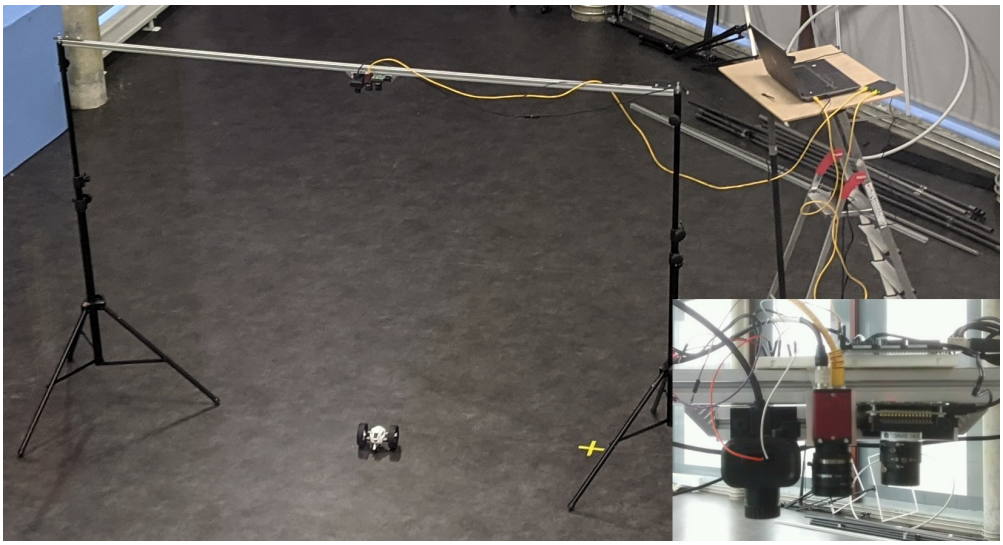


Figure A.1 – Setup used for the recording of the optical flow dataset. The Parrot Jumping Sumo minidrone is visible in the center, and a zoomed view of the cameras (Prophesee Gen4 / Allied Vision Mako G-192C / DAVIS240C) is given at the bottom right.

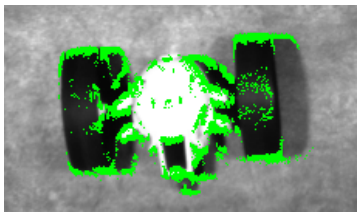


Figure A.2 – Superimposed frame from the Mako camera (in grayscale) and events from the Prophesee camera (in green).

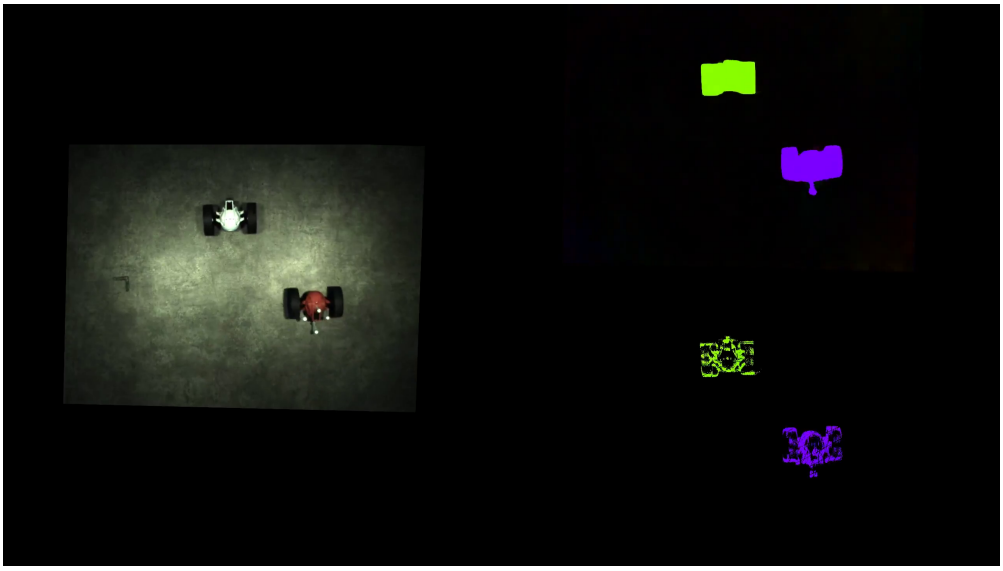


Figure A.3 – Example results on the dataset. Left is the view from the RGB camera (after applying the homography), top right is the frame-based optical flow from RAFT, and bottom right is our event-based optical flow.

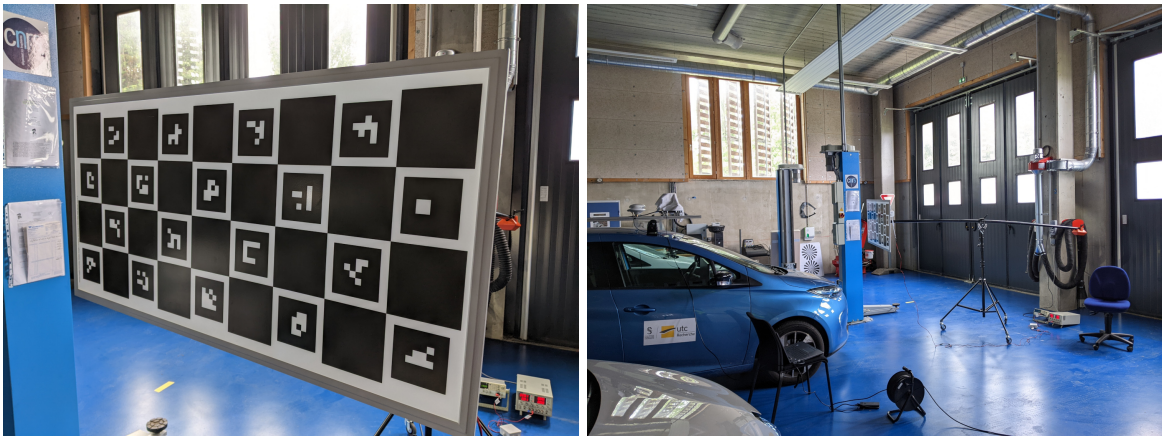


Figure A.4 – Left: detailed view of the back-illuminated calibration board, with the ChArUco pattern. Right: larger view, showing the board positioned in front of the Zoe car, with the mechanical arm used for manipulating it.

A.1.2 *For Depth Estimation*

As described in Section 6.3, one of the parallel works of the thesis was conducted on the potential recording of a dataset using the robotized Renault Zoe cars of the lab, for depth estimation. Such a dataset was never actually recorded in the end, but we describe here the works conducted on calibration and synchronization.

Calibration

One of the first issues that we encountered while trying to record sequences was on the projection of LiDAR points into the frame of the event camera. Doing so required a calibration between the two sensors, an issue which had not been explored in details in the literature at the time (only a single method was available [137] but with moderate accuracy, more works on that topic have been published since [138, 139]). Fortunately, the Hesai Pandora LiDAR that we used also contained five cameras as an all-in-one sensor, with all modalities being factory-calibrated both intrinsically and extrinsically. Therefore, our problem was simplified to a frame- to event-based calibration (while keeping in mind that this event-to-frames-to-LiDAR solution might be less precise than a direct LiDAR-to-event calibration).

However, this problem remains quite difficult: frame-based calibration often relies on a static pattern (e.g., a chessboard) being detected, but such a static pattern would not be visible for the event camera, making the calibration impossible. Also, for automotive applications, the focus of the camera is set further away than for more traditional indoor applications, meaning that a larger calibration board is required for its accurate detection. In order to solve these issues, we manufactured here a large, back-illuminated calibration board, with a ChArUco pattern on it. This calibration board is illustrated in Fig. A.4.

The main advantage of this setup is that, by making the back-light of the board blink at a high frequency, the event camera is able to detect this blinking for the white areas only, and therefore reconstruct an image of the board without any motion. As

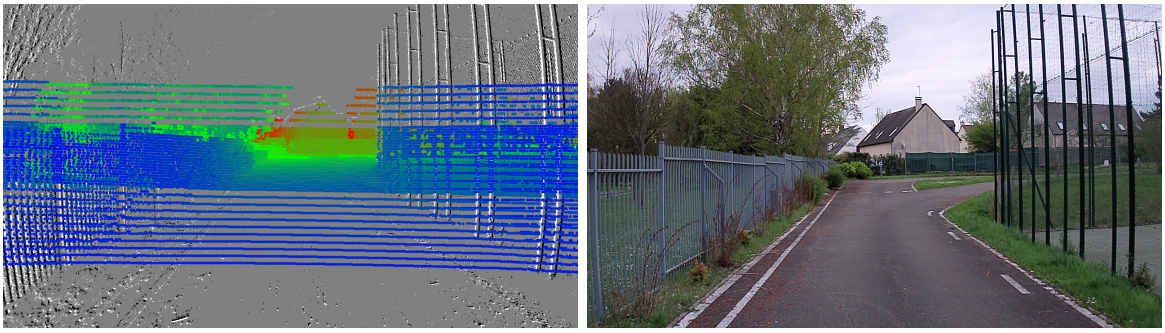


Figure A.5 – Left: superimposed event and LiDAR data following the calibration process. Right: reference RGB view of the scene, given for a better understanding.

for the frame-based camera, this blinking is invisible due to its high frequency, and therefore does not disturb the board recognition. The choice of using a ChArUco pattern was also motivated by the fact that calibration can still be conducted even when the board is not fully visible for both cameras (which happened often in the configuration illustrated in Fig. A.4) thanks to the unique tags, and by the ability of the Prophesee Gen4 camera to detect these tags perfectly thanks to its high-resolution.

Post-calibration results are shown in Fig. A.5, where the event and LiDAR data appear to be adequately superimposed.

Synchronization

For the recording of the dataset, four main sensors were intended to be used. In addition to the Prophesee camera and the Pandora LiDAR described in the previous paragraphs, two high-speed RGB cameras were also planned to be used, for being able to construct dense stereo-based ground truth depth maps, as well as for making comparisons possible between event-based and frame-based methods on the same scenes. However, for both these purposes, an accurate synchronization of the sensors was necessary.

While the Pandora LiDAR and the RGB cameras were all PTP (IEEE-1588) compatible, the Prophesee Gen4 camera transmits its data through USB, and can only be synchronized via external triggers. Therefore, as illustrated in Fig. A.6, while both the RGB and LiDAR sensors were synchronized through PTP, the left RGB camera was physically connected to the Prophesee camera, and was configured to send a signal on this link every time a frame was captured. Upon reception, the Prophesee camera would mark this signal as a special "trigger" event, which could be easily identified. Following recording, a post-processing step was applied, to associate each trigger event to its corresponding image, and therefore its corresponding timestamp (as images are PTP-timestamped). A simple interpolation was finally needed, to correct the timestamp of each individual event to its corresponding PTP timestamp.

The main issue of this method is that, in case of missing data (images or trigger events not being recorded), discrepancies in the timestamps would have been

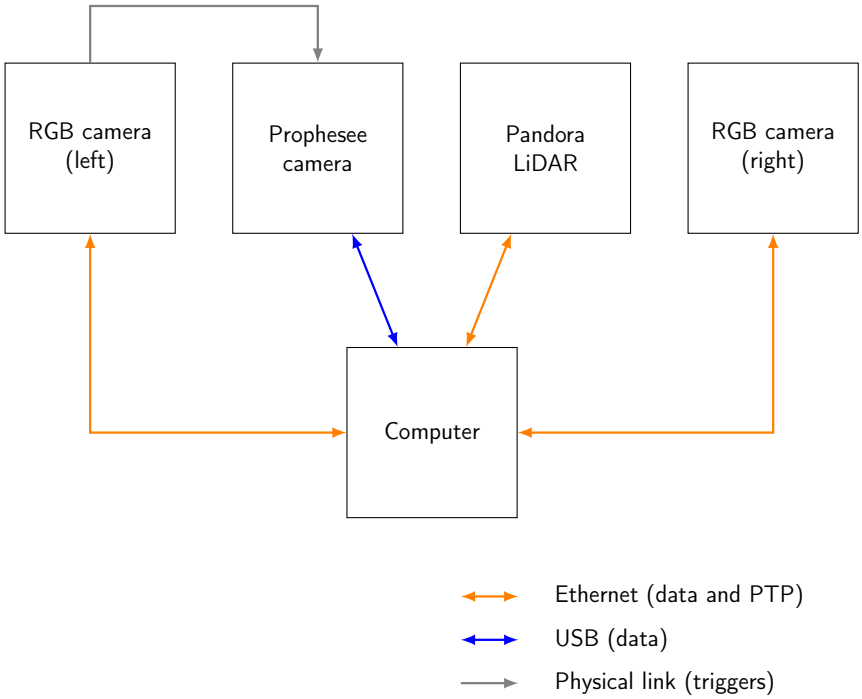


Figure A.6 – The synchronization system between the four sensors and the computer (which collects data and acts as the PTP master).

introduced, making the synchronization incorrect. During the tests we conducted, this case was often observed, with the recording containing a few more trigger events than images. However, a finer analysis showed that the additional triggers were always located at the very end of the sequence, and were in reality due to the RGB camera continuing to capture a few frames after being told to stop recording (and thus sending trigger signals), but not actually transmitting these frames.

A.2 EXTENSIONS TO THE SLED DATASET

A.2.1 *Ground Truth for Instance Segmentation*

During our tests for making the sparse attention-based depth estimation networks work in Chapter 5, we tried among other ideas to add a supervision loss on attention values (as described in Section 5.4.3). The idea here was that, if we would be able to force the network to put in relation events and LiDAR points belonging to the same object, then it would learn how to group them, and produce better depth estimations (especially for the sky, which was never recognized as being a special entity).

For that purpose, we tried adding a ground truth instance segmentation to the SLED dataset, for being able to identify independent objects, and therefore regroup events and LiDAR points correctly for the supervision loss. However, as shown in Fig. A.7, the instance segmentation maps in CARLA suffer from several issues (invisible

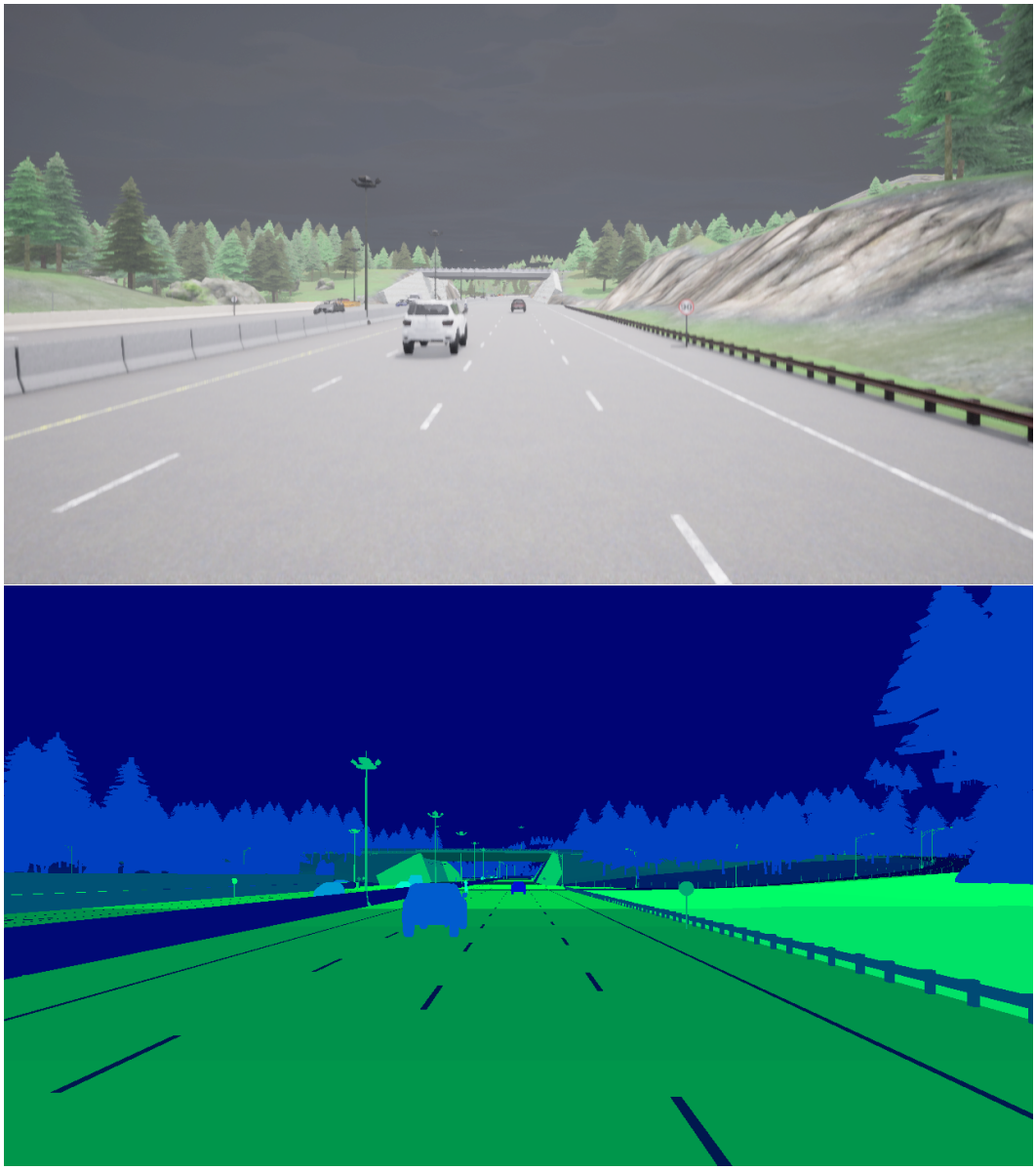


Figure A.7 – Illustration of the segmentation issue in CARLA. Top: RGB view of the scene. Bottom: corresponding segmentation map; note how the hills on the left and right extremities are missing in this segmentation map, making objects behind them appear, and how rough the segmentation is for the leaves of the trees.

objects, lack of precision for dynamic objects), and therefore they could not be used as part of this work.

As an alternative, we tried computing instance segmentation by using external tools like the Segment Anything method from Meta [177] on the RGB images of the dataset, but this approach yielded suboptimal results, due to a lack of temporal stability of the segmented areas, and due to a lack of precision at the edges of the objects.

Figure A.8 – Overview of the `Town12` map. A residential area can be seen on the left, a large body of water on the right, and buildings can be seen at the back.

A.2.2 *Additional Maps*

As noted in Chapter 4 (Table 4.1), our SLED dataset was built from data collected in the `Town01` to `Town07` and `Town10` maps of CARLA (`Town08` and `Town09` being unseen maps used as part of the evaluation of CARLA's autonomous driving leaderboard[1]). However, following the initial publication of our dataset, several maps have been added to CARLA, namely, `Town12` (in beta in version 0.9.14, and officially released in version 0.9.15), and `Town13 and \verb`Town15 (in beta in version 0.9.15). These three maps (and especially `Town12` and `Town13`) offer new, large environments, with some novel unique features, as illustrated in Fig. A.8. Being able to integrate them in SLED would allow for a more complete and a more diverse dataset, and could allow for a better training and evaluation.

However, as of the writing of this thesis, these maps still suffer from technical issues, and have not been integrated in SLED yet. One such issue, for instance, is the inability to have any pedestrian in `Town12`, as no spawn location has been provided for them (`https://github.com/carla-simulator/carla/issues/6552`). Therefore, like for the instance segmentation, we are waiting for fixes to these issues to be published, in order to upgrade the dataset in the future.

1 `https://leaderboard.carla.org`

B ADDITIONAL RESULTS FOR CHAPTER 4

In this second appendix, we give some additional quantitative and qualitative results for the Chapter 4 of this thesis.

B.1 DETAILED RESULTS ON OUR SLED DATASET

As a complement to the summarized results shown in Table 4.2 on our SLED dataset, we provide here the full results of ALED_{SL} for every recording on both maps of the testing set of SLED, `Town01` and `Town03`. These results are given in Tables B.1 and B.2 respectively.

B.2 ADDITIONAL DENSE DEPTHS RESULTS ON OUR SLED DATASET

We give in Figs. B.1 to B.4 additional visualizations of the results of ALED_{SL} on the SLED dataset. We showcase in Figs. B.1 and B.2 scenes with accurate estimations, but also some failure cases in Figs. B.3 and B.4.

B.3 ADDITIONAL DENSE DEPTHS RESULTS ON THE MVSEC DATASET

We showcase in Fig. B.5 some additional qualitative results on the MVSEC dataset for $\text{ALED}_{\text{SL}\rightarrow\text{MV}}$. We show here that, despite the low amount of LiDAR points and the sparse and/or noisy event input, we are still able to predict dense depth maps accurately.

B.4 ADDITIONAL DEPTH CHANGE MAPS RESULTS ON OUR SLED DATASET

We present in Fig. B.6 additional qualitative results for the thresholded depth change maps.

Sequence	Cutoff	Dense depths errors				Sparse depths errors				Depth change map errors	
		On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Absolute error	Correctly classified events (with a threshold of ±1m)
		Raw	AbsRel	Raw	AbsRel	NN	$ALED_{SL}$	NN	$ALED_{SL}$		
Town01_00	10m	1.13m	22.35%	1.23m	24.58%	1.00m	1.02m	1.73m	1.15m	1.35m	94.84%
	20m	3.58m	34.69%	3.82m	37.37%	1.46m	1.44m	2.62m	1.63m	3.55m	87.88%
	30m	4.47m	36.48%	4.77m	39.26%	1.66m	1.75m	2.97m	2.02m	4.30m	85.19%
	100m	5.37m	34.57%	5.85m	37.37%	2.42m	3.08m	4.54m	3.82m	5.33m	82.03%
	200m	6.09m	23.39%	6.59m	25.28%	7.14m	4.17m	9.71m	5.35m	7.08m	80.31%
Town01_01	10m	0.85m	12.58%	0.82m	12.27%	1.58m	1.27m	2.45m	1.43m	1.68m	94.23%
	20m	2.50m	23.27%	2.74m	24.94%	4.94m	5.82m	6.68m	6.50m	4.69m	89.96%
	30m	3.24m	25.24%	3.53m	27.05%	4.73m	6.67m	6.36m	7.45m	5.04m	88.35%
	100m	9.87m	34.86%	10.25m	36.56%	4.69m	6.62m	6.34m	7.36m	5.33m	86.03%
	200m	8.23m	22.50%	8.55m	23.61%	13.74m	9.88m	16.33m	11.39m	8.00m	79.91%
Town01_02	10m	0.51m	9.52%	0.53m	9.69%	0.48m	0.51m	0.72m	0.61m	0.73m	93.64%
	20m	1.00m	10.34%	1.04m	10.65%	0.47m	0.65m	0.80m	0.71m	1.48m	90.29%
	30m	1.96m	12.73%	2.03m	13.14%	0.57m	0.92m	0.98m	0.99m	2.54m	85.05%
	100m	4.23m	15.80%	4.30m	16.14%	1.05m	1.76m	1.61m	1.84m	4.13m	79.08%
	200m	5.40m	14.54%	5.47m	14.84%	1.57m	2.09m	2.17m	2.19m	6.11m	78.12%
Town01_03	10m	0.66m	10.23%	0.69m	11.09%	0.66m	0.53m	0.98m	0.64m	0.59m	94.96%
	20m	1.16m	11.44%	1.22m	12.21%	0.76m	0.62m	1.34m	0.75m	1.46m	91.42%
	30m	1.69m	12.79%	1.73m	13.44%	0.86m	0.75m	1.54m	0.89m	2.71m	87.88%
	100m	2.70m	12.15%	2.80m	12.73%	1.39m	1.47m	2.51m	1.70m	4.01m	82.84%
	200m	2.85m	9.13%	2.93m	9.55%	2.47m	1.90m	3.84m	2.16m	6.05m	81.50%
Town01_04	10m	0.95m	17.55%	1.17m	22.09%	0.94m	0.96m	1.85m	1.40m	2.22m	84.93%
	20m	1.29m	16.70%	1.56m	20.64%	1.28m	1.35m	2.37m	1.74m	2.80m	80.03%
	30m	1.54m	16.58%	1.82m	20.28%	1.44m	1.64m	2.62m	2.03m	2.98m	78.99%
	100m	2.14m	16.60%	2.45m	20.07%	2.04m	2.37m	3.45m	2.82m	3.44m	77.08%
	200m	2.05m	10.38%	2.25m	12.48%	4.01m	3.15m	5.63m	3.63m	4.94m	75.44%
Town01_05	10m	0.42m	7.40%	0.52m	9.83%	0.96m	0.78m	1.93m	1.15m	0.99m	93.63%
	20m	0.88m	8.85%	0.96m	10.61%	0.79m	1.13m	1.51m	1.33m	1.41m	88.30%
	30m	1.55m	10.45%	1.59m	11.77%	0.84m	1.44m	1.61m	1.60m	2.15m	80.29%
	100m	3.41m	12.68%	3.49m	13.74%	1.76m	2.64m	2.94m	2.92m	4.59m	71.36%
	200m	3.67m	10.88%	3.87m	11.84%	4.76m	4.02m	6.17m	4.79m	6.35m	70.24%
Town01_06	10m	0.53m	8.25%	0.63m	10.33%	0.72m	0.89m	1.44m	1.25m	1.95m	90.47%
	20m	1.02m	10.18%	1.17m	12.19%	0.84m	1.28m	1.64m	1.64m	2.94m	85.40%
	30m	1.53m	11.33%	1.69m	13.19%	0.98m	1.91m	1.91m	2.26m	3.78m	80.78%
	100m	2.96m	13.01%	3.24m	14.88%	1.66m	3.85m	3.07m	4.48m	5.64m	76.59%
	200m	3.49m	8.83%	3.79m	10.06%	12.55m	7.46m	13.97m	8.47m	9.00m	74.50%
Town01_07	10m	0.99m	16.01%	1.11m	17.89%	0.41m	0.74m	0.76m	0.90m	1.32m	86.15%
	20m	1.76m	18.57%	1.93m	20.61%	0.55m	0.93m	1.00m	1.06m	2.61m	80.28%
	30m	2.02m	18.45%	2.19m	20.31%	0.77m	1.27m	1.30m	1.40m	3.16m	78.14%
	100m	2.72m	18.43%	2.92m	20.17%	1.32m	2.12m	2.06m	2.31m	4.06m	75.73%
	200m	2.45m	13.42%	2.64m	14.67%	3.11m	2.67m	4.13m	2.94m	5.28m	75.12%
Town01_08	10m	1.06m	17.74%	1.25m	20.76%	0.85m	0.90m	1.84m	1.21m	3.66m	88.92%
	20m	1.69m	19.34%	1.90m	22.09%	1.04m	1.45m	2.13m	1.80m	5.79m	80.47%
	30m	2.17m	19.47%	2.39m	21.96%	1.23m	2.05m	2.46m	2.41m	6.51m	76.56%
	100m	3.48m	19.63%	3.77m	21.92%	2.05m	4.06m	3.80m	4.69m	8.25m	72.25%
	200m	4.50m	14.57%	4.79m	16.20%	11.21m	8.22m	13.02m	9.16m	12.81m	70.12%
Town01_09	10m	0.58m	11.49%	0.68m	14.03%	1.61m	1.26m	3.53m	1.71m	2.23m	87.48%
	20m	1.15m	12.58%	1.30m	14.83%	3.03m	2.45m	4.80m	2.92m	4.46m	80.01%
	30m	1.92m	14.38%	2.05m	16.28%	4.99m	3.81m	6.96m	4.22m	6.74m	70.78%
	100m	2.69m	14.98%	2.84m	16.72%	6.69m	5.57m	8.88m	6.03m	8.07m	67.77%
	200m	4.20m	10.91%	4.38m	12.04%	10.05m	8.53m	12.91m	9.29m	10.19m	62.17%
Town01_10	10m	3.24m	47.08%	3.29m	47.53%	0.66m	0.68m	0.98m	0.78m	1.96m	94.32%
	20m	6.01m	61.31%	6.14m	62.28%	0.73m	0.87m	1.17m	0.97m	3.94m	92.32%
	30m	6.99m	60.05%	7.15m	61.07%	0.83m	1.06m	1.37m	1.19m	4.17m	91.54%
	100m	11.98m	61.20%	12.18m	62.20%	1.25m	1.83m	2.04m	2.00m	4.58m	89.85%
	200m	10.98m	44.63%	11.14m	45.36%	3.93m	2.46m	4.82m	2.70m	6.01m	88.36%
Town01_11	10m	0.96m	13.78%	1.07m	15.43%	1.72m	1.31m	2.19m	1.40m	1.64m	93.85%
	20m	3.95m	30.55%	4.31m	33.37%	2.07m	1.79m	2.80m	1.94m	2.65m	89.14%
	30m	6.32m	37.96%	6.88m	41.40%	2.41m	2.38m	3.45m	2.60m	3.87m	85.82%
	100m	7.36m	37.58%	8.05m	41.06%	3.34m	3.45m	5.07m	4.02m	4.96m	83.09%
	200m	9.51m	28.98%	10.46m	31.71%	8.61m	5.50m	11.29m	6.93m	6.89m	81.48%
Town01_12	10m	2.22m	40.37%	2.58m	47.49%	0.50m	0.98m	0.99m	1.23m	1.62m	84.69%
	20m	2.99m	39.47%	3.30m	45.12%	1.56m	2.58m	2.58m	2.93m	2.64m	81.92%
	30m	3.48m	39.46%	3.78m	44.69%	2.01m	3.83m	3.28m	4.16m	3.32m	80.22%
	100m	3.82m	38.13%	4.11m	43.02%	2.39m	4.40m	3.81m	4.71m	3.74m	79.38%
	200m	3.48m	28.58%	3.77m	32.24%	8.63m	5.92m	10.74m	6.56m	5.18m	78.55%
Town01_13	10m	1.81m	29.12%	1.99m	32.24%	4.31m	3.96m	6.71m	4.61m	3.00m	92.10%
	20m	2.80m	30.46%	3.02m	33.27%	3.59m	3.85m	5.92m	4.41m	4.42m	82.45%
	30m	3.50m	30.39%	3.73m	32.91%	3.75m	4.49m	6.22m	4.98m	5.27m	77.92%
	100m	4.26m	28.98%	4.56m	31.38%	4.42m	5.55m	7.46m	6.19m	6.54m	73.83%
	200m	4.43m	21.75%	4.83m	23.62%	13.17m	7.64m	17.04m	9.09m	8.72m	72.78%
Town01_14	10m	1.09m	17.97%	1.28m	21.84%	0.90m	1.09m	1.91m	1.37m	1.92m	91.20%
	20m	1.42m	17.89%	1.62m	21.19%	1.33m	1.40m	2.63m	1.68m	2.65m	88.08%
	30m	1.51m	17.72%	1.71m	20.91%	1.53m	1.58m	2.92m	1.88m	2.89m	87.25%
	100m	2.27m	16.25%	2.51m	18.99%	2.58m	3.03m	4.49m	3.45m	3.79m	83.32%
	200m	2.16m	10.10%	2.34m	11.74%	6.32m	4.65m	8.85m	5.17m	5.57m	80.79%
Town01_15	10m	1.01m	20.03%	1.10m	21.58%	0.92m	1.23m	1.72m	1.49m	1.25m	91.74%
	20m	1.34m	19.14%	1.45m	20.60%	1.39m	1.62m	2.51m	1.91m	2.28m	86.31%
	30m	1.67m	19.04%	1.82m	20.55%	1.43m	1.75m	2.62m	2.05m	2.60m	84.81%
	100m	2.40m	18.03%	2.61m	19.46%	2.15m	2.71m	3.66m	3.16m	3.42m	81.41%
	200m	2.13m	11.60%	2.31m	12.53%	4.32m	3.39m	6.02m	3.93m	4.81m	80.28%
Town01_16	10m	0.51m	8.67%	0.58m	10.18%	0.96m	1.00m	1.81m	1.37m	1.50m	88.09%
	20m	0.87m	9.42%	0.99m	10.94%	0.99m	1.11m	1.82m	1.40m	2.25m	84.43%
	30m	1.14m	9.72%	1.26m	11.13%	1.13m	1.38m	2.11m	1.68m	2.71m	81.61%
	100m	2.34m	10.97%	2.51m	12.27%	1.95m	2.82m	3.26m	3.15m	5.41m	76.30%
	200m	2.19m	7.28%	2.33m	8.12%	4.89m	3.60m	6.46m	4.03m	7.75m	75.00%
Town01_17	10m	3.84m	66.60%	4.07m	71.45%	8.69m	12.36m	12.83m	14.03m	13.05m	87.20%
	20m	3.80m	57.35%	4.02m	61.41%	5.08m	9.33m	7.61m	10.43m	10.31m	83.69%
	30m	4.48m	56.21%	4.71m	60.01%	4.21m	11.31m	6.38m	12.28m	9.93m	81.53%
	100m	5.18m	54.61%	5.43m	58.23%	4.00m	12.38m	6.06m	13.31m	9.75m	80.13%
	200m	7.00m	32.38%	7.39m	34.51%	37.01m	20.46m	39.95m	22.17m	14.48m	72.83%
Town01_18	10m	0.93m	17.13%	1.15m	21.68%	1.61m	1.20m	3.35m	1.83m	3.60m	81.59%
	20m	1.10m	15.39%	1.32m	19.13%	1.78m	1.34m	3.44m	1.85m	3.93m	78.11%
	30m	1.27m	15.00%	1.49m	18.46%	1.86m	1.48m	3.52m	1.99m	4.18m	76.12%
	100m	3.09m	15.07%	3.47m	17.96%	3.02m	3.26m	5.47m	4.62m	6.45m	67.96%
	200m	3.35m	11.42%	3.79m	13.58%	5.04m	4.87m	7.86m	6.93m	9.06m	67.02%
Town01_19	10m	0.37m	6.11%	0.41m	6.94%	0.77m	0.74m	1.33m	1.02m	1.18m	90.21%
	20m	0.96m	8.47%	1.02m	9.25%	0.68m	0.90m	1.16m	1.05m	2.33m	78.79%
	30m	1.57m	10.21%	1.66m	10.98%	0.82m	1.35m	1.38m	1.51m	3.32m	75.04%
	100m	2.59m	11.56%	2.70m	12.29%	1.27m	2.31m	1.94m	2.49m	4.39m	71.75%
	200m	2.40m	8.63%	2.50m	9.18%	2.64m	2.79m	3.26m	2.98m	5.84m	71.37%

Table B.1 – Detailed results of $ALED_{SL}$ for the SLED dataset (on Town01).

Sequence	Cutoff	Dense depths errors				Sparse depths errors				Depth change map errors	
		On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Absolute error	Correctly classified events
		Raw	AbsRel	Raw	AbsRel	NN	$ALED_{SL}$	NN	$ALED_{SL}$		(with a threshold of ±1m)
Town03_00	10m	0.63m	11.61%	0.71m	13.23%	0.45m	0.25m	0.45m	0.30m	0.14m	99.93%
	20m	3.27m	26.39%	3.31m	27.51%	0.45m	0.32m	0.49m	0.40m	0.22m	99.76%
	30m	4.87m	31.32%	4.91m	32.29%	0.48m	0.36m	0.54m	0.44m	0.30m	99.62%
	100m	8.39m	35.65%	8.53m	36.67%	0.87m	0.89m	1.01m	0.99m	0.73m	98.06%
	200m	10.28m	25.56%	10.29m	26.17%	3.90m	2.51m	4.14m	2.66m	1.32m	92.99%
Town03_01	10m	0.59m	11.69%	0.63m	12.29%	0.44m	0.43m	0.45m	0.49m	0.14m	99.65%
	20m	1.60m	15.63%	1.68m	16.43%	0.60m	0.62m	0.77m	0.66m	1.76m	97.13%
	30m	3.19m	21.06%	3.34m	22.05%	0.88m	1.02m	1.22m	1.07m	2.94m	94.33%
	100m	5.29m	22.74%	5.67m	24.01%	1.87m	2.40m	2.96m	2.81m	4.71m	88.91%
	200m	7.08m	16.06%	7.41m	16.92%	5.15m	3.91m	6.91m	4.82m	7.17m	85.29%
Town03_02	10m	0.29m	4.45%	0.33m	4.99%	0.72m	0.41m	1.17m	0.59m	1.03m	96.81%
	20m	0.65m	6.10%	0.73m	6.81%	1.06m	0.65m	2.08m	0.88m	2.80m	91.20%
	30m	0.95m	6.95%	1.09m	7.86%	1.74m	1.10m	3.26m	1.43m	3.61m	87.28%
	100m	1.83m	7.45%	2.07m	8.37%	2.93m	2.72m	5.36m	3.48m	4.97m	75.93%
	200m	1.75m	5.37%	2.00m	6.05%	4.62m	3.47m	7.31m	4.49m	6.40m	75.65%
Town03_03	10m	0.31m	4.41%	0.38m	5.46%	0.63m	0.38m	0.95m	0.46m	1.12m	96.29%
	20m	1.88m	13.42%	2.08m	15.21%	0.70m	0.63m	1.36m	0.79m	3.83m	79.65%
	30m	2.62m	14.96%	2.79m	16.40%	0.82m	1.03m	1.58m	1.17m	3.83m	76.22%
	100m	3.54m	15.53%	3.71m	16.76%	1.65m	2.35m	2.75m	2.53m	4.03m	73.98%
	200m	4.35m	14.96%	4.53m	16.10%	2.48m	2.89m	3.63m	3.10m	5.16m	73.74%
Town03_04	10m	0.41m	5.77%	0.43m	6.19%	0.41m	0.45m	0.66m	0.54m	1.22m	93.05%
	20m	1.01m	9.11%	1.06m	9.67%	0.87m	0.75m	1.43m	0.83m	4.22m	83.42%
	30m	1.56m	10.42%	1.62m	10.97%	1.12m	1.27m	1.88m	1.35m	5.44m	79.69%
	100m	2.95m	12.15%	3.05m	12.70%	1.98m	2.67m	3.14m	2.78m	6.74m	76.06%
	200m	4.18m	10.13%	4.28m	10.56%	4.91m	4.23m	6.30m	4.44m	10.73m	74.61%
Town03_05	10m	7.05m	87.16%	7.20m	88.34%	0.34m	0.55m	0.43m	0.65m	1.82m	93.59%
	20m	10.10m	99.82%	10.32m	101.69%	0.35m	0.83m	0.67m	1.01m	3.30m	88.19%
	30m	10.02m	93.36%	10.19m	94.98%	0.72m	1.48m	1.40m	1.66m	3.87m	84.79%
	100m	11.15m	83.69%	11.36m	85.20%	1.73m	2.87m	3.06m	3.32m	4.94m	76.94%
	200m	12.16m	74.37%	12.37m	75.77%	2.16m	3.41m	3.65m	4.00m	5.99m	74.93%
Town03_06	10m	0.22m	3.59%	0.22m	3.62%	0.57m	0.30m	0.70m	0.39m	0.34m	97.94%
	20m	0.53m	4.90%	0.57m	5.20%	0.72m	0.61m	1.38m	0.79m	1.62m	89.39%
	30m	1.49m	8.00%	1.56m	8.36%	0.87m	1.04m	1.77m	1.22m	2.45m	83.98%
	100m	3.34m	10.69%	3.41m	10.97%	1.95m	3.39m	3.40m	3.57m	4.64m	76.50%
	200m	3.07m	8.51%	3.15m	8.74%	5.86m	4.67m	7.66m	4.92m	6.17m	75.07%
Town03_07	10m	0.30m	4.61%	0.33m	5.10%	0.73m	0.47m	1.37m	0.56m	0.97m	94.43%
	20m	0.69m	6.37%	0.80m	7.29%	1.27m	0.73m	2.69m	0.95m	2.47m	86.80%
	30m	1.42m	8.72%	1.57m	9.77%	1.94m	1.18m	3.87m	1.51m	3.84m	77.83%
	100m	3.14m	10.72%	3.49m	11.91%	3.57m	3.51m	6.50m	4.56m	5.40m	68.94%
	200m	3.07m	8.42%	3.46m	9.39%	5.38m	4.37m	8.57m	5.60m	6.86m	68.81%
Town03_08	10m	0.85m	12.67%	1.15m	16.72%	0.06m	0.57m	0.07m	0.55m	1.95m	74.38%
	20m	1.19m	13.53%	1.50m	17.21%	0.17m	0.66m	0.33m	0.67m	2.11m	74.90%
	30m	1.32m	13.41%	1.61m	16.87%	0.52m	0.89m	0.85m	0.95m	2.38m	75.08%
	100m	1.86m	13.22%	2.16m	16.35%	1.57m	2.15m	2.32m	2.31m	3.05m	73.89%
	200m	2.07m	9.82%	2.32m	12.05%	6.36m	3.56m	7.22m	3.73m	3.89m	72.38%
Town03_09	10m	3.27m	60.78%	3.46m	64.64%	0.17m	1.30m	0.56m	1.54m	1.82m	86.29%
	20m	3.07m	45.57%	3.20m	48.14%	0.16m	1.22m	0.56m	1.38m	1.57m	83.64%
	30m	3.18m	42.72%	3.29m	45.01%	0.27m	1.52m	0.71m	1.68m	1.63m	83.39%
	100m	3.50m	40.82%	3.62m	42.98%	0.38m	1.93m	0.85m	2.09m	2.00m	82.62%
	200m	3.23m	35.29%	3.32m	37.20%	0.53m	1.96m	0.93m	2.12m	2.44m	82.49%
Town03_10	10m	2.50m	30.96%	2.60m	32.49%	0.65m	0.66m	0.85m	0.77m	1.16m	91.32%
	20m	2.71m	27.29%	2.80m	28.47%	0.49m	0.68m	0.73m	0.77m	1.41m	91.30%
	30m	3.63m	29.05%	3.73m	30.17%	0.60m	1.00m	1.02m	1.11m	2.02m	89.91%
	100m	7.85m	33.13%	7.99m	34.18%	1.44m	2.67m	2.51m	2.92m	3.87m	84.03%
	200m	7.38m	26.17%	7.52m	27.03%	4.53m	3.43m	5.85m	3.96m	5.33m	81.81%
Town03_11	10m	0.43m	6.06%	0.47m	6.62%	0.67m	0.43m	0.89m	0.45m	0.99m	98.51%
	20m	2.24m	15.59%	2.38m	16.64%	0.98m	0.66m	1.51m	0.68m	3.07m	89.26%
	30m	4.19m	22.65%	4.38m	23.84%	1.30m	0.94m	1.96m	1.00m	3.90m	81.62%
	100m	4.81m	21.35%	5.04m	22.46%	2.43m	2.64m	3.63m	2.82m	4.49m	76.78%
	200m	3.86m	15.19%	4.03m	15.98%	5.02m	3.41m	6.61m	3.75m	5.43m	75.71%
Town03_12	10m	0.29m	4.44%	0.32m	4.92%	0.61m	0.26m	0.67m	0.30m	0.65m	98.45%
	20m	0.60m	5.64%	0.69m	6.40%	0.65m	0.53m	1.00m	0.65m	1.67m	89.75%
	30m	1.00m	6.72%	1.07m	7.40%	1.14m	0.85m	1.91m	1.03m	2.50m	85.92%
	100m	2.02m	7.93%	2.14m	8.58%	2.75m	2.55m	4.05m	2.81m	3.42m	79.38%
	200m	1.97m	5.65%	2.10m	6.12%	4.65m	3.40m	6.22m	3.78m	4.48m	78.09%
Town03_13	10m	0.30m	4.86%	0.36m	5.71%	0.56m	0.35m	0.82m	0.51m	0.86m	95.75%
	20m	1.68m	12.08%	1.87m	13.72%	0.61m	0.56m	1.21m	0.77m	3.00m	80.59%
	30m	2.54m	14.42%	2.71m	15.75%	0.75m	0.95m	1.49m	1.13m	3.67m	76.82%
	100m	3.43m	15.21%	3.58m	16.36%	1.58m	2.44m	2.50m	2.61m	4.41m	74.22%
	200m	4.02m	14.06%	4.14m	15.08%	3.05m	2.94m	4.05m	3.13m	6.02m	73.78%
Town03_14	10m	0.65m	12.46%	0.64m	11.99%	0.80m	0.57m	1.06m	0.62m	0.87m	98.33%
	20m	1.76m	17.26%	1.75m	16.98%	1.24m	1.16m	2.00m	1.27m	4.09m	88.79%
	30m	2.25m	17.83%	2.27m	17.70%	1.61m	1.73m	2.62m	1.94m	5.83m	82.77%
	100m	3.89m	18.38%	4.00m	18.47%	2.85m	3.90m	4.68m	4.33m	8.77m	76.20%
	200m	5.40m	14.13%	5.62m	14.26%	8.66m	6.74m	10.78m	7.58m	14.13m	74.72%
Town03_15	10m	0.79m	18.94%	0.80m	18.87%	0.52m	0.52m	0.86m	0.60m	0.74m	97.04%
	20m	1.94m	22.27%	2.13m	23.34%	0.92m	0.77m	2.06m	0.93m	2.90m	90.03%
	30m	2.32m	22.17%	2.55m	23.36%	1.26m	1.04m	2.69m	1.28m	3.62m	85.86%
	100m	3.83m	19.45%	4.19m	20.63%	2.37m	2.90m	4.88m	3.66m	6.04m	75.59%
	200m	3.77m	15.36%	4.17m	16.35%	5.29m	3.77m	8.18m	4.96m	8.06m	74.93%
Town03_16	10m	0.42m	6.33%	0.50m	7.64%	0.86m	0.68m	1.46m	0.92m	1.81m	91.42%
	20m	1.02m	9.35%	1.13m	10.61%	1.06m	0.92m	1.90m	1.12m	4.97m	80.90%
	30m	1.45m	10.35%	1.54m	11.43%	1.50m	1.20m	2.43m	1.39m	6.10m	78.03%
	100m	2.41m	10.50%	2.51m	11.34%	2.28m	2.73m	3.49m	2.93m	6.62m	75.20%
	200m	3.02m	8.29%	3.11m	8.91%	4.76m	3.72m	6.01m	3.95m	9.95m	74.01%
Town03_17	10m	0.34m	6.59%	0.37m	7.17%	0.45m	0.46m	0.72m	0.48m	0.33m	98.56%
	20m	0.51m	6.38%	0.55m	6.91%	0.96m	0.92m	1.95m	1.10m	2.65m	89.09%
	30m	0.97m	7.36%	1.03m	7.89%	0.98m	1.16m	2.13m	1.36m	3.13m	83.00%
	100m	1.65m	7.86%	1.72m	8.32%	1.61m	2.44m	3.13m	2.67m	3.48m	79.44%
	200m	2.14m	6.02%	2.21m	6.34%	6.77m	3.89m	8.50m	4.20m	4.68m	77.12%
Town03_18	10m	0.38m	5.52%	0.48m	6.80%	0.37m	0.42m	0.45m	0.48m	1.15m	92.67%
	20m	0.76m	7.12%	0.86m	8.19%	0.49m	0.86m	0.84m	0.94m	1.88m	85.87%
	30m	1.02m	7.57%	1.13m	8.56%	0.86m	1.22m	1.47m	1.35m	2.12m	82.31%
	100m	4.09m	12.24%	4.16m	12.93%	1.68m	2.39m	3.10m	2.75m	4.08m	73.94%
	200m	4.49m	10.76%	4.69m	11.40%	4.01m	3.34m	5.60m	4.19m	5.34m	73.31%
Town03_19	10m	13.25m	180.69%	13.33m	181.88%	0.28m	1.24m	0.49m	1.44m	2.87m	89.69%
	20m	12.80m	150.50%	12.77m	150.61%	0.35m	1.26m	0.70m	1.39m	2.82m	85.70%
	30m	11.72m	133.06%	11.68m	133.11%	0.51m	1.73m	0.89m	1.83m	2.68m	84.81%
	100m	11.19m	120.74%	11.17m	120.93%	0.73m	2.32m	1.36m	2.45m	2.74m	83.57%
	200m	9.94m	105.93%	9.94m	106.27%	0.82m	2.34m	1.40m	2.47m	3.23m	83.49%

Table B.2 – Detailed results of $ALED_{SL}$ for the SLED dataset (on Town03).

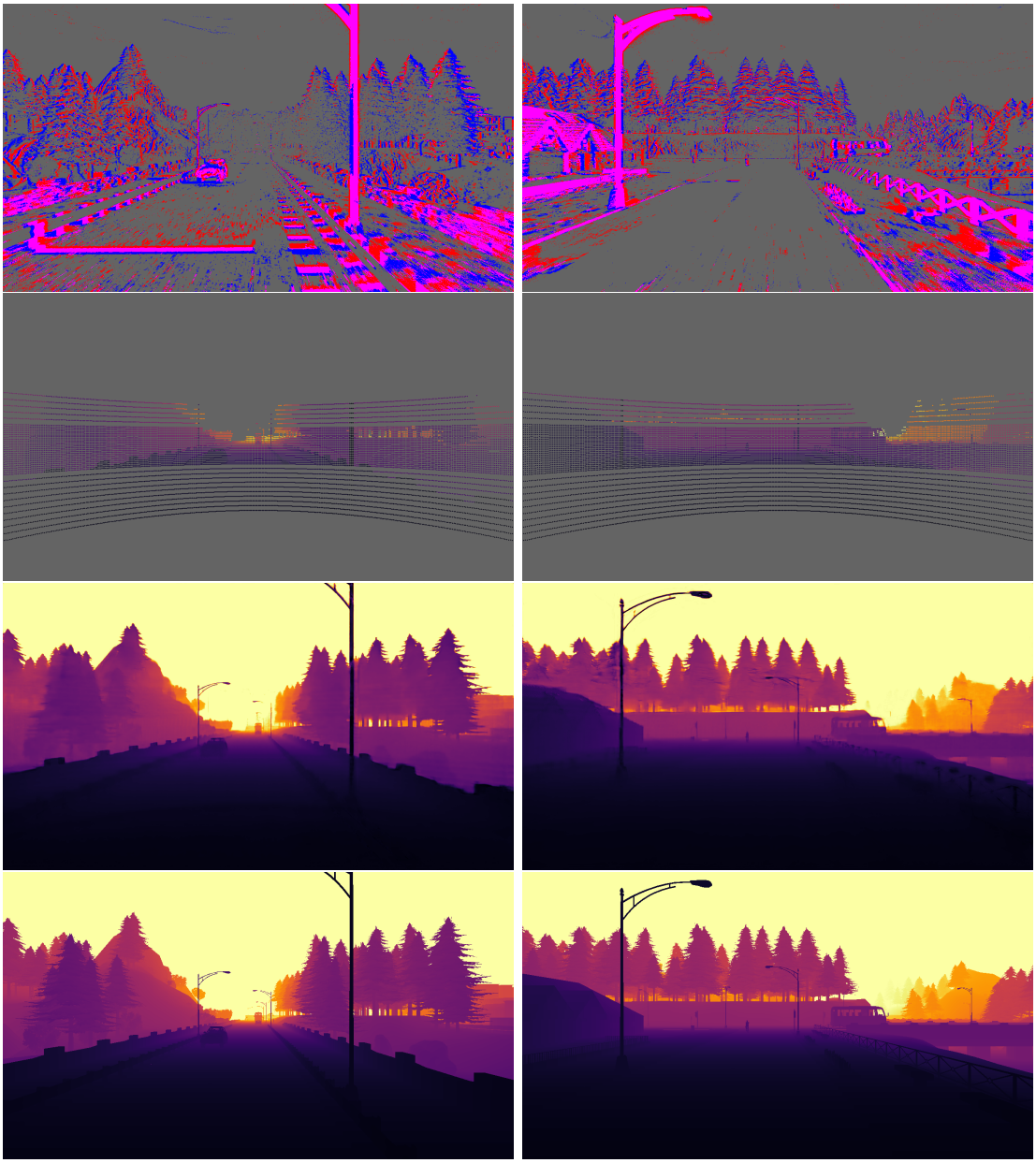


Figure B.1 – Additional dense depths results on the SLED dataset, on sequences `Town01_03` and `Town01_05`. From top to bottom: events, LiDAR, our prediction ($ALED_{SL}$), ground truth.

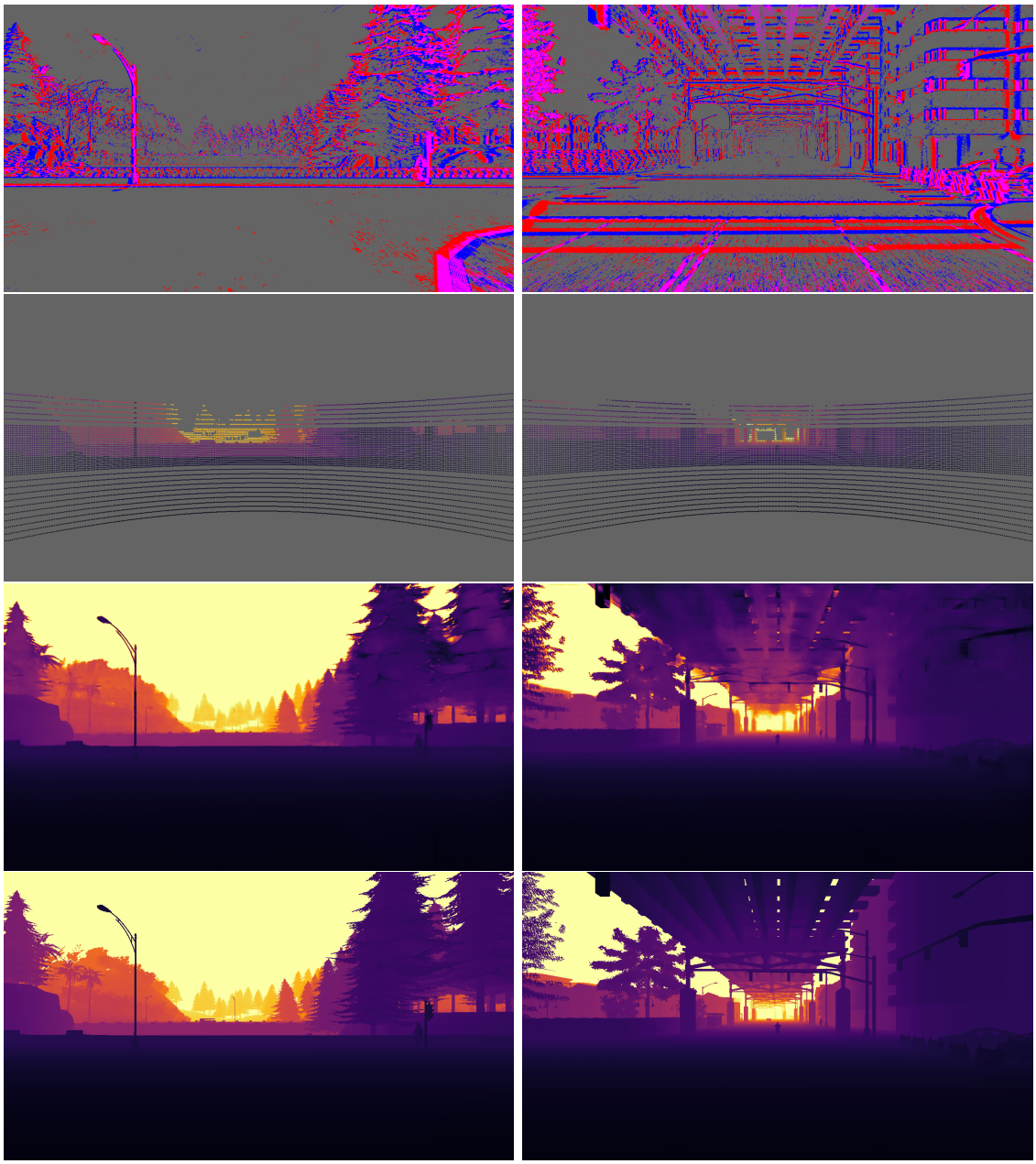


Figure B.2 – Additional dense depths results on the SLED dataset, on sequences `Town01_18` and `Town03_13`. From top to bottom: events, LiDAR, our prediction ($ALED_{SL}$), ground truth.

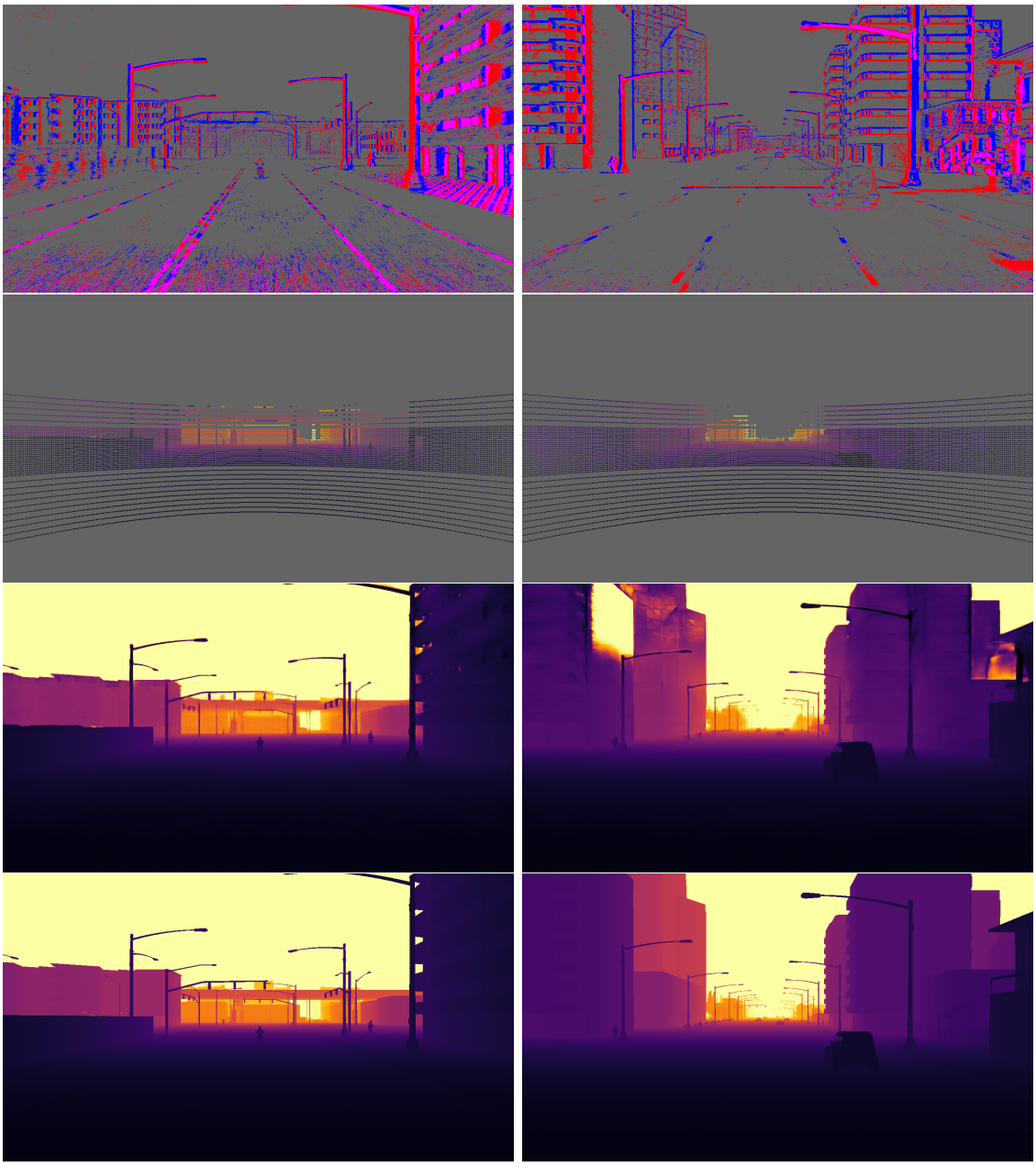


Figure B.3 – Additional dense depths results on the SLED dataset, on sequences `Town03_02` and `Town03_06`. From top to bottom: events, LiDAR, our prediction (ALED_{SL}), ground truth. Shown here on the right is a failure case, where the side of the building got mistaken with the sky.

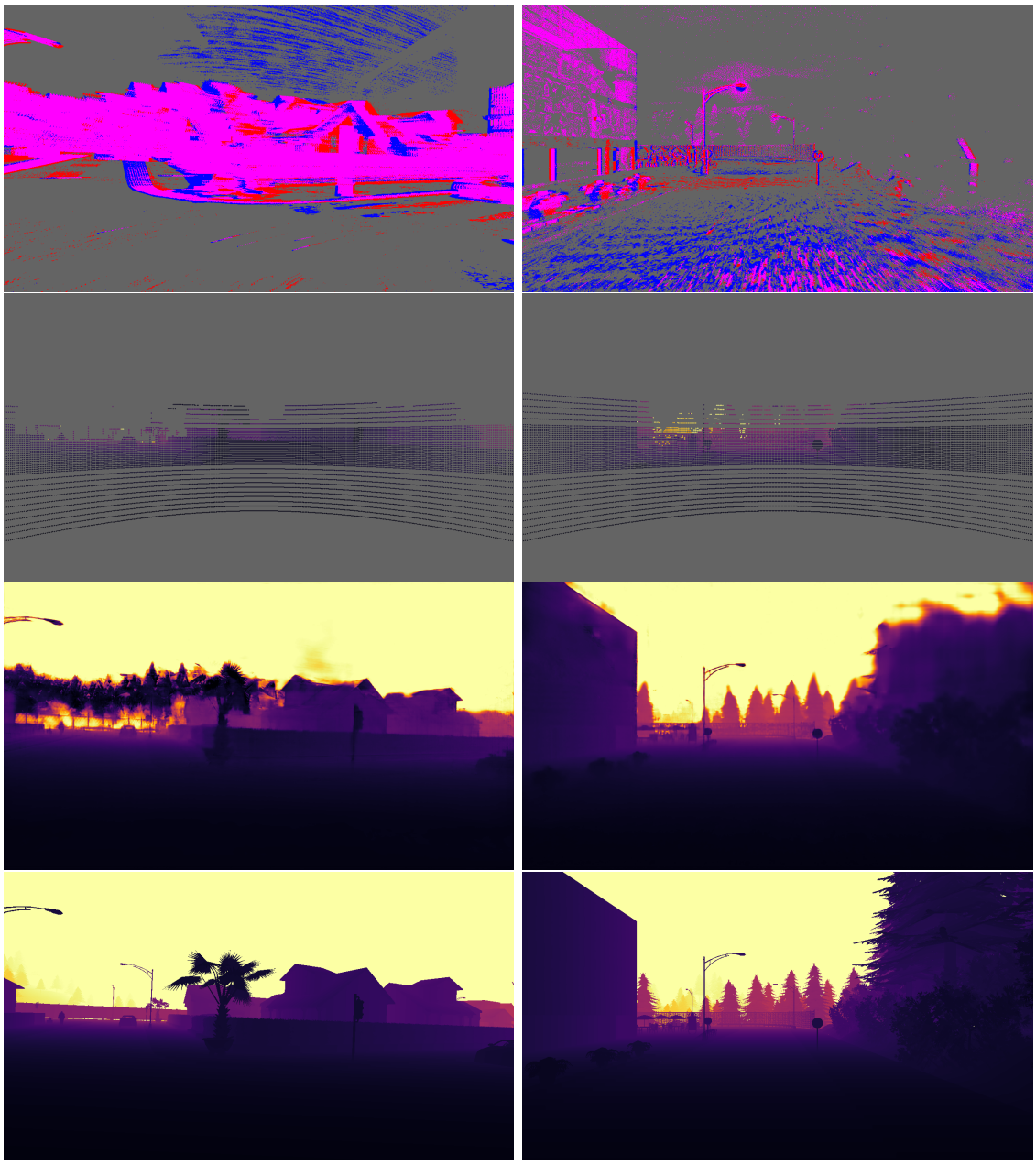


Figure B.4 – Additional dense depths results on the SLED dataset, on sequences `Town01_08` and `Town01_11`. From top to bottom: events, LiDAR, our prediction (ALED_{SL}), ground truth. Illustrated here are two failure cases. Left: due to a sharp turn at high speed, accumulated events become too blurry, resulting in an incorrect prediction for distant objects. Right: night scene, where the trees on the right side are too dark to be seen even by the event camera, resulting in a partially blurry prediction.

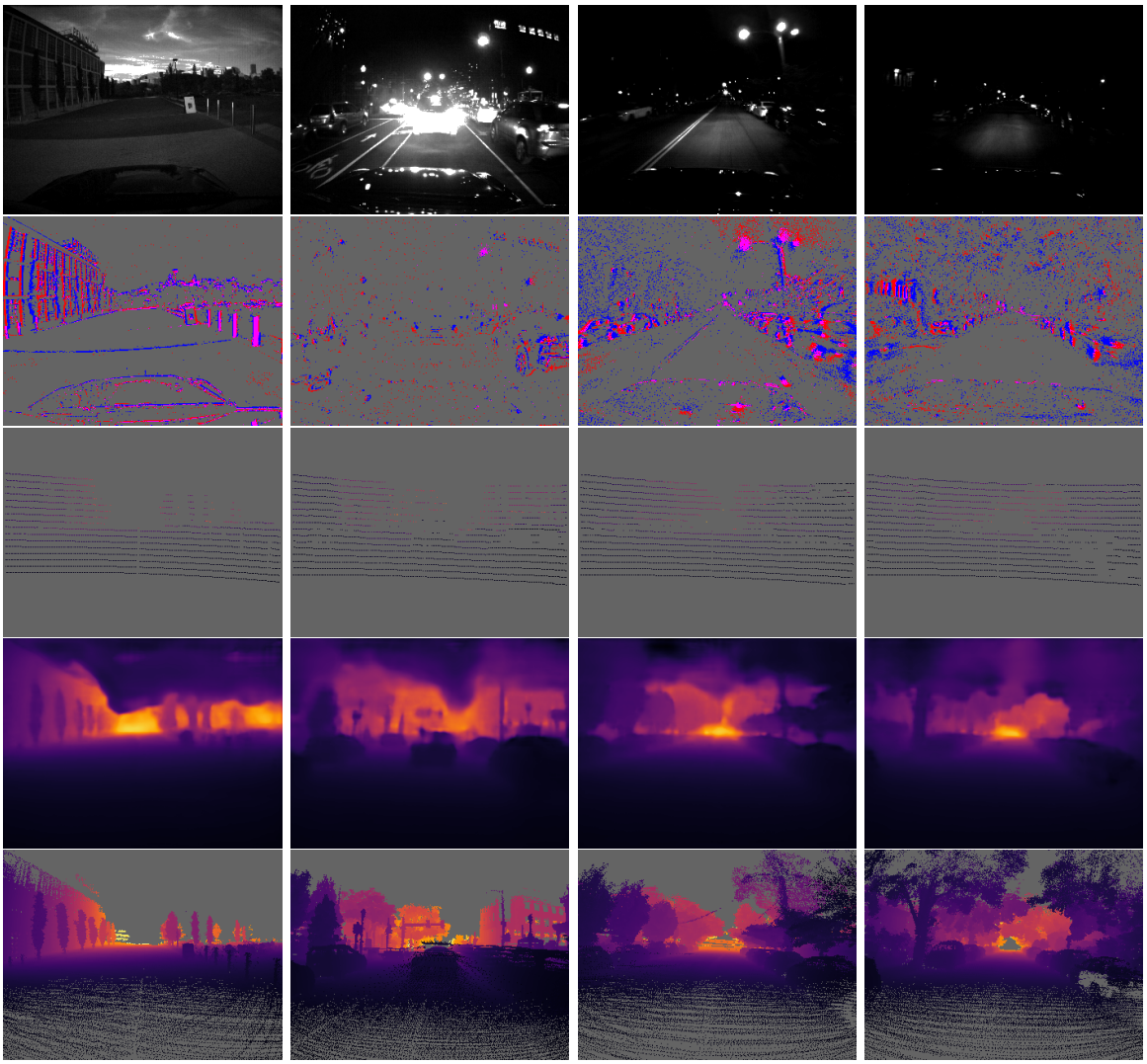


Figure B.5 – Additional dense depths results on the MVSEC dataset. From left to right: `outdoor_day_1`, `outdoor_night_1`, `outdoor_night_2`, `outdoor_night_3`. From top to bottom: grayscale reference, events, LiDAR, our prediction ($\text{ALED}_{\text{SL}\to\text{MV}}$), ground truth.

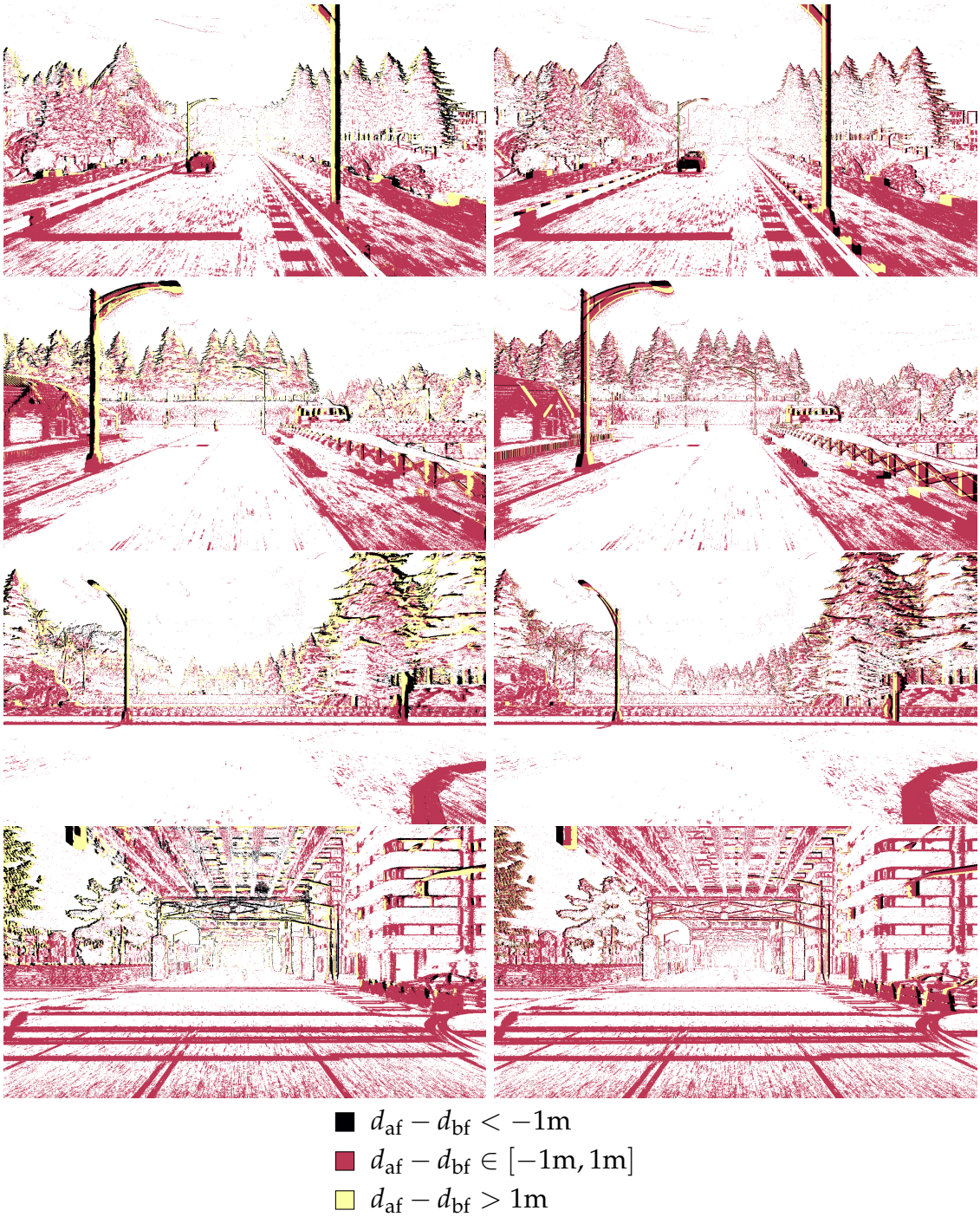


Figure B.6 – Additional thresholded depth change map results on SLED, using the events as a mask. Left: prediction ($ALED_{SL}$). Right: ground truth.

C ADDITIONAL RESULTS FOR CHAPTER 5

We give here some additional quantitative and qualitative results for Chapter 5.

C.1 DETAILED RESULTS ON OUR SLED DATASET

As a complement to the summarized results shown in Table 5.2 on our SLED dataset, we provide here the full results of DELTA_{SL} for every recording on both maps of the testing set of SLED, `Town01` and `Town03`, in Tables C.1 and C.2 respectively.

In addition to the results of Chapter 5, some additional elements can be noted, which are not visible when only looking at the average results shown in Table 5.2:

- as observed on the MVSEC dataset, DELTA_{SL} appears to struggle slightly more on the night sequences (sequences finishing by `_00`, `_01`, `_02`, `_10`, `_11`, and `_12`) than ALED, especially for long cutoff distances; once again, this behavior is probably due to the higher noise profile, but also because of the limitations of the event camera simulation in the CARLA simulator, as shown in Fig. C.4;
- results at the max cutoff range are especially worse when compared to ALED_{SL} on sequences `Town01_13`, `Town01_17`, and `Town03_12`:
  - for `Town01_13` and `Town01_17`, this is explained by the presence of a wire fence alongside the road, which creates difficult patterns in the event stream, in the LiDAR data, and in the ground truth depth maps;
  - as for `Town03_12`, this is due to a bad initial propagation of the depths, leading to some incorrect depth values in the sky;
- on the contrary, DELTA_{SL} has much better results than ALED_{SL} on sequences `Town03_05`, `Town03_09`, and `Town03_19`, as they all showcase a tunnel section, which ALED_{SL} clearly appears to struggle with.

C.2 ADDITIONAL VISUAL RESULTS ON THE SLED DATASET

Additional qualitative results on SLED are given in Figs. C.1 to C.4: scenes with accurate estimations in Figs. C.1 and C.2, and some failure cases in Figs. C.3 and C.4.

C.3 ADDITIONAL VISUAL RESULTS ON THE MVSEC DATASET

Additional qualitative results on the MVSEC dataset are given in Fig. C.5.

C.4 ADDITIONAL VISUAL RESULTS ON THE M3ED DATASET

Additional qualitative results on the M3ED dataset are given in Figs. C.6 and C.7.

Sequence	Cutoff	Dense depths errors				Sparse depths errors				Depth change map errors	
		On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Absolute error	Correctly classified events
		Raw	AbsRel	Raw	AbsRel	NN	$DELTA_{SL}$	NN	$DELTA_{SL}$		(with a threshold of ±1m)
Town01_00	10m	0.54m	9.03%	0.55m	9.76%	1.00m	0.68m	1.73m	0.71m	1.13m	95.55%
	20m	3.06m	23.99%	3.10m	24.61%	1.46m	1.24m	1.63m	1.36m	2.84m	89.40%
	30m	4.59m	29.30%	4.69m	30.13%	1.66m	1.58m	2.97m	1.77m	3.71m	86.55%
	100m	5.50m	28.54%	5.70m	29.48%	2.42m	2.67m	4.54m	3.07m	4.65m	83.92%
	200m	6.71m	19.83%	6.86m	20.44%	7.14m	3.90m	9.71m	4.52m	6.64m	82.09%
Town01_01	10m	0.48m	6.69%	0.51m	6.92%	1.58m	1.01m	2.45m	1.20m	1.76m	94.95%
	20m	2.36m	20.19%	2.50m	21.31%	4.94m	4.80m	6.68m	5.16m	4.08m	90.74%
	30m	2.97m	21.83%	3.14m	23.00%	4.73m	5.49m	6.36m	5.87m	4.52m	89.03%
	100m	13.80m	42.50%	13.95m	43.46%	4.69m	5.54m	6.34m	5.86m	4.90m	86.95%
	200m	11.35m	27.57%	11.49m	28.20%	13.74m	10.26m	16.33m	10.86m	7.47m	81.96%
Town01_02	10m	0.15m	2.54%	0.16m	2.62%	0.48m	0.29m	0.72m	0.32m	0.49m	94.33%
	20m	0.43m	3.79%	0.44m	3.92%	0.47m	0.39m	0.80m	0.42m	1.27m	91.46%
	30m	1.28m	6.71%	1.31m	6.86%	0.57m	0.50m	0.98m	0.53m	2.38m	86.36%
	100m	3.43m	10.59%	3.44m	10.69%	1.05m	1.13m	1.61m	1.13m	3.97m	80.79%
	200m	5.65m	10.52%	5.64m	10.60%	1.57m	1.58m	2.17m	1.60m	6.20m	79.64%
Town01_03	10m	0.26m	4.11%	0.28m	4.36%	0.66m	0.36m	0.98m	0.40m	0.55m	95.53%
	20m	0.68m	6.01%	0.70m	6.17%	0.76m	0.55m	1.34m	0.59m	1.56m	91.94%
	30m	1.32m	8.20%	1.33m	8.33%	0.86m	0.69m	1.54m	0.75m	3.03m	88.29%
	100m	2.54m	9.04%	2.59m	9.22%	1.39m	1.55m	2.51m	1.69m	4.47m	82.79%
	200m	3.14m	7.15%	3.15m	7.26%	2.47m	2.17m	3.84m	2.32m	6.98m	81.20%
Town01_04	10m	0.64m	11.31%	0.67m	11.74%	0.94m	0.80m	1.85m	0.90m	2.42m	85.27%
	20m	1.02m	11.92%	1.10m	12.63%	1.28m	1.29m	2.37m	1.42m	2.92m	82.20%
	30m	1.29m	12.25%	1.38m	12.96%	1.44m	1.62m	2.62m	1.74m	3.15m	81.17%
	100m	2.01m	12.77%	2.11m	13.46%	2.04m	2.56m	3.45m	2.71m	3.62m	78.94%
	200m	2.18m	8.20%	2.22m	8.61%	4.01m	3.49m	5.63m	3.64m	5.47m	77.05%
Town01_05	10m	0.29m	5.95%	0.34m	6.95%	0.96m	0.66m	1.93m	0.80m	1.17m	93.52%
	20m	0.60m	6.57%	0.65m	7.36%	0.79m	0.67m	1.51m	0.77m	1.42m	88.38%
	30m	1.23m	8.21%	1.29m	8.93%	0.84m	0.79m	1.61m	0.90m	2.11m	80.55%
	100m	3.13m	11.00%	3.16m	11.49%	1.76m	1.99m	2.94m	2.16m	4.74m	70.85%
	200m	4.04m	9.82%	4.11m	10.25%	4.76m	4.05m	6.17m	4.32m	6.90m	69.87%
Town01_06	10m	0.40m	6.24%	0.41m	6.31%	0.72m	0.69m	1.44m	0.74m	2.13m	92.13%
	20m	0.76m	7.67%	0.80m	7.98%	0.84m	1.17m	1.64m	1.28m	3.16m	87.43%
	30m	1.16m	8.61%	1.23m	9.00%	0.98m	1.62m	1.91m	1.77m	4.07m	83.30%
	100m	2.17m	9.68%	2.29m	10.14%	1.66m	3.07m	3.07m	3.37m	5.93m	78.83%
	200m	5.19m	7.88%	5.23m	8.15%	12.55m	8.28m	13.97m	8.50m	10.01m	75.31%
Town01_07	10m	0.46m	7.18%	0.47m	7.36%	0.41m	0.51m	0.76m	0.52m	1.05m	90.76%
	20m	0.91m	8.95%	0.95m	9.33%	0.55m	0.68m	1.00m	0.71m	2.02m	87.19%
	30m	1.20m	9.57%	1.24m	9.94%	0.77m	0.93m	1.30m	0.97m	2.73m	84.29%
	100m	1.88m	10.25%	1.96m	10.65%	1.32m	1.75m	2.06m	1.85m	3.78m	81.11%
	200m	2.21m	7.78%	2.24m	8.04%	3.11m	2.53m	4.13m	2.68m	5.48m	79.91%
Town01_08	10m	0.55m	8.38%	0.58m	8.79%	0.85m	0.74m	1.84m	0.82m	3.78m	90.76%
	20m	1.16m	11.45%	1.24m	12.12%	1.04m	1.44m	2.13m	1.62m	6.28m	83.06%
	30m	1.68m	12.55%	1.76m	13.20%	1.23m	1.90m	2.46m	2.11m	7.09m	79.42%
	100m	2.89m	13.58%	3.01m	14.23%	2.05m	3.58m	3.80m	3.95m	8.82m	74.39%
	200m	4.93m	10.84%	4.99m	11.28%	11.21m	8.08m	13.02m	8.39m	14.22m	71.74%
Town01_09	10m	0.25m	4.78%	0.27m	5.34%	1.61m	1.07m	3.53m	1.30m	2.14m	89.06%
	20m	0.94m	7.98%	0.97m	8.48%	3.03m	3.10m	4.80m	3.28m	4.36m	82.51%
	30m	1.99m	11.56%	2.04m	12.06%	4.99m	4.99m	6.96m	5.22m	6.64m	74.69%
	100m	2.93m	12.98%	3.01m	13.50%	6.69m	6.76m	8.88m	7.08m	8.05m	70.85%
	200m	4.46m	9.68%	4.52m	9.99%	10.05m	9.30m	12.91m	9.68m	10.42m	64.91%
Town01_10	10m	1.57m	20.95%	1.59m	21.35%	0.66m	0.33m	0.98m	0.35m	1.46m	94.80%
	20m	3.99m	35.26%	4.05m	35.99%	0.73m	0.50m	1.17m	0.54m	2.96m	93.16%
	30m	4.78m	35.72%	4.84m	36.36%	0.83m	0.65m	1.37m	0.72m	3.22m	92.51%
	100m	9.33m	40.21%	9.39m	40.76%	1.25m	1.31m	2.04m	1.39m	3.68m	91.09%
	200m	12.79m	31.61%	12.81m	32.00%	3.93m	2.10m	4.82m	2.19m	5.31m	89.58%
Town01_11	10m	0.73m	10.50%	0.77m	10.95%	1.72m	1.28m	2.19m	1.26m	1.49m	94.76%
	20m	3.86m	28.16%	3.95m	28.97%	2.07m	1.67m	2.80m	1.70m	2.34m	91.43%
	30m	6.90m	38.64%	7.00m	39.39%	2.41m	2.18m	3.45m	2.19m	3.25m	88.42%
	100m	8.20m	39.18%	8.37m	40.01%	3.34m	3.18m	5.07m	3.32m	4.21m	85.43%
	200m	10.84m	30.48%	11.13m	31.18%	8.61m	5.07m	11.29m	5.42m	6.45m	83.44%
Town01_12	10m	0.50m	9.26%	0.52m	9.67%	0.50m	0.54m	0.99m	0.64m	0.70m	89.67%
	20m	1.18m	12.78%	1.24m	13.38%	1.56m	2.28m	2.58m	2.52m	2.13m	87.66%
	30m	1.67m	14.21%	1.74m	14.85%	2.01m	3.45m	3.28m	3.75m	2.95m	85.62%
	100m	2.08m	14.41%	2.16m	15.03%	2.39m	4.07m	3.81m	4.37m	3.45m	84.51%
	200m	2.92m	11.44%	2.95m	11.88%	8.63m	6.78m	10.74m	7.03m	5.43m	83.06%
Town01_13	10m	1.09m	15.73%	1.18m	17.20%	4.31m	3.57m	6.71m	4.10m	3.70m	93.63%
	20m	1.53m	16.05%	1.60m	17.13%	3.59m	3.73m	5.92m	4.06m	4.31m	88.49%
	30m	1.89m	16.00%	1.98m	17.01%	3.75m	4.16m	6.22m	4.45m	4.83m	83.86%
	100m	2.51m	15.61%	2.63m	16.58%	4.42m	5.10m	7.46m	5.40m	5.80m	79.89%
	200m	7.53m	14.33%	7.55m	15.01%	13.17m	9.82m	17.04m	10.14m	9.20m	77.60%
Town01_14	10m	0.43m	7.87%	0.49m	9.24%	0.90m	0.72m	1.91m	0.92m	1.77m	93.80%
	20m	0.84m	9.55%	0.89m	10.61%	1.33m	1.21m	2.63m	1.36m	2.59m	90.81%
	30m	1.00m	9.95%	1.05m	10.95%	1.53m	1.53m	2.92m	1.67m	2.87m	89.66%
	100m	1.99m	10.07%	2.06m	10.93%	2.58m	3.11m	4.49m	3.30m	3.84m	85.43%
	200m	2.49m	6.72%	2.52m	7.22%	6.32m	5.41m	8.85m	5.58m	6.08m	82.34%
Town01_15	10m	0.38m	6.61%	0.43m	7.48%	0.92m	0.79m	1.72m	0.99m	1.40m	92.09%
	20m	0.79m	8.58%	0.87m	9.53%	1.39m	1.39m	2.51m	1.60m	2.45m	88.16%
	30m	1.16m	9.54%	1.25m	10.47%	1.43m	1.57m	2.62m	1.77m	2.73m	86.85%
	100m	2.04m	10.09%	2.15m	10.95%	2.15m	2.65m	3.66m	2.86m	3.55m	83.22%
	200m	2.30m	6.86%	2.33m	7.38%	4.32m	3.60m	6.02m	3.82m	5.29m	81.84%
Town01_16	10m	0.38m	6.64%	0.39m	6.65%	0.96m	0.88m	1.81m	0.98m	1.66m	89.29%
	20m	0.72m	7.65%	0.76m	7.89%	0.99m	1.21m	1.82m	1.32m	2.21m	86.82%
	30m	1.00m	8.16%	1.03m	8.38%	1.13m	1.49m	2.11m	1.59m	2.58m	84.92%
	100m	2.10m	9.40%	2.16m	9.65%	1.95m	2.85m	3.26m	2.96m	5.59m	78.89%
	200m	2.71m	6.64%	2.72m	6.78%	4.89m	4.19m	6.46m	4.30m	8.54m	76.81%
Town01_17	10m	2.48m	36.50%	2.48m	36.58%	8.69m	9.95m	12.83m	9.85m	17.60m	87.55%
	20m	2.60m	33.19%	2.58m	33.12%	5.08m	7.66m	7.61m	7.52m	14.62m	86.83%
	30m	2.74m	31.66%	2.72m	31.55%	4.21m	7.40m	6.38m	7.20m	13.20m	85.30%
	100m	3.02m	30.46%	3.02m	30.42%	4.00m	7.50m	6.06m	7.35m	12.60m	83.51%
	200m	10.31m	21.37%	10.12m	21.26%	37.01m	29.56m	39.95m	28.39m	19.78m	75.36%
Town01_18	10m	0.74m	13.55%	0.74m	13.30%	1.61m	1.12m	3.35m	1.31m	4.24m	84.36%
	20m	0.97m	12.89%	1.00m	12.92%	1.78m	1.46m	3.44m	1.64m	4.43m	82.14%
	30m	1.20m	12.88%	1.24m	12.98%	1.86m	1.67m	3.52m	1.85m	4.64m	79.90%
	100m	2.91m	13.54%	3.02m	13.75%	3.02m	3.12m	5.47m	3.53m	6.56m	71.26%
	200m	3.32m	10.35%	3.43m	10.52%	5.04m	4.31m	7.86m	4.91m	9.18m	70.03%
Town01_19	10m	0.19m	3.16%	0.20m	3.25%	0.77m	0.63m	1.33m	0.67m	1.25m	90.72%
	20m	0.64m	5.32%	0.66m	5.44%	0.68m	0.74m	1.16m	0.78m	2.14m	82.66%
	30m	1.15m	6.93%	1.18m	7.10%	0.82m	0.93m	1.38m	0.98m	3.16m	78.91%
	100m	1.98m	8.17%	2.01m	8.33%	1.27m	1.57m	1.94m	1.63m	4.36m	74.88%
	200m	2.34m	6.38%	2.34m	6.49%	2.64m	2.26m	3.26m	2.29m	6.36m	73.90%

Table C.1 – Detailed results of $DELTA_{SL}$ for the SLED dataset (on Town01).

Sequence	Cutoff	Dense depths errors				Sparse depths errors				Depth change map errors	
		On D_{bf}		On D_{af}		On D_{bf}		On D_{af}		Absolute error	Correctly classified events
		Raw	AbsRel	Raw	AbsRel	NN	$DELTA_{SL}$	NN	$DELTA_{SL}$		(with a threshold of ±1m)
Town03_00	10m	0.17m	2.43%	0.17m	2.51%	0.45m	0.07m	0.45m	0.07m	0.03m	99.96%
	20m	3.06m	21.17%	3.00m	20.85%	0.45m	0.18m	0.49m	0.20m	0.12m	99.74%
	30m	4.68m	26.65%	4.61m	26.30%	0.48m	0.25m	0.54m	0.27m	0.19m	99.60%
	100m	7.05m	28.67%	7.01m	28.41%	0.87m	0.92m	1.01m	0.97m	0.66m	97.80%
	200m	12.06m	22.40%	12.01m	22.21%	3.90m	2.70m	4.14m	2.78m	1.16m	94.27%
Town03_01	10m	0.09m	1.83%	0.10m	1.86%	0.44m	0.09m	0.45m	0.09m	0.03m	99.93%
	20m	0.70m	5.15%	0.76m	5.50%	0.60m	0.28m	0.77m	0.32m	1.33m	98.32%
	30m	2.05m	10.35%	2.25m	11.29%	0.88m	0.65m	1.22m	0.75m	2.28m	96.18%
	100m	4.10m	13.42%	4.47m	14.63%	1.87m	1.94m	2.96m	2.19m	3.77m	91.54%
	200m	5.64m	9.97%	5.82m	10.70%	5.15m	3.52m	6.91m	3.87m	6.20m	88.55%
Town03_02	10m	0.22m	3.08%	0.23m	3.15%	0.72m	0.32m	1.17m	0.49m	1.21m	97.05%
	20m	0.62m	5.35%	0.67m	5.63%	1.06m	0.69m	2.08m	0.82m	3.40m	92.58%
	30m	0.89m	6.12%	0.97m	6.57%	1.74m	1.35m	3.26m	1.54m	4.29m	88.89%
	100m	1.87m	6.90%	1.94m	7.22%	2.93m	3.13m	5.36m	3.40m	5.43m	78.24%
	200m	2.86m	5.55%	2.80m	5.72%	4.62m	3.99m	7.31m	4.32m	7.80m	76.92%
Town03_03	10m	0.18m	2.39%	0.17m	2.29%	0.63m	0.32m	0.95m	0.33m	1.10m	97.10%
	20m	1.52m	9.97%	1.57m	10.18%	0.70m	0.59m	1.36m	0.63m	3.54m	84.01%
	30m	2.42m	12.73%	2.49m	13.04%	0.82m	0.79m	1.58m	0.82m	3.60m	81.45%
	100m	3.13m	13.16%	3.19m	13.43%	1.65m	1.79m	2.75m	1.84m	3.77m	79.52%
	200m	6.40m	13.97%	6.38m	14.19%	2.48m	2.53m	3.63m	2.59m	5.81m	78.95%
Town03_04	10m	0.30m	3.76%	0.29m	3.62%	0.41m	0.38m	0.66m	0.40m	1.73m	93.18%
	20m	0.88m	7.39%	0.89m	7.41%	0.87m	0.74m	1.43m	0.76m	5.63m	84.12%
	30m	1.35m	8.63%	1.38m	8.72%	1.12m	1.16m	1.88m	1.22m	6.83m	80.92%
	100m	2.70m	10.39%	2.76m	10.54%	1.98m	2.46m	3.14m	2.55m	8.15m	76.49%
	200m	5.01m	9.32%	5.00m	9.41%	4.91m	4.57m	6.30m	4.70m	13.09m	74.54%
Town03_05	10m	1.45m	18.43%	1.47m	18.68%	0.34m	0.21m	0.43m	0.27m	1.05m	98.57%
	20m	2.02m	20.05%	2.15m	21.14%	0.35m	0.34m	0.67m	0.48m	1.90m	94.40%
	30m	2.31m	19.93%	2.46m	21.06%	0.72m	0.74m	1.40m	0.91m	2.42m	90.72%
	100m	4.49m	21.79%	4.65m	22.78%	1.73m	1.85m	3.06m	2.16m	3.59m	83.54%
	200m	9.60m	22.33%	9.66m	23.18%	2.16m	2.42m	3.65m	2.82m	5.36m	80.96%
Town03_06	10m	0.10m	1.67%	0.10m	1.59%	0.57m	0.19m	0.70m	0.19m	0.17m	98.77%
	20m	0.28m	2.49%	0.28m	2.47%	0.72m	0.60m	1.38m	0.61m	1.09m	93.93%
	30m	0.54m	3.23%	0.54m	3.23%	0.87m	0.94m	1.77m	0.96m	1.68m	91.09%
	100m	2.04m	5.48%	2.05m	5.50%	1.95m	2.97m	3.40m	3.02m	3.99m	83.82%
	200m	2.48m	4.72%	2.50m	4.74%	5.86m	4.79m	7.66m	4.82m	5.87m	81.48%
Town03_07	10m	0.25m	3.94%	0.26m	4.03%	0.73m	0.39m	1.37m	0.44m	1.11m	96.82%
	20m	0.78m	6.62%	0.80m	6.80%	1.27m	0.73m	2.69m	0.85m	2.91m	87.48%
	30m	1.75m	9.98%	1.78m	10.17%	1.94m	1.28m	3.87m	1.48m	4.01m	78.46%
	100m	3.89m	12.87%	3.94m	13.03%	3.57m	3.42m	6.50m	3.78m	5.05m	71.74%
	200m	4.09m	10.27%	4.10m	10.40%	5.38m	4.54m	8.57m	4.92m	7.15m	71.14%
Town03_08	10m	0.16m	2.28%	0.14m	2.04%	0.06m	0.20m	0.07m	0.20m	0.35m	97.83%
	20m	0.29m	2.84%	0.27m	2.64%	0.17m	0.31m	0.33m	0.33m	0.76m	96.68%
	30m	0.42m	3.25%	0.41m	3.08%	0.52m	0.56m	0.85m	0.61m	1.19m	95.68%
	100m	1.00m	4.18%	0.99m	4.03%	1.57m	2.01m	2.32m	2.07m	2.10m	92.29%
	200m	1.89m	3.63%	1.89m	3.52%	6.36m	3.60m	7.22m	3.61m	3.31m	89.41%
Town03_09	10m	1.36m	28.68%	1.39m	29.22%	0.17m	1.24m	0.56m	1.43m	0.45m	95.98%
	20m	1.42m	22.08%	1.45m	22.50%	0.16m	0.85m	0.56m	1.00m	0.49m	95.75%
	30m	1.47m	20.65%	1.50m	21.04%	0.27m	0.91m	0.71m	1.08m	0.61m	95.31%
	100m	1.64m	19.76%	1.67m	20.14%	0.38m	1.01m	0.85m	1.17m	1.15m	94.03%
	200m	1.92m	17.28%	1.89m	17.61%	0.53m	1.07m	0.93m	1.21m	1.91m	93.70%
Town03_10	10m	0.98m	11.93%	1.02m	12.62%	0.65m	0.44m	0.85m	0.48m	0.62m	96.05%
	20m	1.04m	10.15%	1.08m	10.67%	0.49m	0.43m	0.73m	0.47m	0.90m	95.75%
	30m	1.75m	12.22%	1.80m	12.77%	0.60m	0.65m	1.02m	0.67m	1.42m	94.18%
	100m	6.13m	19.19%	6.22m	19.78%	1.44m	1.77m	2.51m	1.82m	2.88m	88.31%
	200m	7.03m	15.96%	7.10m	16.43%	4.53m	2.69m	5.85m	2.80m	4.33m	85.74%
Town03_11	10m	0.27m	3.67%	0.26m	3.51%	0.67m	0.35m	0.89m	0.32m	0.81m	98.77%
	20m	2.56m	16.28%	2.61m	16.55%	0.98m	0.64m	1.51m	0.55m	2.79m	91.72%
	30m	4.31m	22.59%	4.39m	22.93%	1.30m	1.03m	1.96m	0.96m	3.60m	85.27%
	100m	5.51m	22.86%	5.58m	23.19%	2.43m	2.78m	3.63m	2.74m	4.31m	81.05%
	200m	4.71m	16.44%	4.73m	16.65%	5.02m	3.74m	6.61m	3.78m	5.42m	79.85%
Town03_12	10m	0.17m	2.50%	0.17m	2.47%	0.61m	0.13m	0.67m	0.13m	0.77m	98.73%
	20m	0.39m	3.40%	0.40m	3.48%	0.65m	0.49m	1.00m	0.51m	1.84m	91.70%
	30m	0.85m	4.93%	0.87m	5.04%	1.14m	0.86m	1.91m	0.90m	2.63m	87.62%
	100m	2.19m	7.18%	2.20m	7.26%	2.75m	2.96m	4.05m	2.99m	3.57m	81.65%
	200m	5.51m	6.88%	5.40m	6.88%	4.65m	4.03m	6.22m	4.11m	5.68m	80.11%
Town03_13	10m	0.15m	2.26%	0.14m	2.13%	0.56m	0.23m	0.82m	0.26m	0.67m	95.78%
	20m	1.45m	9.53%	1.49m	9.70%	0.61m	0.50m	1.21m	0.55m	2.65m	85.34%
	30m	2.41m	12.61%	2.48m	12.91%	0.75m	0.71m	1.49m	0.76m	3.61m	81.08%
	100m	3.32m	13.76%	3.39m	14.06%	1.58m	2.14m	2.50m	2.17m	4.41m	78.21%
	200m	5.29m	13.46%	5.23m	13.66%	3.05m	2.95m	4.05m	2.96m	6.78m	77.37%
Town03_14	10m	0.15m	2.14%	0.15m	2.20%	0.80m	0.28m	1.06m	0.31m	1.11m	98.77%
	20m	0.82m	6.19%	0.85m	6.40%	1.24m	1.29m	2.00m	1.40m	5.51m	89.25%
	30m	1.34m	7.90%	1.37m	8.15%	1.61m	1.89m	2.62m	2.01m	7.45m	83.52%
	100m	2.82m	9.68%	2.88m	9.94%	2.85m	3.53m	4.68m	3.72m	10.36m	76.37%
	200m	5.69m	8.63%	5.71m	8.79%	8.66m	7.09m	10.78m	7.33m	17.14m	74.35%
Town03_15	10m	0.28m	6.81%	0.27m	6.74%	0.52m	0.23m	0.86m	0.24m	0.79m	98.22%
	20m	0.65m	7.95%	0.69m	8.17%	0.92m	0.68m	2.06m	0.74m	2.24m	93.02%
	30m	0.94m	8.53%	1.00m	8.79%	1.26m	1.04m	2.69m	1.16m	2.79m	89.62%
	100m	2.36m	9.09%	2.48m	9.44%	2.37m	2.67m	4.88m	2.97m	5.32m	79.43%
	200m	3.36m	7.79%	3.38m	8.03%	5.29m	3.78m	8.18m	4.23m	8.09m	77.99%
Town03_16	10m	0.33m	4.85%	0.34m	4.94%	0.86m	0.63m	1.46m	0.71m	2.08m	92.95%
	20m	0.96m	8.37%	0.99m	8.62%	1.06m	0.94m	1.90m	1.01m	5.67m	83.91%
	30m	1.35m	9.34%	1.38m	9.56%	1.50m	1.28m	2.43m	1.34m	6.78m	80.75%
	100m	2.13m	9.30%	2.14m	9.43%	2.28m	2.49m	3.49m	2.56m	7.09m	77.62%
	200m	3.74m	7.88%	3.70m	7.96%	4.76m	3.96m	6.01m	4.00m	11.49m	75.74%
Town03_17	10m	0.06m	1.08%	0.06m	1.08%	0.45m	0.23m	0.72m	0.22m	0.25m	98.88%
	20m	0.25m	2.02%	0.25m	2.04%	0.96m	0.83m	1.95m	0.83m	2.79m	92.46%
	30m	0.56m	3.05%	0.58m	3.16%	0.98m	1.03m	2.13m	1.03m	3.07m	90.86%
	100m	1.33m	4.46%	1.36m	4.56%	1.61m	2.36m	3.13m	2.37m	3.55m	85.97%
	200m	2.25m	3.82%	2.23m	3.87%	6.77m	4.16m	8.50m	4.14m	5.18m	82.52%
Town03_18	10m	0.17m	2.53%	0.17m	2.56%	0.37m	0.22m	0.45m	0.23m	0.58m	97.26%
	20m	0.42m	3.71%	0.45m	3.93%	0.49m	0.42m	0.84m	0.46m	1.56m	91.31%
	30m	0.59m	4.13%	0.64m	4.42%	0.86m	0.66m	1.47m	0.75m	1.79m	87.73%
	100m	3.02m	8.05%	3.09m	8.33%	1.68m	1.71m	3.10m	1.99m	3.60m	79.12%
	200m	4.36m	7.69%	4.38m	7.90%	4.01m	2.94m	5.60m	3.37m	5.32m	78.08%
Town03_19	10m	1.85m	33.56%	1.88m	33.86%	0.28m	0.77m	0.49m	0.84m	0.75m	96.35%
	20m	1.98m	27.41%	2.02m	27.80%	0.35m	0.70m	0.70m	0.79m	0.92m	94.49%
	30m	1.97m	24.81%	2.01m	25.15%	0.51m	0.87m	0.89m	0.94m	1.05m	93.58%
	100m	2.10m	22.98%	2.14m	23.31%	0.73m	1.06m	1.36m	1.15m	1.29m	91.99%
	200m	2.25m	20.35%	2.22m	20.64%	0.82m	1.10m	1.40m	1.17m	2.24m	91.68%

Table C.2 – Detailed results of $DELTA_{SL}$ for the SLED dataset (on Town03).

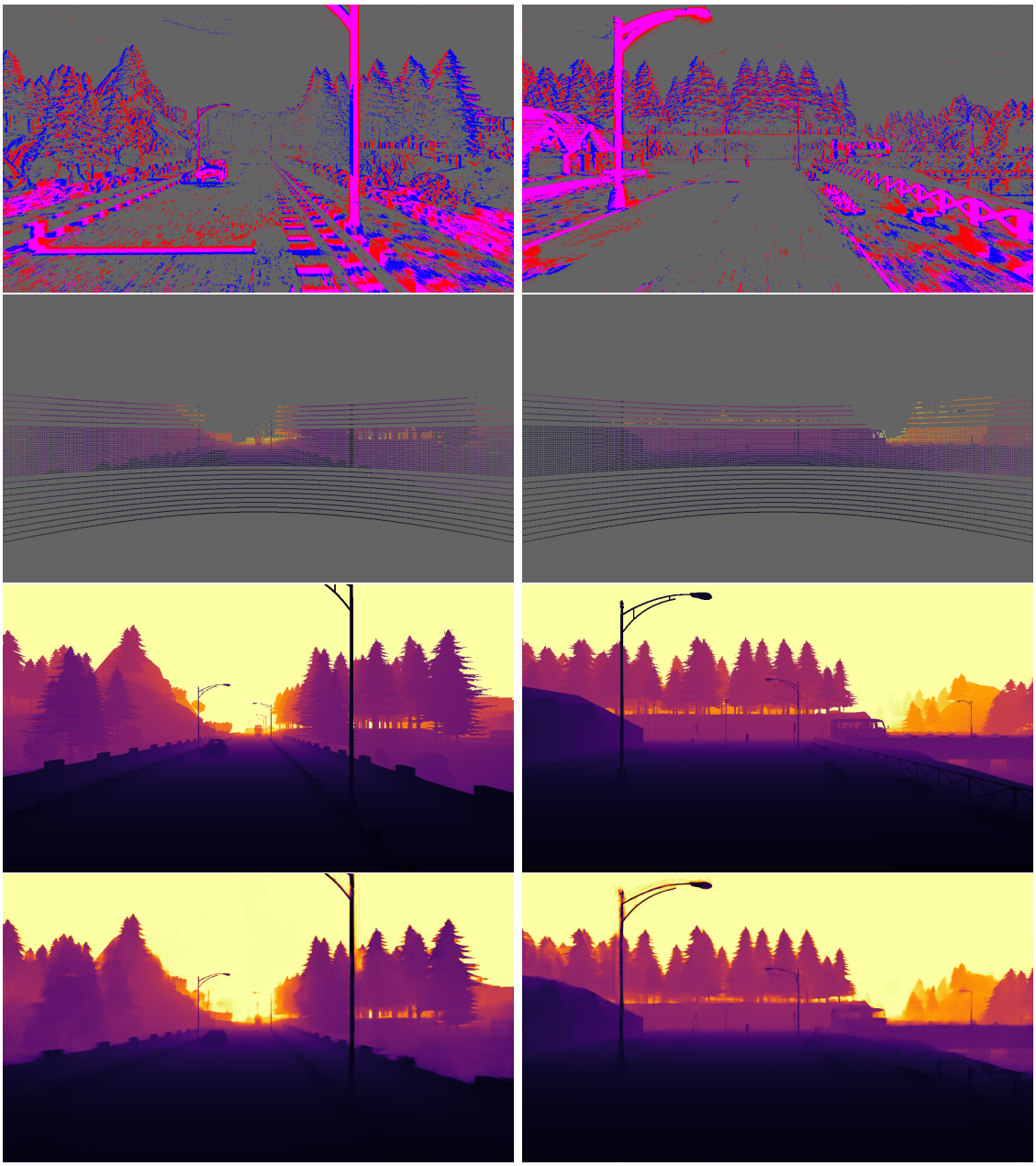


Figure C.1 – Additional results on the SLED dataset, on sequences `Town01_03` and `Town01_05`. From top to bottom: events, LiDAR projection, ground truth, our predictions (DELTA_{SL}).

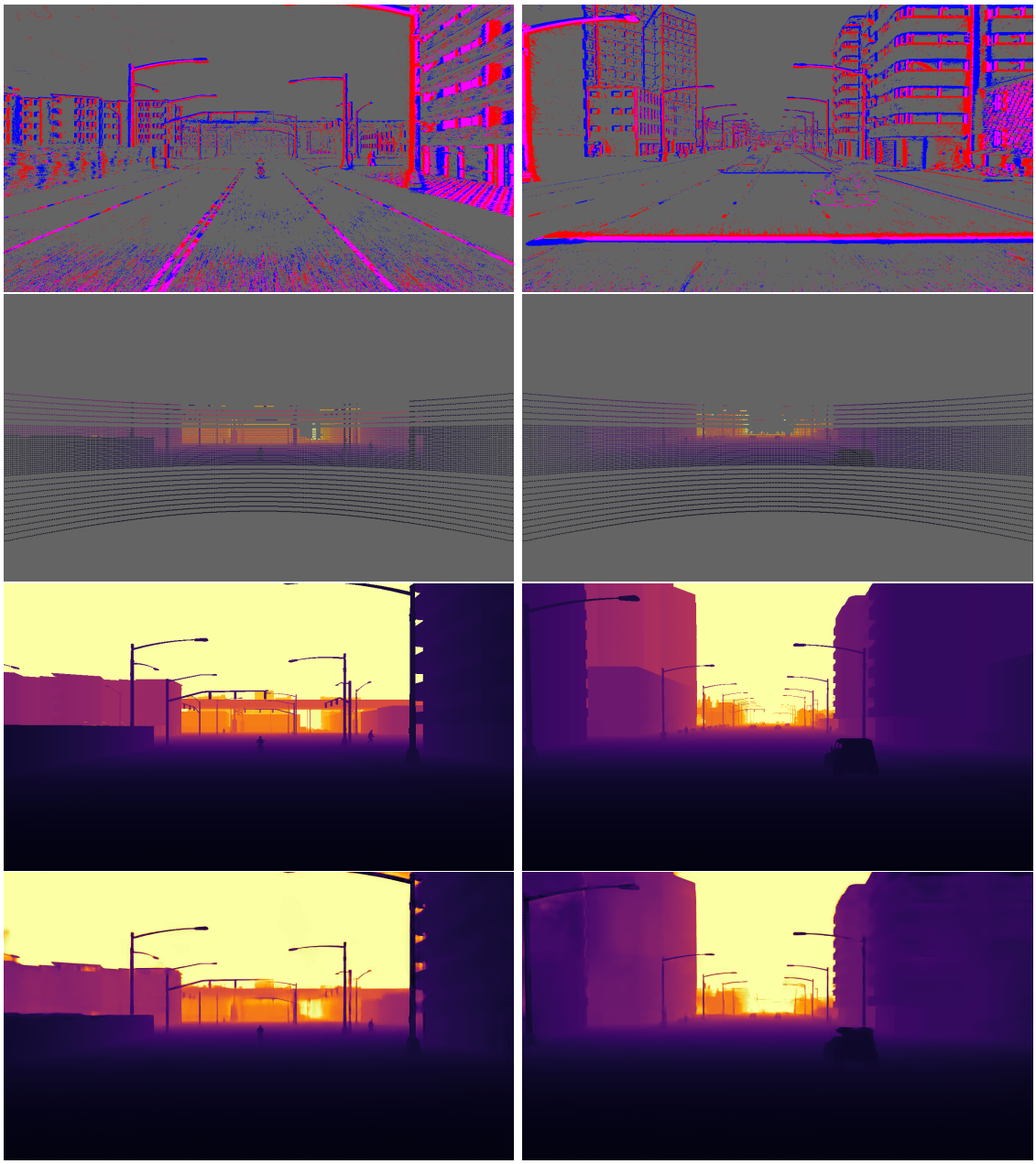


Figure C.2 – Additional results on the SLED dataset, on sequences `Town03_02` and `Town03_06`. From top to bottom: events, LiDAR projection, ground truth, our predictions (DELTA_{SL}).

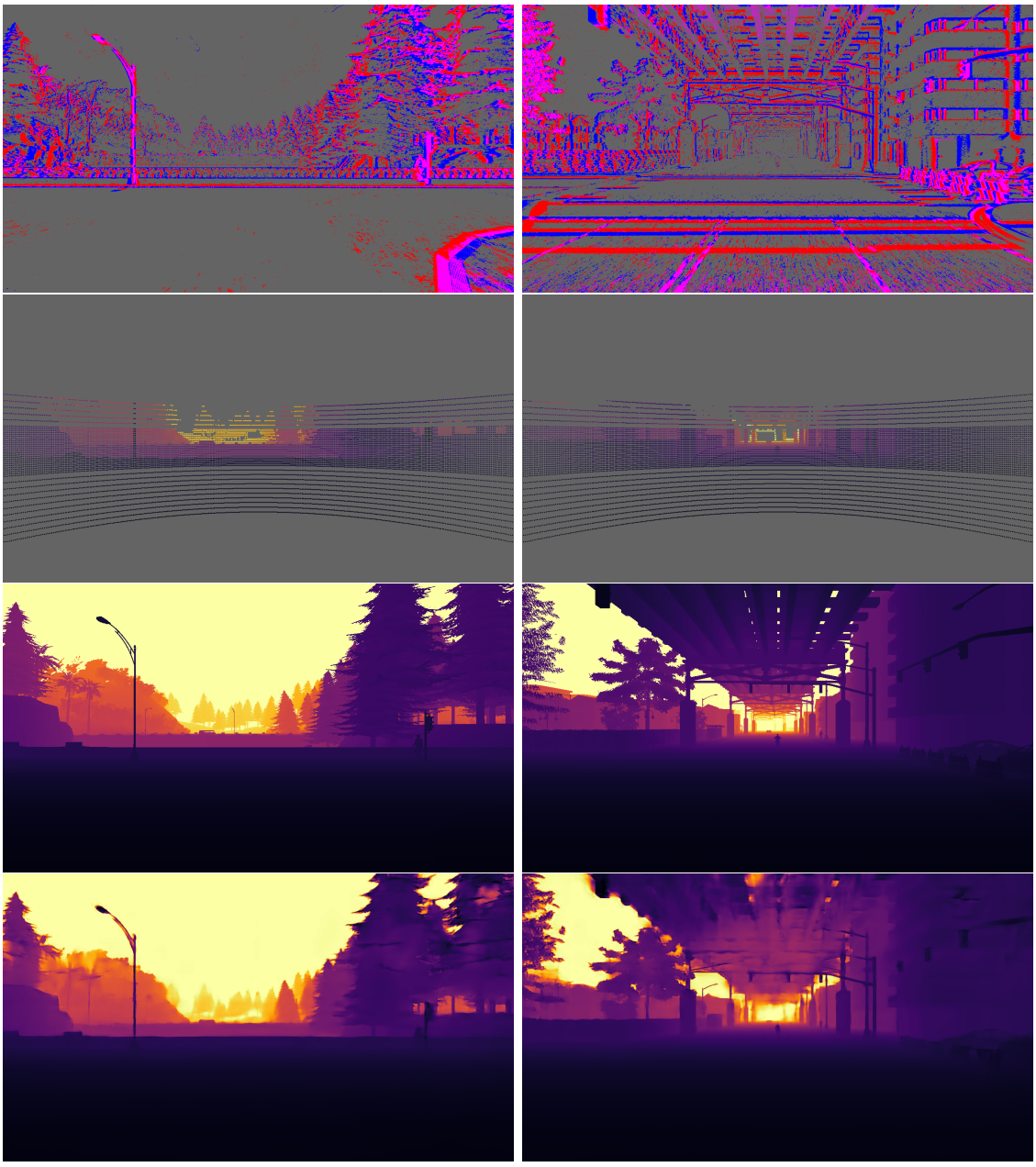


Figure C.3 – Additional results on the SLED dataset, on sequences `Town01_18` and `Town03_13`. From top to bottom: events, LiDAR projection, ground truth, our predictions (DELTA_{SL}). Shown here are two cases where DELTA_{SL} displays moderate errors for objects in the upper part of the depth maps, like the trees on the left and on the top right for the left column, and the suspended railway on the top for the right column.

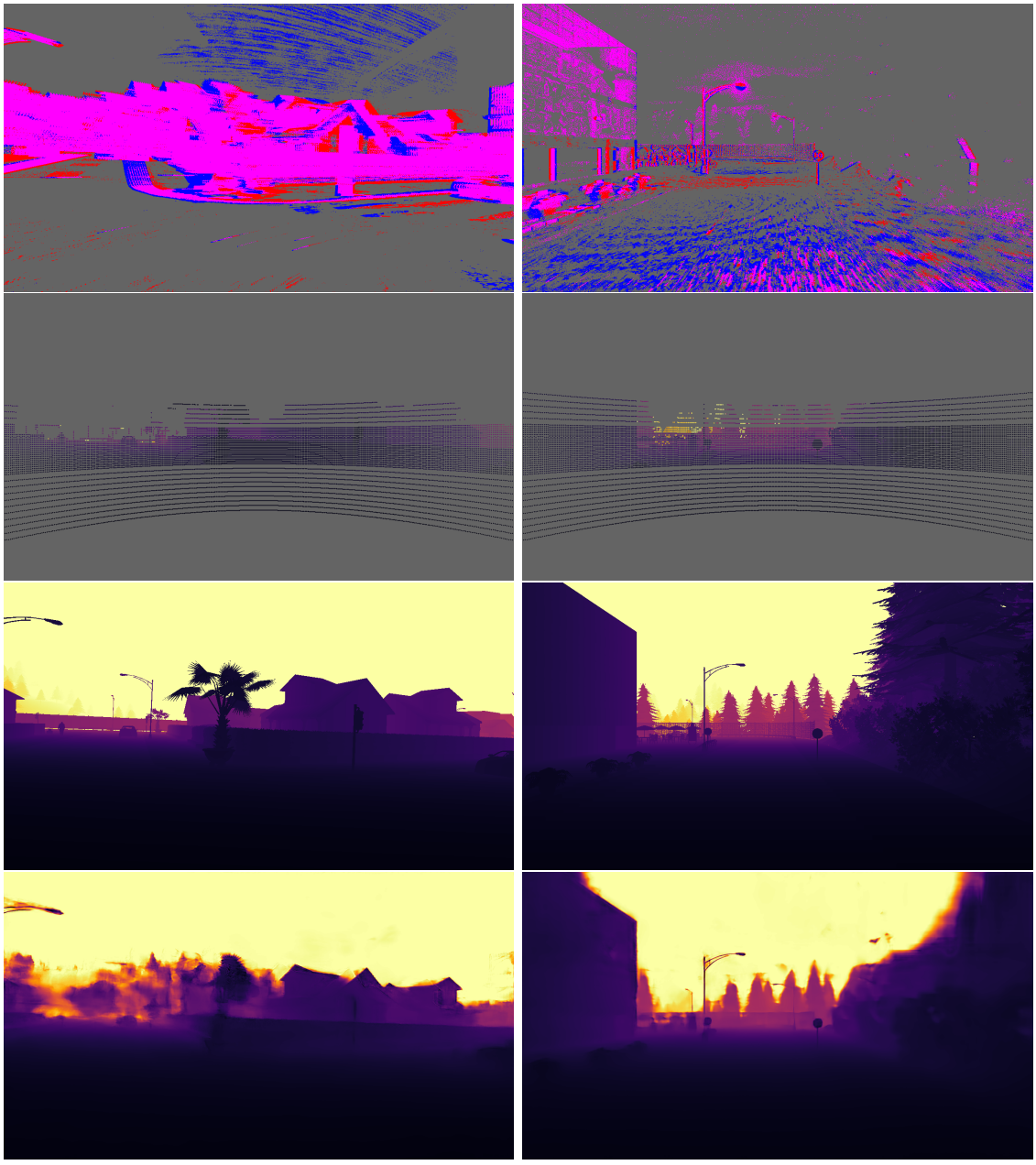


Figure C.4 – Additional results on the SLED dataset, on sequences `Town01_08` and `Town01_11`. From top to bottom: events, LiDAR projection, ground truth, our predictions (DELTA_{SL}). Shown here are two failure cases where DELTA_{SL} displays large errors. Left: due to a high-speed sharp turn, a very high quantity of events is produced in the time window of accumulation, leading to information being lost due to the formulation of the event volume, and thus leading to an inaccurate depth estimation. Right: due to the limitations of the event camera in the CARLA simulator, dark objects in a night scene like the trees in the middle and on the right of the scene are not captured in the event stream of the SLED dataset, resulting in an incorrect depth estimation.

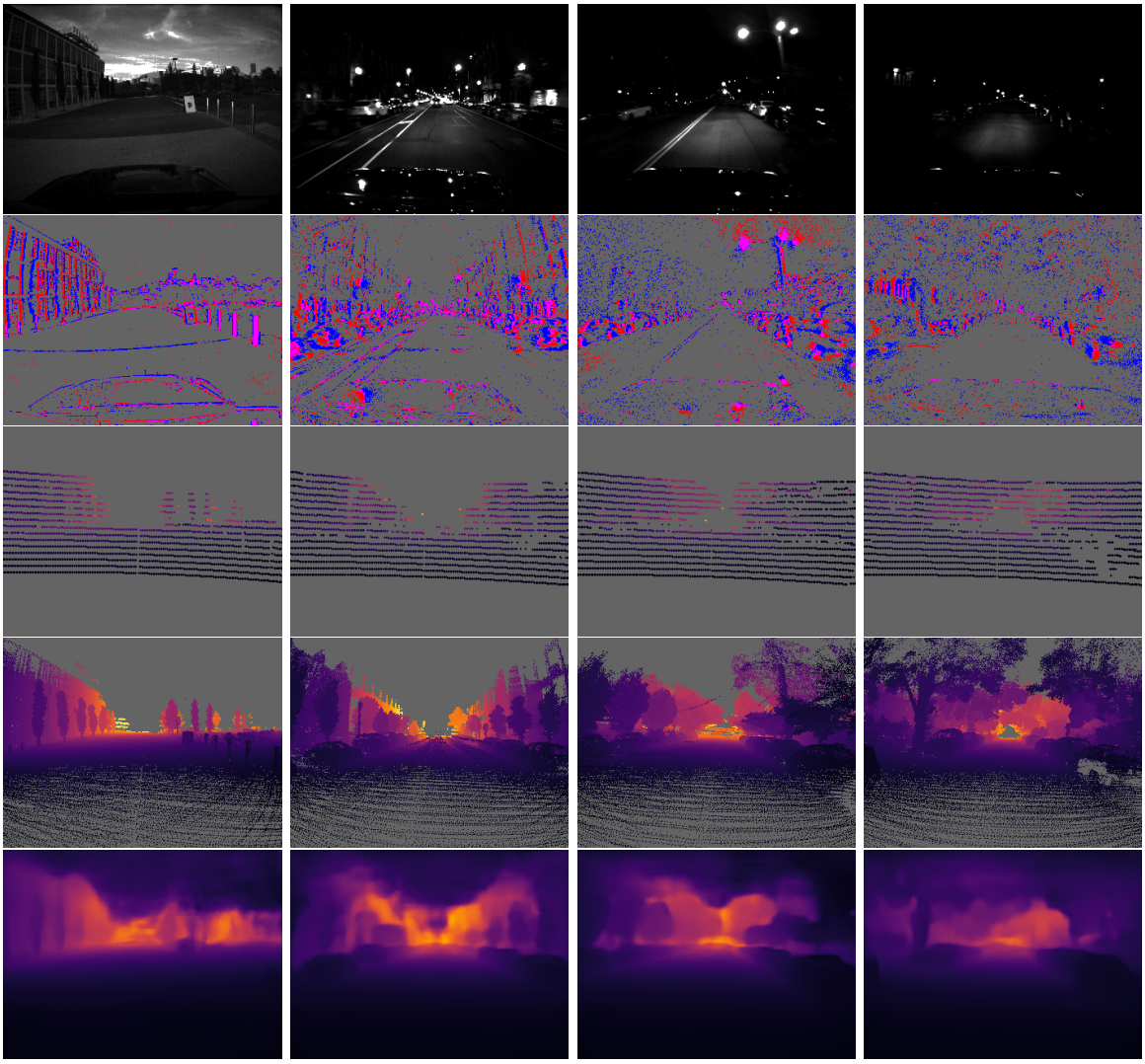


Figure C.5 – Additional results on the MVSEC dataset. Sequences shown, from left to right: `outdoor_day_1`; `outdoor_night_1`; `outdoor_night_2`; `outdoor_night_3\Verb|`.From| top to bottom: reference image of the scene; events; LiDAR projection (with size of points increased for a better visibility); ground truth; our results ($\text{DELTA}_{\text{SL}\rightarrow\text{MV}}$).

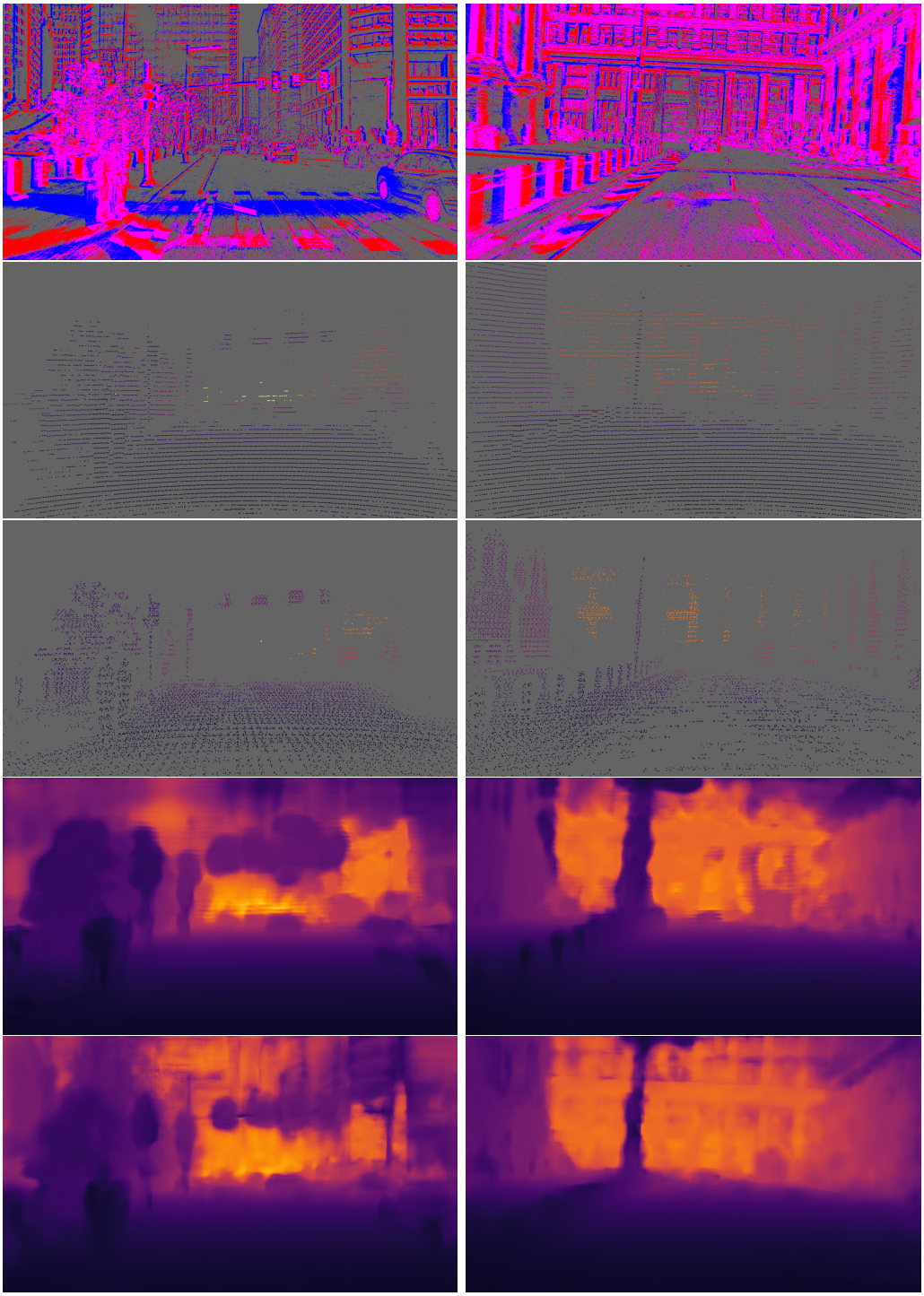


Figure C.6 – Additional results on the M3ED dataset, for the `city_hall_day` sequence. From top to bottom: events, LiDAR projection, ground truth, our results (DELTA_{M3}, $\text{DELTA}_{\text{SL}\rightarrow\text{M3}}$).

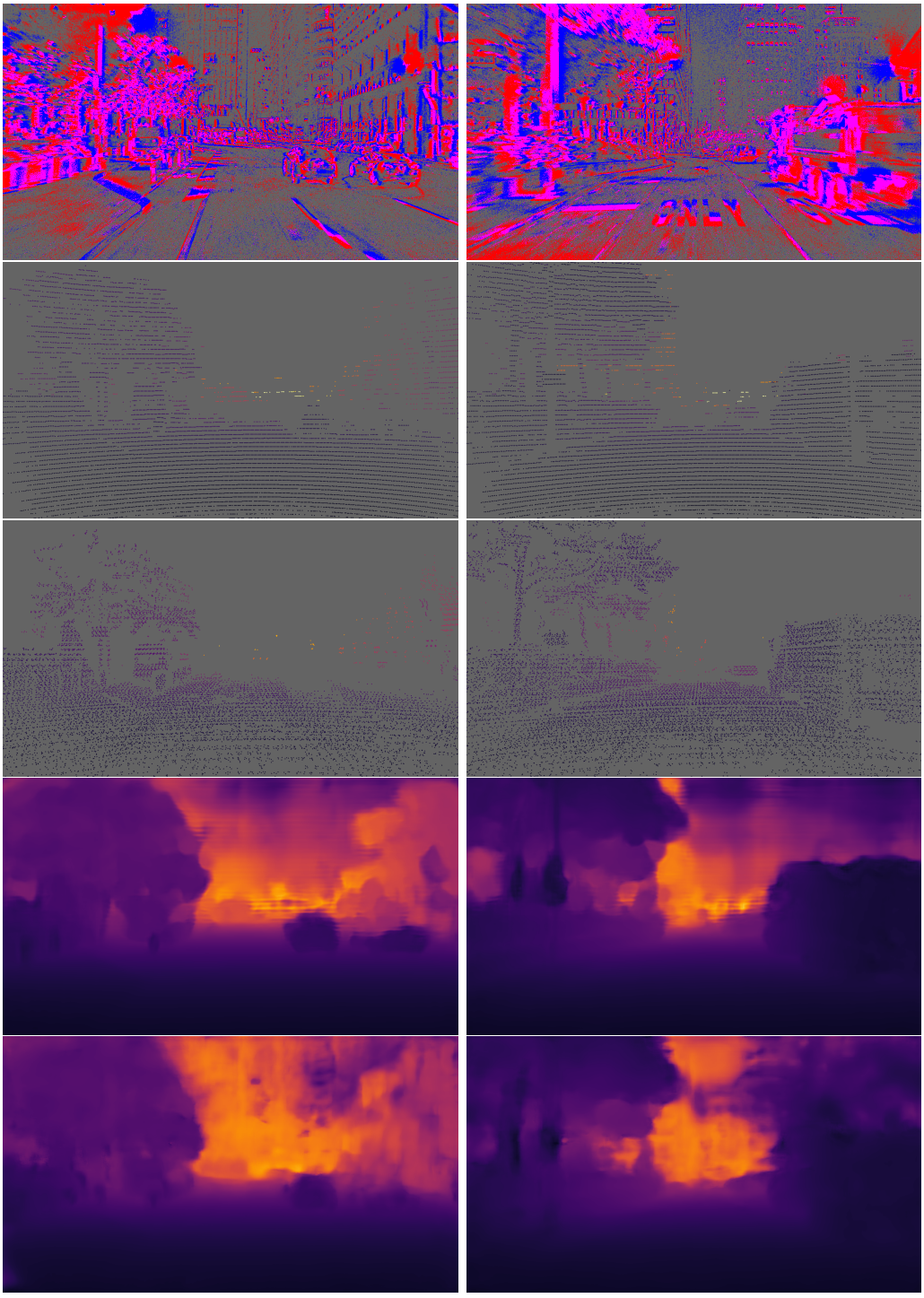


Figure C.7 – Additional results on the M3ED dataset, for the `city_hall_night` sequence. From top to bottom: events, LiDAR projection, ground truth, our results (DELTA_{M3}, $\text{DELTA}_{\text{SL}\rightarrow\text{M3}}$).

GLOSSARY

hot pixel broken pixel in an event camera which constantly detects events. 11

shot noise in optics, noise originating from the fluctuations of the number of photons detected by the sensor. 11

ACRONYMS

AAE Average Angular Error. 39, 40

AEE Average Endpoint Error. 34–37, 39, 40, 45

ALED Asynchronous LiDAR and Events Depths densification network. v, vii, viii, 5, 50, 53, 55, 57, 60, 62, 63, 69, 70, 75, 83, 84, 86, 87, 93, 96, 98

AR Augmented Reality. 17

CPU Central Processing Unit. 25, 26, 28, 46

DELTA Dense depths from Events and LiDAR using Transformer's Attention. v, viii, 5, 69, 70, 80–84, 87, 91, 93, 94, 97, 98

EBOF Event-Based Optical Flow. 21, 23–25, 34–37, 42, 44, 45, 47, 48

EOS End of Sentence. 70

FPGA Field-Programmable Gate Array. 48

FPS Frames Per Second. 7, 10, 11, 16

FWL Flow Warping Loss. 35, 37, 40, 42, 44, 101

GNN Graph Neural Network. 14

GPU Graphics Processing Unit. 11, 14, 25, 26, 28, 32, 33, 46, 48

GRU Gated Recurrent Unit. 82, 93

GT Ground Truth. 38

HD High Definition. 11, 33, 80

HDR High Dynamic Range. 11, 15, 25

LSTM Long Short-Term Memory. 70

MEPS Millions of Events Per Second. 16, 17

NLP Natural Language Processing. 69–71

NN Nearest Neighbor. 60, 64, 74, 78, 79

PTP Precision Time Protocol. 104, 105

RSU Road Side Unit. 64

RTEF Real-Time Event-based Flow. 21, 25, 33, 40, 46, 98

SLAM Simultaneous Localization and Mapping. 50, 51

SLED Synthetic LiDAR Events Depths. 50, 52, 56, 57, 59–62, 64–66, 78, 80, 83–87, 89, 91, 93, 96, 99, 105, 107, 109, 119

SNN Spiking Neural Network. 14, 23

UAV Unmanned Aerial Vehicle. 19

VR Virtual Reality. 17

BIBLIOGRAPHY

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

[2] Misha A. Mahowald. "VLSI analogs of neuronal visual processing: A synthesis of form and function". PhD thesis. California Institute of Technology, 1992.

[3] Edison Gerena. "6-DoF Optical-driven Micro-robots with Force Feedback Capabilities for Interactive Bio-manipulation". PhD thesis. Sorbonne Université, 2020.

[4] Xiang Li et al. "Intelligent Machinery Fault Diagnosis With Event-Based Camera". In: *IEEE Transactions on Industrial Informatics* (2023).

[5] Kamil Bialik et al. "Fast Object Counting with an Event Camera". In: *Pomiary Automatyka Robotyka* (2023).

[6] Guang Chen et al. "Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception". In: *IEEE Signal Processing Magazine* 37 (2020), pp. 34–49.

[7] Christian E. Willert and Joachim Klinner. "Event-based imaging velocimetry: an assessment of event-based cameras for the measurement of fluid flows". In: *Experiments in Fluids* 63 (2022).

[8] Tat-Jun Chin et al. "Star Tracking Using an Event Camera". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), pp. 1646–1655.

[9] Western Sydney University. *World first technology led by Western now in orbit*. 2021. URL: https://www.westernsydney.edu.au/newscentre/news_centre/story_archive/2021/world_first_technology_led_by_western_now_in_orbit (visited on 11/14/2023).

[10] World Heath Organization. *Global Status Report on Road Safety*. World Health Organization, 2018. ISBN: 9789241565684.

[11] Barry A. T. Brown and E. Laurier. "The Trouble with Autopilots: Assisted and Autonomous Driving on the Social Road". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017).

[12] Tabitha S. Combs et al. "Automated Vehicles and Pedestrian Safety: Exploring the Promise and Limits of Pedestrian Detection." In: *American journal of preventive medicine* 56 1 (2019), pp. 1–7.

[13] T. Finateu et al. "A 1280x720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86µm Pixels, 1.066GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline". In: *ISSCC* (2020), pp. 112–114.

[14] Boyang Li et al. "Enhancing 3-D LiDAR Point Clouds With Event-Based Camera". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–12.

[15] Christian Brandli et al. "A 240×180 130dB 3µs Latency Global Shutter Spatiotemporal Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 49 (2014), pp. 2333–2341.

[16] Guillermo Gallego et al. "Event-based Vision: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

[17] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS". In: *IEEE Journal of Solid-State Circuits* 46 (2011), pp. 259–275.

[18] Xueye Zheng et al. "Deep Learning for Event-based Vision: A Comprehensive Survey and Benchmarks". In: *ArXiv* abs/2302.08890 (2023).

[19] 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.

[20] Guillermo Gallego et al. "Event-Based, 6-DOF Camera Tracking from Photometric Depth Maps". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (2016), pp. 2402–2412.

[21] Amélie Gruel et al. "Neuromorphic Event-Based Spatio-temporal Attention using Adaptive Mechanisms". In: *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (2022), pp. 379–382.

[22] Tobias Delbrück. "Frame-free dynamic digital vision". In: *Proceedings of the International Symposium on Secure-Life Electronics, Advanced Electronics for Quality Life and Society* (2008), pp. 21–26.

[23] Xavier Lagorce et al. "HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2017), pp. 1346–1359.

[24] A. Z. Zhu et al. "Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion". In: *CVPR* (2019), pp. 989–997.

[25] E. Perot et al. "Learning to Detect Objects with a 1 Megapixel Event Camera". In: *34th Conference on Neural Information Processing Systems (NeurIPS 2020)* (2020).

[26] Daniel Gehrig et al. "End-to-End Learning of Representations for Asynchronous Event-Based Data". In: *ICCV* (2019), pp. 5632–5642.

[27] Nikola Zubi'c et al. "From Chaos Comes Order: Ordering Event Representations for Object Recognition and Detection". In: *ICCV* (2023).

[28] Cedric Scheerlinck, Nick Barnes, and Robert E. Mahony. "Continuous-time Intensity Estimation Using Event Cameras". In: *Asian Conference on Computer Vision*. 2018, pp. 308–324.

[29] Henri Rebecq et al. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021), pp. 1964–1980.

[30] Daniel Dauner. "Image Reconstruction from Event Cameras for Autonomous Driving". In: *International Conference on Learning Representations (ICLR)* (2023).

[31] R. Benosman et al. "Event-Based Visual Flow". In: *IEEE Transactions on Neural Networks and Learning Systems* 25 (2014), pp. 407–417.

[32] A. Z. Zhu et al. "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras". In: *Proceedings of Robotics: Science and Systems*. 2018.

[33] Daniel Gehrig et al. "Combining Events and Frames Using Recurrent Asynchronous Multimodal Networks for Monocular Depth Prediction". In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 2822–2829.

[34] F. Paredes-Vallés, Jesse J. Hagenaars, and Guido C.H.E. de Croon. "Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks". In: *Neural Information Processing Systems*. 2021.

[35] Ulysse Rançon et al. "StereoSpike: Depth Learning With a Spiking Neural Network". In: *IEEE Access* 10 (2021), pp. 127428–127439.

[36] Yusra Alkendi et al. "Neuromorphic Camera Denoising using Graph Neural Network-driven Transformers". In: *IEEE transactions on neural networks and learning systems* PP (2021).

[37] Simon Thomas Schaefer, Daniel Gehrig, and Davide Scaramuzza. "AEGNN: Asynchronous Event-based Graph Neural Networks". In: *CVPR* (2022), pp. 12361–12371.

[38] Timo Stoffregen and L. Kleeman. "Simultaneous Optical Flow and Segmentation (SOFAS) using Dynamic Vision Sensor". In: *Australasian Conference on Robotics and Automation (ACRA)* (2017).

[39] Mathias Gehrig et al. "E-RAFT: Dense Optical Flow from Event Cameras". In: *International Conference on 3D Vision (3DV)*. 2021.

[40] Haotian Liu et al. "TMA: Temporal Motion Aggregation for Event-based Optical Flow". In: *ICCV* (2023).

[41] Stephan Schraml et al. "Dynamic stereo vision system for real-time tracking". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (2010), pp. 1409–1412.

[42] Stephan Schraml, Ahmed Nabil Belbachir, and Horst Bischof. "An Event-Driven Stereo System for Real-Time 3-D 360° Panoramic Vision". In: *IEEE Transactions on Industrial Electronics* 63 (2016), pp. 418–428.

[43] Yeongwoo Nam et al. "Stereo Depth from Events Cameras: Concentrate and Focus on the Future". In: *CVPR* (2022), pp. 6104–6113.

[44] David Weikersdorfer et al. "Event-based 3D SLAM with a depth-augmented dynamic vision sensor". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 359–364.

[45] Mingyue Cui et al. “Dense Depth-Map Estimation Based on Fusion of Event Camera and Sparse LiDAR”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–11.

[46] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. “Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera”. In: *European Conference on Computer Vision*. 2016.

[47] Javier Hidalgo-Carrió, Daniel Gehrig, and Davide Scaramuzza. “Learning Monocular Dense Depth from Events”. In: *2020 International Conference on 3D Vision (3DV)* (2020), pp. 534–542.

[48] F. Paredes-Vallés and G. D. Croon. “Back to Event Basics: Self-Supervised Learning of Image Reconstruction for Event Cameras via Photometric Constancy”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 3446–3455.

[49] Lei Yu et al. “Learning to See Through With Events”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2022), pp. 8660–8678.

[50] Burak Ercan et al. “HyperE2VID: Improving Event-Based Video Reconstruction via Hypernetworks”. In: *ArXiv* abs/2305.06382 (2023).

[51] Rafael Serrano-Gotarredona et al. “CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory-Processing-Learning-Actuating System for High-Speed Visual Object Recognition and Tracking”. In: *IEEE Transactions on Neural Networks* 20 (2009), pp. 1417–1438.

[52] Georg Wiesmann et al. “Event-driven embodied system for feature extraction and object recognition in robotic applications”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2012), pp. 76–82.

[53] Amos Sironi et al. “HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 1731–1740.

[54] Mathias Gehrig and Davide Scaramuzza. “Recurrent Vision Transformers for Object Detection with Event Cameras”. In: *CVPR* (2022), pp. 13884–13893.

[55] G. Orchard et al. “Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades”. In: *Frontiers in Neuroscience* 9 (2015).

[56] Cheston Tan, Stéphane Lallée, and G. Orchard. “Benchmarking neuromorphic vision: lessons learnt from computer vision”. In: *Frontiers in Neuroscience* 9 (2015).

[57] Daniel Gehrig et al. “Video to Events: Recycling Video Datasets for Event Cameras”. In: *CVPR* (2019), pp. 3583–3592.

[58] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbrück. “v2e: From Video Frames to Realistic DVS Events”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2021), pp. 1312–1321.

[59] Damien Joubert et al. “Event Camera Simulator Improvements via Characterized Parameters”. In: *Frontiers in Neuroscience* 15 (2021).

[60] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. "Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details". In: *Frontiers in Neuroscience* 9 (2015).

[61] Arnon Amir et al. "A Low Power, Fully Event-Based Gesture Recognition System". In: *CVPR* (2017), pp. 7388–7397.

[62] A. Z. Zhu et al. "The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception". In: *IEEE Robotics and Automation Letters (RA-L)* 3 (2018), pp. 2032–2039.

[63] Mathias Gehrig et al. "DSEC: A Stereo Event Camera Dataset for Driving Scenarios". In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 4947–4954.

[64] Kenneth Chaney et al. "M3ED: Multi-Robot, Multi-Sensor, Multi-Environment Event Dataset". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2023), pp. 4016–4023.

[65] Elias Mueggler et al. "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM". In: *The International Journal of Robotics Research* 36 (2016), pp. 142–149.

[66] Wenbin Li et al. "InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset". In: *BMVC*. 2018.

[67] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conference on Robot Learning*. 2018.

[68] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[69] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbrück. "A 128x128 120dB 15µs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43 (2008), pp. 566–576.

[70] Levi 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: *ArXiv* (2022).

[71] Prophesee. *Introducing the world's smallest and most power-efficient Event-based Vision sensor ever released*. 2023. URL: `https://www.prophesee.ai/event-based-sensor-genx320/` (visited on 11/16/2023).

[72] Zhaoning Sun et al. "ESS: Learning Event-based Semantic Segmentation from Still Images". In: *ECCV*. 2022.

[73] Daniel Gehrig and Davide Scaramuzza. "Pushing the Limits of Asynchronous Graph-based Object Detection with Event Cameras". In: *ArXiv* (2022).

[74] Vincent Brebion, Julien Moreau, and Franck Davoine. "Real-Time Optical Flow for Vehicular Perception With Low- and High-Resolution Event Cameras". In: *IEEE Transactions on Intelligent Transportation Systems* 23.9 (2022), pp. 15066–15078.

[75] Vincent Brebion, Julien Moreau, and Franck Davoine. "Estimation de flot optique basé évènements en temps réel". In: *Congrès Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP 2022)*. Vannes, France, July 2022.

[76] C. Braillon et al. "Real-time moving obstacle detection using optical flow models". In: *IEEE Intelligent Vehicles Symposium* (2006), pp. 466–471.

[77] Muhammad Kamal Hossen and Sabrina Hoque Tuli. "A surveillance system based on motion detection and motion estimation using optical flow". In: *5th International Conference on Informatics, Electronics and Vision (ICIEV)*. 2016, pp. 646–651.

[78] Cheng Chuanqi et al. "Monocular visual odometry based on optical flow and feature matching". In: *29th Chinese Control And Decision Conference (CCDC)* (2017), pp. 4554–4558.

[79] S. Galic and S. Lončarić. "Spatio-temporal image segmentation using optical flow and clustering algorithm". In: *Proceedings of the First IWISPA* (2000), pp. 63–68.

[80] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. "Optical flow modeling and computation: A survey". In: *Computer Vision and Image Understing* 134 (2015), pp. 1–21.

[81] Berthold K. P. Horn and Brian G. Schunck. "Determining Optical Flow". In: *Artificial Intelligence* 17 (1981), pp. 185–203.

[82] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *International Joint Conference on Artificial Intelligence*. 1981.

[83] E. Meinhardt, J. Pérez, and D. Kondermann. "Horn-Schunck Optical Flow with a Multi-Scale Strategy". In: *Image Processing On Line* 3 (2013), pp. 151–172.

[84] J.-Y. Bouguet. *Pyramidal implementation of the Lucas Kanade feature tracker*. Tech. rep. Intel Corporation, Microprocessor Research Labs, 1999.

[85] Michael J. Black and P. Anandan. "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields". In: *Computer Vision and Image Understanding* 63 (1996), pp. 75–104.

[86] Deqing Sun, S. Roth, and Michael J. Black. "A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them". In: *International Journal of Computer Vision* 106 (2013), pp. 115–137.

[87] A. Dosovitskiy et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *ICCV* (2015), pp. 2758–2766.

[88] S. Meister, Junhwa Hur, and S. Roth. "UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[89] Zachary Teed and Jun Deng. "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow". In: *ECCV*. 2020, pp. 402–419.

[90] Haofei Xu et al. "Unifying Flow, Stereo and Depth Estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2022), pp. 13941–13958.

[91] Guillermo Gallego, Henri Rebecq, and D. Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation". In: *CVPR* (2018), pp. 3867–3876.

[92] Daqi Liu, Álvaro Parra Bustos, and Tat-Jun Chin. "Globally Optimal Contrast Maximisation for Event-Based Motion Estimation". In: *CVPR* (2020), pp. 6348–6357.

[93] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. "Secrets of Event-Based Optical Flow". In: *European Conference on Computer Vision*. 2022.

[94] F. Paredes-Vallés et al. "Taming Contrast Maximization for Learning Sequential, Low-latency, Event-based Optical Flow". In: *ICCV* (2023).

[95] F. Paredes-Vallés, Kirk Y. W. Scheper, and G. de Croon. "Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception". In: *IEEE TPAMI* 42 (2020), pp. 2051–2064.

[96] Javier Cuadrado et al. "Optical flow estimation from event-based cameras and spiking neural networks". In: *Frontiers in Neuroscience* 17 (2023).

[97] M. Almatrafi et al. "Distance Surface for Event-Based Optical Flow". In: *IEEE TPAMI* 42 (2020), pp. 1547–1556.

[98] A. Z. Zhu, N. Atanasov, and Kostas Daniilidis. "Event-based feature tracking with probabilistic data association". In: *ICRA* (2017), pp. 4465–4470.

[99] C. G. Harris and M. Stephens. "A Combined Corner and Edge Detector". In: *Alvey Vision Conference*. 1988.

[100] Jun Nagata, Yusuke Sekikawa, and Y. Aoki. "Optical Flow Estimation by Matching Time Surface with Event-Based Cameras". In: *Sensors* 21 (2021).

[101] Chankyu Lee et al. "Spike-FlowNet: Event-based Optical Flow Estimation with Energy-Efficient Hybrid Neural Networks". In: *ECCV*. 2020.

[102] Liyuan Pan, Miaomiao Liu, and R. Hartley. "Single Image Optical Flow Estimation With an Event Camera". In: *CVPR* (2020), pp. 1669–1678.

[103] Bodo Rueckauer and Tobi Delbruck. "Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor". In: *Frontiers in Neuroscience* 10 (2016).

[104] 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)* (2021), pp. 6504–6510.

[105] H. Akolkar, S. Ieng, and R. Benosman. "See before you see: Real-time high speed motion prediction using fast aperture-robust event-driven visual flow". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[106] C. Ramamoorthy and H. F. Li. "Pipeline Architecture". In: *ACM Computing Surveys* 9 (1977), pp. 61–102.

[107] Yang Feng et al. "Event Density Based Denoising Method for Dynamic Vision Sensor". In: *Applied Sciences* 10 (2020), p. 2024.

[108] D. Coeurjolly, A. Montanvert, and J. Chassery. "Distances discrètes". In: *Géométrie discrète et images numériques*. Hermès Paris, 2007. Chap. 5, pp. 123–145.

[109] J. Adarve and R. Mahony. "A Filter Formulation for Computing Real Time Optical Flow". In: *IEEE Robotics and Automation Letters (RA-L)* 1 (2016), pp. 1192–1199.

[110] Michael J. Black. "Robust incremental optical flow". PhD thesis. Yale University, 1992.

[111] Timo Stoffregen et al. "Reducing the Sim-to-Real Gap for Event Cameras". In: *ECCV*. 2020.

[112] Andreas Geiger, Philip Lenz, and R. Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *CVPR* (2012), pp. 3354–3361.

[113] Xinglong Luo et al. "Learning Optical Flow from Event Camera with Rendered Dataset". In: *ICCV* (2023).

[114] Chengxi Ye et al. "Unsupervised Learning of Dense Optical Flow, Depth and Egomotion with Event-Based Sensors". In: *IROS* (2020), pp. 5831–5838.

[115] Yijin Li et al. "BlinkFlow: A Dataset to Push the Limits of Event-based Optical Flow Estimation". In: *ArXiv* (2023).

[116] M. Liu and T. Delbrück. "Adaptive Time-Slice Block-Matching Optical Flow Algorithm for Dynamic Vision Sensors". In: *BMVC*. 2018.

[117] Jae Hyung Jung and Chan Gook Park. "Constrained Filtering-based Fusion of Images, Events, and Inertial Measurements for Pose Estimation". In: *ICRA* (2020), pp. 644–650.

[118] Wachirawit Ponghiran, Chamika M. Liyanagedera, and Kaushik Roy. "Event-based Temporally Dense Optical Flow Estimation with Sequential Learning". In: *ICCV* (2023).

[119] Zhexiong Wan et al. "RPEFlow: Multimodal Fusion of RGB-PointCloud-Event for Joint Optical Flow and Scene Flow Estimation". In: *ICCV* (2023).

[120] Vincent Brebion, Julien Moreau, and Franck Davoine. "Learning to Estimate Two Dense Depths from LiDAR and Event Data". In: *Image Analysis - 22nd Scandinavian Conference, SCIA 2023, Sirkka, Finland, April 18-21, 2023, Proceedings, Part II*. Vol. 13886. Lecture Notes in Computer Science. Springer, 2023, pp. 517–533.

[121] Jonas Uhrig et al. "Sparsity Invariant CNNs". In: *2017 International Conference on 3D Vision (3DV)* (2017), pp. 11–20.

[122] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. "Deep Convolutional Compressed Sensing for LiDAR Depth Completion". In: *Asian Conference on Computer Vision*. 2018, pp. 499–513.

[123] Zixuan Huang et al. "HMS-Net: Hierarchical Multi-Scale Sparsity-Invariant Network for Sparse Depth Completion". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 3429–3441.

[124] Jason Ku, Ali Harakeh, and Steven L. Waslander. "In Defense of Classical Image Processing: Fast Depth Completion on the CPU". In: *2018 15th Conference on Computer and Robot Vision (CRV)* (2018), pp. 16–22.

[125] Maximilian Jaritz et al. "Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation". In: *2018 International Conference on 3D Vision (3DV)* (2018), pp. 52–60.

[126] Wouter Van Gansbeke et al. "Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty". In: *2019 16th International Conference on Machine Vision Applications (MVA)* (2019), pp. 1–6.

[127] Yan Xu et al. "Depth Completion From Sparse LiDAR Data With Depth-Normal Constraints". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 2811–2820.

[128] William P. Maddern and Paul Newman. "Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 2181–2188.

[129] Liyuan Pan et al. "Bringing a Blurry Frame Alive at High Frame-Rate With an Event Camera". In: *CVPR* (2019), pp. 6813–6822.

[130] Genady Paikin et al. "EFI-Net: Video Frame Interpolation from Fusion of Events and Frames". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2021), pp. 1291–1301.

[131] Beat Kueng et al. "Low-latency visual odometry using event-based feature tracks". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 16–23.

[132] Daniel Gehrig et al. "EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames". In: *International Journal of Computer Vision* 128 (2019), pp. 601–618.

[133] Zhuangyi Jiang et al. "Mixed Frame-/Event-Driven Fast Pedestrian Detection". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 8332–8338.

[134] Hu Cao et al. "Fusion-Based Feature Attention Gate Component for Vehicle Detection Based on Event Camera". In: *IEEE Sensors Journal* 21 (2021), pp. 24540–24548.

[135] Abhishek Tomy et al. "Fusing Event-based and RGB camera for Robust Object Detection in Adverse Conditions". In: *2022 International Conference on Robotics and Automation (ICRA)* (2022), pp. 933–939.

[136] Yuhuang Hu et al. "DDD20 End-to-End Event Camera Driving Dataset: Fusing Frames and Events with Deep Learning for Improved Steering Prediction". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (2020), pp. 1–6.

[137] Rihui Song et al. "Calibration of Event-based Camera and 3D LiDAR". In: *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)* (2018), pp. 289–295.

[138] Kevin Ta et al. "L2E: Lasers to Events for 6-DoF Extrinsic Calibration of Lidars and Event Cameras". In: *ArXiv* (2022).

[139] Jianhao Jiao et al. "LCE-Calib: Automatic LiDAR-Frame/Event Camera Extrinsic Calibration With A Globally Optimal Solution". In: *ArXiv* abs/2303.09825 (2023).

[140] Mario A. V. Saucedo et al. "Event Camera and LiDAR based Human Tracking for Adverse Lighting Conditions in Subterranean Environments". In: *ArXiv* abs/2304.08908 (2023).

[141] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CVPR* (2016), pp. 770–778.

[142] Mennatullah Siam et al. "Convolutional gated recurrent networks for video segmentation". In: *IEEE International Conference on Image Processing (ICIP)* (2017), pp. 3090–3094.

[143] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1026–1034.

[144] Xingang Pan et al. "Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net". In: *European Conference on Computer Vision*. 2018.

[145] Benjamin Ummenhofer et al. "DeMoN: Depth and Motion Network for Learning Monocular Stereo". In: *CVPR* (2017), pp. 5622–5631.

[146] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)* (2015).

[147] Alberto Sabater, Luis Montesano, and Ana Cristina Murillo. "Event Transformer+. A multi-purpose solution for efficient event data processing". In: *ArXiv* abs/2211.12222 (2022).

[148] Nico Messikommer et al. "Event-based Asynchronous Sparse Convolutional Networks". In: *European Conference on Computer Vision*. 2020.

[149] R. W. Baldwin et al. "Time-Ordered Recent Event (TORE) Volumes for Event Cameras". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2021), pp. 2519–2532.

[150] Ashish Vaswani et al. "Attention is All you Need". In: *Neural Information Processing Systems*. 2017.

[151] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. "Multi-Modal Fusion Transformer for End-to-End Autonomous Driving". In: *CVPR* (2021), pp. 7073–7083.

[152] Kashyap Chitta et al. "TransFuser: Imitation With Transformer-Based Sensor Fusion for Autonomous Driving". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2022), pp. 12878–12895.

[153] Elizbar Nadaraya. "On Estimating Regression". In: *Theory of Probability and Its Applications* 9 (1964), pp. 141–142.

[154] Geoffrey Stuart Watson. "Smooth regression analysis". In: *Sankhyā: The Indian Journal of Statistics*. Vol. 26. 4. 1964, pp. 359–372.

[155] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *International Conference on Learning Representations (ICLR)* (2014).

[156] Angelos Katharopoulos et al. "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention". In: *International Conference on Machine Learning*. 2020.

[157] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *International Conference on Learning Representations (ICLR)*. 2016.

[158] Uday Kamal, Saurabh Dash, and S. Mukhopadhyay. "Associative Memory Augmented Asynchronous Spatiotemporal Representation Learning for Event-based Perception". In: *International Conference on Learning Representations (ICLR)*. 2023.

[159] Sharan Narang et al. "Do Transformer Modifications Transfer Across Implementations and Applications?" In: *ArXiv* abs/2102.11972 (2021).

[160] Yi Tay et al. "Scaling Laws vs Model Architectures: How does Inductive Bias Influence Scaling?" In: *ArXiv* abs/2207.10551 (2022).

[161] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations (ICLR)*. 2021.

[162] Zhihao Li, M. Salman Asif, and Zhan Ma. "Event Transformer". In: *ArXiv* abs/2204.05172 (2022).

[163] Zuowen Wang, Yuhuang Hu, and Shih-Chii Liu. "Exploiting Spatial Sparsity for Event Cameras with Visual Transformers". In: *IEEE International Conference on Image Processing (ICIP)* (2022), pp. 411–415.

[164] Alberto Sabater, Luis Montesano, and Ana Cristina Murillo. "Event Transformer. A sparse-aware solution for efficient event data processing". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2022), pp. 2676–2685.

[165] Yi Tian and J. Andrade-Cetto. "Event Transformer FlowNet for optical flow estimation". In: *BMVC*. 2022.

[166] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. "Event-based Video Reconstruction Using Transformer". In: *ICCV* (2021), pp. 2543–2552.

[167] Stefano Chiavazza, Svea Marie Meyer, and Yulia Sandamirskaya. "Low-latency monocular depth estimation using event timing on neuromorphic hardware". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2023), pp. 4071–4080.

[168] Urbano Miguel Nunes, Laurent Udo Perrinet, and Sio-Hoi Ieng. "Time-to-Contact Map by Joint Estimation of Up-to-Scale Inverse Depth and Global Motion using a Single Event Camera". In: *ICCV* (2023).

[169] Hoonhee Cho, Jegyeong Cho, and Kuk-Jin Yoon. "Learning Adaptive Dense Event Stereo from the Image Domain". In: *CVPR* (2023), pp. 17797–17807.

[170] Suman Ghosh and Guillermo Gallego. "Multi-Event-Camera Depth Estimation and Outlier Rejection by Refocused Events Fusion". In: *Advanced Intelligent Systems* 4 (2022).

[171] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *ECCV*. 2020.

[172] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241.

[173] Anton Trusov et al. "Fast Implementation of 4-bit Convolutional Neural Networks for Mobile Devices". In: *25th International Conference on Pattern Recognition (ICPR)* (2020), pp. 9897–9903.

[174] Tim Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs". In: *ArXiv* abs/2305.14314 (2023).

[175] Amaro Da Costa Luz Carneiro. "Asynchronous Event-Based 3D vision". PhD thesis. Université Pierre et Marie Curie, 2014.

[176] Sio-Hoi Ieng, João Carneiro, and Ryad B. Benosman. "Event-Based 3D Motion Flow Estimation Using 4D Spatio Temporal Subspaces Properties". In: *Frontiers in Neuroscience* 10 (2017).

[177] Alexander Kirillov et al. "Segment Anything". In: *ArXiv* abs/2304.02643 (2023).