Asynchronous Stereo Vision

Event-based Stereo Matching and Tracking for Dynamic Vision Sensors

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der Technischen Wissenschaften

by

M.Sc. Ewa Alicja Piątkowska
Registration Number 1125736

to the Faculty of Informatics

at the TU Wien

Advisor: Ao. Univ. Prof. Dr. Margrit Gelautz

The dissertation has been reviewed by:

Ao. Univ. Prof. Dr. Josef Scharinger	Prof. Dr. Ing. Pavel Zemčík

Vienna, 10th October, 2018

Ewa Alicja Piątkowska

Erklärung zur Verfassung der Arbeit

M.Sc. Ewa Alicja Piątkowska
Strudlhofgasse 14/6, 1090 Wien

Wien, 10. Oktober 2018

Ewa Alicja Piątkowska

I dedicate this thesis to my wonderful parents whose love, patience, sacrifice, and continuous support made it possible for me to complete this thesis.

Moim wspaniałym rodzicom, za ich miłość, cierpliwość, poświęcenia i wsparcie, które pozwoliły mi uwierzyć w siebie i dokończyć tę pracę.

Acknowledgements

First of all, I would like to express sincere gratitude to my supervisor, Margrit Gelautz, who guided and supported me during my doctoral studies. I am very thankful for her insightful feedback and thorough review of this thesis. I am grateful to Jürgen Kogler for sharing his knowledge and experience in the field, and for his contribution to the evaluation section by providing the ground-truth dataset. I would also like to thank Josef Scharinger and Pavel Zemčík, who kindly agreed to be external reviewers of this thesis.

I would like to acknowledge the FEMtech project CHANGES, which partially funded this PhD research. In particular, I would like to thank the project coordinator, Helga Gartner, who supported me in numerous aspects and made the first steps in the new country easier.

I would like to express my appreciation to Nabil Belbachir and Heinrich Garn, who gave me the opportunity to work in the field of dynamic vision sensors at the AIT Austrian Institute of Technology. I would like to thank Stephan Schraml for inspiring discussions and sharing the software for event simulation, which was very useful in the evaluation of the algorithms. I would also like to thank Michael Hofstätter for his encouragement to finalise this PhD.

I am deeply grateful to all colleagues and friends that I had a pleasure to work with in both SNI and ICT-Security group at AIT. In particular, I would like to thank my dear friends: Helen Smith, Aneta Czetina (Nowakowska) and Georg Wiesmann, who were always understanding, supportive and eager to offer help, time or a kind word to cheer me up. Special thanks go to Paul Smith for his guidance, kindness and creating a wonderful working environment where I could grow as a researcher.

Last and most importantly, I would like to thank my family for supporting and motivating me along the way. I am deeply grateful to my beloved sister Weronika, who has been a great inspiration in my life and introduced me to the field of computer vision. I would also like to thank her and her husband Marcin for the best birthday present which, although meant for a different purpose, has allowed me to perform the most computationally heavy experiments. I would like to express my deepest gratitude to my wonderful parents, not only because they made it possible for me to get educated, but also in many ways supported every step in my life with encouragement and unconditional love.

Abstract

This thesis addresses the problem of stereo reconstruction from a stream of events provided by two dynamic vision sensors (DVS) in a stereo configuration. Dynamic vision sensors consist of self-spiking pixels that independently and in continuous time react to relative light intensity changes by generating ‘spikes’ encoded in Address Event Representation (AER). In result, the output of the sensor is not a sequence of frames as in conventional cameras, but an asynchronous stream of events indicating captured intensity changes. The main advantages of these types of sensors are high temporal resolution (better than $10\mu s$) and wide dynamic range ($> 120dB$). Several approaches for stereo matching have been introduced for dynamic vision sensors, including the application of conventional stereo algorithms by operating on ‘pseudo frames’ built from the address event stream (image-based methods). Although the image-based algorithms are acceptable in performance, they do not exploit the sensor’s specific capabilities. Only few efforts have been invested so far in stereo processing techniques that can be applied directly to the stream of events (event-based methods). These methods preserve the asynchronous aspect of events, thus are better suited for keeping the advantages of dynamic vision sensors. However, there are still various challenges to tackle in event-based stereo matching.

In this thesis, we investigate the feasibility of fully asynchronous stereo vision tailored for dynamic vision sensors. We start out with a thorough analysis of event data from dynamic vision sensors in the context of stereo analysis with a focus on the events’ coincidence in time. We find that single event-to-event matching with the use of timing information as a matching score lacks reliability while dealing with complex scenes and challenging conditions. As the main contribution of this thesis we propose an adaptive dynamic cooperative network, which is constantly updated while events are generated, making it feasible to preserve the data-driven aspect of the sensor. We develop two cooperative stereo matching algorithms with the first employing simple time-based event matching as an input to the cooperative network. In the second algorithm, we suggest using the spatio-temporal neighbourhood of the event as matching primitive and a novel similarity measure, which is a combination of time-based correlation and polarity. Extensive evaluation of the proposed cooperative stereo algorithms demonstrates that the results are comparable or better than competing algorithms in the field. Furthermore, we propose an asynchronous tracking method that is realised by clustering events in three-dimensional space with Gaussian mixture models and demonstrate its performance in conjunction with the cooperative stereo matching results.

Kurzfassung

Diese Arbeit befasst sich mit dem Problem der Stereo-Rekonstruktion aus Event-Daten, die von zwei Dynamic Vision Sensoren (DVS) in Stereo-Konfiguration geliefert werden. Dynamic Vision Sensoren bestehen aus Pixeln, die eigenständig und zeitkontinuierlich auf relative Änderungen in der Lichtintensität reagieren, indem sie sogenannte Spikes erzeugen, welche in Address Event Representation (AER) dargestellt werden. Dadurch ist die Ausgabe des Sensors keine Bildsequenz wie bei herkömmlichen Kameras, sondern ein asynchroner Strom von Events, welche die erfassten Intensitätsänderungen anzeigen. Die Hauptvorteile dieser Art von Sensoren sind die hohe zeitliche Auflösung (besser als $10\mu s$) und ein großer Dynamikbereich ($> 120dB$). Mehrere Ansätze für Stereo-Matching wurden für Dynamic Vision Sensoren vorgestellt, einschließlich der Anwendung von konventionellen Stereo-Algorithmen, welche auf sogenannten Pseudoframes arbeiten, die aus dem Strom der Address Events aufgebaut werden (bildbasierte Methoden). Obwohl die Leistung der bildbasierten Algorithmen akzeptabel ist, nützen sie die speziellen Fähigkeiten des Sensors nicht aus. Bisher wurden nur wenige Versuche unternommen Stereo-Verarbeitungstechniken zu entwickeln, welche direkt auf die Event-Ströme angewendet werden können (Event-basierte Methoden). Diese Methoden bewahren den asynchronen Aspekt der Events und sind daher besser geeignet, die Vorteile von Dynamic Vision Sensoren zu erhalten. Allerdings gilt es noch verschiedene Herausforderungen beim Event-basierten Stereo-Matching zu bewältigen.

In dieser Dissertation untersuchen wir die Machbarkeit von vollständig asynchroner Stereo Vision, die auf Dynamic Vision Sensoren zugeschnitten ist. Wir beginnen mit einer ausführlichen Analyse der Event Daten von Dynamic Vision Sensoren im Kontext der Stereoanalyse mit einem Fokus auf die zeitliche Koinzidenz der Ereignisse. Es zeigt sich, dass einzelnes Event-zu-Event Matching unter Verwendung der Zeitinformation als Matching-Maß nicht zuverlässig ist, wenn man mit komplexen Szenen und herausfordernden Bedingungen konfrontiert ist. Als wichtigsten Beitrag dieser Arbeit schlagen wir ein adaptives dynamisches kooperatives Netzwerk vor, welches während der Generierung von Events ständig aktualisiert wird, wodurch der datengetriebene Aspekt des Sensors erhalten bleibt. Wir entwickeln zwei kooperative Stereo-Matching Algorithmen, wobei der erste Algorithmus einfaches zeitbasiertes Event-Matching als Input für das kooperative Netzwerk verwendet. Im zweiten Algorithmus schlagen wir die Verwendung einer räumlich-zeitlichen Nachbarschaft des Events als Matching-Basis und ein neuartiges Ähnlichkeitsmerkmal, welches zeitabhängige Korrelation und Polarität kombiniert, vor.

Umfangreiche Auswertung der vorgestellten kooperativen Stereo-Algorithmen zeigt, dass die Ergebnisse vergleichbar oder besser als konkurrierende Algorithmen auf dem Gebiet sind. Darüber hinaus schlagen wir eine asynchrone Tracking-Methode vor, welche durch Event-Clustering im dreidimensionalen Raum mit Hilfe von Gaußschen Mischmodellen realisiert wird, und demonstrieren ihre Leistungsfähigkeit in Verbindung mit den kooperativen Stereo-Matching-Ergebnissen.

Contents

CHAPTER 1

Introduction

1.1 Background

The performance of the biological nervous system in terms of processing visual information is outstanding. Considering aspects such as efficiency, power consumption and size, computers are by no means a competition to the abilities of the human brain. The mainstream, conventional vision systems invest efforts in highly sophisticated computational models. However, computers in their current form cannot achieve the performance of the human brain, especially regarding tasks such as vision, hearing and learning [99, 14]. The research field of bio-inspired vision explores ways of improving conventional image based computer vision, by trying to implement in hardware the mechanisms observed in biological retina, e.g. features extraction. Recently, bio-inspired Dynamic Vision Sensors (DVSs) [83, 115] have gained a lot of recognition, mainly due to their advantages such as high-temporal resolution, wide dynamic range and low power consumption. These characteristics make them suitable for new trends in robotics, such as high-speed motion analysis and tracking or autonomous navigation in uncontrolled environments.

One of the fundamental abilities of the human biological vision system is three-dimensional perception of the surrounding world. The ability to perceive depth is explained by the fact of having two eyes, i.e. *binocular (stereo) vision*. Objects in the scene, when observed from two viewpoints, are projected on each of the two views at slightly different positions. From such displacement, in the stereo literature called *disparity*, the depth information can be inferred. Stereo vision is a well-researched field in conventional computer vision and a vast amount of work has been proposed to calculate the disparity from two images. However, dynamic vision sensors differ from conventional digital cameras in their construction with respect to chip architecture and functionality. The sensors consist of self-spiking pixels, which react to relative contrast by firing an event upon a detected change in light intensity.

In Figure 1.1, a schematic overview of the depth estimation procedure is presented. The stereo system consists of two dynamic vision sensors, which generate events upon activity in the scene, in this case two people walking in front of the sensor. We can observe that the position of the left projection of each person is different from the right one. Knowing the correspondence between the events (symbolically depicted as arrows, differentiated with colour by object), and hence the magnitude of an object's disparity, allows for depth calculation.

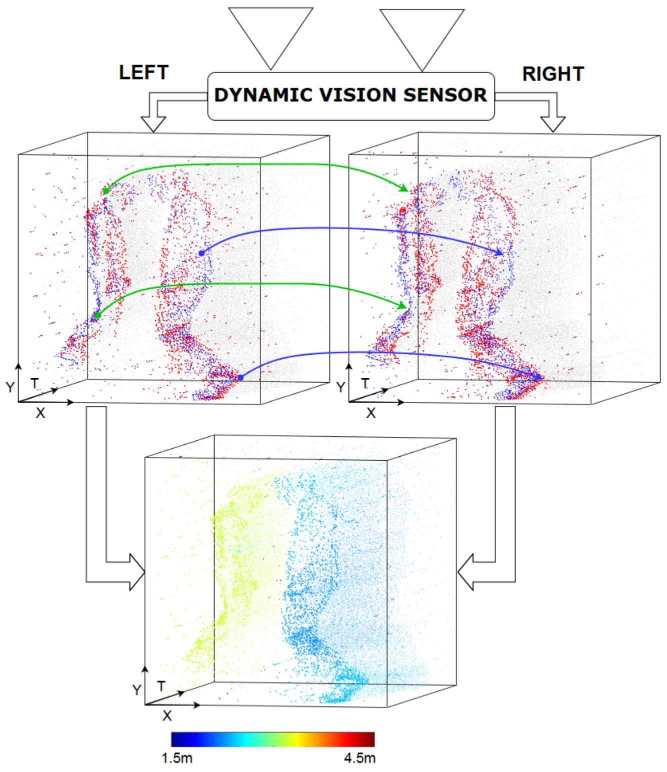


Figure 1.1: Schematic description of the stereo matching task. The motion of two walking people captured by a stereo DVS system is represented by a spatio-temporal stream of events. Correspondence between the left and right view is established to calculate the depth information, encoded in colour.

While dynamic vision sensors offer numerous advantages, they also introduce several challenges to the way the data has to be processed afterwards. Most of the well-established computer vision algorithms operate on pixels' values within images, therefore they cannot be directly applied to the stream of events. In this thesis, we investigate methods for efficient stereo matching tailored to the nature of event-based vision. We focus on preserving the asynchronous aspect of the data processing to fully exploit the capabilities of the sensor, especially in terms of temporal resolution.

1.2 Contributions

The main objective of our research is to provide an asynchronous approach to stereo matching of event-data from dynamic vision sensors. The research questions and contributions of this PhD thesis can be summarized as follows.

What challenges do dynamic vision sensors introduce to the task of stereo matching? There are two major differences between image-based and event-based vision, the first one is connected with the dimensionality of the input data, i.e. spatiotemporal data, and the second one is associated with the visual representation of a captured scene, i.e. sparse data. We analyse how these differences affect the formulation of the stereo matching problem. We demonstrate how asynchronous stereo matching helps to preserve the temporal resolution and depth accuracy regardless of the speed of objects moving in the scene. We show that asynchronous processing cannot assume any fixed time aggregation of the events, hence stereo matching should be performed on an event basis. We define a set of requirements for an efficient event-based stereo algorithm.

Is it possible to find correspondences between single events within asynchronous data streams and what are the constraints? Operating directly on the stream of events is important to keep the processing asynchronous. Most of the approaches use the events' coincidence in time to establish correspondence. By theoretical and experimental data analysis, we investigate how the representation of the same object varies across different sensors, and show that using only time and polarity as matching constraints might not be sufficient to handle more complex, dynamic scenes.

What could be an appropriate model for asynchronous stereo matching? Event-based matching should include additional constraints to improve the matching accuracy, i.e. smoothness constraint. Building on early studies of stereo computation [93], we propose a novel approach, the adaptive dynamic cooperative network, as a possible model for asynchronous stereo matching. The cooperative network is constantly updated while events are generated, making it feasible to preserve the data-driven aspect of the sensor. The network uses cooperative and inhibitory mechanisms to optimise the results of time-based event matching.

How could the reliability of event-based matching be improved? Matching events by their timings is prone to errors, thus we investigate ways of improving the accuracy of event-based matching. We show that by using an event's context, the ambiguity in correspondence finding can be reduced. We suggest using a spatiotemporal neighbourhood of the event as a matching primitive. The matching score is a combination of time-based correlation and polarity. These define a novel matching function of our second cooperative stereo matching algorithm.

Can asynchronous stereo matching achieve an accuracy comparable to image-based approaches applied to event data? We provide an extensive evaluation of our cooperative stereo algorithms. Using a ground-truth dataset, we show that asynchronous stereo can achieve comparable or better results than competing image-based algorithms in the field.

In addition, we evaluate the robustness of the cooperative algorithms to changing scene appearance, complexity and dynamics. Although we have designed our algorithms assuming a static sensor setup, we further demonstrate their applicability to data sequences captured by a moving DVS observing a stationary scene.

What could be a feasible approach for asynchronous multiple object tracking? When the DVS is in a static position, only scene dynamics is captured and therefore data provided by the sensor represent the trajectories of moving objects. While tracking a single object can be considered rather trivial, dealing with multiple objects occluding each other makes the task of tracking a challenging problem. We propose a novel asynchronous tracking algorithm realised by clustering events with Gaussian mixture models. The algorithm addresses the challenge of multiple objects tracking in the presence of occlusions. The occlusions are tackled by clustering events in three-dimensional space, using as input the spatial location of events and depth estimates, calculated by asynchronous stereo matching algorithms. We evaluate the suitability of both cooperative stereo matching algorithms to the task of tracking.

1.3 Resulting Publications

The work presented in this thesis resulted in the following publications:

- Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz: *Spatiotemporal Multiple Persons Tracking Using Dynamic Vision Sensor*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 35–40, 2012.
- Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz: *Asynchronous Stereo Vision for Event-Driven Dynamic Stereo Sensor Using an Adaptive Cooperative Approach*, IEEE International Conference on Computer Vision (ICCV) Workshops, 45–50, 2013.
- Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz: *Cooperative and Asynchronous Stereo Vision for Dynamic Vision Sensors*, Measurement Science and Technology, Volume 25, Number 5, pp. 1-8, 2014.
- Ewa Piatkowska, Jürgen Kogler, Ahmed Nabil Belbachir, and Margrit Gelautz: *Improved Cooperative Stereo Matching for Dynamic Vision Sensors with Ground Truth Evaluation*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 370–377, 2017.

1.4 Thesis Organisation

The subsequent chapters of this thesis are structured as follows: Chapter 2 provides background information about neuromorphic vision, including characteristics of dynamic vision sensors and event-based visual processing. Chapter 3 briefly describes the fundamentals of stereo vision, and presents the state-of-the-art in the field of depth estimation using dynamic vision sensors. Chapter 4 analyses two main challenges of asynchronous stereo vision, which are related to the aspect of event-based matching and asynchronous processing. In Chapter 5, the dynamic and adaptive cooperative network is presented as a feasible method for asynchronous event-based stereo matching and two cooperative stereo matching algorithms are described. An extensive evaluation of both algorithms and their comparative analysis are presented in Chapter 6. Furthermore, the suitability of the proposed algorithms is evaluated in higher-level computer vision tasks, such as multiple object tracking, which is described in Chapter 7. Chapter 8 summarises the major outcomes of our research and provides an outlook on future directions in the field.

CHAPTER 2

Neuromorphic Vision Sensors

Dynamic vision sensors (DVSs) stem from the field of neuromorphic vision, which focuses on the design of sensors that replicate the physical architecture and functionality of biological vision systems. In this chapter, we shortly introduce the goal and means of neuromorphic engineering, and describe main efforts made in replicating the retina. Next, we describe the architecture of dynamic vision sensors in more detail and present the specifics of event-based vision, including its benefits and application areas.

2.1 Neuromorphic Vision

Neuromorphic vision is a subdomain of neuromorphic engineering, a research discipline introduced by Carver Mead in the late 1980s. Neuromorphic engineering focuses on the emulation of neural architectures and functionality of the biological nervous system in the electronic devices [99]. The functional implementation is commonly made in silicon using analog Very Large Scale Integration (aVLSI) [82]. The VLSI technology is used due to the similarities to neural systems [15]. Both systems operate on millions of inexpensive, failure-prone elements and since neurons use electrical signalling, biological processing can be simulated by electronic circuits.

The human retina achieves a wide dynamic range of approx. 200dB (while conventional cameras only 70dB) and high spatial resolution reaching the size of 2.6 μm in the fovea (comparable with cameras resolution) [83]. Furthermore, humans can perform such complex visual tasks as image identification in around 100-150ms of processing [149]. Motivated by the remarkable performance of biological vision systems, engineers have investigated how a neuromorphic approach can help overcoming the limitations of traditional cameras. Furthermore, the sensory systems are well documented by anatomists and, thus, easier to 'morph' in silicon [14]. In the following section, we will briefly describe the human retina and discuss the key features that are assumed to play an important role in its efficiency.

2.1.1 Biological Retina

The design of the retina is highly connected with its functionality. As depicted in Figure 2.1, the retina has a layered structure. The processing starts at the back, in the photoreceptors layer. In the human vision system, we can find at least two types of photoreceptors, namely rods and cones. The former are used for low-light vision, whereas the latter are adjusted for bright-coloured, daylight vision. The photoreceptors

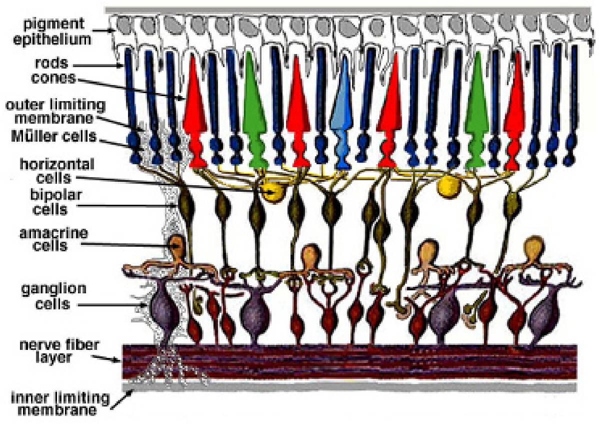


Figure 2.1: The layout of the human retina [74].

are specialised in sensing particular parts of the spectrum, which plays an important role in achieving a wide dynamic range. In addition, the human eye eliminates the information about the absolute light illumination, leaving only the relative local ratio that describes the scene regardless of the light conditions [83]. The key to the efficiency of the retina is the parallel computation in networks of cells, starting from photoreceptors through the layers of bipolar, horizontal, amacrine and ganglion cells. These cells are designed to filter out redundant information and detect the most prominent features to be conveyed to the brain for further processing. Except from the fovea, there is only one optic nerve fibre for each 100 photoreceptors, therefore processing in the retina could be regarded as a very sophisticated compression of the visual scene [149]. The contrast detection is realised by receptive fields of ganglion cells, which are sensitive either to the light areas on darker background (ON) or dark areas on lighter background (OFF). Axons of ganglion cells bundled together into an optic nerve are passed to the Lateral Geniculate Nuclei (LGN), which is structured into two pathways: *magnocellular* and *parvocellular* path. The former provides fast, transient response and focuses on object boundaries and detection to form a basis for depth and motion perception. The latter pathway has sustained response and is oriented on colour and fine details of the objects' appearance. The signals along these two pathways are passed further to the primary visual cortex for higher level visual processing [149]. Such functional division into two pathways captures an important aspect of biological systems efficiency, i.e. the processing is done in parallel over the networks specialised in particular tasks [82].

2.1.2 Silicon Retina

The term *silicon retina* has been introduced in the context of the seminal work of Mahowald and Mead [90]. The proposed vision sensor implements 'spiking' pixels, which are sensitive to illumination contrast. The sensor was a demonstration of concept, however due to performance limitations, it was not usable in real-world conditions.

Different aspects of biological retina have been investigated in the design of newer generation silicon retinas. We can find implementations of the functionality of ganglion cells and LGN pathways to perform contrast detection, including sustained cells for spatial contrast (edge detection) [3, 123] and transient cells for temporal contrast (motion detection) [92, 76, 83]. We can also observe a trend towards replication of both LGN pathways in order to combine a fast response for motion and change detection with more detailed spatial appearance features. Some approaches [167, 168, 115] implement these functionalities on the pixel level, whereas more recent works [20, 80] use separate pixels for dynamic and colour vision. Another group of sensors [64, 123] proposed the emulation of horizontal cells to enable the detection of oriented movement. Different ways of encoding visual information have been proposed, e.g. the order of events could denote the contrast magnitude [3], or absolute pixel illumination, e.g. Time-To-First-Spike imager [116] and Time-Based image sensor [87]. Some experiments have also been proposed to include the *fovea* in the design of silicon retina [2], by using different architectures for central pixels than for the peripheral ones.

Asynchronous Communication

Apart from the emulation of the retina's functionality in terms of feature extraction, the neuromorphic vision systems intend to replicate the communication schemes similar to those between the neurons. This task has shown to be particularly challenging in two aspects, namely connectivity and synchronisation. There is a huge amount of wiring between the neurons and the number of inputs and outputs in neural systems is around several thousands, whereas in Very Large Scale Integration (VLSI) circuits significantly less (around 10) [14]. Due to space limitations, it is impossible to directly implement the neural connectivity pattern in VLSI. The synchronisation issues in VLSI are solved by an external clock. However, the sequential readout controlled by the clock is not applicable for the design of neuromorphic chips because the communication in biology is driven by the occurrence of events. The closest solution to tackle the connectivity and synchronization issues is an asynchronous digital multiplexing technique that uses Address Event Representation (AER) to encode events ('spikes') [90, 78, 38].

The communication scheme between two chips is depicted in Figure 2.2. The pixels are self-timed and generate a temporal sequence of events upon visual stimuli. Whenever the pixel spikes, its address is communicated through a multiplexing circuitry to the receiver chip, where it is decoded into a sequence of events at corresponding locations. Such communication scheme preserves detailed timing of events and an address event protocol allows for arbitrary connectivity patterns.

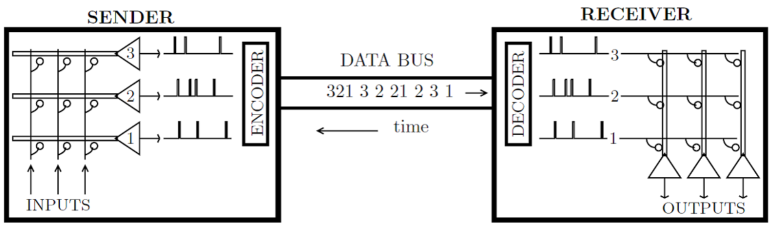


Figure 2.2: Address event representation [90].

2.2 Dynamic Vision Sensors

In the literature, various names have been used to denote a vision sensor that is sensitive to temporal contrast such as transient vision sensor, temporal contrast sensor, silicon retina, event-based vision sensor, dynamic vision sensor or, more recently, event camera. Throughout this work, we mostly use the term dynamic vision sensor (DVS). In the context of our work, two types of devices have been used for data acquisition, both developed at the AIT Austrian Institute of Technology (see Figure 2.3). The first one is the Universal COunting Sensor (UCOS) [125, 134], which is a stereo system consisting of two 128×128 DVS sensors chips [82]. The second device, the CARE system [6], consists of two Asynchronous Time-based Image Sensor (ATIS) sensor chips of 304×240 spatial resolution [115]. The chips of both stereo systems are fabricated in the Complementary Metal Oxide Semiconductor (CMOS) process.

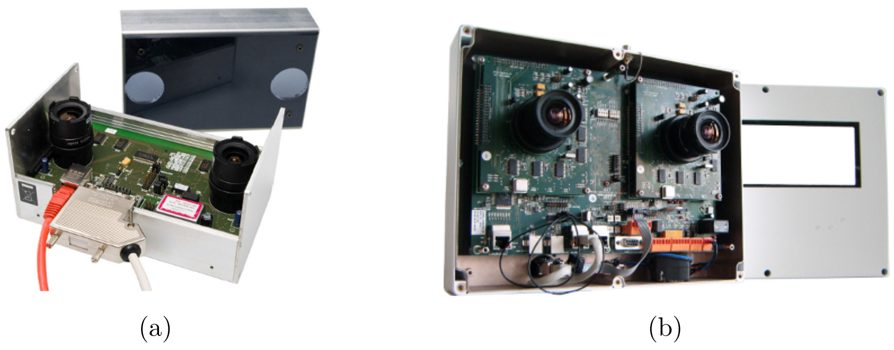
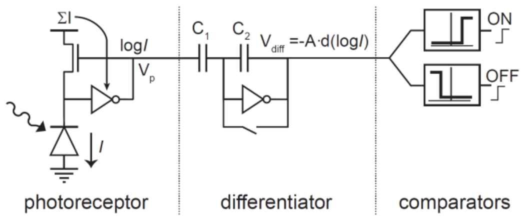


Figure 2.3: Stereo sensors used for data acquisition: (a) UCOS (image from [119]) and (b) CARE.

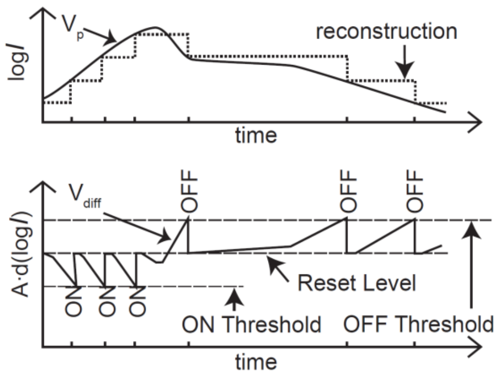
DVS Pixel Design

Dynamic vision sensors are an efficient implementation of transient (*magnocellular*) pathway. Based on the observation that measuring the magnitude of irradiance and detecting its change over time may result in lower dynamic range and higher latency of response [83], the DVS pixels are designed to handle only the temporal visual information.

The design of the DVS pixel is outlined in Figure 2.4a. The pixel's integrated circuit consists of three parts: photoreceptor, differentiator and comparators.



(a)



(b)

Figure 2.4: DVS pixel design: (a) abstract pixel core schematic, (b) principle of operation [82].

The photoreceptor circuit controls the individual pixel gain and responds to changes in illumination. The photocurrent I from the photodiode is proportional to the pixel illumination. It is converted logarithmically into voltage V_p and passed to the differentiator. The logarithmic response encodes the relative illumination (contrast). Analogically to the biological system, the relative instead of absolute values are used, which improves the wide dynamic range of the pixel.

The differentiator amplifies the changes in the photoreceptor output V_p resulting in V_{diff}, defined in (2.1).

$$\Delta V_{diff} = \frac{C_1}{C_2} \cdot \Delta V_p \equiv A \Delta V_p \tag{2.1}$$

The closed-loop gain of the differencing circuit is denoted as A. The polarity of the change is detected by discrimination between positive and negative irradiance fluctuations. If the voltage V_{diff} exceeds the ON or OFF threshold, respectively, the ON or OFF comparator switches and issues an event. After the event is generated, the differentiator is balanced to a reset level. Like in biological neurons, there is a refractory period in which the reset switch is held closed, preventing the pixel from generating another event. In Figure 2.4b the principle of a pixel's operation is illustrated. The output of the photoreceptor V_p as a function of time is shown in the top diagram. Below, the corresponding output of the differencing circuit V_{diff} is depicted with ON and OFF events marked.

Figure 2.5 depicts a schematic overview of a stereo DVS system, comprising two arrays of DVS pixels. As discussed above, each DVS pixel acts independently and in continuous time. Whenever the change in illumination is detected, the pixel 'spikes', i.e. it generates an event which is communicated off-chip using the Address Event Representation (AER).

The AER communication scheme is arbitrated, i.e. the pixel requests to access the shared bus are queued. The generated events are communicated to the peripheral Address Event (AE) circuits using a four-phase AE handshaking with the asynchronous row and column arbiters. The address of the pixel is communicated in parallel. The communication

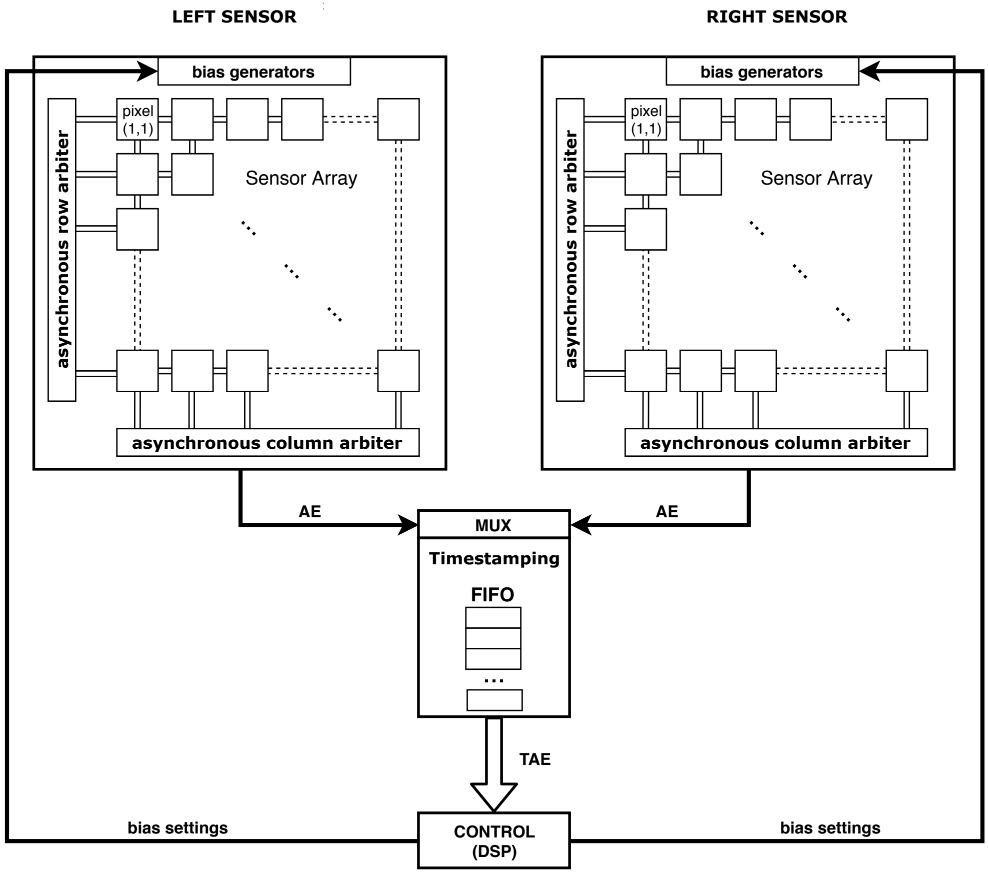


Figure 2.5: Schematic architecture of a stereo DVS system.

ensures the non-greedy arbitration in which row (column) can be serviced again only after requests from all other rows (columns) are serviced. Events from the left and right sensors are passed through the multiplexer unit (MUX) and then sent over a first-in-first-out (FIFO) buffer where they are timestamped.

Furthermore, a DVS is equipped with an integrated programmable bias generator. This allows configuring the internal pixel parameters (bias settings) to the desired functionality of the sensor (e.g. high speed motion analysis) and according to environmental conditions (e.g. night time or daytime vision). The most commonly used settings for the performance adjustment are biases that control the bandwidth of the photoreceptor or follower circuit, the contrast sensitivity thresholds and refractory period.

2.3 Event-based Vision

Dynamic vision sensors provide data in the Address Event Representation (AER), i.e. each event $e \in E$ is defined by a tuple, as shown in Equation 2.2:

$$\begin{aligned} e = (e_x, e_y, e_t, e_p, e_c) \mid e_x, e_y \in \mathbb{N}, e_t \in \mathbb{R}, \\ e_p, e_c \in \{0, 1\} \end{aligned} \tag{2.2}$$

where (e_x, e_y) is the pixel address, e_t is the event timestamp (time of occurrence), e_p is the polarity (ON or OFF), and in the case of a stereo configuration, e_c denotes the source of the event (the left or right sensor).

In Figure 2.6, a stream of events generated by one DVS upon a single edge motion (rotation) is shown. The events are visualised in space and time. The trajectory of the rotational motion is clearly visible in the data.

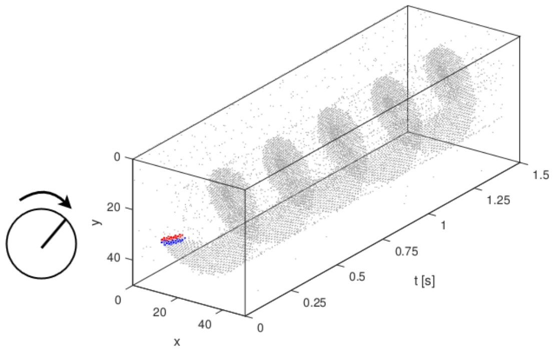


Figure 2.6: Sequence depicting a moving edge represented in space and time.

For processing and visualising purposes, the stream of events can be transformed into image-like representation, so-called ‘pseudo-frames’. The value assigned to each pixel of a ‘pseudo-frame’ is the accumulated sum of the events generated by the pixel over a period of time. Some methods consider aggregation with respect to the polarity of events. In Figure 2.7 a visual data sequence (person walking in front of the sensor), recorded by a conventional camera, is compared with the image representation of the event data. As we can observe, only edges of the moving person are captured. The polarity of events is encoded as follows: white for ON events; black for OFF. Another way to treat events is to consider them in their original dimensionality, i.e. space and time. In Figure 2.8 a sequence of events is plotted as three-dimensional space-time cloud. For visualisation purposes, the events of corresponding frames from Figure 2.7 are highlighted. The polarity of events is depicted in colour with blue and red for OFF and ON events, respectively. The cloud of events forms a trajectory of the moving person.

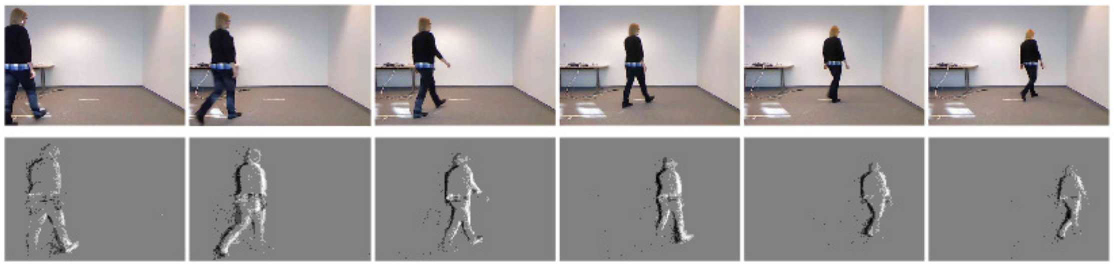


Figure 2.7: Motion of a person walking in a room, captured by a conventional camera (top row) and DVS (bottom row). The events are displayed in image-like representation.

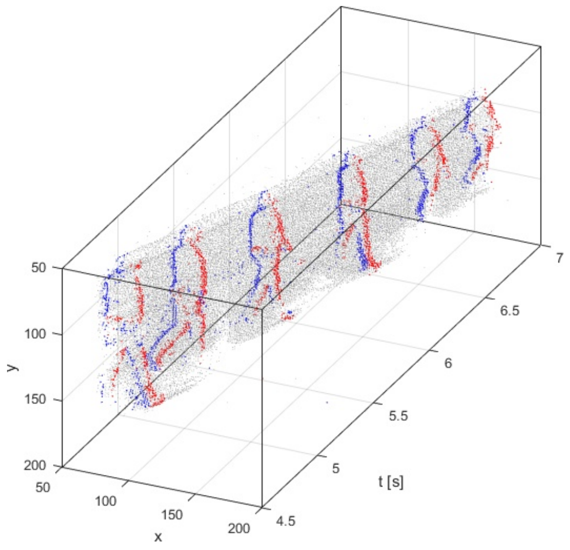


Figure 2.8: Sequence depicting moving person (the same as above) represented in space and time.

2.3.1 Advantages of Dynamic Vision Sensors

Differences between a conventional digital camera and dynamic vision sensor output are significant and, due to them, dynamic vision sensors benefit from temporal redundancy reduction, high temporal resolution, and wide dynamic range.

Data volume Conventional imagers capture a visual scene at a predetermined frame rate, reading the information from all pixels even if their values have not changed since the last frame. This leads to a high redundancy of the data and consumes unnecessary power, as well as time and resources of acquisition and data processing. Considering video content captured by a stationary camera, the most informative regions typically correspond to movements (foreground) while the background, in most cases static, tends to be highly redundant. In contrary, assuming a static DVS setup (excluding ego motion) only the scene dynamics are captured by the sensor. As shown in Figure 2.7, the events are only generated upon changes in the scene, only the foreground is captured. The data volume is reduced even further because the pixels are sensitive to the temporal contrast, which results in detecting only moving edges. The background is neglected. Moreover, temporal redundancy in DVS is handled on-chip since pixels do not react to static content, and in contrary to frame differencing techniques, the temporal resolution is not limited to the frame rate.

Temporal resolution Another advantage of dynamic vision sensors is the high temporal resolution achieved by the asynchronous data generation. Unlike clocked cameras sampling the scene at a fixed frame rate, the DVS pixels operate in continuous time. The fast response of the sensor makes it suitable for high-speed motion analysis. The temporal resolution can reach the equivalence of 100kfps (at illumination of more than 100lx, timestamp resolution 500ns) [115]. As illustrated in Figure 2.9, the fast motion of a

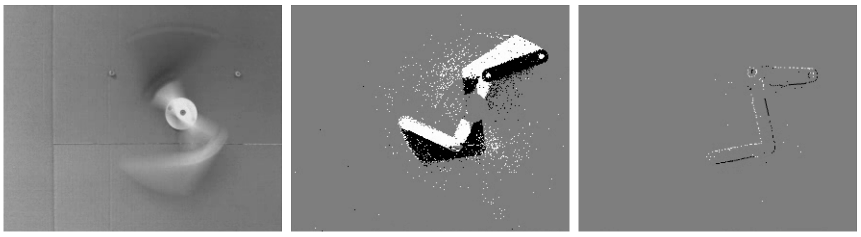

(a) (b) (c)

Figure 2.9: Pendulum moving at high speed represented by a frame from conventional camera (a) and DVS events accumulated over 17ms (b) and 0.5ms (c).

moving pendulum is recorded by an ATIS sensor and a digital camera under the scene illumination of 100lux[1]. In Figure 2.9a a frame captured by the digital camera at 30fps (exposure time of 33ms) is shown. For comparison, corresponding DVS data are visualised as images, where events are accumulated over (b) 17ms (60fps) and (c) 0.5ms (2000fps). Due to the high speed of the pendulum we can observe motion blur in Figure 2.9a-2.9b, but not in Figure 2.9c where the accumulation period is shorter.

Dynamic range Dynamic vision sensors have a wide dynamic range due to the design of the pixel and its sensitivity to temporal contrast rather than absolute illumination. As mentioned before, dynamic vision sensors can reach the value of 240dB while standard digital cameras typically around 70dB. This ability of the sensor is illustrated in Figure 2.10, where the output of the dynamic vision sensor is roughly the same regardless of the illumination of the scene. In Figure 2.10a-2.10b a scene is captured under normal illumination (approx. 100lx), whereas in Figure 2.10c-2.10d the same scene is captured under strong light (1000lx). We can observe that the output of the DVS is essentially the same in terms of data quality and appearance, while the standard camera cannot cope with the high illumination; a part of the image is saturated and the object is not visible.

[1]Images in Figure 2.9-2.10 are from the ATIS demonstration, AIT Austrian Institute of Technology

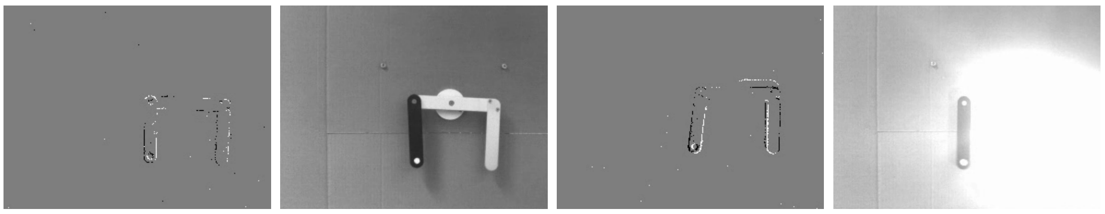


Figure 2.10: Comparison of DVS (a,c) and digital camera (b,d) with respect to scene captured at normal, ambient light (a,b) and strong light (c,d).

2.3.2 Application Areas

The advantages of dynamic vision sensors can be exploited in numerous application domains. Depending on the sensor setup, we can distinguish two types of applications: (1) monitoring and surveillance, assuming that the sensor is static, and (2) navigation and control, if the sensor is moving. In the former, motion analysis is naturally supported by the sensor due to the redundancy suppression and prominent feature selection done by pixels (edge extraction). The generated output from the sensor denotes the scene dynamics. Among possible applications we can find gesture recognition [79], object recognition [138], people counting [134], tracking [133, 112, 101], including fall detection [6], and traffic control [84]. Using DVS for surveillance applications has the additional advantage of being more privacy-preserving than digital cameras. Due to the limited spatial resolution of the sensor, the reconstruction of prominent facial features of the tracked people is close to impossible. Dynamic vision sensors have also been used in other fields, such as microscopy [105], high-speed tracking [77, 105] or space imaging for space situational awareness [33]. The second type of applications focuses mostly on robotic vision. Aspects such as efficiency and low power consumption are especially demanded in robotic vision, and we can find several works being proposed for navigation and control [39, 36]. The most popularity, however, dynamic vision sensors have achieved once applied to Unmanned Aerial Vehicle (UAV) navigation [103, 68], where conventional cameras are rather problematic, due to the motion blur and sensitivity to light condition changes. We can find methods for visual odometry [104, 174], pose estimation [52], and Simultaneous Localisation and Mapping (SLAM) [156]. Moreover, Dual Line Sensors (DLSs) [114] have been applied in industrial machine vision applications, e.g. for quality control, when objects moving at very high speed are being 'scanned' by the line sensor. In addition, DLSs have also been successfully applied for panoramic scene acquisition [7] and 3D panoramic reconstruction from events generated by each line of the sensor, as presented in [131]. Panoramic vision has also been achieved via event-based mosaicking [67] generated from scenes captured by moving DVS.

2.4 Summary

Advances in neuromorphic vision allow overcoming limitations of conventional digital cameras, such as redundancy, speed and limited dynamic range. Dynamic vision sensors have become a promising alternative to cameras whenever the speed of response, efficiency and robustness to scene illumination are required. Recently, a lot of effort is being invested in event-based vision algorithms to exploit the capabilities of the dynamic vision sensors. Current trends in data processing, including artificial deep neural networks, are encouraging and indicate that neuromorphic models in both software and hardware are expected to get more and more attention in the future.

CHAPTER 3

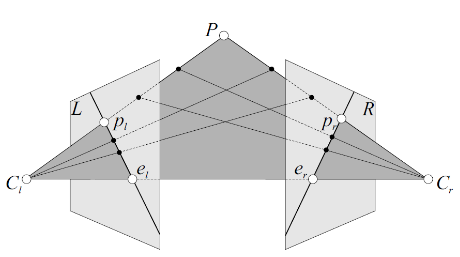
Stereo Vision and Related Work

Depth perception is essential in many vision tasks, e.g. navigation, localisation and manipulation. One of the most established methods for depth estimation is stereo vision. In this section, we briefly describe the fundamentals of stereo vision and give an overview of conventional approaches to solving the stereo correspondence problem. Afterwards, we present the state-of-the-art research in the field of depth estimation using dynamic vision sensors.

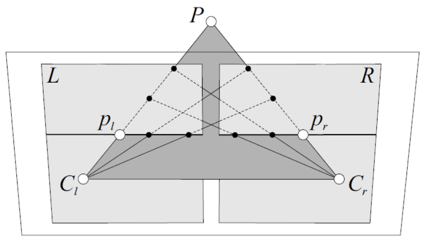
3.1 Stereo Vision

The concept of stereo vision is inspired by biology following the observation that depth perception is achieved by fusing the visual information captured by a pair of eyes. Stereo vision, also called binocular vision, assumes that two cameras are located close to each other to provide sufficient overlap between the views. The position of a 3D scene point projected onto the left view is slightly different from that in the right view, due to the difference in viewpoints. The displacement (disparity) of an object across the views is inversely proportional to its distance to the cameras. The closer the object to the camera, the bigger is the disparity between its projections.

The geometrical interpretation of the depth computation from two images of the same scene relies on *epipolar geometry*. As shown in Figure 3.1a, the three-dimensional point P is projected onto the left image as two-dimensional point p_l. Having only one projection of point P it is not possible to determine the Z coordinate, since P can lie anywhere on the projection ray connecting the left camera centre C_l and point P. Therefore, the second projection p_r of point P is required, since the intersection of the projection rays from the left and right view determines uniquely the position of the point P. The task of stereo computation is to find corresponding projections of points in the scene and calculate the depth based on their positions in the left and right view. The plane connecting both camera centers (C_l, C_r) and the projected point P is called *epipolar*



(a)



(b)

Figure 3.1: Stereo geometry before (a) and after (b) the epipolar rectification [12].

plane. The intersections of the *epipolar plane* with the left and right image planes are known as *epipolar lines*. The projections of the opposite camera centre, i.e. *epipoles*, are denoted as e_l and e_r for the left and right projection, respectively. The corresponding projections of the point P are known to lie on respective *epipolar lines*. This allows for reducing the correspondence search space to one dimension. In order to restrict the search space to horizontal lines, an *epipolar rectification* is performed. The left and right camera image planes are projected onto one plane, so that the corresponding *epipolar lines* are parallel to the baseline (see Figure 3.1b). After the rectification, corresponding points have the same vertical coordinate ($y_l = y_r$) and the disparity is expressed as the difference in horizontal coordinate $d = |x_l - x_r|$.

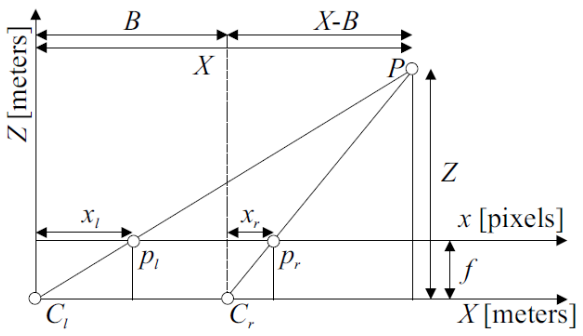


Figure 3.2: Depth reconstruction by triangulation [12].

The depth (Z coordinate) of the point P can be calculated via triangulation knowing the position of its projections onto the left and right view and the parameters of the stereo system, specifically, the baseline B, i.e. the distance between the two camera centres, and the focal length f, i.e. the distance between the cameras' focal points and their optical centres. The relation between depth and disparity is shown in Figure 3.2. Following the rule of similar triangles, the depth Z can be calculated as presented in Equation 3.1.

$$\frac{x_l}{f} = \frac{X}{Z} \; and \; \frac{x_r}{f} = \frac{X - B}{Z} \Rightarrow Z = \frac{f \cdot B}{|x_l - x_r|} = \frac{f \cdot B}{d} \tag{3.1}$$

The 3D reconstruction with known disparities and camera parameters is, in principle, a straightforward calculation. The most challenging part in stereo computation, however, is finding correct correspondences across the views.

Correspondence Search

Solving the correspondence problem is based on finding matching points in two or more views of the same scene. Specifically, for each point from the reference view we search for the best matching candidate in the opposite view. The quality of the matching candidate at position (x, y) for disparity d is defined by the matching cost C as follows:

$$C(x, y, d) = \rho(\tau(x, y), \tau(x - d, y)) \,, \tag{3.2}$$

where ρ is the (dis-) similarity measure (e.g. sum of absolute differences), which takes as argument two matching primitives τ (e.g. pixel, patch, edge). The correct disparity is selected out of all possible disparities in range D, as the one with the minimal matching cost:

$$d(x, y) = \operatorname*{argmin}_{d \in D} C(x, y, d) \tag{3.3}$$

Finding correspondences between two images is a common task in computer vision for locating similar regions across a collection of images. Apart from stereo vision, the task of correspondence search is present also in motion estimation or tracking. In the context of stereo vision, the correspondence search is referred to as *stereo matching.*

3.2 Conventional Approaches to Stereo Matching

A variety of different approaches have been proposed to perform stereo matching in conventional, image-based computer vision. In this section, a general outline of the main trends in stereo vision is given. A systematic overview of stereo matching algorithms can be found in [145, 129, 26].

Sparse and Dense Stereo Correspondence

In general, stereo matching methods can be divided into sparse and dense approaches. The former perform matching on previously extracted prominent features of the scene, whereas in the latter every pixel is considered in the stereo matching. Sparse stereo was more common in early stereo vision approaches, e.g. [93, 56, 98], mainly due to limited computational resources. Later, sparse correspondences were applied in surface fitting algorithms [137, 136] or as seeds to grow additional matches within dense stereo methods [171]. Dense stereo has gained more attention due to the demand for image-based rendering and 3D modelling. Dense disparity maps are more informative than just a partial (sparse) scene reconstruction.

In their taxonomy of binocular dense stereo correspondence, Scharstein and Szeliski [129] observe that stereo matching algorithms usually can be decomposed into four steps, namely

(1) matching cost computation, (2) cost (support) aggregation, (3) disparity computation and optimisation, and finally (4) disparity refinement. Examples of approaches applied at each of these steps are described in the following.

Matching cost computation The similarity of two pixels is expressed in their similarity in intensity level. Commonly used measures in pixel-based matching are Squared Difference (SD) (e.g. [97]) and Absolute Difference (AD) [65]. In addition, some efforts have been invested in designing measures which are insensitive to camera gain or radiometric differences, e.g. dense feature descriptors [150], census transform [166], gradient [126] or entropy-based measures [178]. A more detailed review and evaluation of matching cost methods can be found in [146, 59].

Aggregation Matching cost is calculated for all pixels in the reference image at all possible disparities and stored in a Disparity Space Image (DSI). The aggregation is performed by summing or averaging over the support region in the DSI, following the assumption that pixels within a certain neighbourhood should have similar disparity values. Support regions can be two-dimensional, if mostly front-to-parallel surfaces are considered or three-dimensional [177], admitting also slanted surfaces. Local cost aggregation tends to perform poorly at object boundaries (depth discontinuities), where the smoothness assumption does not apply. To address this problem, adaptive window techniques have been introduced, including adaptations of the window position [16], size [107] or shape [147]. Another solution is to use fixed windows but with weighted support, where the weight of each pixel in the support region is assigned individually and based on its similarity [165], or geodesic distance [60] to the central pixel. Very fast and accurate results have been obtained by using the guided filter applied to stereo cost volume (DSI) filtering [61].

Disparity calculation / optimisation Stereo matching algorithms can be divided by their way of disparity calculation into local and global approaches. Local stereo methods rely on an implicit smoothness assumption realised by matching cost aggregation. The correct disparity is determined by the local minimal matching cost, i.e. Winner-Takes-All (WTA). In contrast, the global methods make explicit smoothness assumptions and model the stereo matching as optimisation problem. In that case, the goal is to find a solution d that minimizes a global energy, as formulated in Equation 3.4

$$E(d) = E_{data}(d) + \lambda \cdot E_{smooth}(d) \tag{3.4}$$

$E_{data}(d)$ is the matching cost and $E_{smooth}(d)$ is the smoothness cost. A discontinuity-preserving energy function used to express the smoothness assumption can be modelled with Markov Random Fields (MRFs) [54]. In addition, following the observation that disparity and intensity edges are likely to coincide, E_{smooth} can be dependent on the intensity differences by assigning lower penalties at depth discontinuities [16, 19]. Different energy minimisation procedures have been employed in global stereo including: simulated annealing [5], probabilistic (mean-field) diffusion [128], expectation maximisation [11], graph cuts [19] or belief propagation [144]. Global energy functions are computationally expensive and rather slow, therefore dynamic programming methods have been used for finding the global minimum within separate scanlines [106, 153, 13]. Another approach is to perform semi-global matching, for example, by evaluating the cumulative cost function in eight directions, as presented in [58]. In general, when combined with sophisticated aggregation strategies and robust cost functions, dynamic programming stereo can produce fast and high-quality results. An interesting alternative to global disparity optimisation are cooperative algorithms [93, 95, 177, 155]. A global optimum is achieved by iterative local update of disparity estimates using non-linear operations. The global energy function is not specified; the assumptions of uniqueness and smoothness are incorporated in the neighbourhood feedback mechanisms of the cooperative network.

Disparity refinement The direct output of stereo computation, whether local or global matching, can sometimes be rather unappealing. Many errors in disparity maps are caused by occlusions, i.e. parts of the scene visible only in one view. One way of detecting occluded pixels is to perform a left-right consistency check [50, 32]. Disparities of corresponding pixels from the left and right disparity maps are compared and, if not equal, they are considered as invalid (occluded) and may be assigned with the disparity value of the spatially closest background pixel on the same scanline. The quality of the resulting disparity maps can also be improved by smoothing filters, e.g. median filter [127]. Further enhancement of the disparity map can be achieved by subpixel refinement methods [66, 124, 97].

Space-time stereo In the literature, we can find several methods that extend the classical stereo vision technique into the time domain, motivated by the fact that the correspondence of points is easier to establish using both spatial and temporal constraints. Dynamic information can help in resolving the ambiguity in static stereo matching, due to the scene variation over time. Space-time stereo can be achieved, for example, by local aggregation using spatiotemporal integration windows [169], by combining motion and stereo formulated as partial differential equations [141], or extending the Markov Random Field (MRF) to three-dimensions, assuming that points should be spatially and temporally smooth [157].

3.3 Depth Estimation using Dynamic Vision Sensors

Although the previous section focused mostly on stereo matching approaches, in the following we summarise main directions in a broader field of depth estimation using event data, not limited to stereo matching.

In Figure 3.3 we present a taxonomy of different methods for depth estimation using DVSs. In the literature, depending on the setup, we can distinguish active and passive depth reconstruction methods. The former assume combining dynamic vision sensors with active depth sensing, e.g. laser scanner, as proposed in [21] for terrain reconstruction. Application of DVS to active stereo helps to overcome conventional cameras' shortcomings, such as limited bandwidth, sensitivity to highly reflective surfaces or ambient light [96]. The passive methods rely only on the data captured by vision sensors, measuring radiance reflected from scene objects. Among passive methods we can find depth

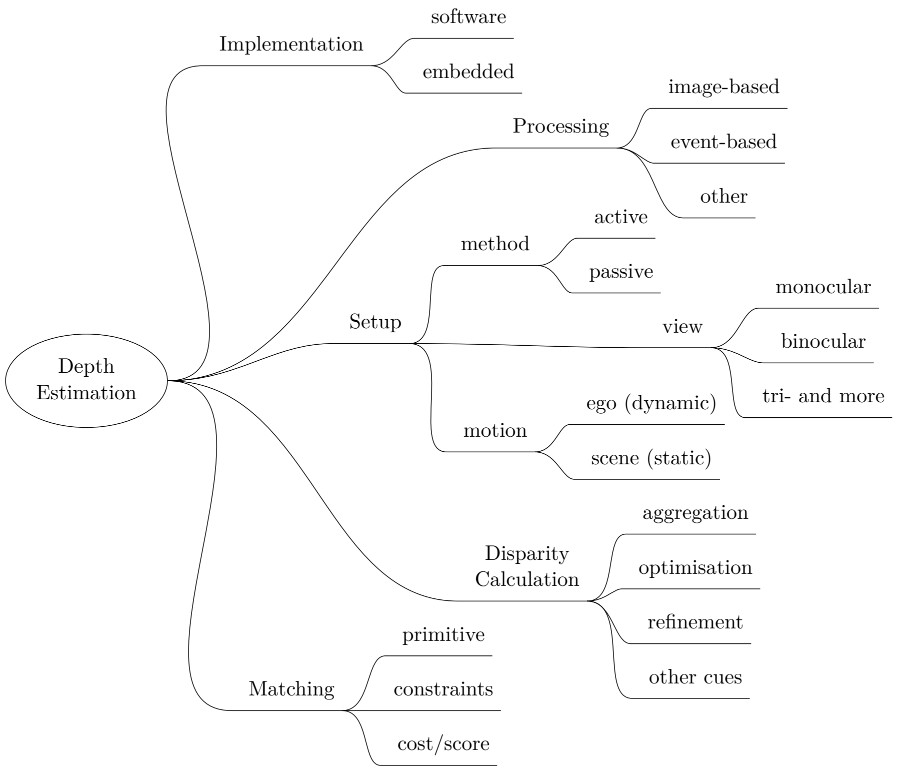


Figure 3.3: Taxonomy of depth estimation methods for dynamic vision sensors.

reconstruction from two or more views of the scene, captured simultaneously, i.e. *shape-from-stereo* [71, 111], or monocular depth estimation from a moving sensor, i.e. *structure-from-motion* [68, 118]. Since DVS reacts to temporal illumination changes, there are two ways of generating events, namely upon motion of the objects in the scene or upon the sensor' ego-motion. Further distinction could be made within the choice of handling the events. Thus, we distinguish methods that operate on an image representation of events (image-based) [134], directly on the stream of events (event-based) [121, 111] or using other data structures (other) [175]. Regarding the correspondence search, a variety of matching functions were proposed. In Figure 3.3, we organise them by three aspects, matching primitive, constraints and cost/score calculation. We can also distinguish several disparity calculation methods, such as aggregation, optimisation, post-filtering and use of other cues, e.g. greyscale images [156] or motion characteristics [68, 27]. Finally, the algorithms can be grouped by means of implementation, including software and hardware solutions. The software applications are designed to run on a Central Processing Unit (CPU) or Graphics Processing Unit (GPU). The embedded implementations are deployed in hardware, e.g. on a Field Programmable Gate Array (FPGA), Digital Signal Processor (DSP) or dedicated neuromorphic platforms. In the following, we provide an overview of the main research directions and approaches to depth estimation using dynamic vision sensors, organised according to our taxonomy.

The first AER stereo chip and cooperative stereo The first prototypes of stereo *silicon retina* were developed by Mahowald and Delbruck [91] in the late 1980s. The authors tested the cooperative computation [94] using single line sensors. The analog CMOS circuit was designed to find regions of correspondence between two one-dimensional images. The correspondence circuit has been employed in two kinds of chips based either on static or dynamic (temporal) contrast features. The sensors consist of two one-dimensional pixel arrays of 40 pixels, connected to an array of correlators of dimension 40×3. A simplified view of the stereo chip architecture is shown in Figure 3.4.

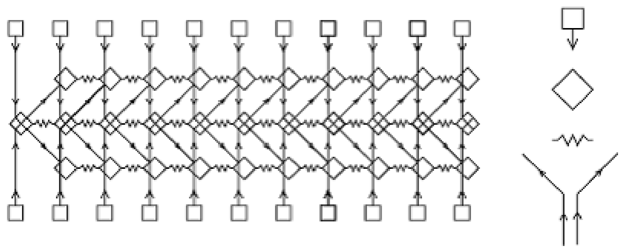


Figure 3.4: A simplified stereo chip architecture [91].

The nonlinear combination of inputs from both *silicon retinas* is passed to the correlators, which detect if the inputs refer to similar contrast and polarity. In order to suppress false matches, feedback connections are used to perform positive coupling among correlators at the same disparity. The correlators at the true disparity tend to be active and receive more positive feedback than the surrounding correlators. The correct matches are said to have high values. Therefore, a *winner-takes-all* (WTA) function is applied to select the maximally driven correlator, which is assumed to be the true match. The complexity of the circuit increases drastically for two-dimensional image matching. A great amount of correlator units would be required to deal with higher dimensionality and resolution as well. Since most of the parameters are realised by the chip architecture, a lack of flexibility was considered the main limitation of the on-chip stereo approach.

Motivated by the success of the on-chip cooperative stereo computation, a similar approach was implemented in software by Hess [57] in 2006. The algorithm performs disparity estimation based on events from two DVS. Events of two corresponding rows are matched by their coincidence in time, and accumulated over 20ms in a two-dimensional *matching matrix*, where each diagonal represents a disparity level. The diagonal with the highest sum of matches is considered the correct disparity. One limitation of the algorithm is that it estimates only a dominant disparity in the scene. Since the local feedback operations are not performed, this solution is not considered a cooperative stereo implementation.

Among the stereo matching methods proposed prior to this thesis (listed in Table 3.1), we can distinguish two major ways of processing visual event data: image-based (also called frame-based) and event-based algorithms.

Method	*Setup*			*Platform*	*Matching Function*	*Disparity Refinement*
Mahowald [91]	B	S	E	VLSI	T,P, ev	1D match., Coop.Net.
Hess [57]	B	S	E	CPU	T,P, ev	aggregation, WTA
Schraml [134]	B	S	I	CPU, DSP	P, F, px(win), NSAD	aggregation, WTA
Kogler[73]	B	S	E	CPU	T,P, ev	aggregation, WTA
	B	S	I	CPU	P, px(win), Hamming	aggregation, WTA
	B	S	I	CPU	T,P, px(win), SAD	aggregation, WTA
Eibensteiner [47]	B	S	I	FPGA	T,P, ev	aggregation, WTA
Benosman [8]	B	S	E	CPU	T,P,G, ev	
Rogister [121]	B	S	E	CPU	T,P,Ord,G, ev	
Carneiro [30]	Tr	S	E	CPU	T,P,Ord,G, ev	Bayesian filt. in 3D

Table 3.1: Taxonomy of event stereo matching methods proposed until 2013. Methods are listed in chronological order and described by categories: (1) setup (B-binocular, Tr-trinocular, S-static, I-image or E-event based processing), (2) matching constraints (T-time, P-polarity, F-frequency, G-geometry, Ord-order), primitive and cost, (3) disparity refinement.

Image-based approaches to stereo matching Image-based methods explore the possibility of adapting events to conventional computer vision algorithms. Events can be converted to images by aggregation over a specific time period, as presented in [133, 73], or reconstructed to grey-scale images by more sophisticated algorithms, as shown in [37] or [4]. Once the events are encoded into an image form, conventional stereo algorithms can be applied. Schraml et al. [133] evaluated window-based stereo matching with different matching cost variants, achieving the best performance with Normalized Sum of Absolute Differences (NSAD). Kogler et al. [73] proposed a window-based stereo matching technique applied to a tri-logic image representation that stores the polarity of the most recent event generated at a particular position.

Event-based approaches to stereo matching The second way of dealing with event data, the asynchronous processing, is to operate directly on the stream of events. The main challenge in event-based stereo matching is finding correspondences between events. In conventional stereo, two pixels are compared by their values (greyscale, colour). Since events do not convey the absolute brightness values, the task becomes more difficult. In most of the approaches, events are compared by their occurrence in time, following the assumption that the same stimulus should trigger corresponding pixels in the left and right view at the same time. In the method proposed by Rogister et al. [121], matching candidates selected from events within a defined temporal window are compared by their Euclidean distance to the epipolar line. Additional constraints have been proposed to reduce matching ambiguity, e.g., matching only events of the same polarity and orientation or eliminating wrong matches by an ordering constraint [121]. Furthermore, in order to smooth the final results, the disparities of events can be averaged over time [73]. Carneiro et al. [30] apply Bayesian filtering to the initial matches projected into 3D space.

The above-mentioned methods constitute the state-of-the-art in stereo matching using DVS at the beginning of this PhD research. Our main goal was to improve the quality of event-based methods, to make them comparable with image-based algorithms' accuracy and performance, and at the same time exploit the data driven aspect of the sensors. As a feasible model for asynchronous stereo matching, we propose a dynamic cooperative network that adapts the disparity estimates with each incoming event. We develop two variants of the cooperative stereo, which are described further in Chapter 5.

Several methods developed in parallel with this thesis are included in Table 3.2. Kogler et al. [71] suggested an adapted Belief Propagation (BP) and a Two-Stage Filter (2SF) technique, and demonstrated that the latter gave better results. Belief propagation has also been used by Xie [158] to refine the initial results of event matching [121] with smoothness and uniqueness constraints. Zou et al. [179, 180] proposed a novel matching primitive to enable context based event matching. The local, spatial context of each event is constructed from the spatial distribution of neighbouring events in the signed (polarity) distance space. Each descriptor is a 24-bin histogram of events frequency in 12 directions (in 30 degrees steps), calculated per each polarity. The matching cost is a binwise difference between the histograms. Additional assumptions have been used to improve the quality of event to event matching. Camuñas-Mesa et al. [28, 29] proposed to use the orientation of edges accumulated over a fixed time period (e.g. 50ms). The orientations are calculated by a bank of Gabor filters implemented as a network of convolution modules on an FPGA.

Monocular dynamic setup Since many of the computer vision applications are highly connected with tasks of navigation (robotics, UAV), and Simultaneous Localisation and Mapping (SLAM), also in the context of event-vision we can recently find more and more methods designed to work in a dynamic sensor setup (ego-motion). Among 3D reconstruction methods, we can find multi-view stereo from a single sensor [118] or simultaneous 3D reconstruction and 6 degrees of freedom (6DoF) motion tracking [68]. The former applies a space-sweep method [34] and performs depth estimation directly in the 3D space. The latter method applies probabilistic estimation using e.g. Bayesian or Kalman filters to the events with reconstructed greyscale values. Gallego et al. [53] proposed a contrast maximisation framework for event-based vision tasks such as motion, depth and optical flow estimation. The depth estimation is achieved by multiple-view plane-sweep and the patch of warped events is created for different depth levels. The correct depth is denoted by the highest contrast, i.e. best alignment of events to edges.

Binocular dynamic setup Unlike the above-mentioned methods, which assumed monocular vision, there are also methods designed for systems which generate visual information from ego-motion, however are equipped with two sensors. An example of such approach is the panorama stereo vision system [7]. Depth estimation from stereo panoramas was proposed by Schraml et al. [131, 132]. The stereo matching considers *spike trains*, i.e. event sequences and novel matching cost based on comparing event distributions using inter-event distances. Very recently, Zhu et al. [175] proposed matching applied to time synchronised event disparity volumes, using the motion of the camera to synchronise the

Method	*Setup*			*Platform*	*Matching Function*	*Disparity Refinement*
Piatkowska [110]	B	S	E	CPU	T,P, ev	dynamic Coop.Net.
Camuñas [29]	B	S	E	FPGA,CPU	T,P,Ord,G,Or, ev	WTA
Kogler [71]	B	S	I	CPU	P,F, px(win), SAD	aggregation, BP, 2SF
Eibensteiner [45]	B	S	I	FPGA	T,P, seg, SAD	aggregation, WTA
Schraml [131]	B	D	E	CPU	T,P, ev.seq.	dynamic prog.
Kim [68]	M	D	O	CPU	T,P,I, ev	Kalman Filter
Rebecq [118]	M	D	E	CPU	T,P, ev	PS
Zou [180]	B	S	I	CPU	P,F, ev.orient.hist	binwise hist.dist., WTA
Piatkowska [113]	B	S	E	CPU	T,P, spatiotemp.win	dynamic Coop.Net.
Osswald [108]	B	S	E	ROLLS	T, ev	SpikingNet.
Dikov [41]	B	D	E	SpiNNaker	T, ev	Coop.SpikingNet.
Xie [158]	B	S	E	CPU	T,G, ev	BP
Andreopolous [1]	M	S	E	TrueNorth	T,P, ev	WTA
Gallego [53]	M	D	O	CPU	T,P, ev	PS, contrast max.
Zhu [175]	B	D	O	GPU	ev.disp.volume	focus/defocus max.

Table 3.2: Taxonomy of event depth reconstruction methods proposed since 2013. Methods are listed in chronological order and described by categories: (1) setup (B-binocular, M-monocular, S-static, D-dynamic, I-image or E-event based processing, O-other), (2) matching constraints (T-time, P-polarity, F-frequency, G-geometry, Ord-order, Or-orientation, I-intensity), primitive (ev-event, px-pixel, seg.-segment, seq.-sequence) and cost, (3) disparity refinement (BP-belief propagation, WTA-winner takes all, 2SF-two-stage filter, PS-plane sweep).

events streams in time. The correct disparities of the event image pixels are determined by the minimal motion blur. The methods have been evaluated on the Multi Vehicle Stereo Event Camera dataset [176].

Hardware implementations Some of the above-mentioned algorithms have already been realised on hardware platforms. The work presented in [134] was implemented on a DSP and later also on an FPGA [44]. Regarding event-based matching, the time correlation algorithm from [73] was realised on a DSP by Sulzbachner et al. [143] and FPGA by Eibensteiner et al. [47]. More recent FPGA implementations include the segmentation based stereo matching proposed by Eibensteiner et al. [45]. Furthermore, more efforts are being invested into the development of algorithms in dedicated neuromorphic hardware implementations. Neuromorphic architectures, such as TrueNorth [100] and Spiking Neural Network Architecture (SpiNNaker) [51] have been shown to be highly efficient image processing platforms, and especially well suited to be coupled with dynamic vision sensors. Dikov et al. [41] used six SpiNNaker processor boards to implement a 106×106 cooperative network. Osswald et al. [108] proposed an implementation of a variant of a cooperative, spiking neurons network on a Reconfigurable On-Line Learning Spiking neuromorphic processor (ROLLS) platform [117]. Andreopoulos et al. [1] implemented a full stereo pipeline on an IBM cluster of TrueNorth neurosynaptic processors. The algorithm comprises data acquisition and rectification, multi-scale spatiotemporal stereo matching and disparity map regularisation. The algorithm is estimated to be capable of producing 2000 disparity maps per second.

3.3.1 Summary

Stereo vision is an extensively researched field in conventional computer vision. Over the last couple of years, a growing number of methods has been proposed for depth estimation using dynamic vision sensors. At the starting point of this thesis, the best performance was achieved by image-based methods, whereas event-based stereo matching was still in its very early stage. The main challenge of this thesis was to provide an efficient event-based stereo matching algorithm, which is tailored to event data and achieves a performance comparable to or better than image-based stereo. The recently growing interest in event-based stereo vision has also led to newly published event-based ground-truth datasets and benchmarks, which facilitate further research in the field.

CHAPTER 4

Challenges of Asynchronous Event-based Stereo Vision

As presented in the previous chapter, stereo matching is a challenging task in computer vision, and numerous methods have been proposed to improve its accuracy, robustness and performance. Whereas there are several solutions proposed for handling the ambiguity in conventional stereo matching, only very little attention has been devoted to the analysis of the problem in the event stereo domain. In this chapter, we focus on the challenges introduced by the nature of dynamic vision sensors. We have identified two main aspects that have to be considered when designing a stereo algorithm for dynamic vision sensors: providing asynchronous processing and establishing similarity between the events. We investigate the challenges associated with these two tasks, and based on that we derive a set of requirements for a well-designed asynchronous stereo matching algorithm for event-based stereo vision.

4.1 Asynchronous Stereo Processing

As mentioned in Section 3.3, there are two principal methods of processing event data, frame-based and event-based. The former assumes fixed time aggregation of events into 'frames' (images), whereas the latter considers matching applied directly to the event stream. Frame-based processing is less computationally demanding, therefore eligible for real-time performance, which is required for many applications. In addition, transforming events into image representation allows for exploiting the variety of available robust conventional stereo methods. Nonetheless, the accuracy of the frame-based methods is highly dependent on the quality of the input frames. Fixed-time aggregation requires finding an optimal frame length to build a good visual representation of the captured motion. In Figure 4.1, frames of different duration have been generated to compare the quality of the moving hand motion representation. We can observe that in this example the optimal frame length can be found around 25ms. If the aggregation time is too short, e.g. 5ms (see Figure 4.1a), the edges of the hand are quite ‘weak’, whereas too long frames may result in ‘motion blur’ (see Figure 4.1e).

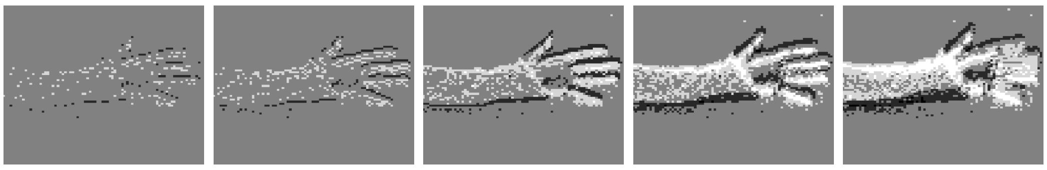

(a) 5ms (b) 10ms (c) 25ms (d) 50ms (e) 75ms

Figure 4.1: Influence of the frame length on the visual representation of a moving hand.

Dynamic vision sensors are data-driven and their output depends on a number of factors, such as the scene illumination, the projected speed of objects or their appearance in terms of relative contrast to the background. The first factor can often be assumed uniform within a particular period of time, however, the appearance and speed are likely to vary across objects in the scene. These two last factors influence the density and frequency of the events generated upon the objects’ motion.

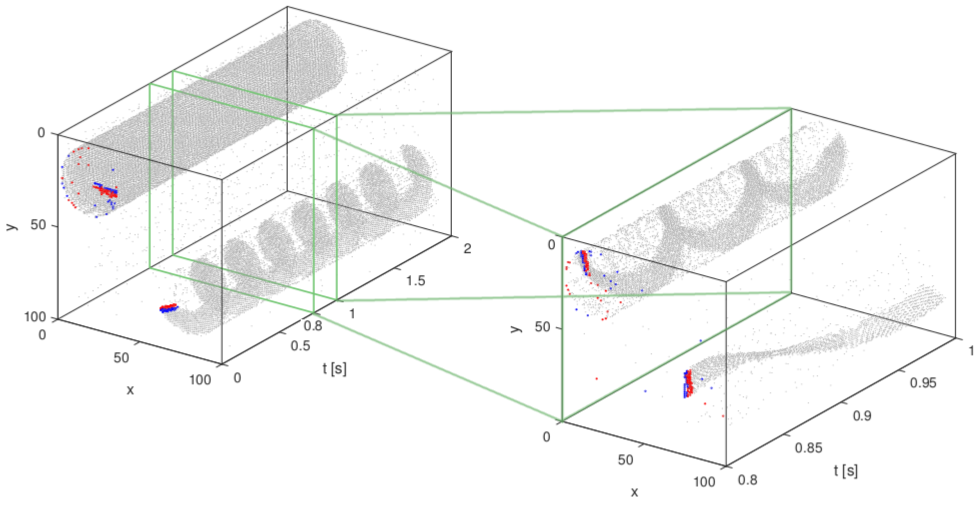
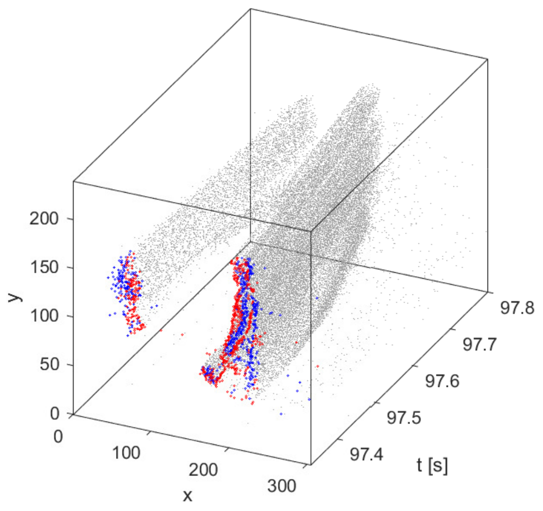


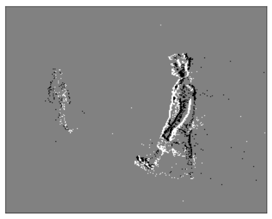
Figure 4.2: Events generated upon the motion of two edges rotating at different speeds. The events are plotted in space and time over a 2s (left) and 0.2s (right) period.

Figure 4.2 depicts the events generated by two edges rotating at different speeds. Each edge is presented by the polarity plot in the first time window; the aggregation time of 2ms and 5ms has been selected to represent the fast and slow edge, respectively. We can observe the influence of speed on the frequency (density) of the generated events; intuitively, the faster the object, the higher the frequency. On the left, the events are plotted in space and time over a period of 2 seconds. We can recognise the rotational motion trajectory for the slower edge (at the bottom of the scene), but not the fast edge, which forms a dense 'tube' of events. However, if we zoom-in and plot part of the sequence over a 0.2s period, then we can see that whilst the slower edge made only a half of the rotation, the faster edge made three within the same time. Smaller aggregation time is beneficial for fast objects, however longer time periods are required to obtain better representation of slow objects. Therefore, the aggregation of events should be adjusted to the object's speed individually for every object in the scene. This is a potential limitation of the fixed-time aggregation.

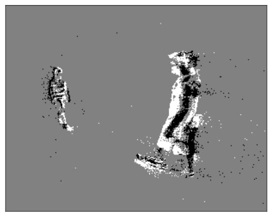
Figure 4.3 presents another situation where a single optimal frame duration cannot be determined for all objects in the scene. Two people are walking at the same speed but at different distances to the sensor. In this case, the event representation varies across objects not due to their speed but the projected velocity of their motion. We can observe in Figure 4.3a that the event 'cloud' generated by closer person is denser than the other one.



(a)



(b) 10ms

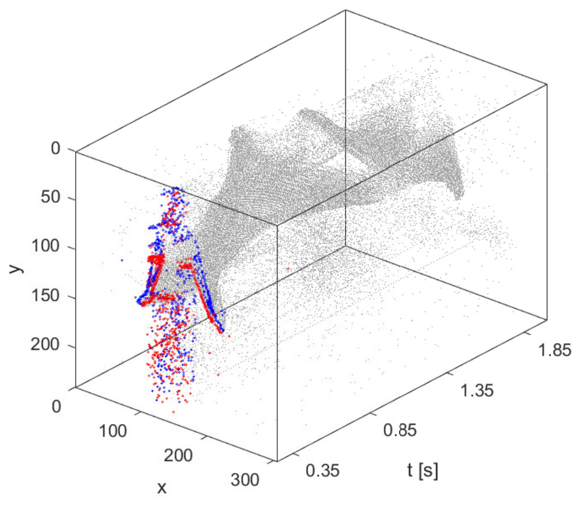


(c) 30ms

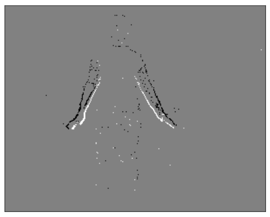
Figure 4.3: Events generated upon motion of two people walking at different distances from the sensor, presented with a spatiotemporal plot (a), and frames adjusted to the person closer (b) and farther (c) to the sensor.

Figures 4.3b–4.3c present frames of duration 10ms and 30ms, which were found optimal for the closer and farther person, respectively. As can be observed, in the shorter frame the events generated by the farther person are not sufficient to form clear edges, as in 30ms frame. On the contrary, the closer person's motion is projected as faster, therefore we can observe a 'motion blur' in the longer 30ms frame. In this example, the difference in speed is not that prominent, thus the stereo matching would be able to handle the slight motion blur of the closer person. Nonetheless, it is easy to imagine situations where the difference in velocity could be more significant, e.g. in a scene containing cars and pedestrians.

Furthermore, even considering a single object in the scene we cannot make any assumptions about the expected velocity, since the velocity could change over time (e.g. a car speeding up or slowing down). Dealing with human motion, we need to take into account that different parts of the body could be moving at varying speed, e.g. the torso moving slower than hands or legs. Figure 4.4 depicts a person standing and moving arms and hands in front of the sensor. Looking at the spatiotemporal view of the events, we can see that the upper part (containing the arms) is significantly more dense than the lower part containing torso and legs. However, the slight motion of the rest of the body is still captured by the sensor. The optimal aggregation time for the hands is 10ms (see Figure 4.4b), however at the same time the body is represented only by few events. Nonetheless, these sparse events, when aggregated for a longer time period, e.g. 100ms, can form clear contours of the body, legs, or face (see Figure 4.4c).



(a)


(b) 10ms

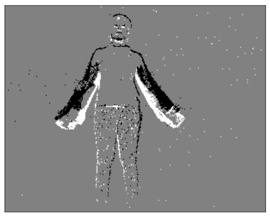

(c) 100ms

Figure 4.4: Events generated upon motion of one person moving arms presented as spatiotemporal plot (a), and frames accumulated over 10ms (b) and 100ms (c) period.

Although frame-based methods proved efficient and accurate when applied to well-chosen frame lengths, they lack robustness to the speed variation, as it has been demonstrated by the examples above. Fixed-time aggregation assumes single frame length, which might not be optimal for all the objects in the scene. This could have a direct impact on the stereo matching results. If the frame is too short, hence not sufficient to build a good representation of the object, then the object might be mismatched or even filtered out as noise. On the contrary, if the frame is too long, the transition of the edges in time is captured within one frame. Due to that, the depth of multiple edges is averaged over the frame duration, which causes loss of precision and lowers the accuracy of depth estimation. The loss of precision is quite visible when dealing with complex objects (e.g. spherical surfaces, human body), since details of moving edges are lost and averaged over the captured trajectory. Moreover, it can happen that within one frame the trajectories of fast objects might overlap, i.e. one object occludes the other, which in result introduces unnecessary ambiguity to the stereo matching.

As mentioned previously (Section 3.3), one way of addressing the challenge of asynchronous stereo matching is to avoid aggregation and apply processing directly to the stream of events [73, 121]. Such methods perform event-to-event matching using constraints such as time, polarity or orientation. However, the accuracy of event-to-event stereo matching is rather low and can be compared to pixel-to-pixel matching in conventional stereo vision. Additional constraints, such as a smoothness, would be necessary to improve the quality of the results. Nevertheless, the smoothness constraint requires events aggregation over time. An attempt to provide adaptive events aggregation has been proposed by [119]. The frame duration is adjusted according to the local density and timestamp thresholds using local buffers for every pixel. This method, however, suffers from noise, which is accumulated together with the other events and, due to a lack of spatial context, is difficult to be detected and removed.

4.2 Event-based Matching

As we have learned from the previous section, event aggregation into frames may limit the temporal resolution of the sensor, thus correspondence between data from the left and right DVS should be established in an event-wise manner. In the following, we revise three fundamental assumptions used in solving the correspondence problem in conventional stereo vision, i.e. compatibility, smoothness, and uniqueness, and investigate their applicability to event-based matching.

Compatibility of two corresponding points should be reflected in their similarity. In conventional computer vision, compatibility is measured by the similarity of objects' appearance — commonly it is the difference in pixels' intensities. Since events denote change of intensity, they do not carry the information about the absolute intensity values. A common approach for comparing events is to use their timing according to the assumption that the object moving in the scene should trigger the left and right sensor at the same time. Moreover, apart from location and time, another feature is carried in each event, namely the polarity, which denotes the direction of the detected change, i.e. whether the event was generated upon increase or decrease of the light intensity. The polarity can be used as a matching constraint since corresponding edges (events) are expected to be of the same polarity.

Local smoothness assumptions are derived from the coherence of the matter, i.e. points that belong to an object occupying a certain area are likely to vary smoothly in depth. In the context of events, the assumption of local smoothness would apply in the spatial and temporal domain. Events that represent the same edge are assumed to be locally smooth since the edge was generated by the same object. Due to the high temporal resolution of the sensor, the events form the trajectory of the moving object, therefore their depth values are also assumed to vary gradually.

Uniqueness assumes that a point on the 3D surface occupies one position at a particular time, therefore there could only be one corresponding candidate in the other view, which denotes the correct disparity (depth) estimate. This assumption could be problematic in event-based matching. Unlike conventional pixels, which represent the projection of a scene point at the time the frame is captured, the DVS events reflect the change of pixel's intensity over time. This raises the question whether it is reasonable to assume that, at the event level, the representation of a change in a point's location will be the same in the left and right view. It is also worth investigating if we can assume a unique mapping of events, meaning that for every left event there is at most one corresponding right event.

In addition to the above-mentioned three assumptions, other matching constraints could be applied to event data. Employing geometrical constraints to sparse event data seems applicable. A basic geometrical assumption is the use of epipolar geometry, i.e. events that lie on the corresponding epipolar line are considered in matching. Furthermore, assumptions about object representation characteristics could be used, e.g. line-fitting algorithms. Moreover, the assumption about the order of objects across the views is valid also in even-based matching, excluding rare situations, e.g. thin foreground objects, which could appear in different order in the left and right view. The temporal order of events, however, cannot be used as a matching constraint because time and order in which the edges are formed on the left and right view may vary due to the fact that pixels are independent and generate events in an asynchronous manner. Another feature that can be used in matching events is orientation, which is usually defined as the orientation of a line fitted to the local neighbourhood of the event. Events of similar orientation are assumed to be possible matching candidates. Calculating the orientation for each event, however, can be a computationally demanding task.

Solving the correspondence problem becomes challenging when dealing with more complex scenes captured by dynamic vision sensors such as objects moving at different speeds, in highly textured, or 'cluttered' scenes with many moving objects, which are often occluding each other. Just like in conventional stereo, each of the aforementioned assumptions could be violated in real world conditions. This may lead to *erroneous* similarity scores between two candidates, when incorrect matches score higher than true ones. In some cases, a similarity measure could be *ambiguous*, when more than one matching candidate in the opposite view achieves the same score.

There are several factors that influence the response of a pixel, including *(i)* external conditions, such as lighting, speed and direction of the moving object, or the colour of the object (contrast); and *(ii)* internal parameters, including the hardware-specific variations and bias settings that control the pixel's response time, sensitivity, spike rate, etc. It is very likely that these factors will not result in identical responses of the pixels from the left and right sensor. In what follows, we investigate how much these factors can influence the sensor response, and how much ambiguity needs to be handled by the stereo algorithm.

4.2.1 Internal Parameters that Influence Left and Right Response

We present an experimental analysis of the data recorded by the left and right sensor upon the same visual stimuli. The test recordings were performed under a controlled laboratory setup to suppress the influence of external factors on the left and right sensor response. We analyse two aspects of the recorded data, (1) the similarity in temporal characteristics of the left and right DVS response, and (2) the behaviour of sensors under different bias settings, in terms of overall similarity of the left and right response.

Temporal Event Data Characteristics

It has been observed [121] that commonly used assumption about events coincidence in time is rarely true in real DVS recordings, due to the latency in a pixel's response (which varies across pixels and boards), and jitter in the event's timing caused by off-chip transmission over a shared digital bus. In the following, we investigate *(i)* how much variation could be introduced by hardware specific factors, *(ii)* how similar the temporal information generated by corresponding pixels is, and *(iii)* if for each event there is a corresponding one in the other view. In our experiment, we used the ATIS stereo sensor [6] with time resolution set to 1ms. As a visual stimulus we used a green Light-Emitting Diode (LED) blinking at a frequency of 10Hz. We measured the response of five different pixel locations, marked as *top*, *bottom*, *left*, *right*, *center*. Additionally, for the *center* pixel, we have recorded data at different distances from the sensor: 0.15m, 0.45m, 0.7m, and 1.95m. The duration of all data sequences is 10 seconds. The summary of the events statistics for each test sequence is given in Table 4.1. We compared the number of events

	distance	#events		#events ratio		difference
	(m)	left	right	left	right	(%)
bottom	0.45	7154	6762	0,514	0,486	2,8
top	0.45	11588	12515	0,481	0,519	3,8
left	0.45	12786	14266	0,473	0,527	5,5
right	0.45	11267	10371	0,521	0,479	4,1
center	0.15	15388	16564	0,482	0,518	3,7
	0.45	19330	20599	0,484	0,516	3,2
	0.70	17897	15134	0,542	0,458	8,4
	1.95	14555	17371	0,456	0,544	8,8

Table 4.1: Summary of the recorded test data sequences.

generated by the left and right sensors. We calculated the ratio of the left (right) events to all generated events, and the percentage of the difference in number of the left and right events. Figure 4.5 shows the time histograms of 1ms bins, built separately for ON and OFF events that were generated by two corresponding pixels (*center* at 0.45m).

Although the responses of the left and right sensors are similar, we can observe that some of the events cannot be matched (denoted by green circles).

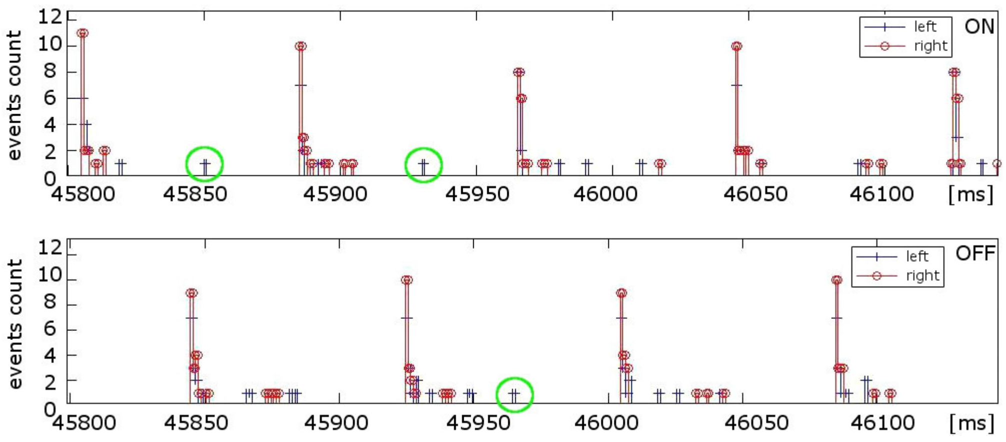


Figure 4.5: The stereo DVS response to blinking light stimuli: time histograms of ON (top) and OFF (bottom) events from the left (blue) and right (red) DVS. The histograms show the number of events generated in 1ms.

Next, we analyse the histograms of the pixels' responses averaged over a sequence of 1s duration, as shown in Figure 4.6. The shapes of the left and right distributions are similar. However, we observe that the distributions are shifted in time due to a delay between left and right response. Additionally, the histograms usually tend to differ in height, meaning that one of the pixels generates more events than the other.

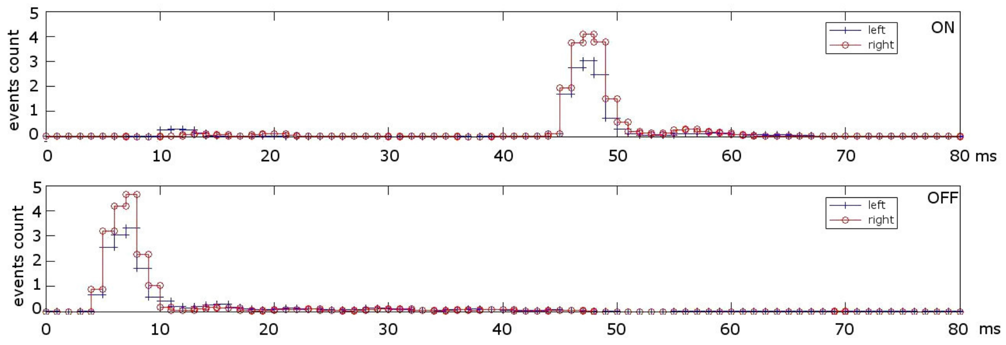


Figure 4.6: The stereo DVS response to blinking light stimuli averaged over 1s sequences: time histograms of ON (top) and OFF (bottom) events from the left (blue) and right (red) sensors.

We have inspected time histograms for all test sequences. In general, there is a clear similarity between the response of the left and right sensor in terms of the shape of distribution, as well as their agreement in time. Nonetheless, every histogram indicated some events that cannot be matched; these are considered to be noise. Furthermore, there is no regularity in response order or delay. Sometimes left events occur later than the right ones and vice versa. We have not observed any delay or dissimilarity correlated with the address (location) of the pixel, however we have noted that the response time tends to vary more with increasing distances of the object to the sensor. The number of events generated by the left and right sensor board differs on average by around 5%. Moreover, the disproportion increases with the distance, e.g. at 1.95m it reaches 8.8% of difference.

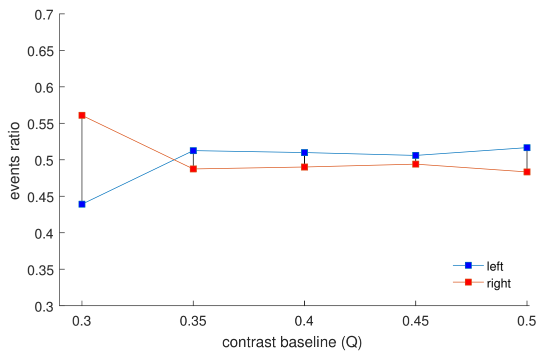
Sensor's Response Under Different Bias Settings

The second part of the data analysis is focused on the impact of bias settings on the similarity of the sensors' response. The sensor is controlled by a set of parameters, called bias settings. These parameters are set by numerical values, which are then translated to voltage values by the internal bias generator. It has been observed that, depending on the sensor board, numerical settings might translate to slightly different voltage values. Hence, there might be some variations in the behaviour of the sensors, even under the same numerical bias settings. We investigate if by an appropriate choice of the bias values, the similarity between left and right sensors' output can be improved.

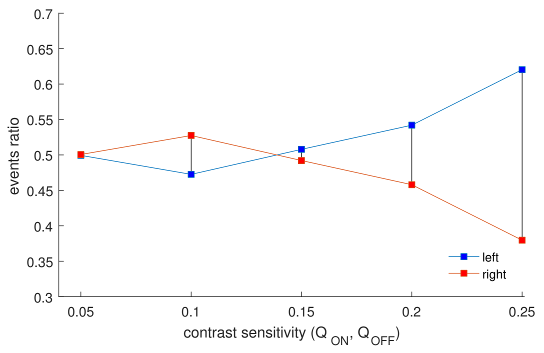
In our experiments the test data were recorded with the UCOS sensor [125, 134]. As a test pattern we used two horizontal black bars on white background, placed on a drum rotating at a speed of 2.25 m/s. The tests were recorded for 15s, separately for each sensor, under exactly the same conditions, such as position, distance, angle of view, lighting, speed of visual stimuli, and bias settings. We have selected several biases for further investigation in the experimental tests. We start with an analysis of the influence of the contrast sensitivity parameter, controlled by Q, Q_{ON}, and Q_{OFF}. The smaller the difference between $|Q_{\mathrm{ON}} - Q|$ or $|Q_{\mathrm{OFF}} - Q|$, the higher the sensitivity of a pixel to contrast change. We have also used the B_{Refr} bias, which is responsible for the refractory period, i.e. time until the pixel can spike again. The length of the refractory period has an impact on the data rate. The last two biases considered in our tests are B_{Fo} and B_{Pr}, which are responsible for bandwidth control, and usually used for noise reduction.

We have recorded four bias tests (for more details see Appendix C), each focusing on varying particular bias values, while keeping the other settings constant and default. The results of the bias tests are presented in Figure 4.7. We can observe how the similarity in the amount of events generated by the left and right sensor changes upon different bias settings. Biases controlling the contrast sensitivity tend to have quite significant influence on the sensors' response, and careful selection of $Q, Q_{\mathrm{ON}}, Q_{\mathrm{OFF}}$ could help in reducing the disproportion between the left and right events. In the context of B_{Refr}, although the best similarity of the left and right response is achieved for small bias value settings, this setting is not optimal for the quality of the acquired data. Different bandwidth settings show little impact on the similarity of the left and right response.

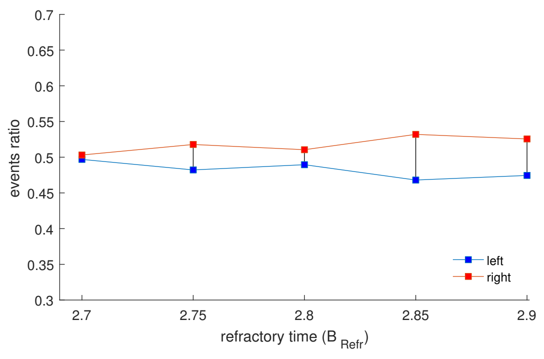
In addition, we have performed a so-called *noise test*, in which the recordings are made with an idle scene (no visual stimuli), hence all of the generated events are considered noise. Table 4.2 presents the statistics of the *noise test* and a *default-test* with a moving



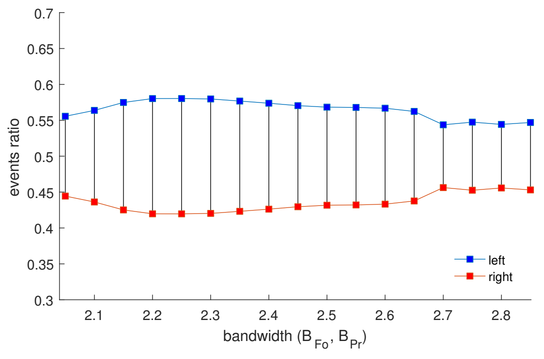
(a)



(b)



(c)



(d)

Figure 4.7: Bias tests with varying settings for the contrast baseline (a), contrast sensitivity (b), refractory period (c), and bandwidth (d), and their influence on the ratio of the left and right events.

object; both data recordings were made under optimal bias settings. We can observe that the sensor platforms vary also in the generated noise ratio, expressed in almost 3% difference between left and right sensor's response. Moreover, there is a clear imbalance in the ON and OFF events, indicating that the majority of the noise events are of OFF polarity.

	Difference between the left and right response						
	all events		ON events		OFF events		duration
	(kEv)	(%)	(kEv)	(%)	(kEv)	(%)	(s)
default-test	75,3	1,98	31,2	1,66	44,2	2,29	15
noise-test	127,7	2,89	89,4	34,18	200,2	4,88	238

Table 4.2: Summary of the results for *default-test* and *noise-test*.

In general, we can observe that under different bias settings, sensors can generate a quite imbalanced amount of events, in the worst case reaching up to 25% of difference between the number of the left and right events (see Table C.2). Nonetheless, with a well chosen set of bias parameters the imbalance can be reduced rather significantly, e.g. to 1.98% in *default-test*. The tests indicate an average 8% of overall difference between left and right events. We consider this value as an indicator for how much ambiguity could be introduced by hardware specific cross-platform variations, and which should be handled by the stereo algorithm.

4.2.2 Ambiguities Introduced by External Conditions

Apart from the internal parameters that affect the DVS behaviour, we also need to take into account how external conditions influence the difference between the left and right events. In the following, we describe two potential causes of ambiguity in event stereo matching, the first one associated with the difference of an object's representation due to the projected motion velocity, and the second related to the variation of the scene's appearance depending on the viewing angle.

Relativity of Motion Speed

As presented in Section 4.1, the appearance of moving objects depends on their projected velocity, i.e. affect the density of events in space and time. Objects closer to the sensor are naturally projected as bigger, and their relative motion velocity as well as event rate are higher. On the contrary, objects that are farther from the sensor are projected as smaller and generate fewer events. Figure 4.8a–4.8b shows two people walking at different distances; the events are generated by the left and right sensor, respectively. We selected a horizontal patch (marked by green dotted lines) to analyse in more detail the time history of the collected events. The time history for the left and right view are depicted in Figure 4.8c and 4.8d. The event's timing is encoded in colour ranging from cold (for old events) to hot colours for the most recent events. We can observe that more events are generated for the person closer to the camera, and the events show more consistent temporal information along the edges. In fact, we can actually see that the edges are changing position over time, as indicated by temporal transition from blue, green up to red. This is mainly because the temporal window for the 'closer' person is too long. When we consider the second, 'farther', person, he/she is not only represented by a fewer events, but also the timing of events is less consistent.

We have selected a left event e_{left} generated by the 'farther' person and collected all possible matching candidates from the opposite (right) view. In Figure 4.8e we plot the matching score, i.e. the inverse of the timestamp difference between reference and candidate event, for all candidates. We can observe that the corresponding event, e_{right}, achieves a relatively small score, when compared to candidates from the 'closer' person. In such cases, the single event-to-event temporal matching is likely to produce erroneous results, because events generated by the slow object are matched to the events of the fast object, as they are more 'recent', and thus achieve a higher temporal matching score.

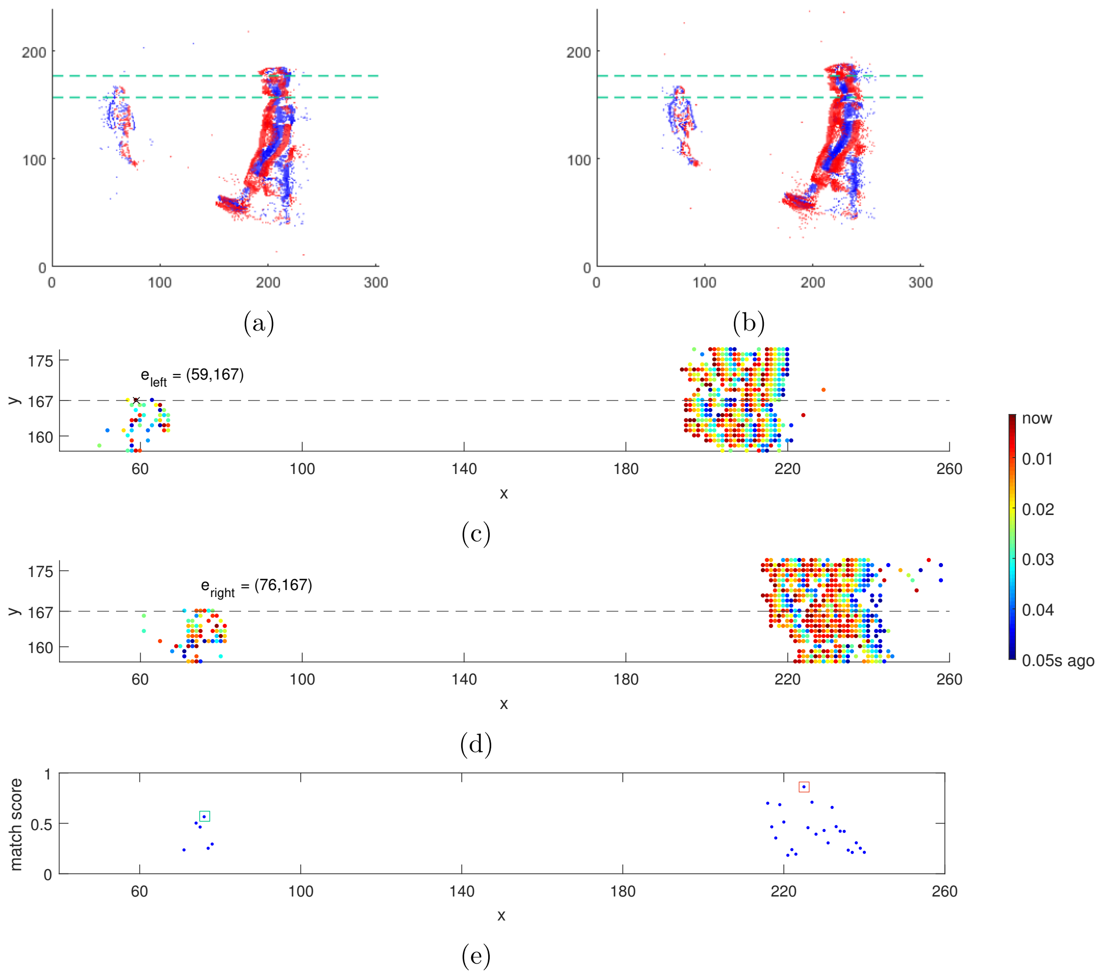


Figure 4.8: Ambiguity in event matching caused by relative motion velocity. Polarity plots of two people moving at different depths are presented for the left (a) and right (b) view, where red indicates ON and blue OFF events. The time history for a selected patch is shown for the left (c) and right (d) view. For the reference event e_{left}, the temporal matching scores for all possible candidates are plotted in (e).

Viewpoint Difference

There are several situations where the viewpoint makes a difference, i.e. the scene appearance varies while viewed from different angles. Firstly, parts of the scene which are visible in one view might not be visible in the opposite view. This situation is referred to as half-occlusions. Secondly, the view point also influences how the geometrical features of the object are projected onto the image plane. For instance, the frontal view of a cube would be represented by events formed into two edges, whereas while seen from a different angle, the cube could appear as three edges (in conventional imaging, the cube would appear as a square or two or three slanted planes, respectively).

Finally, the viewpoint might also influence the perceived contrast magnitude, hence the generated event sequence would differ in the left and right sensor. Events depend on the relative contrast between the colour of the moving object and the background. The perceived background may vary across the views, as illustrated in Figure 4.9a.

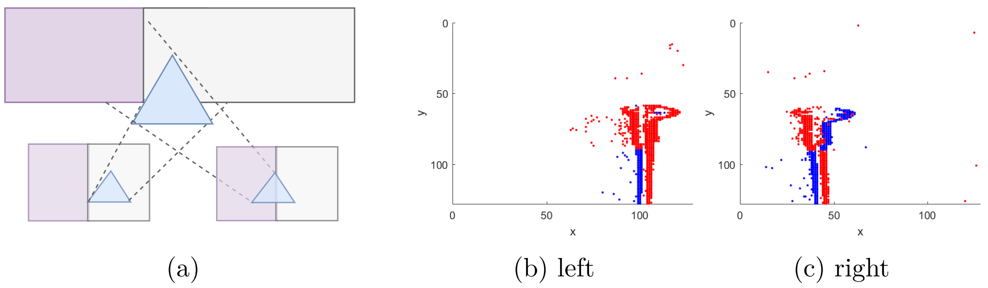


Figure 4.9: Schematic illustration of polarity mismatch (a) caused by the difference in background the object is projected onto in the left and right view. Demonstration on DVS events presented by polarity plots for the left (b) and right (c) view.

This situation may lead to a significantly different contrast magnitude, thus amount and frequency of generated events. In an extreme case, not only the magnitude is different but also direction of change, which is expressed in polarity. As we can observe in Figure 4.9, the left edge of the object (triangle) in the left view is projected onto the relatively lighter background, whereas in the other view onto the darker one. Therefore, assuming that the object is moving to the right, then in the left view the polarity of the edge is ON (change from darker to lighter), whereas in the right view it is OFF (lighter to darker). We refer to this situation as ‘polarity mismatch’. In Figure 4.9b–4.9c, the polarity mismatch is demonstrated in the DVS event sequence.

4.3 Requirements for Asynchronous Event-based Stereo Matching

The main motivation for asynchronous stereo vision is to preserve the high temporal resolution provided by the sensor. It is also preferred to have data-driven processing, meaning that only parts of the scene that are active are being considered in processing. Finally, the asynchronous stereo approach can be a way of preserving the quality of the matching regardless of the type and speed of captured motion. Finding an appropriate model for asynchronous stereo matching is not a trivial task. In the course of our investigation, we have identified a couple of aspects, which should be considered in the design of asynchronous stereo algorithms.

Stream based processing Events should be processed as they are generated, preserving their asynchronous character. In order to perform stream based stereo processing, the correspondence search is done under uncertainty. There is no guarantee that for a given time when the left event arrives, the corresponding right event is already available or if there is a sufficient amount of local context information that could be used in matching. Hence, the asynchronous matching should be able to handle such uncertainty, either by sliding temporal windows, or adaptive aggregation.

Dynamic event aggregation to enable using local context for stereo matching. In conventional image-based stereo methods, the quality of the initial matching costs is usually improved through the smoothness constraint. The smoothness constraint can be employed implicitly by local support aggregation or explicitly by global optimisation. Whether it is the local aggregation or global optimisation, the set of initial matches on which the smoothness constraint is employed have to be collected over time, e.g. stored or buffered. The challenge is to find an efficient data structure to perform aggregation, which dynamically adapts to generated events without limiting the temporal resolution.

Event-wise correspondence in finding correspondence between the left and right event stream. The fundamental assumption in correspondence search is that matching points should have similar appearance regarding intensity values, context colour or texture. Since events denote the temporal intensity change, they do not carry the information about the absolute intensity values. The challenge is to find an appropriate matching primitive and formulate similarity measures to establish correspondence between two events, while tackling the matching ambiguities listed in Section 4.2.

Performance trade-off Asynchronous processing assumes event-based processing which in most cases will be more computationally demanding than applying algorithms for image-like representation of event data. The challenge is to find an efficient way of processing events, e.g. by using suitable data structures and parallel processing techniques.

4.4 Summary

The special characteristics of the data provided by dynamic vision sensors introduce new challenges to the problem of stereo matching. We demonstrated by several examples that fixed-time aggregation might limit the temporal resolution of DVS, and motivated our analysis towards asynchronous stereo matching. Since asynchronous processing can be only achieved by operating directly on the stream of events, we discussed the challenges posed by event-based correspondence search. We presented an experimental analysis of possible factors that might influence the variation between left and right response. Based on our analysis, we have derived a set of requirements that a good event-based stereo matching algorithm should address.

CHAPTER 5

Adaptive Dynamic Cooperative Stereo Matching

A good event-based stereo matching algorithm needs to fulfil two major requirements (1) allow for asynchronous processing, and (2) handle the ambiguities in event-based matching. We have considered these two requirements in the design of the stereo matching algorithms. We revisit an early model of stereo computation, the cooperative stereo [94] and demonstrate that it serves as a good model for asynchronous event matching. The applicability of the cooperative approach to event data has already been proven in [91, 57]. We extend the basic algorithm from [94] to provide the dynamic adaptation of the cooperative network and hence, enable the asynchronous matching. In the following sections, we present two cooperative stereo algorithms, the first one is focused on performing asynchronous cooperative optimisation. The second algorithm addresses the second requirement, namely finding an efficient matching function to handle the ambiguities in event-based correspondence search.

5.1 Asynchronous Cooperative Stereo Matching Model

Marr and Poggio [94] model the problem of stereo matching with the cooperative network where each node corresponds to an intersection of the left and right sightline (see Figure 5.1). The nodes of the network denote all possible matches and the task of stereo matching is to distinguish which nodes represent true matches (depicted in black in Figure 5.1). In order to do that, the authors defined two constraints: uniqueness and smoothness. The former reflects the fact that an object can occupy only one physical position at a particular time, hence there is only one true match for particular point in the scene. The latter constraint, says that disparity varies smoothly almost everywhere due to the coherence of the matter. Both constraints are incorporated in the cooperative network as local neighbourhood operations. Nodes at the same disparity level support each other according to the smoothness rule, whereas nodes along the sightline inhibit each other due to the uniqueness constraint. The network iteratively performs these local operations in order to find a global optimum.

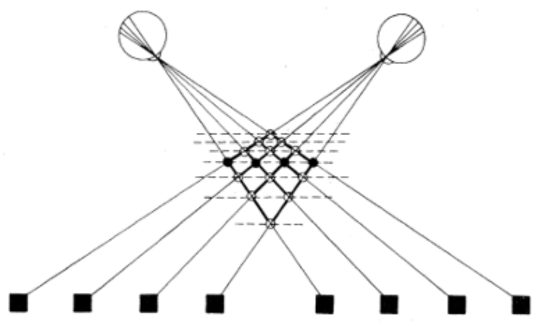


Figure 5.1: Schematic description of correspondence solving between projections from the left and right eye [94].

We notice several aspects that make the cooperative approach very suitable for the events generated by dynamic vision sensors. Firstly, this method is to a significant extent based on implications from biological stereopsis, and therefore, adequate for the bio-inspired DVS sensor. Secondly, the use of neural mechanism to achieve global optimum through multiple local neighbourhood operations can be easily employed for dynamic event processing. Finally, the cooperative stereo was designed for matching identical features thus it can be considered very eligible for matching events.

In its original form, the cooperative network performs optimisation through a number of iterations per stereo image pair. In our work, we propose to dynamically update the cooperative network with each event generated by the sensor and mapped to the network. In this way, the network acts as a dynamic history of previous matching results. This allows for fully asynchronous matching because each event from the stream can infer its disparity on the basis of information found in the network.

A schematic view of the cooperative network is given in Figure 5.2. Incoming events from the left and right sensor are matched by their co-occurrence, i.e. the assumption that quality of matching candidate is reflected in smaller time difference between their timestamps. The stereo matching is done symmetrically for both views. A time-based cost function can be used to assign the initial weights while mapping the possible matches into the network. Additional constraints for the suppression of false matches are incorporated in the network by the positive and negative feedback from local neighbours. Neighbours at the same disparity level implement a cooperative process and give positive feedback to the node, whereas the nodes across the disparity planes compete with each other by negative feedback. Finally, a *winner-takes-all* (WTA) is used to derive the disparity estimates. Nodes exceeding the activation threshold and having the highest weight within the competing neighbourhood denote the correct matches. The cooperative network can be considered a prototype of disparity space image (DSI), term commonly used in stereo

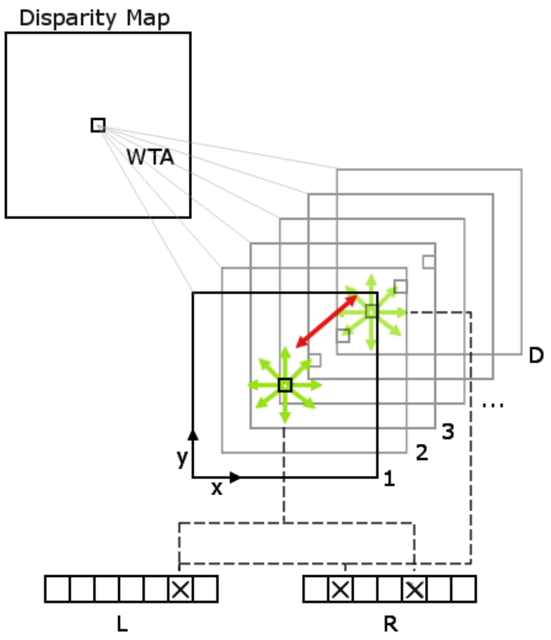


Figure 5.2: Schematic illustration of the cooperative network with positive (green) and negative (red) neighbourhood.

matching literature. Similarly to nodes in cooperative network, pixels in DSI store the matching score/cost values per each matching candidate. However, the major difference between DSI and the proposed cooperative network is that cooperative nodes interact between each other and dynamically adapt their weights based on incoming data.

5.2 Asynchronous Cooperative Stereo Matching Algorithm

In this section, we present our first stereo matching algorithm, further referred to as *Coopv1* algorithm, which implements the dynamic cooperative network. The proposed algorithm can be divided into two processing steps, as illustrated in Figure 5.3.

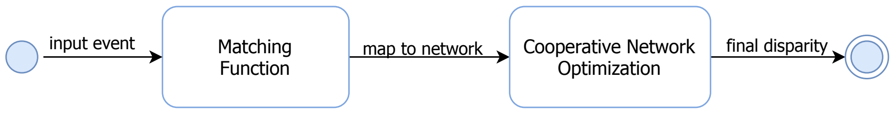


Figure 5.3: Block diagram of the proposed algorithm.

Each event from the input stream is passed to the matching function, in order to find a set of possible matches and to calculate an initial weight (matching score). We assume that events have been previously rectified such that left and right views are geometrically aligned and possible matches can be found on the corresponding epipolar lines. From the matching function, the weighted matching candidates are mapped to the cooperative network. Subsequently, the network feedback is calculated for each possible match and, finally, a *winner-takes-all* (WTA) is applied on the nodes to retrieve the correct disparity. In the following sections, we described the algorithm's steps in more detail.

Matching Function and Mapping Events to the Cooperative Network

Let $e = (e_x, e_y, e_t, e_p, e_c)$ represent the event by its location (x, y), time t, polarity p and camera c.

$$\begin{aligned} e = (e_x, e_y, e_t, e_p, e_c) \mid\ & e_x, e_y \in \mathbb{N}, e_t \in \mathbb{R}, \\ & e_p, e_c \in \{0, 1\} \end{aligned} \tag{5.1}$$

The set of all events in the input stream is denoted by E. For each event, we search for the set of possible matching candidates M_e among events of the other view within a given disparity range (d_{min}, d_{max}), as defined in Equation 5.2.

$$\begin{aligned} \forall_{e \in E} \quad M_e = \{ & m \mid m \in E, \\ & d_{min} < |m_x - e_x| < d_{max}, \\ & m_y = e_y, m_t < e_t, m_c = \neg e_c \} \end{aligned} \tag{5.2}$$

Matching is done symmetrically for the left and right events. Additionally, according to the canonical stereo setup, we assume that the true match for any of the left events will always appear in the right view on the right-hand side from the reference event's position; analogically for the right events, the corresponding match is expected to appear on the left-hand side in the opposite view.

In most event-based matching algorithms (e.g. [121, 57]) matching candidates are weighted by their similarity in terms of timestamps to the reference event e as defined in Equation 5.3.The similarity function ρ_t for each of the candidates is computed as:

$$\rho_t(e, m) = \frac{1}{\alpha \cdot |e_t - m_t| + 1} \tag{5.3}$$

The parameter α controls the slope of the matching score function. The lower the value of α, the more restrictive is the matching function, i.e. weights drop more drastically with higher distance in time.

In addition, the polarity constraint can be added to the matching function. As explained in Section 4.2.2), the corresponding edges sometimes might have opposite polarities (the so-called polarity mismatch), thus we propose to use polarity confidence *pconf* parameter, as shown in Equation 5.4.

$$\rho(e, m) = \begin{cases} \rho_t(e, m), & \text{when}\, e_p = m_p \\ pconf \cdot \rho_t(e, m), & \text{when}\, e_p \neq m_p \end{cases} \tag{5.4}$$

The matches of opposite polarities are still considered in the matching cost calculation but with lower confidence. In this way, in case of polarity mismatch, the corresponding events are not immediately discarded and could still contribute to the score of correct cooperative node (true match).

Cooperative Network Structure, Update and Disparity Calculation

For each incoming event, the set of possible matching candidates, weighted as described before, is mapped to the cooperative network as follows:

$$\begin{aligned} \forall_{m \in M_e} \quad C^*_{e_x,e_y,d} =& \rho(e,m), \\ & \text{where } d = |e_x - m_x| \end{aligned} \tag{5.5}$$

The C^* denotes the initial weight of each matching candidate used to update the cooperative network nodes at corresponding locations. The cooperative network is a three-dimensional structure $C = (X, Y, D)$, consisting of nodes $C_{x,y,d}$ for each disparity $d \in D$ and spatial location $(x, y), x \in X, y \in Y$. Nodes act independently and are connected to two types of neighbourhoods: supporting Φ and inhibitory Ψ. The first one (Equation 5.6) implements the smoothness assumption, including support from nodes at the same disparity plane and within a given radius *swin*. Function $\Phi(x, y)$ returns indices (x', y') of nodes in a supporting neighbourhood of the node at position (x, y).

$$\begin{aligned} \Phi(x,y) : (x',y') \mid & \ |x - x'| < swin \\ \wedge & \ |y - y'| < swin \end{aligned} \tag{5.6}$$

The second one, defined in Equation 5.7, realises the uniqueness assumption through competition between the candidate nodes along the disparity dimension. Function $\Psi(d)$ returns indices d' of nodes in an inhibitory neighbourhood of the cooperative node at position (x, y, d).

$$\Psi(d) : \ d' \mid d_{min} < |d - d'| < d_{max} \tag{5.7}$$

The cooperative network is constantly changing as the events are generated. Once matching candidates for an event e are mapped to the network with initial weights C^* from Equation 5.5, the affected nodes $C_{e_x,e_y,d}$ are updated as follows:

$$C^{n+1}_{e_x,e_y,d} = \sum_{x',y' \in \Phi(e_x,e_y),d} C^n_{x',y',d} * C^*_{e_x,e_y,d} - \varepsilon \cdot \sum_{x,y,d' \in \Psi(d)} C^n_{x,y,d'} \tag{5.8}$$

Each event contributes to the local neighbourhood at different disparity levels. Additionally, the node weights decay in time if they are not updated. This is achieved by update of the whole network, performed in predefined periods of time.

5.2.1 Proof of Concept: Experimental Results

In order to verify the proposed adaptive cooperative network, first tests have been performed using synthetic data depicting different configurations of moving edges. We used three sequences: (a) *edge20*, which consists of one moving edge at a disparity of 20 pixels; (b) *2edges*, with two edges at different disparities (5 and 20 pixels) moving in two opposite directions; (c) *changDisp*, with an edge of changing disparity (from 5 to 20 pixels). The synthetic datasets are presented in Figure 5.4.

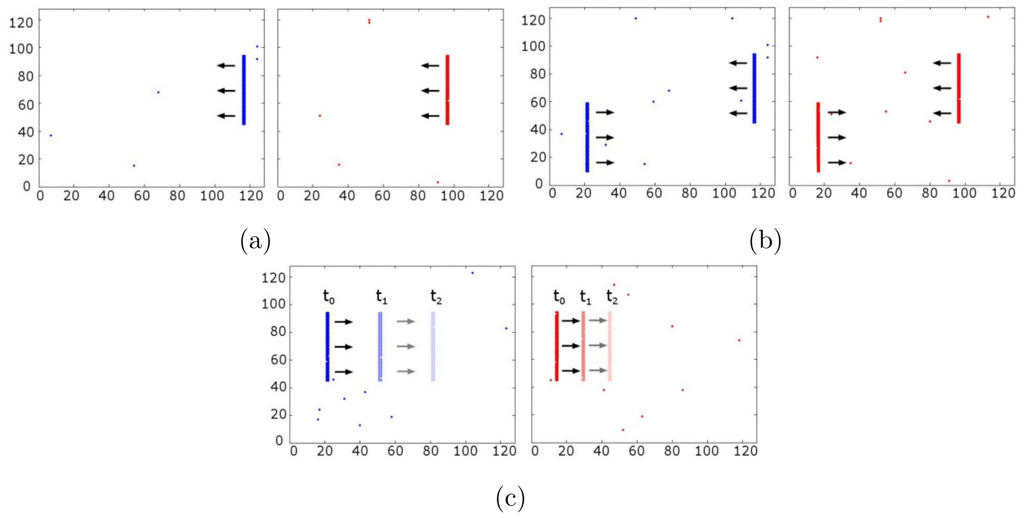


Figure 5.4: Synthetic datasets used in the experiment: (a) *edge20*, (b) *2edges*, (c) *changDisp*. The left view is depicted in blue and the right in red. The direction of motion is indicated by arrows. Additionally, in (c) the change of disparity over time is illustrated by showing the position of edges at three different timestamps (t_0, t_1 and t_2).

A summary of the tests on synthetic data is listed in Table 5.1. The results are given by the accuracy and performance. The accuracy is defined as a percentage of events where

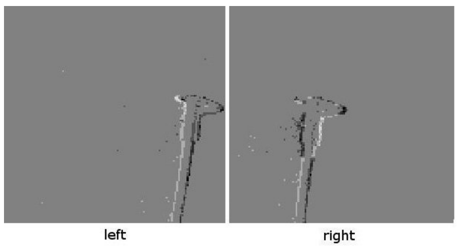
	#events	accuracy %	performance time(s)	performance Ev/s
edge20	51510	98	54.33	953.89
changDisp	51106	97	54.33	940.73
2edges	103020	95	120	858.5

Table 5.1: Results of the proposed cooperative stereo algorithm on synthetic test data.

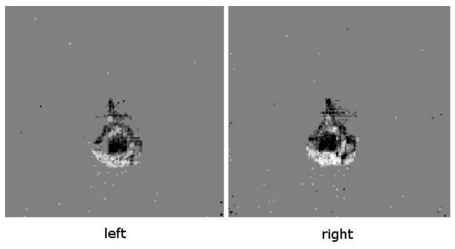
disparity agrees with ground truth, considering up to one pixel difference as a correct result. Our algorithm achieves more than 95% of accuracy. Considering the algorithm's performance, we need to take into account that the processing time is dependent on the events' rate (number of events per second), which may vary across the sequences.

Moreover, the complexity of a scene can influence the processing time due to the higher amount of possible matches to filter out. Therefore, we measured the performance by the number of events that could be processed in one time unit (1 second) and this varies around 900 events per second (Ev/s). These preliminary tests on synthetic data indicate that the proposed *Coopv1* algorithm is suitable for processing events and performing asynchronous stereo matching.

As a next step of the proof of concept, we performed tests of the *Coopv1* algorithm on real DVS recordings which are presented in Figure 5.5. We used two sequences depicting a single object in the scene, a *tool* moving at uniform disparity ($d = 60$) and *person* depicting walking person captured from the sensor in the overhead position.



(a)



(b)

Figure 5.5: Selected frames of the test event sequence *tool* (a) and *person* (b).

The obtained disparity maps were compared with the results of the algorithm proposed in [133] which uses the conventional NSAD matching on image-like event data representation. As we can observe in Figure 5.5a, the corresponding edges have opposite polarity. If polarity is used as a constraint in the stereo algorithm, the correct (true) matches are eliminated and not considered in matching. That leads to wrong disparity estimation as visible in Figure 5.6b. In the *Coopv1* algorithm, we employed a polarity confidence ($pconf$) parameter, hence events of the opposite polarity are still considered in matching but with lower confidence. Therefore, in this case, we obtain better results than [133] which is visible in the disparity histogram. In Figure 5.6c we can observe that depth estimates form high and narrow peak at the correct disparity, whereas in Figure 5.6d the results tend to vary more, and we can distinguish two additional peaks (around disparity 50 and 70), caused by the polarity mismatch.

The results of stereo matching applied to the *person* sequence are depicted in Figure 5.7. In this sequence, we are dealing with a complex object movement with non-uniform disparity. Since the sensor has been placed in an overhead position, we can see that the head of the moving person is the closest to the sensor, so it has a higher disparity (yellow–red) than the shoulders or legs (green–blue). We can observe that the *Coopv1* algorithm is capable of recognising different disparity levels even for quite challenging object shapes. However, it tends to smooth the transition between disparity levels.

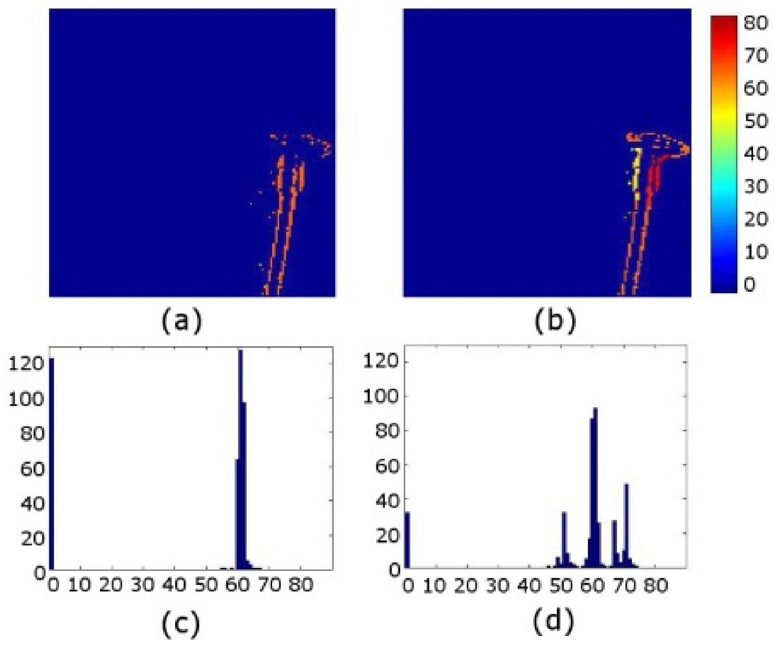


Figure 5.6: Results of the proposed *Coopv1* algorithm applied to the *tool* sequence (a). The results are displayed in image-like representation for visual comparison with the algorithm given in [133] (b). The depth information (disparity in pixels) is encoded in colour. Additionally, the same results are presented by disparity histograms for the *Coopv1* (c) and the algorithm presented in [133] (d). The histograms show the number of events assigned to individual disparities.

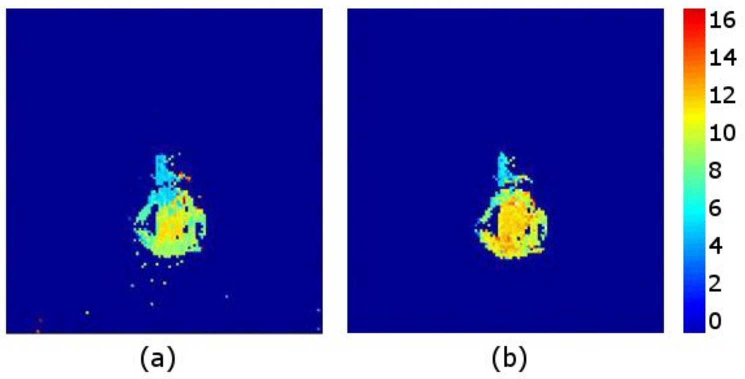


Figure 5.7: Results of the proposed *Coopv1* algorithm applied to the *person* sequence. Comparison of results displayed in image-like representation for our algorithm (a) and the algorithm proposed in [133] (b). The disparity is encoded in colour.

5.3 Enhanced Cooperative Stereo Matching Algorithm

In this section, we investigate the possibility to improve the accuracy of the cooperative stereo by a more reliable way of assigning the initial matching scores to the events, before they are mapped to the cooperative network. As demonstrated in Section 4, event-based stereo matching becomes very challenging while dealing with more complex scenes and uncontrolled environment conditions. Therefore, simple time-based event-to-event matching is in some cases prone to errors and the cooperative optimisation, though effective, might not be sufficient to handle higher ambiguities in matching. In other words, if incorrect initial weights are mapped to the network, then the error might be propagated over time, leading to wrong depth estimates. In this section, we present an enhanced cooperative stereo algorithm, further referred to as *Coopv2* algorithm, which incorporates event-based matching using events' local spatiotemporal neighbourhood.

Spatiotemporal Context in Event-based Matching

Previously, in *Coopv1* algorithm, we have used single event as a matching primitive and measured the similarity based on the timing and polarity of events, as defined in Equation 5.3. We investigate whether using the spatiotemporal neighbourhood of the event can improve the stereo results. The similarity is established per each incoming event, hence, in this way the asynchronous stereo processing is still feasible.

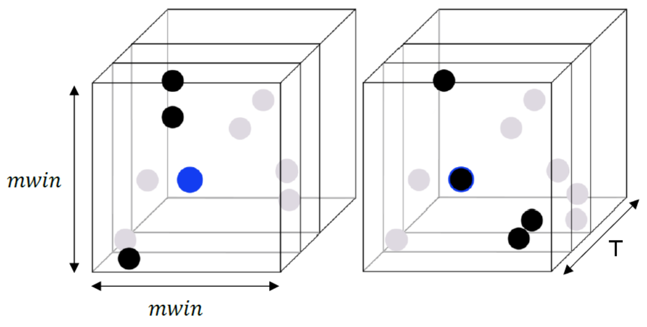


Figure 5.8: Matching events by their spatiotemporal neighbourhood of size $mwin \times mwin \times T$.

As depicted in Figure 5.8, events are considered within their local spatiotemporal neighbourhood. The neighbourhood N_e of event e is described by Equation 5.9. The radius of matching window $mwin$ is given as an algorithm parameter. The temporal dimension of the neighbourhood is limited by T, which is correlated with parameter α, i.e. the matching candidates of significantly low time-based matching score (Equation 5.3)) are discarded.

$$\begin{aligned} N_e = \{n \mid n \in E, \|e_x - n_x\| < mwin, \|e_y - n_y\| < mwin \\ \|e_t - n_t\| < T\} \end{aligned} \tag{5.9}$$

The concept of matching events within their spatiotemporal context is explained by the example illustrated in Figure 5.9. Let there be an event e generated by the left sensor at time $e_t = t4$, at pixel location $(e_x, e_y) = [8, 2]$ with a given history of previous events. There are four matching candidates in the opposite view, at disparities $d = \{2, 3, 4, 5\}$, that is at pixel positions $x = \{6, 5, 4, 3\}$, and $y = 2$. The polarity of the events is indicated by greyscale value, white for ON and grey for OFF events. The reference event (depicted in green) is an OFF event, i.e. $e_p = 0$. The correct match is the candidate $(e_x, e_y, e_t) = (3, 2, t3)$ at disparity $d = 5$, which can be inferred from the similarity in neighbourhood pattern such as matching OFF edge. We can compare the reference event

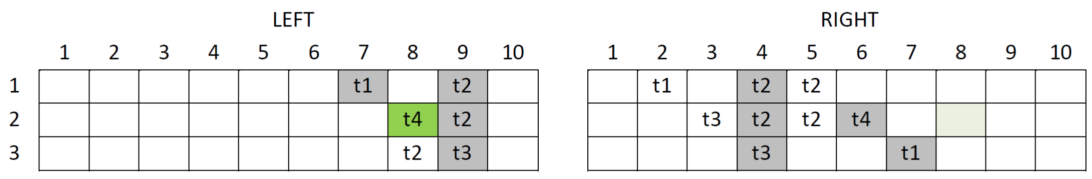


Figure 5.9: Example illustrating the window-based event matching; the history of events in the right and left view.

with its matching candidates considering a 3×3 neighbourhood. The possible comparisons are listed in Figure 5.10, depicted as a difference between each candidate and reference event windows.

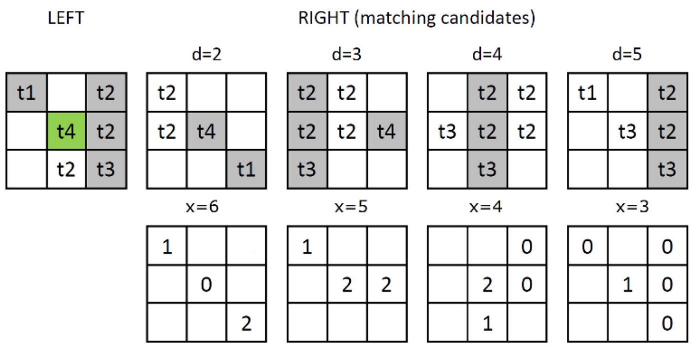


Figure 5.10: Reference event (in green) within its local neighbourhood and matching candidates from the opposite view (top). Time difference between candidate and reference event (bottom).

An example of commonly used similarity measure (matching cost) in window-based stereo is a sum of the pairwise differences in pixels intensity values, Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD). Since measuring the difference in intensity does not apply in matching events, the time and polarity constraints are used. In Figure 5.10, the difference between events within regarded neighbourhoods is expressed numerically, whenever the subtraction of event timestamps is possible to compute, i.e. there is an event available at corresponding locations (addresses). We can observe that mere time-based event-to-event matching, as used in the *Coopv1* approach, would incorrectly assign the highest matching score to the candidate at disparity $d = 2$. On the contrary, using a simple sum of absolute differences over the neighbourhood already allows determining the correct candidate at $d = 5$ by the lowest aggregated matching cost.

In the proposed *Coopv2* algorithm, for each of the matching candidates m, the window-based matching score ρ_{win} is defined in Equation 5.10.

$$\rho_{win}(N_e, N_m) = \frac{\sum\limits_{\{(e,m)\in N_e\times N_m | e_x = m_x \wedge e_y = m_y\}} \rho(e,m)}{|N_e|} \tag{5.10}$$

Individual scores are calculated between each of the events in N_e and the corresponding event at the same position in the candidate's neighbourhood N_m. As can be seen, we are still using Equation 5.3 to compute the similarity between single events within neighbourhoods. The matching scores are afterwards summed and normalised by the amount of events in N_e. The normalisation is necessary to address situations where for the reference event's neighbourhood there are no corresponding events in the candidate's neighbourhood, i.e. $\{(e,m) \in N_e \times N_m | e_x = m_x \wedge e_y = m_y\} = \emptyset$. We demonstrate this

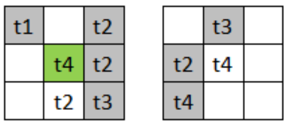


Figure 5.11: Situation where high matching score (without normalisation), does not reflect the similarity of the matching candidate.

situation in Figure 5.11 where none of the events in the reference window can be compared with the candidate window, thus, without normalisation, the resulting matching score equals single time-based correlation. In this case, the achieved high matching score does not reflect the actual similarity of two windows. Therefore, using normalisation allows also for checking the edge alignment within the compared neighbourhoods.

Cooperative Optimisation

Analogically to the *Coopv1*, the second step of the algorithm is the optimisation performed by the cooperative network. For each incoming event, the set of possible matching candidates is weighted by the similarity of their spatiotemporal neighbourhoods, as defined earlier in Equation 5.10. The calculated matching scores are mapped to the cooperative network as follows:

$$\begin{aligned} \forall_{m\in M_e} \quad C^*_{e_x,e_y,d} =& \rho_{win}(N_e, N_m), \\ & \text{where } d = |e_x - m_x| \end{aligned} \tag{5.11}$$

The cooperative nodes are organised into two types of neighbourhoods, supporting Φ and inhibitory Ψ, as defined in Equation 5.6, and Equation 5.7, respectively. Once candidates of an event e are mapped to the network with initial weights C^* from Equation 5.11, the affected nodes $C_{e_x,e_y,d}$ are updated as follows:

$$C^{n+1}_{e_x,e_y,d} = \left(\frac{\sum\limits_{x',y' \in \Phi(e_x,e_y),d} C^n_{x',y',d} * C^*_{e_x,e_y,d}}{\sum\limits_{x,y,d' \in \Psi(d)} C^n_{x,y,d'}} \right)^{\varepsilon} \tag{5.12}$$

We have slightly adjusted the cooperative update function from Equation 5.8, based on conclusions from [177]. Normalisation is achieved by division by the sum of nodes from the inhibitory neighbourhood. The parameter ε controls the amount of inhibition applied to the cooperative nodes values.

Furthermore, in *Coopv1* algorithm noise events were expected to be filtered out by a density threshold incorporated directly in the cooperative network. This method however, was found not to be reliable. In the *Coopv2* algorithm, the additional noise removal filter is employed at the stage of mapping an event to the cooperative network, taking into account the initial weights of the matches and density of the local neighbourhood.

5.4 Summary

In this chapter, we presented two event-based stereo matching algorithms, namely *Coopv1* and *Coopv2*. The first one, *Coopv1*, is focused on finding a feasible model for asynchronous event-based stereo matching. The stereo matching is modelled with a dynamic cooperative network where nodes are organised into two types of neighbourhoods: supporting and inhibitory. Through the local node operations, the cooperative network is performing the optimisation based on smoothness and uniqueness assumptions. The network is adaptive, i.e. it is updated locally upon each event from the input stream, thus adjusting dynamically to the characteristics of the object (e.g. speed, size, appearance). The cooperative network is considered as a refinement for single event-to-event matching as it not only uses the temporal similarity to match events but also applies spatiotemporal smoothness constraint. The second, *Coopv2* algorithm, was developed to improve the accuracy of initial matching score function, thus to address the challenge of tackling ambiguity in event-based correspondence search. We proposed an enhanced cooperative stereo matching technique that calculates similarity over a local neighbourhood of each event pair to compute initial matching weights for the cooperative optimisation.

CHAPTER 6

Evaluation

In this section, we present an extensive evaluation of both *Coopv1* and *Coopv2* stereo matching algorithms. A common way to evaluate stereo algorithms in conventional computer vision is to use available public stereo benchmarks, e.g. Middlebury[1], KITTI[2] or ETH3D[3]. The quality of the stereo method is assessed using a set of ground-truth images and defined metrics of accuracy and performance. However, the well-established benchmarks do not apply in the context of event-based vision. There are several datasets of DVS recordings which are focused on different aspects of event-based vision, e.g. driving datasets [10], optical flow [122], action recognition [63] or pose estimation and SLAM [104]. Until very recently, there have not been any public datasets facilitating the ground-truth evaluation of stereo DVS sequences. The announced multi-vehicle stereo event dataset [176] is filling this gap.

In this work, we applied the method for ground-truth evaluation of DVS algorithms proposed by Kogler et al. [70]. We use the dataset which was made available by the authors for our assessment. We provide a comparative evaluation with competing stereo algorithms. Furthermore, we investigate the influence of the matching function on the overall result of the cooperative stereo. The ground-truth evaluation is accompanied by a series of qualitative results obtained from recorded sequences of human motion. Section 6.2 delivers experiments that demonstrate the robustness of our cooperative stereo algorithms with respect to varying scene complexity and different sensor setup (static and dynamic). We use synthetic data to create challenging test cases, e.g. simultaneous very fast and slow motion, or cluttered scenes. Finally, we also apply our algorithms to a moving camera setup using the pose estimation dataset [104]. The synthetic scenes and three data recordings are used to evaluate whether the cooperative stereo approach also applies in dynamic setup scenarios.

[1]http://vision.middlebury.edu/stereo/data/
[2]http://www.cvlibs.net/datasets/kitti/
[3]https://www.eth3d.net/

Results Assessment and Visualisation

Throughout this chapter, we use several methods to assess the quality of stereo matching results, in both visual and numerical ways. The numerical results are expressed with the following metrics:

Mean distance error which is the average error between the reference (ground truth) and estimated depth at a particular location. It is important to note that all error values are calculated only for locations where ground truth is available. The mean distance error is given in meters.

Relative distance error which is defined as a percentage of the mean distance error with respect to the depth range of the scene.

Matching rate which shows how many events from the input stream were assigned with disparity values. The matching rate is a real value from the $[0, 1]$ interval.

As mentioned in [69], considering only the mean distance error for accuracy assessment might be misleading. Low values of the mean error do not always confirm that the algorithm performs well, since there might be only few events matched which contribute to the error calculation. High matching rate and low mean distance error should be considered together as indicators of an accurate stereo matching algorithm.

Additionally, the quality of the stereo matching will be presented through two types of visualisation: (1) disparity (depth) map, in which the events are plotted with their disparity (depth) encoded in colour, and (2) disparity (depth) histograms, to roughly assess the general distribution of disparity (depth) levels in the scene.

6.1 Algorithm Analysis

In this section, we present a quantitative assessment of the proposed cooperative stereo matching algorithms in terms of parameter selection, accuracy and comparison to competing algorithms. The ground-truth dataset [70] comprises three test scenes,denoted as Scenario A, B, and C, as shown in Figure 6.1.

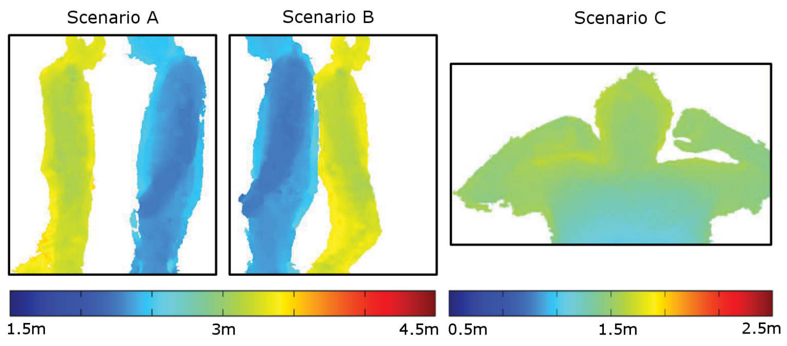


Figure 6.1: Ground truth test scenarios [70].

All test scenarios present indoor scenes captured by a stationary stereo camera setup. In Scenario A and B, two people are walking in opposite directions at different distances from the camera, with a small amount of occlusions present in Scenario B. Scenario C depicts a person sitting relatively close to the camera and moving the upper body and arms. The ground-truth depth for the event data was generated using a conventional stereo system as reference, and its accuracy is estimated to be better than 0.027m for distances up to 2m, with errors increasing up to 0.117m at a distance of 3.5m [70]. For the calibration and rectification of the silicon retina stereo sensor, the calibration toolbox of Bouguet [18] was used, as described in more detail in [46].

As already mentioned before, we distinguish two steps in the proposed cooperative algorithms, the first one being the stereo matching function, and the second one the cooperative network optimisation. To get a better understanding of the parameters

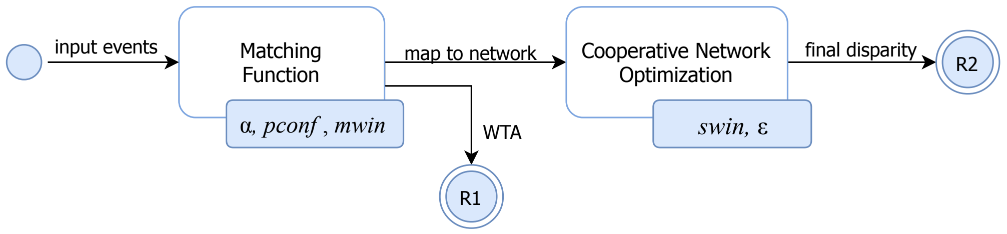


Figure 6.2: Overview of the proposed cooperative algorithms. Results are measured after each step (*R1* and *R2*). The parameters considered in analysis are shown for each step.

and performance of each step of the algorithm, we analyse the output of both steps individually, as indicated in Figure 6.2. *R1* is the result of the matching function, where the initial matching scores are calculated and the best matching candidate (the one with the highest weight) is considered correct. The results of the second step (*R2*) are the overall results of the cooperative stereo matching, where the initial weights are fed into the optimisation network. In the analysis, we consider the proposed cooperative algorithms, *Coopv1* and *Coopv2*, as presented in Chapter 5. We take into account the parameters of the matching function $\{\alpha, pconf, mwin\}$ and cooperative network $\{swin, \varepsilon\}$, which are defined in Section 5.2 and 5.3.

In the following subsections, we investigate *(i)* how different parameters influence the results for each variant of the algorithm, *(ii)* what the optimal values of parameters per each algorithm are, and *(iii)* how the proposed algorithms perform compared to other competing algorithms in the field.

6.1.1 Matching Function Analysis

The matching function, being the first step of the algorithm, finds a set of matching candidates for each incoming event. There are three parameters of the matching function, namely: *(i)* α, which is a factor for measuring the correlation of events in time, *(ii) pconf*, which measures confidence in polarity matching, and *(iii)* $mwin$, which is the size of the matching window. In what follows, we compare how the above-mentioned parameters influence the accuracy of the matching function (*R1*) and overall cooperative stereo algorithms (*R2*). If not stated otherwise, the default parameters settings used in the matching function analysis are as follows: *mwin* and *swin* are set to 9×9, *pconf* is set to 0, α to 0.05, ε to 0.05 and 0.75 for *Coopv1* and *Coopv2*, respectively.

Time Correlation

We start our experiments with an analysis of the parameter α, which is used in the time-based matching score for each matching candidate, as defined in Equation 5.3. The parameter α controls the magnitude of the score depending on the time difference in the events' timestamps. In order to assess the influence of α on the algorithms' performance, we observe how the measured mean distance error varies with different parameter values set to $\{0, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$. The results calculated for all three ground-truth scenarios are shown in Figure 6.3.

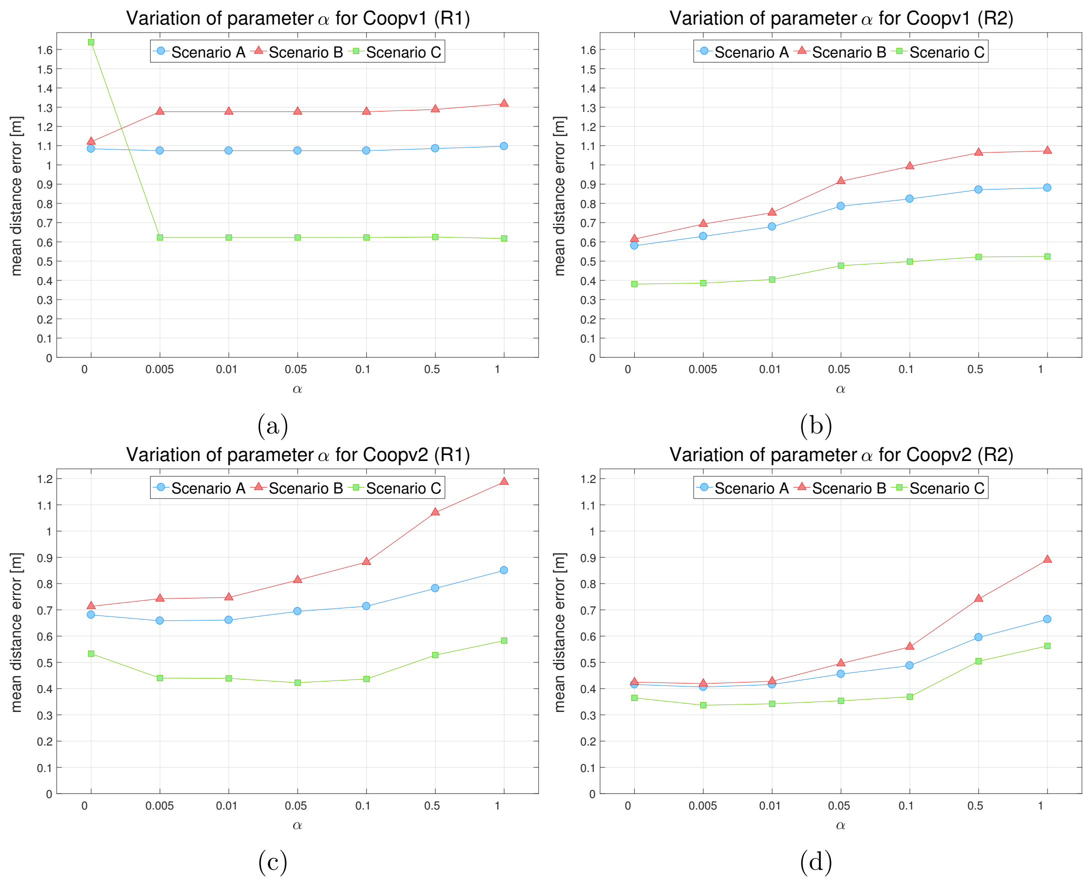
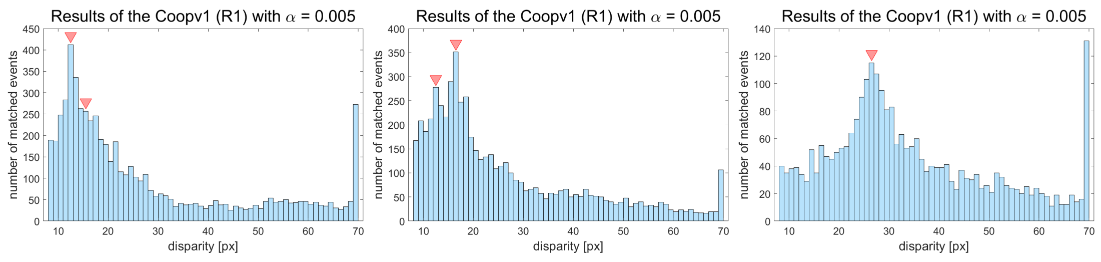


Figure 6.3: Influence of parameter α on the results of the matching function (*R1*) (a,c) and on the overall results after cooperative optimisation (*R2*) (b,d) for the *Coopv1* and *Coopv2* stereo matching algorithms, respectively.

The results show that the mean error increases with higher values of α, hence less restrictive time function. We can assume that the variation of time difference between corresponding events is rather high and it is beneficial to perform less restrictive time correlation. Judging from the results, Scenario B seems to be more challenging than the other two scenarios, as it tends to achieve the highest mean error. Moreover, the lowest error for this scenario can be noted for $\alpha = 0$, hence without time correlation. In the case of Scenario A and C, using time correlation with small α value achieves the best matching function results. Comparing the matching function results (*R1*) with the overall results (*R2*), we can clearly see that the mean error is substantially reduced by the cooperative optimisation. In general, the *R2* results show a steady increase of error with higher values of α, except of the *Coopv2* results in Scenario C, where the best accuracy is achieved with $\alpha = \{0, 0.005\}$ (see Figure 6.3d). Although the optimisation performs very well in reducing ambiguities introduced by temporal correlation, we look at the trends in *R1* to pick the optimal value of $\alpha = 0.005$. As we have noticed before in Scenario B, the time correlation may introduce additional errors into results of matching function (*R1*) and worsen the overall accuracy of the algorithm. In Figures 6.3a–6.3c we can observe that the time constraint improves results only for Scenario C, whereas A and B show slightly higher error values. Omitting the time constraint, i.e. $\alpha = 0$, seems to have little effect on the overall results (*R2*). Even though all matching candidates are assigned with the same matching score, the cooperative optimisation can still achieve reasonable results. This leads to the conclusion that the accuracy of the proposed algorithms is mostly dependent on the cooperative network mechanisms and employed smoothness assumption. We investigate whether the timings of corresponding events convey useful information or just increase the ambiguity in event-based stereo matching. In Figure 6.4 we present histograms of disparity estimates *R1* for the *Coopv1* algorithm with $\alpha = 0.005$.



(a) Scenario A (b) Scenario B (c) Scenario C

Figure 6.4: Results of the matching function for *Coopv1* (*R1*) presented by the histogram of calculated disparities for all test scenarios.

For Scenario A and B, two peaks corresponding to two people walking at different distances are expected in the disparity distribution (indicated by the red arrows). Looking at the histograms, however, the peaks are not very pronounced in Scenario A (Figure 6.4a), and slightly more visible in Scenario B (Figure 6.4b). On the contrary, in Scenario C (Figure 6.4c) the distribution clearly describes the underlying visual data, i.e. one person that spans over a particular depth range. Summing up, the histograms indicate that correlation between events is conveyed in events timestamps, however, the ambiguities in temporal matching increase with the complexity of the scene. Therefore, using additional constraints is necessary to make the event-based stereo matching more reliable.

Polarity Constraint

The polarity of events is taken into account by considering the polarity constraint in the matching function. In the proposed algorithms, the events of opposite polarity are not totally discarded but their weights are decreased according to the polarity confidence, i.e. *pconf* parameter, as defined in Equation 5.4. The higher the value of the *pconf* parameter, the more confidence we put in matches of the opposite polarity. When the *pconf* parameter equals zero, the events of the opposite polarity are not considered in matching. In Figure 6.5 the results of the stereo algorithms *Coopv1* and *Coopv2* are shown for each of the scenarios and varying *pconf* parameter.

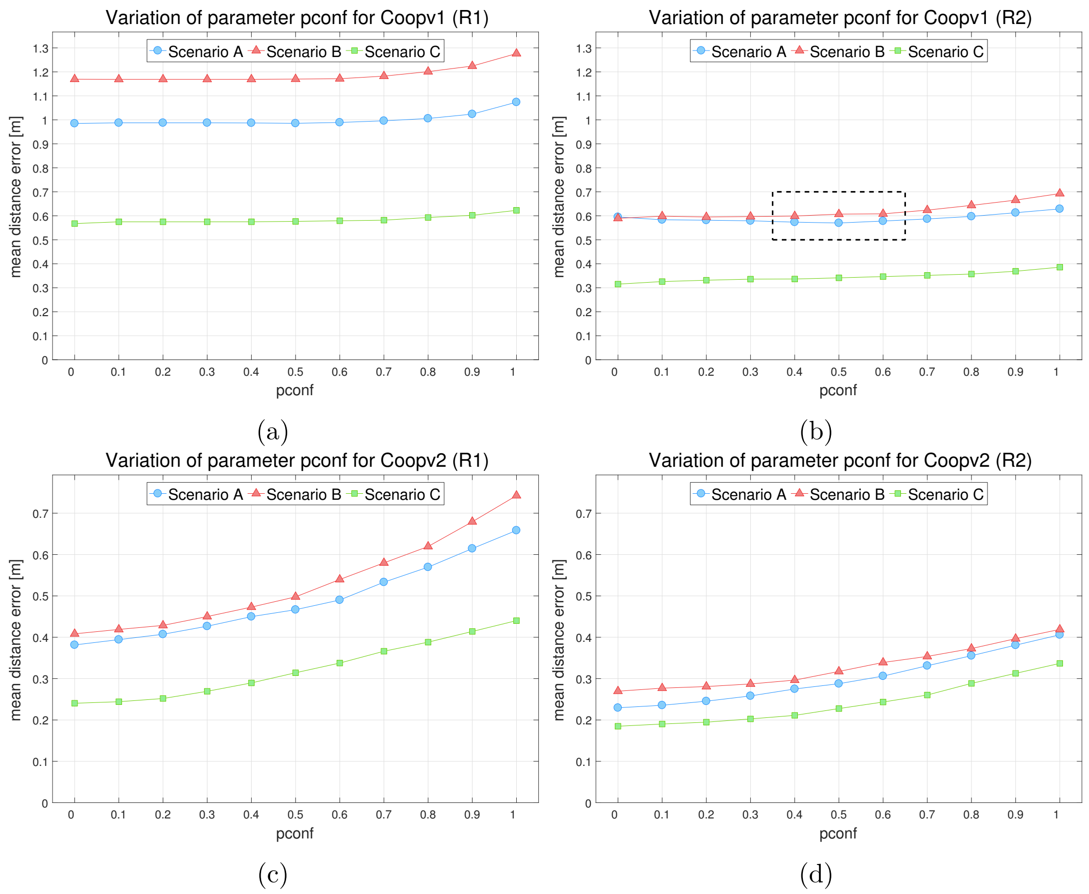


Figure 6.5: Influence of parameter *pconf* on the results of the matching function (*R1*) (a,c) and the overall results after cooperative optimisation (*R2*) (b,d) of the *Coopv1* and *Coopv2* stereo matching algorithms, respectively.

Looking at the *Coopv1* plots (Figure 6.5a–6.5b), we observe that influence of the (*pconf*) parameter does not drastically influence neither the matching function (*R1*) nor the overall (*R2*) results. The mean error does not change much until *pconf*=0.6, where a slight increase in error can be noted. Interestingly, for Scenario A, the overall results of *Coopv1* (Figure 6.5b) show a slight improvement for values between *pconf*=0.4 and *pconf*=0.6. This effect has been investigated in further tests, which confirmed that the optimal value of *pconf* is 0.4 taking into account all three scenarios and optimal parameters for the cooperative network, i.e. *swin* set to 39×39. Compared to *Coopv1*, the results obtained by the *Coopv2* algorithm (see Figure 6.5c) show higher influence of the *pconf* parameter on the resulting mean error in both *R1* and *R2*. In fact, a steady increase of error can be noted, therefore for *Coopv2* the optimal value of the *pconf* parameter is set to 0, as it delivers the best results for all regarded test scenarios.

Matching Window

Finally, the *mwin* parameter is analysed for the *Coopv2* stereo matching algorithm. The accuracy of the window-based matching is expected to increase when higher sizes of matching area are used, as they provide a more distinctive representation of matching candidates. We investigate the behaviour of the cooperative network with a matching window ranging from 3×3 up to 63×63.

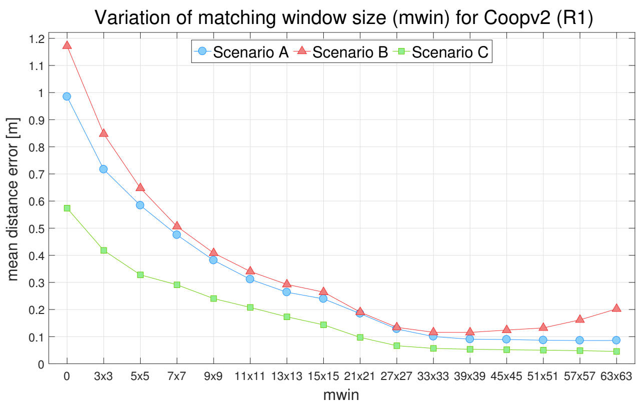


Figure 6.6: Influence of the size of the matching window parameter *mwin* on the results of the matching function (*R1*) of the *Coopv2* algorithm.

The results are shown in Figure 6.6. As expected, the mean error decreases with bigger matching window sizes, reaching an optimum around a window of size 33×33. We can observe that the *Coopv2* algorithm with the optimal *mwin* achieves very accurate results even without cooperative optimisation. This raises the question whether the cooperative optimisation is necessary. We investigate this further in Section 6.1.2 and learn that, even though the numerical results indicate optimal values, the visual results show that adjacent edges of the objects are not well preserved using only the window-based matching function without subsequent optimisation.

Matching Function Variants

The matching cost function can have a big impact on the quality of the overall stereo algorithm results. We have demonstrated how different parameters influence the matching accuracy. Having some intuition about the optimal values of the matching parameters, we test the overall behaviour of the matching function with different combinations of matching constraints. Each parameter is set to its optimal value, as derived above. In Figure 6.7, the results of matching function (*R1*) in terms of mean distance error are

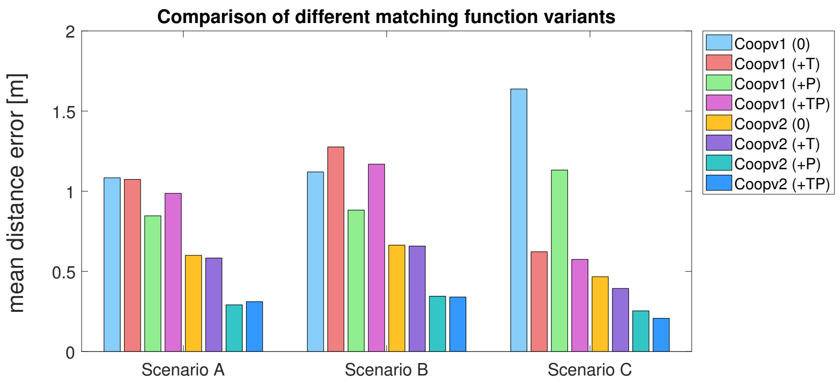


Figure 6.7: Comparison of different variants of matching function, presented for *Coopv1* and *Coopv2* algorithms.

compared for the *Coopv1* and *Coopv2* algorithms. We selected the following configurations of constraints: no constraints (0), time constraint only (+T), polarity constraint only (+P), and combined time and polarity constraints (+TP). We can observe that *Coopv1* achieves better results using only the polarity constraint, than with temporal correlation in Scenario A and B. In Scenario C, however, time combined with polarity constraint improves the results significantly with respect to any other variant of constraints. The results of the *Coopv2* algorithm, even with no constraints, indicate a clear improvement over *Coopv1*. We can also observe that the impact of the polarity constraint on the *Coopv2* results is quite significant. In our further experiments, we use a combination of polarity and time constraints (+TP), with their optimal values per each stereo algorithm.

6.1.2 Cooperative Network Analysis

There are two key parameters within the cooperative network. The first one, *swin*, defines the size of the supporting window (neighbourhood), thus controls the amount of support from the neighbouring cooperative nodes. The second parameter, i.e. ε, controls the amount of inhibition applied to the nodes in the network. Different cooperative update functions are used in the *Coopv1* and *Coopv2* stereo algorithms, thus we do not expect that optimal values of *swin* and ε are the same in both cases. The experimental tests and conclusions are described individually for each cooperative stereo variant. Throughout our experiments we keep the parameters α, *pconf*, d_{min} and d_{max} at constant values of 0.005, 0.4 (for *Coopv1*) or 0 (for *Coopv2*), 7, and 70, respectively.

Supporting Neighbourhood

We start with an analysis of the *Coopv1* stereo matching algorithm, using square support window sizes ranging from 3×3 to 63×63. In Figure 6.8 we can observe that larger sizes of the supporting window result in a significant gain in accuracy. The optimum is achieved around a window size of 39×39. Subsequently, we investigate the influence

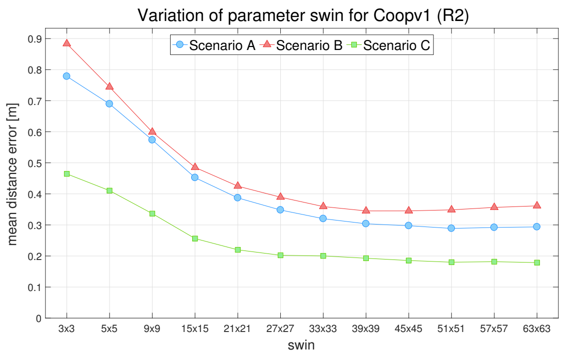
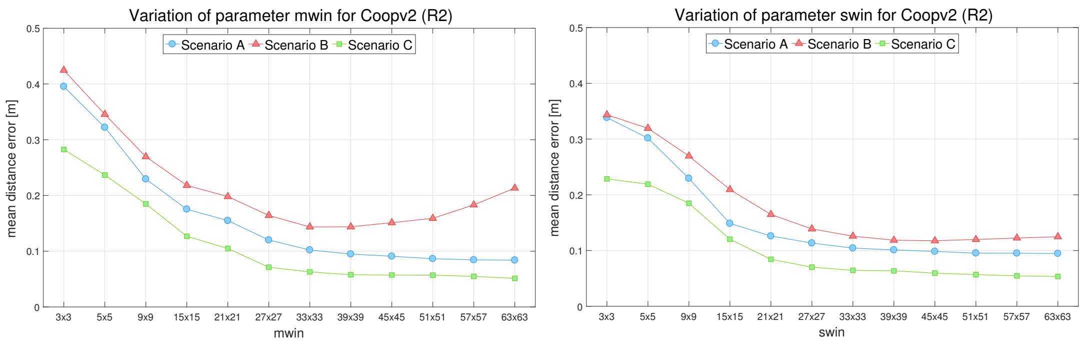


Figure 6.8: Influence of the support window size (*swin*) on the *Coopv1* results.

of the matching and supporting window size on the results of the *Coopv2* algorithm. In Figure 6.9, the window size for one parameter (*mwin* or *swin*) is varied individually from 3×3 to 63×63, while the other parameter is kept constant at 9×9. For Scenario A and C, we can observe an improvement of the accuracy with increasing window sizes over almost the whole tested range. For Scenario B, the plots for *mwin* and *swin* indicate a minimum value for window sizes around 39×39. For values larger than 39×39, further improvements gained for Scenario A and C are also relatively small.



(a) (b)

Figure 6.9: Results of the *Coopv2* stereo algorithm with varying matching window (*mwin*) (a) and supporting window (*swin*) (b) sizes.

The next step of our analysis was to find the best combination of the *mwin* and *swin* values. A well known observation in window-based stereo matching is the so called ‘edge fattening’ problem, caused by larger sizes of the matching window. The matching window should be big enough to cover neighbouring events belonging to the same object, thus efficiently benefiting from the smoothness constraint, and at the same time small enough not to overlap on other, adjacent objects. Dealing with sparse event data, the potential ‘edge fattening’ problem is reduced to situations where two objects are very close to each other (adjacent edges), e.g. in Scenario B. As indicated by the results for Scenario B in Figure 6.9a, after the optimum is reached for *mwin* around 39×39, the mean error starts increasing again with higher *mwin* values. Such an effect is not observed with increasing *swin* values. Therefore, the next test was performed for *swin* set to 39×39 and varying *mwin* values. As we can observe in Figure 6.10, there is an optimum reached around window size *mwin*=11×11.

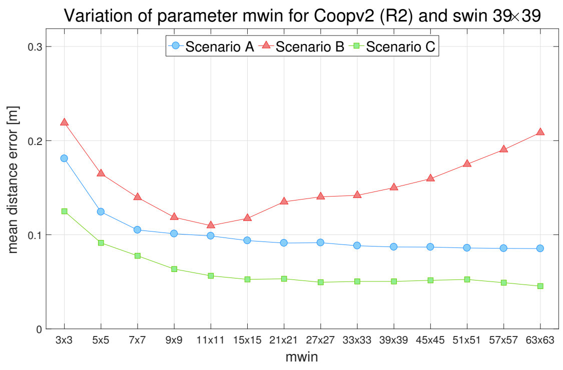


Figure 6.10: Results of the *Coopv2* stereo algorithm with varying matching window (*mwin*) size and supporting window size *swin* set to 39×39.

We investigate in more detail several combinations of *mwin* and *swin*. Table 6.1 gives the mean error (in meters) achieved by the *Coopv2* algorithm with *mwin*={7×7, 11×11, 39×39} and *swin*={11×11, 33×33, 39×39}. During our experiments we have observed that the visual ‘goodness’ of results do not always align with the smallest mean error. That is why three potential parameter combinations have been selected for further inspection, as highlighted in Table 6.1.

swin	11×11			33×33			39×39		
mwin	7×7	11×11	39×39	7×7	11×11	39×39	7×7	11×11	39×39
A	0.221	0.169	0.093	0.106	0.103	0.088	0.105	0.099	**0.087**
B	0.269	0.236	0.142	0.148	0.119	0.148	0.14	**0.11**	0.15
C	0.189	0.145	0.059	0.084	0.06	0.052	0.078	0.056	**0.05**

Table 6.1: Results given in mean distance error [m] of the *Coopv2* algorithm with different sizes of matching window (*mwin*) and supporting window (*swin*). Results are presented for all three test scenarios (A, B, C) and the best result per each scenario is shown in bold.

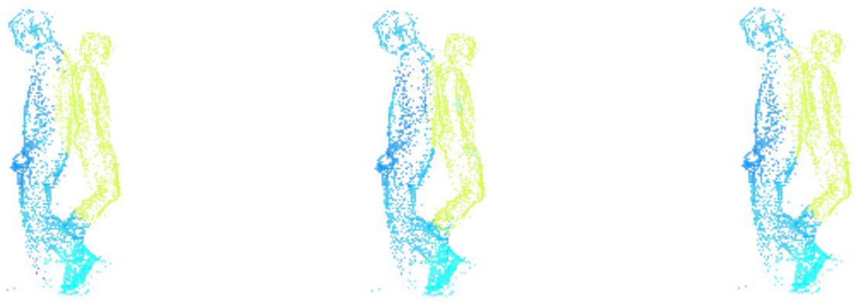
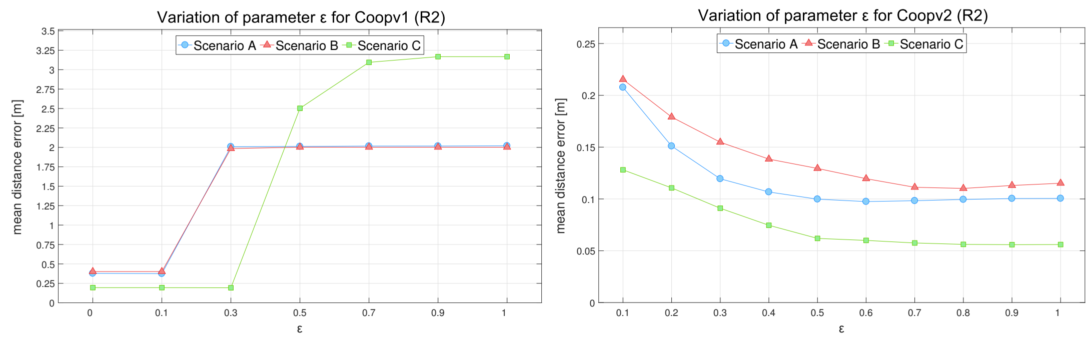

(a) (b) (c)

Figure 6.11: Results of the *Coopv2* stereo matching algorithm on Scenario B with three different window size combinations: (a) $mwin = 39{\times}39$ and $swin = 11{\times}11$ (b) $mwin = 11{\times}11$ and $swin = 39{\times}39$ (c) $mwin = 39{\times}39$ and $swin = 39{\times}39$.

Figure 6.11 depicts the depth maps of the *Coopv2* stereo results on Scenario B. The results clearly show that with high *mwin* values, the adjacent edges of two people are not preserved well. Therefore, in further evaluations, we have selected 11×11 and 39×39 as the size for *mwin* and *swin*, respectively.

Inhibitory Neighbourhood

The second important parameter of the cooperative network is the ε parameter, which controls the amount of inhibition applied to each node in the network. We start with an analysis of the *Coopv1* algorithm. Having set the *swin* parameter to size 39×39, we search for the optimal value of the ε parameter. As depicted in Figure 6.12a, the results show a drastic jump in distance error at a particular ε value, 0.1 for Scenario A and B, and 0.3 for Scenario C. We observed that the best values of the parameter can be found between 0 and 0.1, hence the optimal ε is set to 0.05.



(a) (b)

Figure 6.12: Influence of the ε parameter on results of the *Coopv1* (a) and *Coopv2* (b) stereo matching algorithms.

For the *Coopv2* stereo algorithm, the parameter ε tests are presented in Figure 6.12b. The tests were performed with the parameters *mwin* and *swin* set to 11×11 and 39×39, respectively. We can observe that the mean error decreases with higher values of ε, with an optimum reached between 0.7 and 0.8. Consequently, the ε parameter is set to 0.75.

6.1.3 Comparative Evaluation

We compare results of the proposed cooperative stereo algorithms, *Coopv1* and *Coopv2*, with other frame-based and event-based matching algorithms in Table 6.2. The first algorithm *T-Corr* is a simple event-to-event matching method based on time correlation [72, 69]. The second algorithm *SAD+2SF* was introduced in [71]. It is a frame-based matching approach, which relies on the Sum of Absolute Differences (SAD) as matching function and an additional post-processing technique, the two-stage filter (2SF), that incorporates median filtering. The error rates for *T-Corr* and *SAD+2SF* were taken from the literature [69] and [71], respectively. The third algorithm is the proposed *Coopv1* stereo matching using time-based single event matching. Its parameter *swin* is set to the window size of 39×39. Finally, the results of the enhanced cooperative stereo algorithm, *Coopv2*, with window-based matching used for calculating the initial weights, are listed in the last column of Table 6.2.

	T-Corr [72]	*SAD+2SF* [71]	*Coopv1*	*Coopv2*
Scenario A	0,581	0,119	0.304	**0.089**
Scenario B	0,618	0,222	0.345	**0.152**
Scenario C	0,277	0,088	0.193	**0.069**

Table 6.2: Comparative evaluation of the event-based stereo matching algorithms using the ground-truth test scenarios A, B and C.

As expected, the highest mean distance error in Table 6.2 was found for the basic event-to-event matching technique (*T-Corr*). Comparison of the two right-most columns demonstrates the clear gain in accuracy achieved by the *Coopv2* stereo, with the average error (computed over all three scenarios) dropping by over 50 percent to 0,11m in the final result. Additional window-based matching employed in *Coopv2* caused an increase in runtime of a factor 1.9, when compared to *Coopv1*. The *Coopv2* also outperforms *SAD+2SF* with error differences much smaller in this case. The improvement achieved by *Coopv2* with respect to the initial *Coopv1* algorithm can also be seen on the depth maps in Figure 6.13. The noise visible in *Coopv1* results is noticeably reduced in the results of *Coopv2* algorithm.

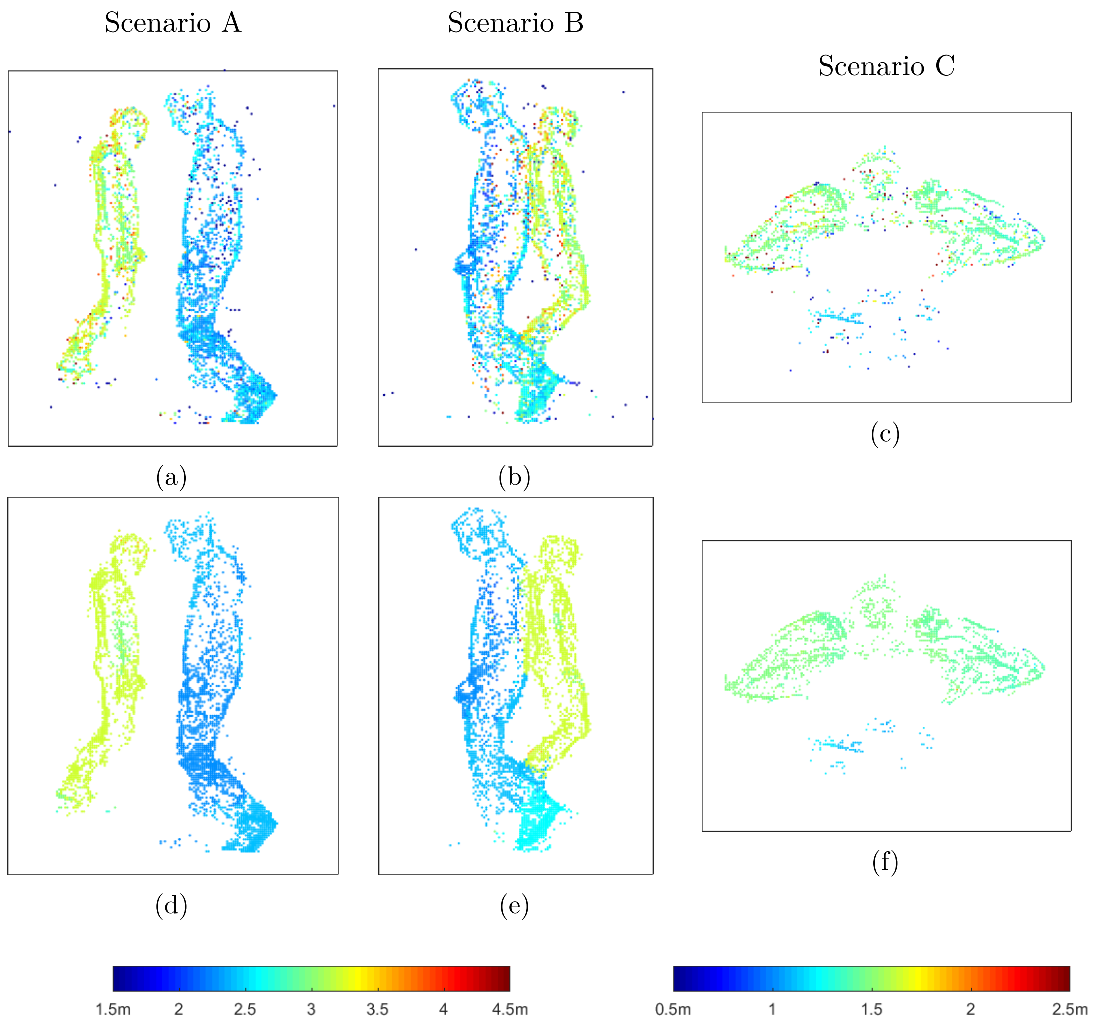
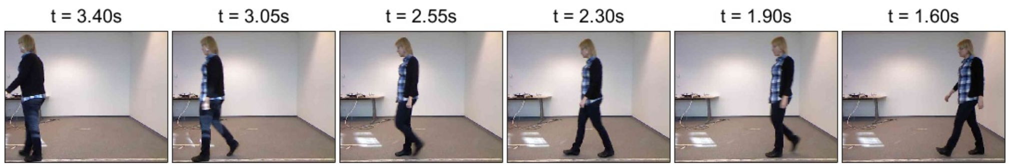


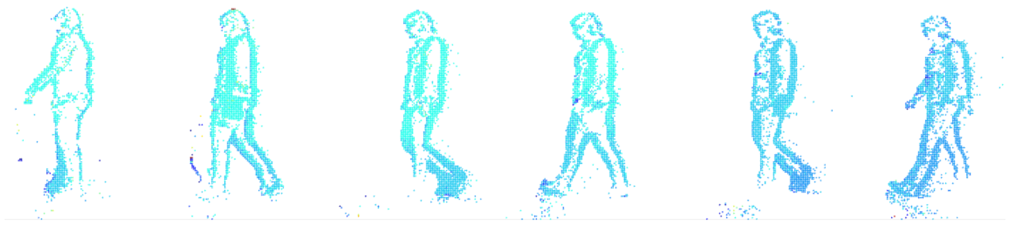
Figure 6.13: Qualitative results (depth maps) of the cooperative stereo matching algorithms: *Coopv1* (a,b,c) and *Coopv2* (d,e,f). The results are presented for all three ground-truth scenarios (Scenario A, B, and C).

6.1.4 People datasets

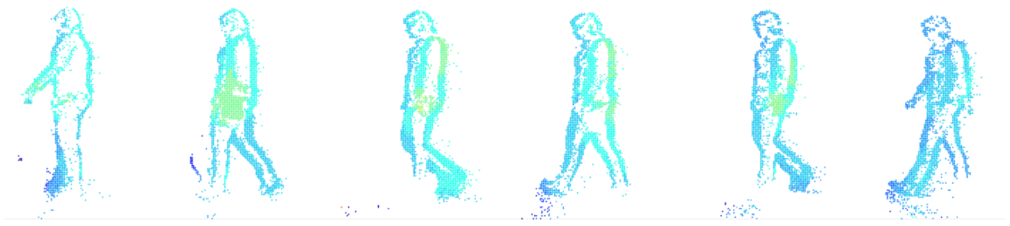
One of the applications intended for the dynamic vision sensor is human motion analysis and people tracking. In this section, we evaluate the proposed algorithms on recordings of walking people. In our experiments, we use several test cases, focusing on different aspects: the first two sequences depict a transition in distance to the sensor of one (*people1*) and two (*people2*) people, whereas the last, *people3*, shows transient occlusions of one person by another. The visualisation used in this section is kept consistent for all test cases. The first row shows the selected key frames from the video camera. Below, the results of the *Coopv1* and *Coopv2* algorithms are presented for the corresponding time steps. In Figure 6.14, the results for the test case *people1-1* is depicted. We can observe



(a) Selected key-frames from *people1-1* data sequence.



(b) Results of the *Coopv1* stereo algorithm.



(c) Results of the *Coopv2* stereo algorithm.

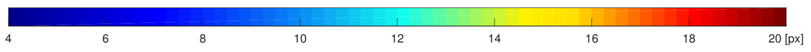
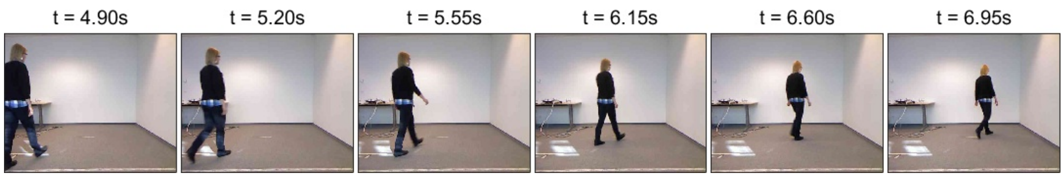
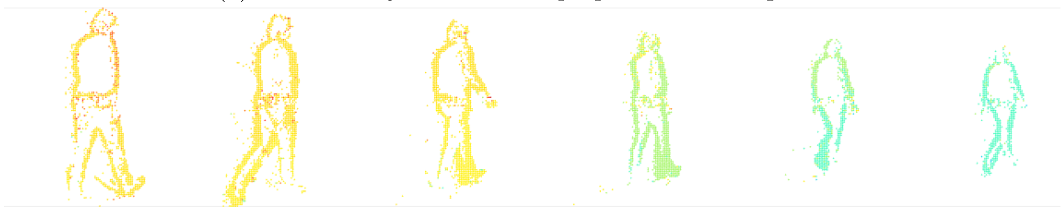


Figure 6.14: Results of the cooperative stereo algorithms on the *people1-1* sequence depicted as colour-coded disparity maps (a) for *Coopv1* (b) and *Coopv2* (c).

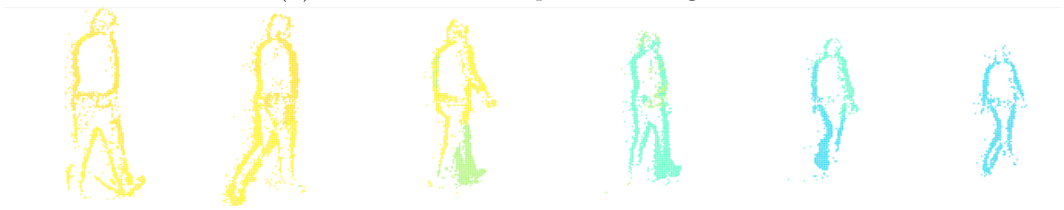
that the *Coopv2* algorithm recognises several disparity levels, e.g., the hand being closer to the camera, whereas the *Coopv1* tends to smooth out such details. Another example can be noticed at time step t = 2.30s, where the frontal leg is correctly assigned with higher disparity (closer to the camera). Further test cases within the *people1* sequence depict a person's transition in distance to the camera, as illustrated in Figure 6.15.



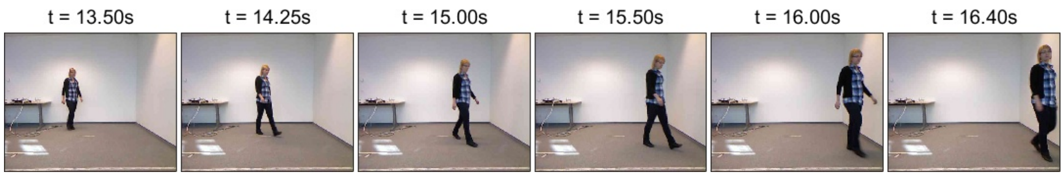
(a) Selected key-frames from *people1-2* data sequence.



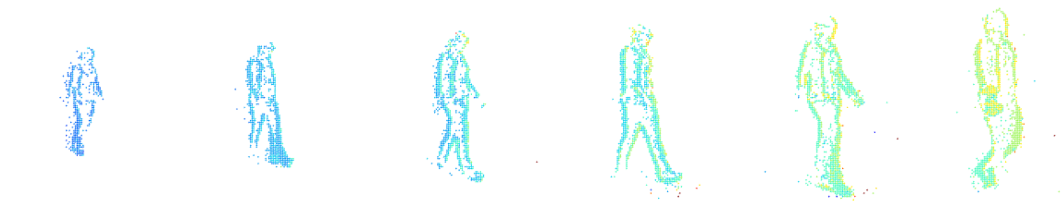
(b) Results of the *Coopv1* stereo algorithm.



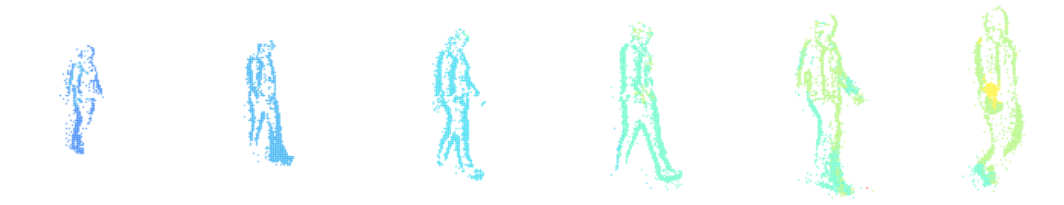
(c) Results of the *Coopv2* stereo algorithm.



(d) Selected key-frames from *people1-3* data sequence.



(e) Results of the *Coopv1* stereo algorithm.



(f) Results of the *Coopv2* stereo algorithm.

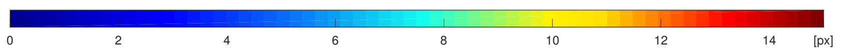
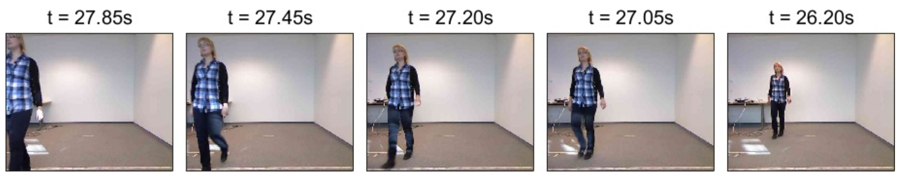


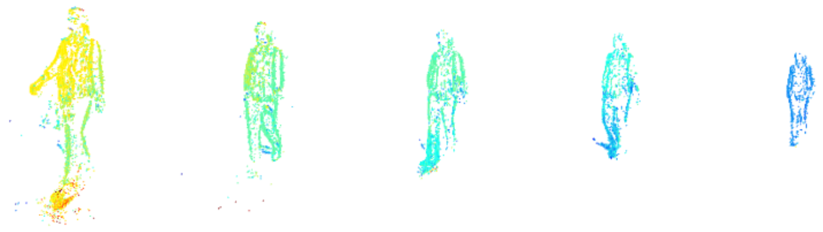
Figure 6.15: Results of the cooperative stereo algorithms on the *people1-2* (a) and *people1-3* (d) sequence depicted as disparity maps for *Coopv1* (b,e) and *Coopv2* (c,f).

Both versions of the cooperative stereo algorithm handle the transition in depth very well, however, the *Coopv2* gives slightly better results in terms of details (e.g. leg in Figure 6.15c at time step 5.55s or hand at time step 16.40s in 6.15f).

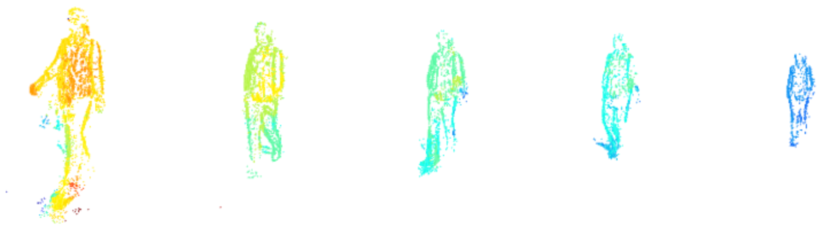
The last part of the *people-1* test case is presented in Figure 6.16. In this example, the person walks towards the sensor up to quite close distance (time step t=27.85s). We can observe that dealing with closer distances, thus higher disparity levels, there is a difference between disparities estimated by the *Coopv1* and *Coopv2* algorithm. The *Coopv1* tends to recognise disparity as lower than *Coopv2*. Moreover, we can observe the *Coopv1* results contain some mismatches within the objects.



(a) Selected key-frames from *people1-4* data sequence.



(b) Results of the *Coopv1* stereo algorithm.



(c) Results of the *Coopv2* stereo algorithm.

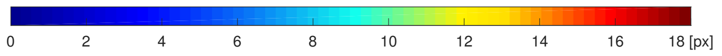
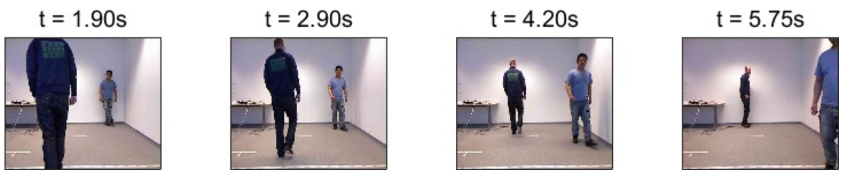
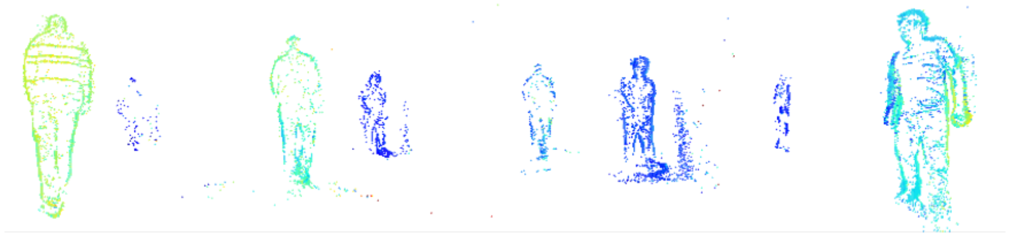


Figure 6.16: Results of the cooperative stereo algorithms on the *people1-4* (a) sequence depicted as disparity maps for *Coopv1* (b) and *Coopv2* (c).

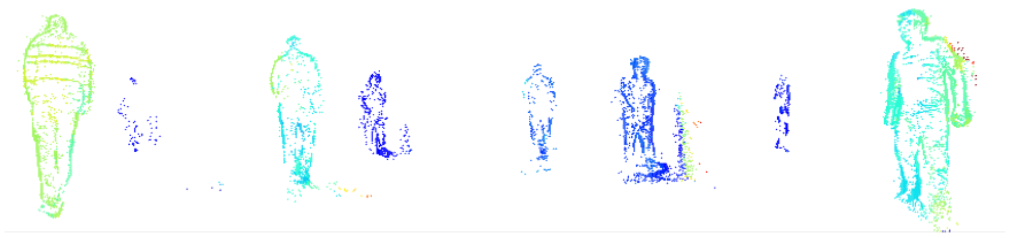
The second test case, *people2*, shows two people approaching and moving away from the sensor. The results of *Coopv1* and *Coopv2* are shown in Figure 6.17. We can observe that both *Coopv1* and *Coopv2* successfully recognise objects at two different disparities. Both algorithms produce some mismatches as well, e.g. the foreground person at time step 5.75s (last frame). There is a slight difference in the character of mismatches; the *Coopv1* results tend to contain single mismatches within the objects, whereas the error



(a) Selected key-frames from *people2-1* data sequence.



(b) Results of the *Coopv1* stereo algorithm.



(c) Results of the *Coopv2* stereo algorithm.

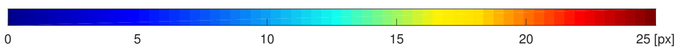
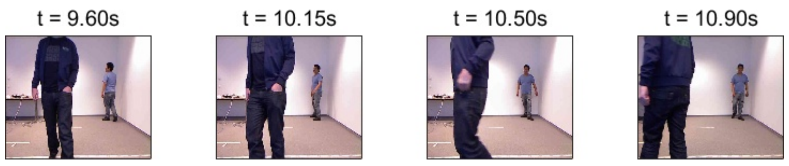


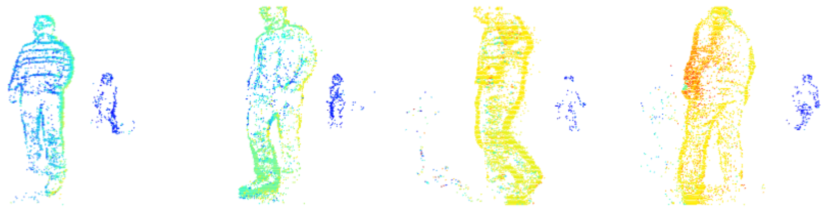
Figure 6.17: Results of the cooperative stereo algorithms on the *people2-1* (a) sequence depicted as disparity maps for *Coopv1* (b) and *Coopv2* (c).

produced by *Coopv2* has more semantic meaning, e.g. high disparities (error) assigned to noise events near the person's shoulder. Furthermore, there are some events generated by the shadow of a person, which are especially visible at time step 4.20s. The events are not filtered out by the noise filter, since they are quite dense and appear in both the left and right event stream. In this case, the *Coopv1* algorithm provides better disparity estimates for the shadow than *Coopv2*, which produces more mismatches, especially in the regions that are not visible in the opposite view (half-occluded).

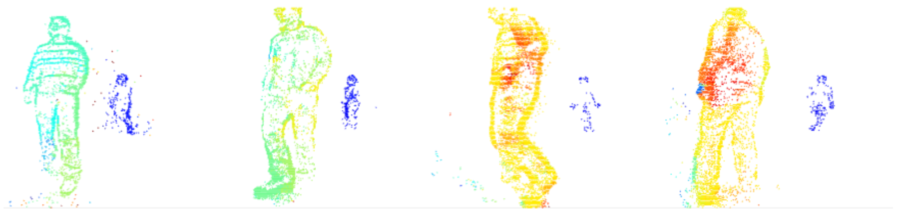
Figure 6.18 shows selected key frames of the *people2* sequence where one person is moving very close to the sensor, whereas the other quite far. The 'foreground' person is more visible in the DVS data, as the field of view of the camera is too small. In the first two frames, it can be clearly seen that *Coopv1* results contain some mismatches within the person, whereas *Coopv2* returns consistent edges and fills out the neighbouring events with correct disparities. We notice that parts of the person which are very close to the sensor (e.g. arm in last two frames) are assigned relatively higher disparities by *Coopv2* than visible in the *Coopv1* results, where the arm has the same disparity as the rest of the upper body.



(a) Selected key-frames from *people2-2* data sequence.



(b) Results of the *Coopv1* algorithm.



(c) Results of the *Coopv2* algorithm.

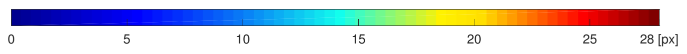
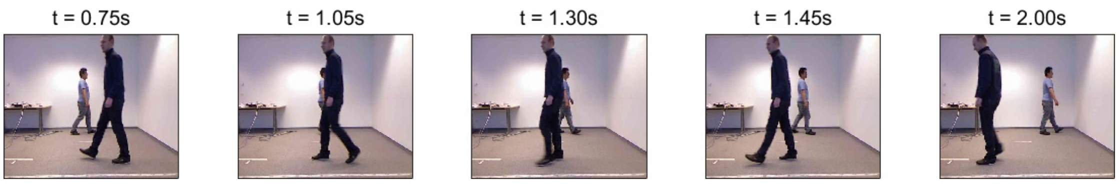
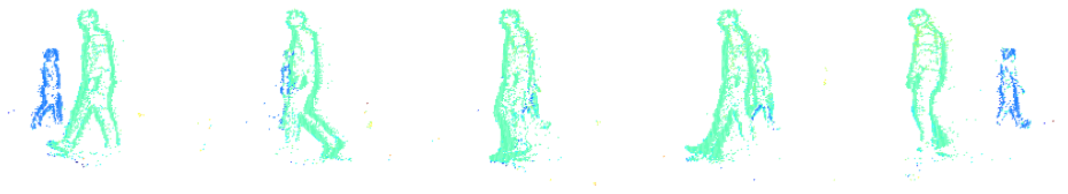


Figure 6.18: Results of the cooperative stereo algorithms on the *people2-2* sequence depicted as disparity maps (a) for *Coopv1* (b) and *Coopv2* (c).

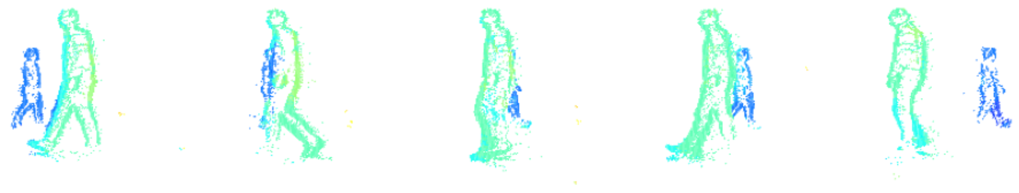
Finally, Figure 6.19 depicts the results of the cooperative stereo algorithms dealing with occlusions in the last *people3* test case. The first set of key frames demonstrates that the *Coopv1* results are erroneous for the occluded person shortly after the occlusion (t = 1.45s). This is most likely caused by high confidence of previous matches assigned to the foreground person. Therefore, it might take a few time steps to recover from occlusion. This 'long recovery' effect is visible in both versions of the cooperative stereo. Another interesting side effect of the cooperative stereo network is the 'see through' effect, where some events from the foreground person are erroneously assigned to the occluded objects in the background, if historical previous matches have strong confidence values. This effect is visible in Figure 6.19e (at t=6.70). In addition, in some cases parts of the adjacent edges of two persons are mismatched (e.g. Figure 6.19c in time step 1.05). We assume that this is related to the trade-off with matching window size. Bigger windows tend to give smooth results, whereas smaller ones preserve more details. In the experiments, we have used the parameters derived in the analysis in Section 6.1.2. However, we can assume that the matching window size should be adjusted to the scene's depth range and focal length of the sensor.



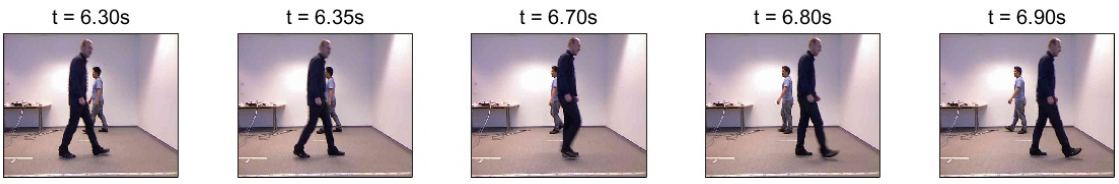
(a) Selected key-frames from *people3-1* data sequence.



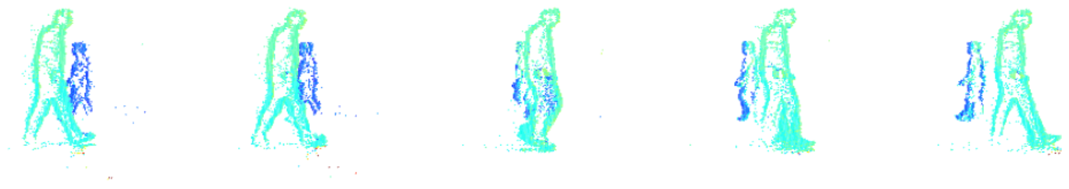
(b) Results of the *Coopv1* algorithm.



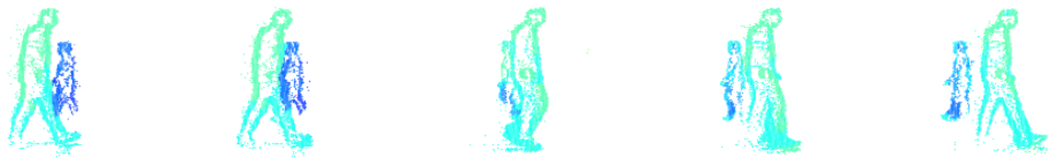
(c) Results of the *Coopv2* algorithm.



(d) Selected key-frames from *people3-2* data sequence.



(e) Results of the *Coopv1* algorithm.



(f) Results of the *Coopv2* algorithm.

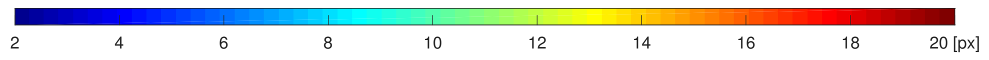


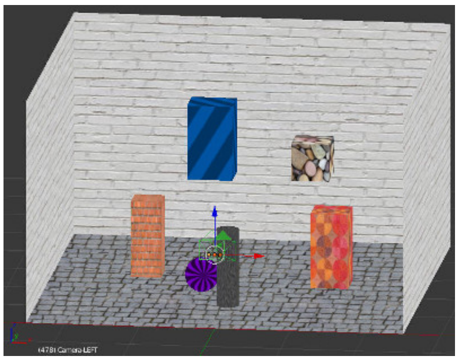
Figure 6.19: Results of the cooperative stereo algorithms on the *people3* (a,d) sequence depicted as disparity maps for *Coopv1* (b,e) and *Coopv2* (c,f) algorithm.

6.2 Experiments

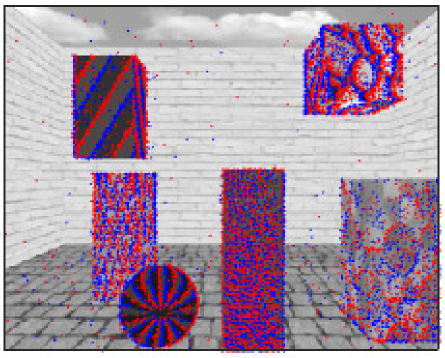
Having derived the optimal parameters for both cooperative stereo algorithms through the analysis presented in the previous section, we demonstrate the applicability of the algorithms using various datasets, including both synthetic and real event sequences. The goal of our experiments is to evaluate the robustness of the proposed cooperative stereo algorithms with respect to varying scene complexity, contrasting speed, and different sensor's setup.

6.2.1 Synthetic Datasets

So far, we have evaluated the performance of the cooperative stereo algorithms mainly on indoors scenes depicting moving people. In this section, we present two synthetic datasets, *synth-Scene1* and *synth-Scene2*, which focus on two different challenges in event-based stereo. The first one is intended to test how well the algorithm tackles cluttered scenes, including objects with complex textures, whereas the second one focuses on the asynchronous aspect of event data, that is, the ability to handle objects moving at drastically different speeds. The synthetic scenes are created from a 3D animation, modelled in 3D studio Blender[4]. The event streams are generated using greyscale images which are rendered separately for the left and right camera. The method used for synthetic data generation is described in detail in Appendix B. A snapshot of the synthetic scene for *synth-Scene1* modelled in blender is shown in Figure 6.20a.



(a)



(b)

Figure 6.20: Simulated dataset *synthScene1*: (a) snapshot of a synthetic 3D model, and (b) rendered greyscale image with generated synthetic events.

The animation depicts six objects of various sizes and textures, moving at a similar speed in different directions. Figure 6.20b is a greyscale image, rendered from the 3D scene for the left sensor (in the stereo rig). The actual synthetic data, namely the generated address events, are plotted over the greyscale image, to show the corresponding edges

[4]https://www.blender.org/

(contrast) that triggered the events. The synthetic event sequence contains some noise events to make it more realistic.

In Figure 6.21 we compare results of the *Coopv1* and *Coopv2* algorithms against the ground truth. As can be observed, both cooperative algorithms obtain very good results, and tackle the complexity of textures well. Details such as boxes transitions (slanted

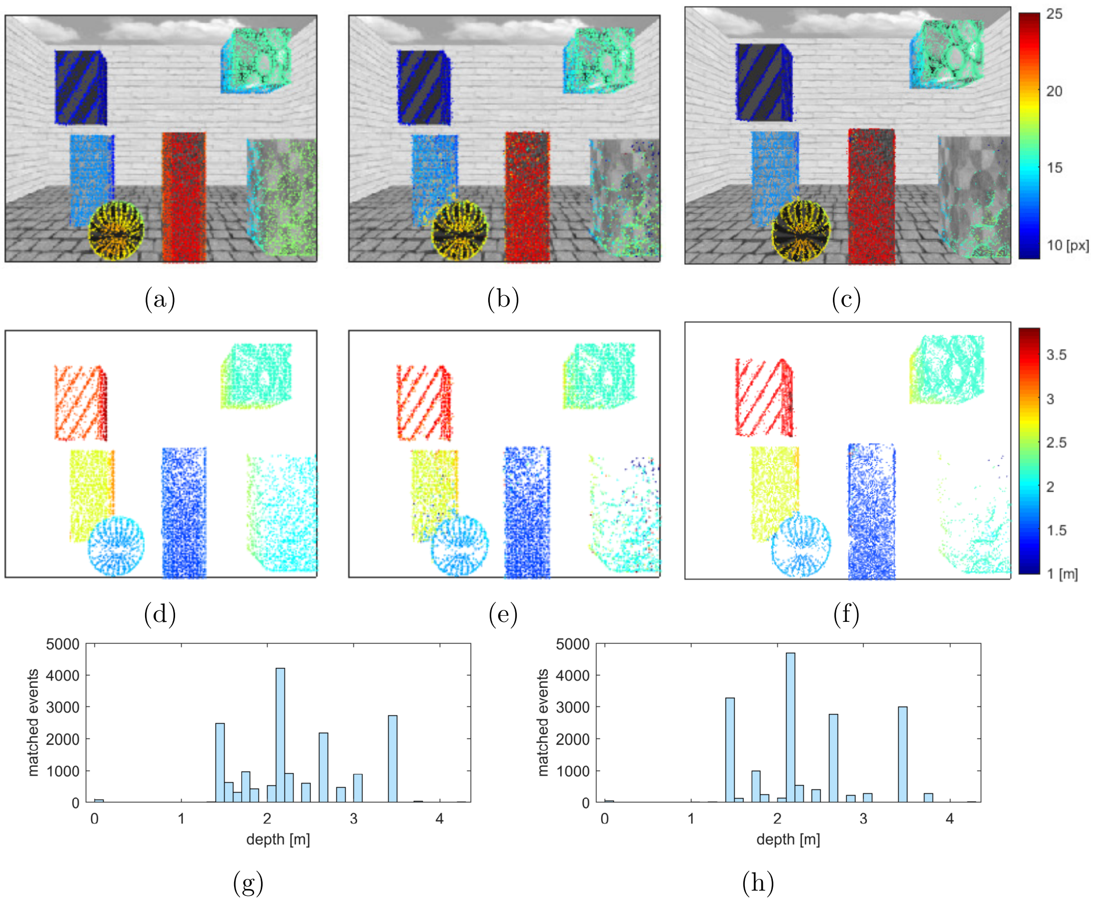
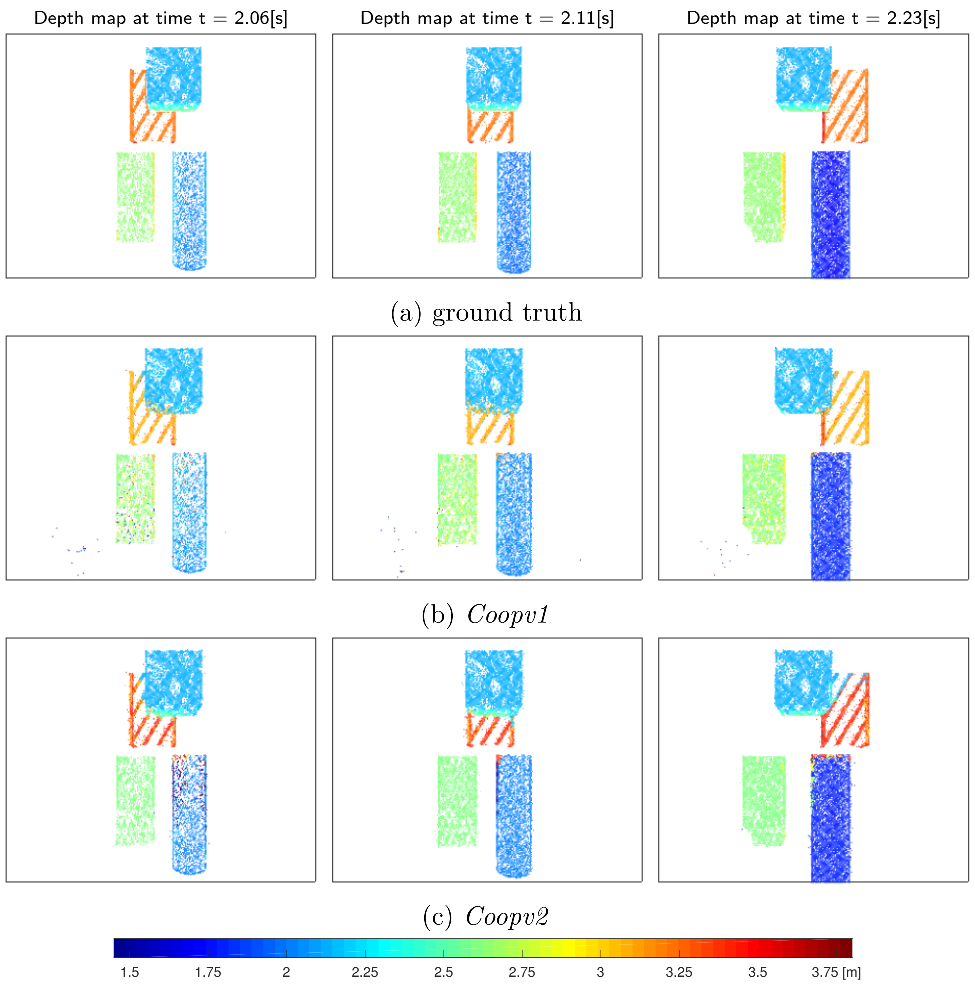


Figure 6.21: Comparison of stereo accuracy on a part of the *synth-Scene1* sequence, compared between ground truth (a,d), *Coopv1* (b,e) and *Coopv2* (c,f). In the first row, the disparity values are plotted over greyscale images. In the second row, the same results are depicted as depth maps. In addition, depth estimates histograms are shown for *Coopv1* (g) and *Coopv2* (h) in the third row.

surfaces) are preserved; *Coopv1* tends to provide slightly less accurate results but smoother transitions between different disparity levels. *Coopv2* gives more accurate results within front-to-parallel surfaces, however the transitions on slanted surfaces are less smooth. This can be also seen in the histograms, where *Coopv1* shows more diverse depths, whereas *Coopv2* results in higher peaks on particular depth values.

In Figure 6.22, we demonstrate how both stereo algorithms tackle occlusions. The results are compared with ground truth, which is shown in the first row. Some objects are better estimated by *Coopv2* than *Coopv1*, e.g. in the foreground upper box, especially its bottom edges. However, the recovery from occlusion tends to be more challenging for *Coopv2* than *Coopv1*. Another interesting observation is that *Coopv1* tends to estimate objects to be farther, whereas *Coopv2* closer. The ground-truth value is in between.



(a) ground truth

(b) *Coopv1*

(c) *Coopv2*

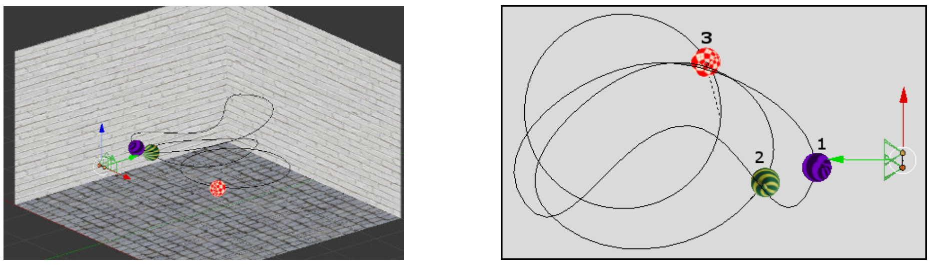
Figure 6.22: Comparison of stereo accuracy on three parts of the *synth-Scene1* sequence: (a) ground truth, (b) *Coopv1* and (c) *Coopv2*.

The numerical results for the whole *synth-Scene1* sequence are presented in Table 6.3. Both algorithms obtain a mean depth error of less than 10cm. Since depth errors depend on the depth range of the scene, we measure the relative error, which is defined as mean depth error divided by the depth range of the scene. Here, the relative error is 1.85% for *Coopv1*, and 1.58% for *Coopv2*. We can also see that *Coopv1* achieves a slightly higher matching rate than *Coopv2*.

	Coopv1	*Coopv2*
mean error [m]	0.089	0.076
relative error [%]	1.85	1.58
matching rate	0.94	0.93

Table 6.3: Results of cooperative stereo algorithms on *synth-Scene1* sequence.

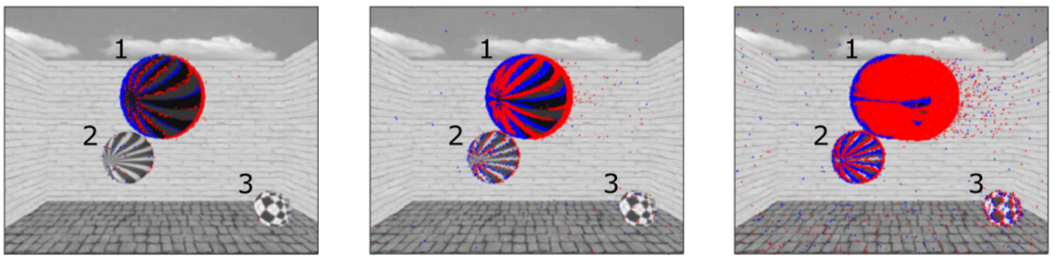
The second synthetic scene is intended to test the algorithms' ability to handle asynchronous data input. Since dynamic vision sensors generate events upon change, the frequency of events is dependent on the speed of the object as well as its proximity to the sensor. The stereo algorithm is required to maintain equal accuracy for all objects, regardless of their speed or appearance. The animation for the *synthScene2* includes three balls moving at different speeds. A snapshot of the modelled 3D scene is shown in Figure 6.23.



(a) (b)

Figure 6.23: Snapshot of a 3D model for *synthScene2*: (a) 3D view of the scene and (b) top-down view on the object trajectories.

In Figure 6.24, event sequences of three different time durations are plotted over the corresponding greyscale image. We can observe that the time window which is optimal for one object would not be sufficient for another. For instance, the fastest moving ball (1),

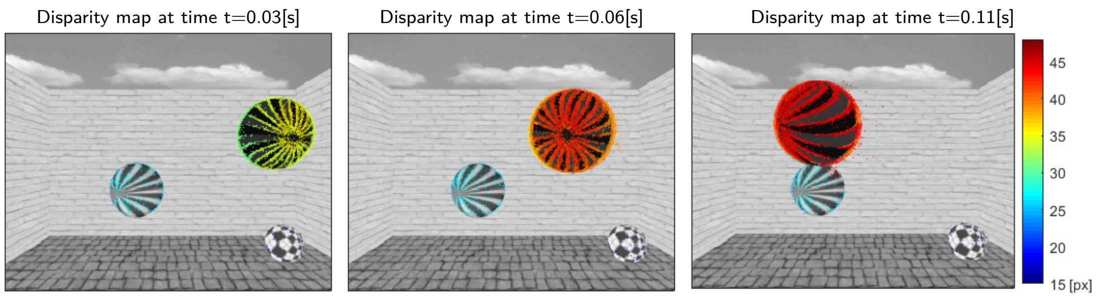


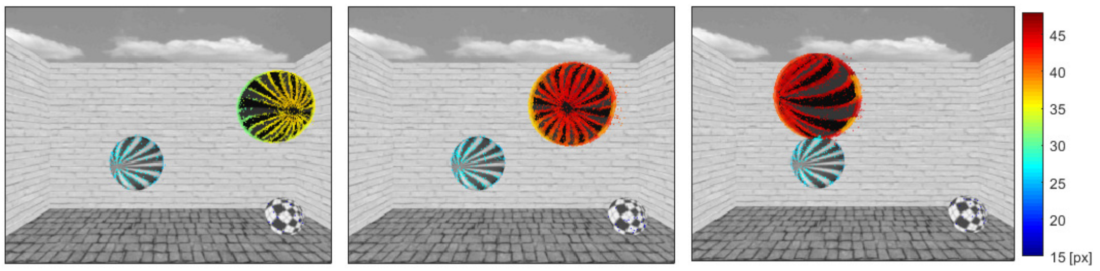
(a) (b) (c)

Figure 6.24: Simulated dataset *synthScene2*. Event sequences of duration (a) $100\mu s$, (b)$1ms$ and (c) $10ms$ are plotted over the rendered greyscale image.

requires less than 1ms time window for accurate processing. This time window, however, would be too short to capture the motion of the slowest ball (3). One or two events triggered by the slowest ball would easily be considered noise in frame-based processing.

We demonstrate that both cooperative stereo algorithms perform asynchronous stereo matching and can correctly estimate the depth of each moving object. In Figure 6.25, a comparison of *Coopv1* and *Coopv2* results is shown in the form of disparity maps displayed over greyscale images. Here, the time window of 1ms is used to present in detail the disparity estimates for the fastest ball (1). As we can observe, the fast ball (1)


Disparity map at time t=0.03[s]
Disparity map at time t=0.06[s]
Disparity map at time t=0.11[s]
45
40
35
30
25
20
15 [px]
(a) *Coopv1*

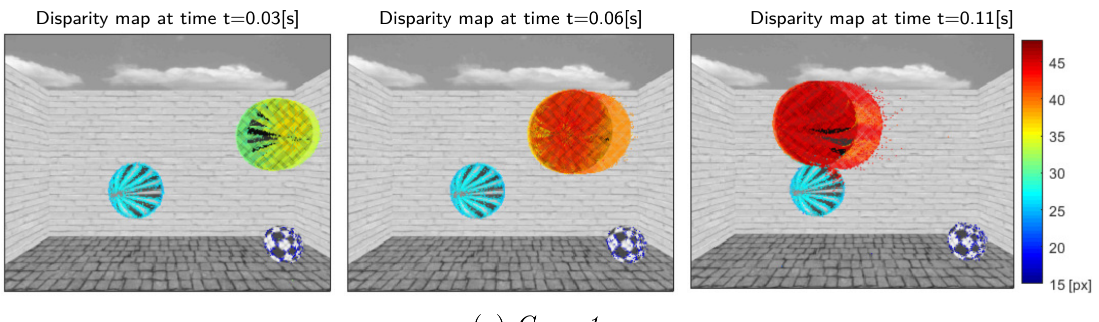


(b) *Coopv2*

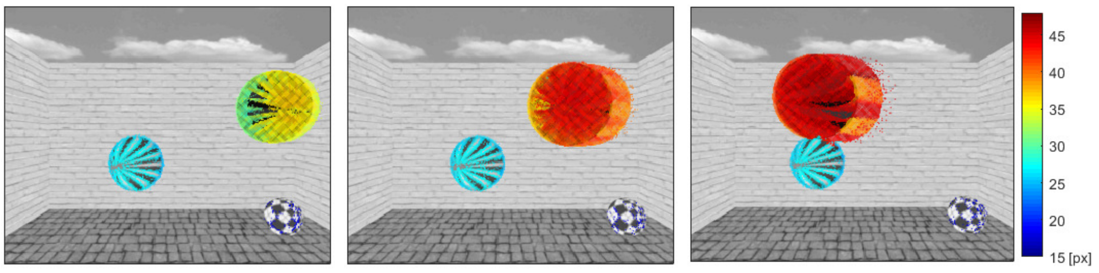
Figure 6.25: Comparison of stereo accuracy on three 1ms parts of the *synth-Scene2* sequence for *Coopv1* (a) and *Coopv2* (b) algorithm.

also rotates whilst it approaches the sensor. We can see that both algorithms give correct disparity estimates and show accurate transitions between disparities within the ball's surface. The second ball, moving at medium speed, is also recognised at the correct disparity, whereas the slowest ball (3) is not visible at all in this time window. In order to see the events of ball (3), the results are presented in ten time longer windows in Figure 6.26.

Table 6.4 presents numerical results calculated per each object individually. As we can see, the *medium* speed ball (2) has the lowest mean error for both stereo algorithms. This might be related to the periodical update in the cooperative network, which is adjusted to a standard object speed. In other words, the parameters of the algorithm are optimal for medium speed objects. Nonetheless, the accuracy of both extreme cases (slow, fast) is not drastically worse. In general, fast objects are less problematic since they


Disparity map at time t=0.03[s]
Disparity map at time t=0.06[s]
Disparity map at time t=0.11[s]
45
40
35
30
25
20
15 [px]

(a) *Coopv1*



(b) *Coopv2*

Figure 6.26: Comparison of stereo accuracy on three 10ms parts of the *synth-Scene2* sequence for *Coopv1* (a) and *Coopv2* (b) algorithm.

modify the cooperative network frequently, and trigger updates within corresponding areas. Contrarily, slow objects are likely to be filtered out or erroneously matched to more active parts of the scene.

	Coopv1			*Coopv2*		
	fast	*medium*	*slow*	*fast*	*medium*	*slow*
mean error [m]	0.064	0.036	0.063	0.032	0.026	0.036
relative error [%]	1.68	0.95	1.65	0.83	0.68	0.96
matching rate	1.00	0.99	0.73	1.00	1.00	0.45

Table 6.4: Results of cooperative stereo algorithms on *synth-Scene2* sequence, calculated individually for each moving object (ball).

6.2.2 Ego-motion Datasets

In the last set of experiments, we test the suitability of the cooperative stereo algorithms on sequences captured by moving stereo sensors. This is partly motivated by the growing interest in applying the dynamic vision sensor to SLAM and visual odometry for pose estimation and mapping of the surrounding from moving platforms, e.g. unmanned aerial vehicles (UAVs). We have used several datasets [104], both synthetic and real, captured and published by the Robotics and Perception Group[5] at University of Zurich.

First experiments are performed on the synthetic datasets. We have extended the available Blender scenes, *3planes* and *3walls*[6]. The camera parameters and scene setup have been kept the same as in the original models. We have added a second camera to simulate a stereo setup, using an appropriate baseline (depending on the scene depth and content). We have used the events generator (see Appendix B) to convert greyscale images into event sequences. A snapshot of the synthetic scene *3planes* modelled in Blender is shown in Figure 6.27.

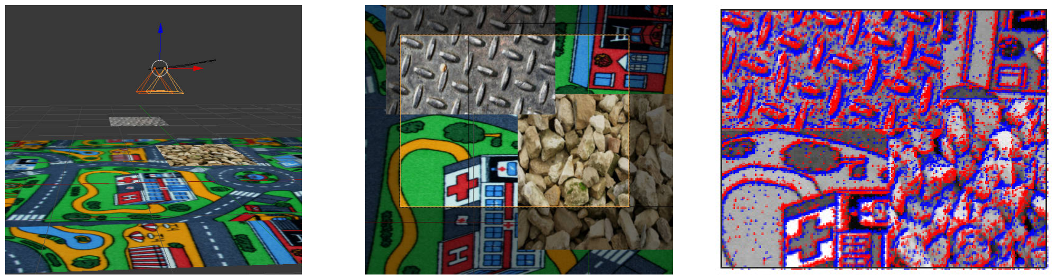

(a) (b) (c)

Figure 6.27: Snapshot of the 3D model of the synthetic scene *3planes*: (a) 3D view on the scene, (b) view on the scene from the left virtual camera, and (c) rendered greyscale image with simulated events plotted over it (ON events are shown in red, OFF in blue).

The only animated object in the scene is the camera, which hovers over three parallel planes, placed at three different distances. The left camera view of the scene is shown in Figure 6.27b. The generated synthetic events are plotted over the rendered greyscale image, as shown in Figure 6.27c.

The results of the cooperative stereo algorithms for the *3planes* scene are depicted in Figure 6.28. Compared to ground truth, both algorithms deliver high-quality disparity estimates. The results show only little difference between the *Coopv1* and *Coopv2* algorithms. Furthermore, the disparity histograms for both algorithms show high similarities. The mismatched events are rare and tend to be scattered in *Coopv1*, whereas in *Coopv2* we observe some mismatched areas.

[5]http://rpg.ifi.uzh.ch/davis_data.html
[6]https://github.com/uzh-rpg/rpg_davis_simulator

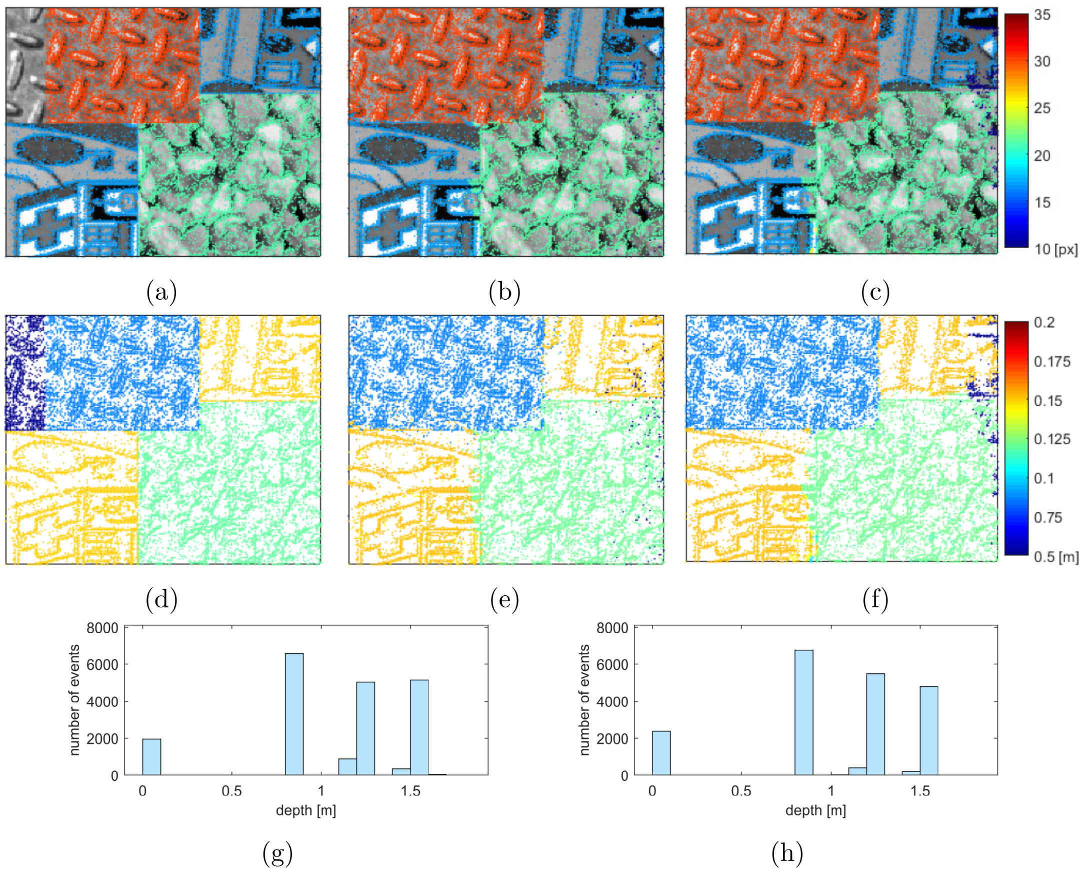


Figure 6.28: Disparity (depth) values of events within a part of the synthetic *3planes* sequence compared between ground truth (a,d), *Coopv1* (b,e) and *Coopv2* (c,f) results. In the first row, the colour-encoded disparity values are plotted over greyscale images. In the second row, the same results are depicted as depth maps. In addition, depth histograms are shown for *Coopv1* (g) and *Coopv2* (h).

The second available synthetic scene *3walls* is presented in Figure 6.29. The scene has been built to resemble the landscape seen from a camera, mounted on a flying vehicle (drone). The virtual stereo camera looks onto three textured walls whilst it is moving along a quite complex motion path, as depicted in Figure 6.29a.

In Figure 6.30, the results of the cooperative algorithms on the *3walls* sequence are depicted. The results show that the slanted surfaces and depth transitions are well captured by each algorithm. From the disparity maps we can see that *Coopv2* recognises well the transitions close to the sensor (areas on the floor), whereas in *Coopv1* there are some mismatches in that area. In contrast, dealing with the farthest parts of the scene (corner), *Coopv1* shows better results than *Coopv2* (see Figure 6.30e-6.30f). Nonetheless, the surfaces in *Coovp1* are not smooth, the events tend to scatter across some disparity levels.

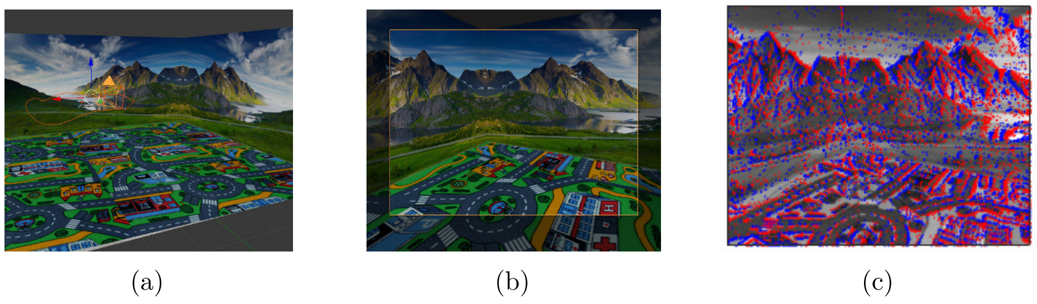


Figure 6.29: Snapshot of the 3D model of the synthetic scene *3walls* in Blender (a), the view on the scene from the left virtual camera (b), and rendered greyscale image with simulated events plotted over it (c).

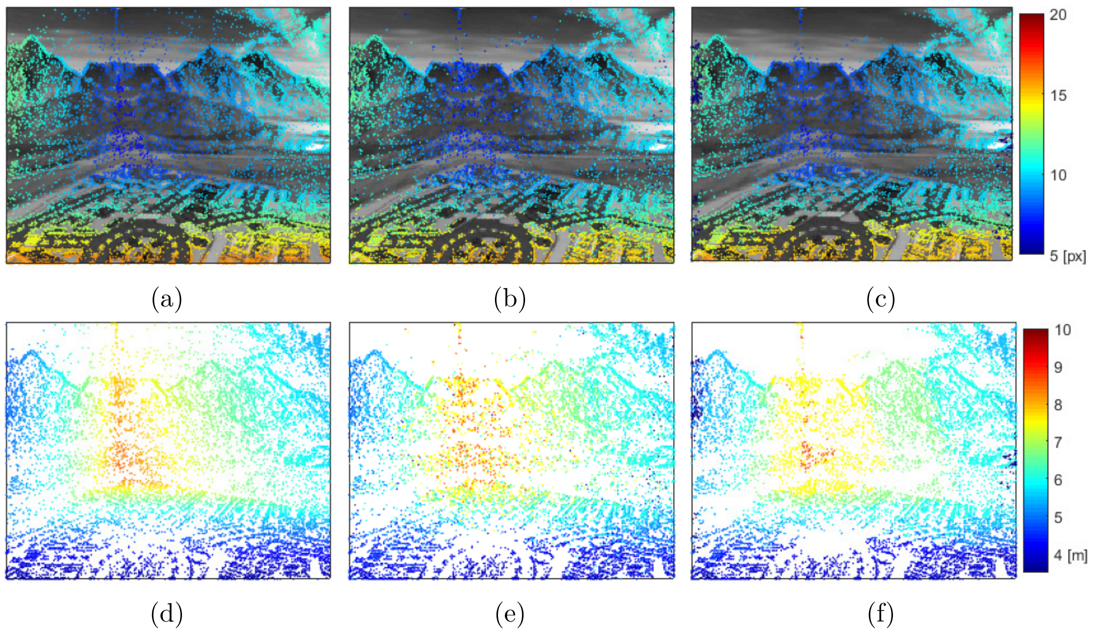


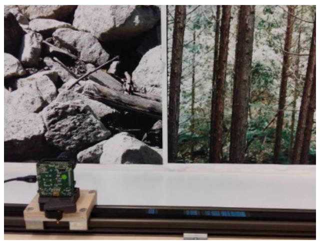
Figure 6.30: Disparity (depth) values for events within a part of the synthetic *3walls* sequence compared between ground truth (a,d), *Coopv1* (b,e) and *Coopv2* (c,f) results. In the first row, the colour-encoded disparity values are plotted over greyscale images. In the second row, the same results are depicted as depth maps.

Table 6.5 summarises the numerical results of the stereo algorithms for both synthetic *3planes* and *3walls* sequences. In terms of accuracy the *Coopv2* algorithm performs slightly better than *Coopv1*. In addition, we list the results of the *EVMS* algorithm as published in [118]. These results, however, cannot be directly compared with the cooperative stereo for two reasons: firstly, the algorithm's input may vary from ours since the synthetic events were generated by a different algorithm. Secondly, the *EMVS*, instead of two, makes use of the multiple views of the same scene.

	3planes			*3walls*		
	Coopv1	*Coopv2*	*EMVS* [118]	*Coopv1*	*Coopv2*	*EMVS*[118]
mean error [m]	0.036	0.032	0.15	0.356	0.201	0.52
relative error [%]	2.30	2.05	11.31	3.56	2.01	6.86
matching rate	1.00	1.00	n/a	0.96	0.95	n/a

Table 6.5: Results of the *Coopv1* and *Coopv2* stereo algorithms on synthetic *3planes* and *3walls* sequences.

Further tests were performed on the data recordings provided in [104]. We have used three sequences: *slider-close*, and *slider-far*, which are recorded by a camera sliding at a particular distance to the wall, and *slider-depth*, sliding in front of a more complex scene of objects. The setup used for these recordings is depicted in Figure 6.31.


(a)

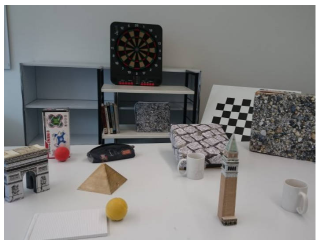

(b)

Figure 6.31: The setup for recording event datasets, for (a) *slider-close*, *slider-far*, and (b) *slider-depth* sequence [104].

For the sequences *slider-close* and *slider-far*, the distance from the sliding sensor to the wall is known and could be used as a ground-truth depth, whereas for *slider-depth* no ground truth is available for reference. Since the event stream is available only for one view (monocular setup), in order to facilitate the stereo configuration, the second view is achieved by translating the event stream in time. The difference in time between the left and right sequence defines the baseline.

Selected parts of the *slider-close* and *slider-far* sequences are presented as polarity plots together with the results of the cooperative stereo algorithms, visualised as colour-encoded depth maps over greyscale images in Figure 6.32 and Figure 6.33, respectively. The results show similar tendencies as in the synthetic data tests. The depth estimates are satisfying for both *Coopv1* and *Coopv2* algorithms. In the former, some singled out, rare mismatches can be observed, whereas in the latter depth estimates are aligned to the planar surface. These observations are also supported by the numerical results, as listed in Table 6.6. The mean error in the *Coopv2* depth estimates is equal or lower than for *Coopv1*. In terms of matching rate, the *Coopv1* achieves slightly higher rate than the *Coopv2* algorithm.

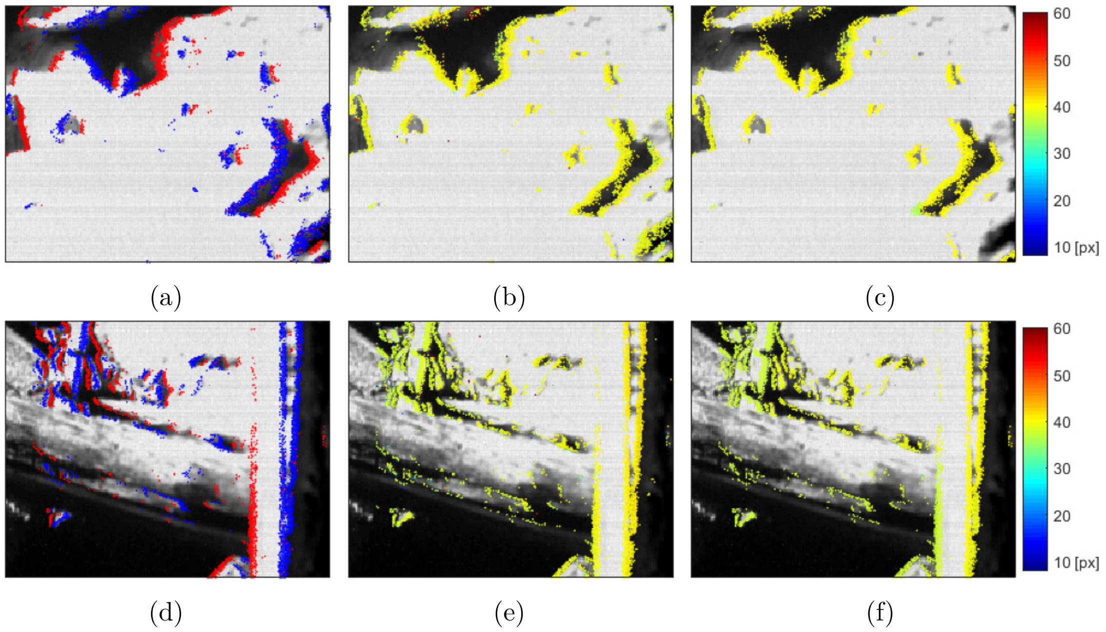


Figure 6.32: Two parts of the *slider-close* sequence depicted as polarity plots (a,d). The stereo results of *Coopv1* (b,e) and *Coopv2* (c,f) algorithms are presented as depth maps.

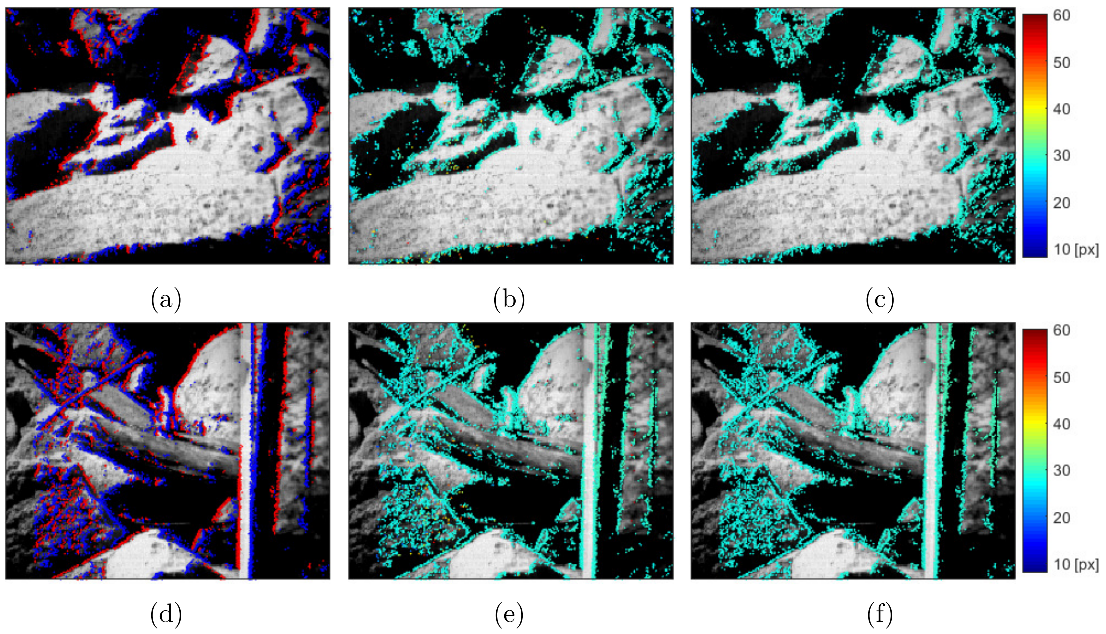


Figure 6.33: Two parts of the *slider-far* sequence depicted as polarity plots (a,d). The stereo results of *Coopv1* (b,e) and *Coopv2* (c,f) algorithms are presented as depth maps.

We compare the cooperative algorithms with the *EMVS* [118], which in this case is feasible, since the algorithms operate on the same event dataset. The *EMVS* compared to the *Coopv1* shows a lower mean error for *slider-close* and *slider-far*. Interestingly, in the latter case, although the mean error indicates that the two algorithms achieve similar accuracy, the relative error shows a more significant difference. We assume that the results *EMVS* considered a different depth range. In our calculations, we have set the depth range to the farthest point in the scene, which is the highest depth in ground-truth estimates, i.e. 0.231m and 0.584m, for *slider-close* and *slider-far*, respectively. The *Coopv2* outscores the other two algorithms on both data sequences.

	slider-close			*slider-far*		
	Coopv1	*Coopv2*	*EMVS* [118]	*Coopv1*	*Coopv2*	*EMVS*[118]
mean error [cm]	1.56	1.07	1.22	2.03	1.55	2.01
relative error [%]	6.76	4.62	5.29	3.48	2.66	4.33
matching rate	0.93	0.83	n/a	0.89	0.85	n/a

Table 6.6: Results of the cooperative stereo, *Coopv1* and *Coopv2*, algorithms and *EMVS* algorithm on *slider-close* and *slider-far* sequence.

The last sequence, *slider-depth*, depicts a more complex scene with several objects placed at different distances from the sliding sensor. In Figure 6.34, the scene captured at selected time steps is presented by polarity plots over greyscale images. The ground-truth depth values for this sequence are not available, therefore the results can only be assessed visually. Judging from the disparity maps, the overall performance of both cooperative algorithms is satisfying. Disparities of different objects are correctly recognised, e.g., details within the triumphal arch, or the edges of the pyramid. We can observe that the *Coopv1* depth estimates contain single high-score errors, such as red and green points in the dartboard (see Figure 6.34b). In this regard, the *Coopv2* performs better. Sometimes, however, the *Coovp1* estimates appear more accurate than *Coopv2*. For instance, in Figure 6.34k, the edge of the tower is quite consistent in disparity values, whilst in Figure 6.34l the edge is slightly de-fragmented (bottom part) and some neighbouring objects (cup and box behind the tower) are erroneously assigned with higher disparities. To better understand the difference between the results achieved by *Coopv1* and *Coopv2*, Figure 6.35 shows events in three-dimensional space (x, y and depth), for the same time steps as in the previous figure. There are two main observations from the results. Firstly, the *Coopv1* algorithm produces estimates scattered over several neighbouring disparity levels, whereas *Coopv2* tends to align them to planar surfaces. Secondly, *Coopv2* seems to have a lower matching rate than the *Coopv1* algorithm. Such an effect might be caused by the difference in matching cost calculation. Since *Coopv2* employs more accurate initial matching scores, only the 'most likely' matches are passed to the cooperative network optimisation. The *Coopv1* algorithm is less restrictive in initial matching scores and admits more matching candidates for further processing, hence, the resulting matching rate may be higher.

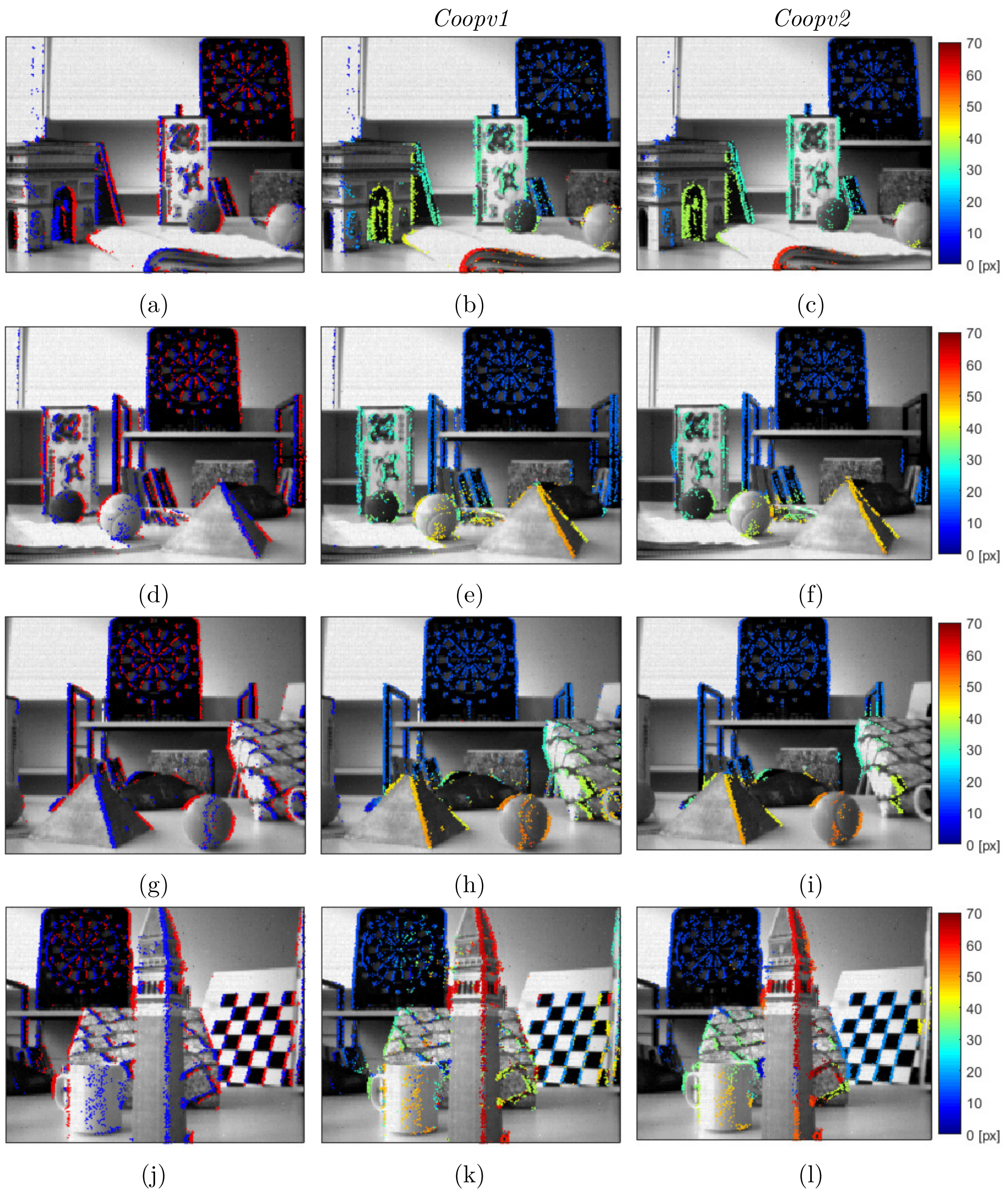


Figure 6.34: Selected time steps of the *slider-depth* sequence depicted as polarity plots (a,d,g,j). Results of *Coopv1* (b,e,h,k) and *Coopv2* (c,f,i,l) are presented as disparity maps over greyscale images.

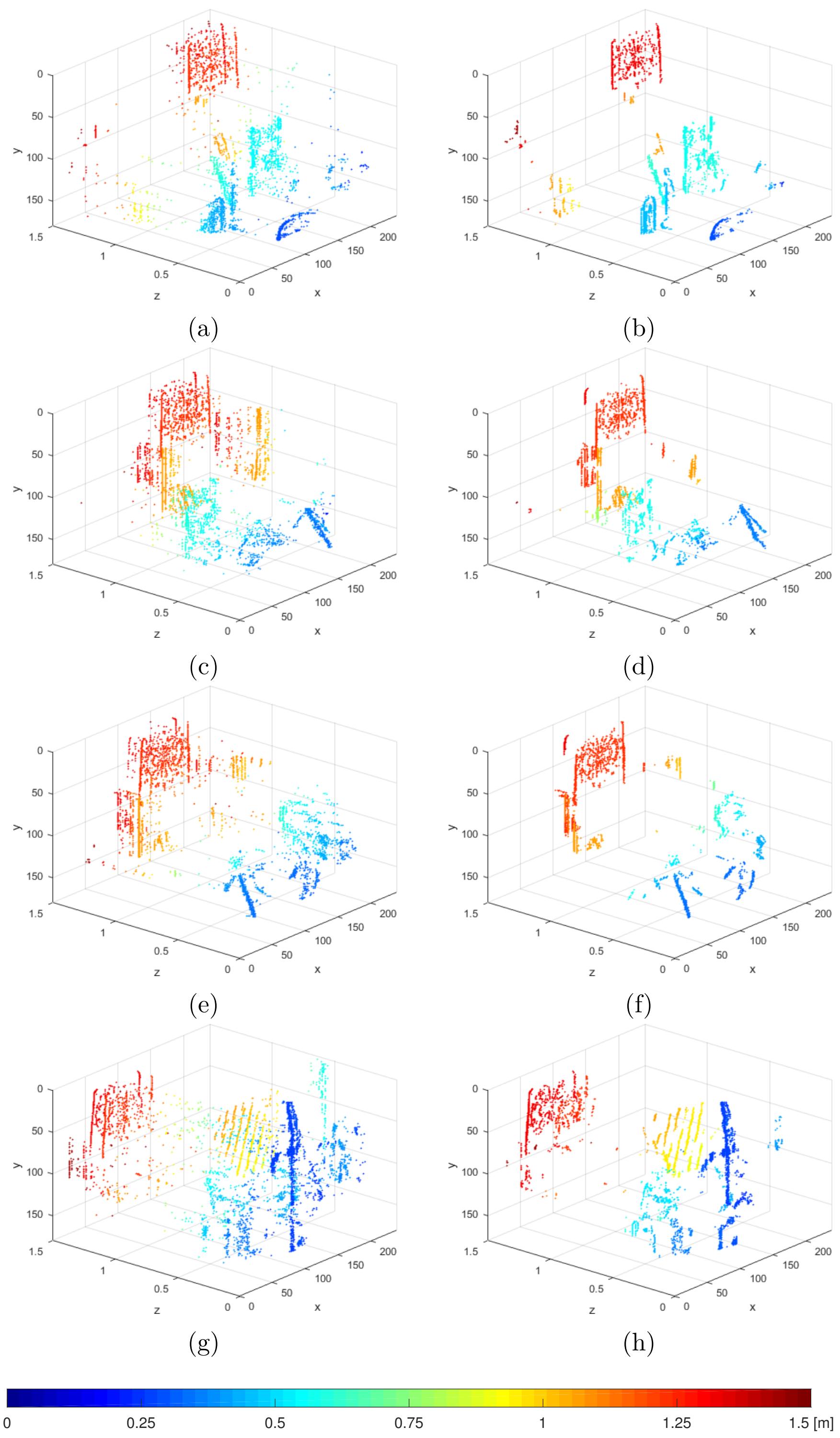


Figure 6.35: Results of the *Coopv1* (a,c,e,g) and *Coopv2* (b,d,f,h) stereo matching algorithms for the selected time steps of the *slider-depth* sequence presented in three-dimensional space (x, y, depth). Depth is also encoded in events colour.

6.3 Summary

In this chapter, we presented an extensive evaluation of the proposed *Coopv1* and *Coopv2* stereo matching algorithms. We started with an analysis of the algorithms' parameters to learn about their influence on the overall accuracy. We found out that the time correlation for event matching is not reliable and depends on the characteristics of the scene/object motion. It is preferred to use higher tolerance in time-based matching, i.e. lower values of α. Additional constraints are useful to mitigate temporal mismatches, e.g. using the polarity constraint. Further improvements have been confirmed for the window-based matching applied in *Coopv2*. We observed that even with a small, 3×3 matching window size, the resulting mean error can be reduced by approx. 25%. Looking at different variants of the matching function, we discovered that *Coopv1* performs remarkably well with no constraints applied in initial matching. The *Coopv2* matching is less sensitive to temporal ambiguities, hence, using both polarity and temporal constraints yields good results. Moreover, the results of the matching function and overall algorithm are measured separately. When compared, the results indicate a significant gain in accuracy achieved by the cooperative network optimisation. A series of tests has been performed to find an optimal choice of matching and supporting window sizes. We have observed that larger sizes of the matching window may lead to 'edge fattening' problems, i.e. when adjacent edges are not preserved well. Therefore, the best results are achieved for bigger support and moderate matching window sizes. The results of the comparative evaluation demonstrated that the *Coopv2* algorithm shows clear improvement in accuracy over *Coopv1*. *Coopv2* also outperforms several competing algorithms in the field.

In addition, we have tested the cooperative algorithms on various datasets, including synthetic and real event sequences. The synthetic datasets have been used to test specific challenges in event stereo vision, such as scenes with complex textures, clutter or occlusions, and objects moving at very different speeds. The former tests the ability to resolve matching ambiguities, whereas the latter focuses on handling the asynchronous aspect of event data. Judging from the results, both algorithms obtain promising results, and successfully tackle the complexity of objects' appearance. In the asynchronous stereo challenge, we measure the accuracy of an algorithm individually per each moving object. The results show that both algorithms obtain better performance for objects moving at regular speed, while still maintaining relatively high accuracy for extremely slow and fast objects.

Finally, we demonstrated the applicability of the proposed cooperative methods to event sequences captured by a moving camera (ego-motion). The results consistently show that *Coopv2* obtains higher stereo accuracy when compared to *Coopv1*, where some high impact single errors increase the overall mean error of the depth estimates. Arguably, the *Coopv1* algorithm would apply for fast and quick approximations of the scene, which is useful in applications where the speed of response is more important than accuracy. In contrary, the *Coopv2* algorithm would be a better choice in applications where the accuracy is more important and response time is less critical.

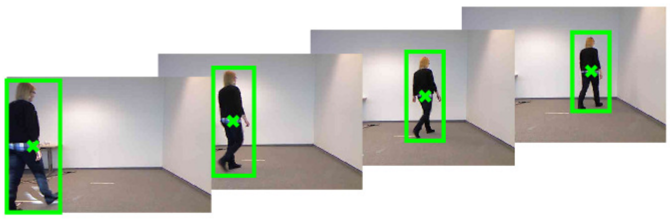
CHAPTER 7

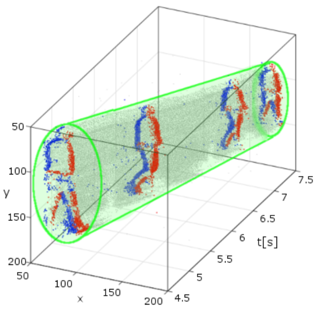
Event-based Spatiotemporal Multiple Object Tracking

An increasing interest in automated video analysis creates a demand for fast and robust tracking algorithms. The task of visual tracking is to detect moving objects in the scene and estimate their motion trajectories based on the change of their location over time. Research on visual tracking is motivated by a number of potential applications including automated visual surveillance [154], human pose estimation [109], action recognition [31], traffic monitoring [75, 84] or autonomous vehicle navigation. Depending on the application, different objects are being tracked, e.g. pedestrians [162], vehicles [84], sport players [86] or groups of animals, e.g. [88]. Visual tracking has proved to be a challenging task due to many factors, e.g. complexity of non-rigid motion, abrupt changes in appearance caused by drastic pose or scale change, occlusions or illumination variations. Tracking multiple objects needs to tackle additional challenges such as configuration changes, that is determining how many objects have left or entered the scene, the ambiguity in identification caused by similar appearance of objects, as well as occlusions and interactions between the objects. To address these challenges, a variety of solutions have been proposed in the literature. In what follows, we briefly describe the main approaches and algorithms in both image- and event-based tracking. Subsequently, we formulate event-based tracking as a task of clustering events in three-dimensional space using a Gaussian Mixture Model (GMM) and describe in detail the proposed algorithm. We address the challenge of occlusions by using depth information to reason about the objects' motion trajectories. Finally, in the experimental section, we demonstrate that the proposed tracking algorithm successfully tracks multiple objects and can tackle full and transient occlusions, as well as changes in configuration, such as entrances, exits, and stops.

7.1 Related Work

In conventional image-based tracking, the object's position is identified in each frame of the video sequence (see Figure 7.1a). The detected objects are then identified across consecutive frames in order to estimate their trajectories. In the context of dynamic vision sensors, however, many computer vision tasks, including tracking, require quite a different approach than the image-based processing. This is mainly due to the specifics of the data provided by the sensors. As explained in detail in Section 2.3, the dynamic vision sensors operate in frame-free mode. In a stationary setup, sensors naturally support the task of tracking by performing an on-chip segmentation of the moving objects from the static background. As depicted in Figure 7.1b, all data provided by the sensor correspond to the contours of the moving object, thus delivering motion information. Thanks to the high temporal resolution of the dynamic vision sensor, the obtained address event stream represents almost continuous motion. Instead of estimating the trajectories from the object's location, event-based tracking deals with discovering which events were generated by a particular moving object.


(a)



(b)

Figure 7.1: Tracking using conventional camera-based vision systems (a). The person is identified in each frame and based on that the motion trajectory is estimated. In DVS data, the trajectory of the moving person is formed by the cloud of address events (b).

We can distinguish three subtasks of multiple object tracking, namely: *(i)* detection and localisation of the moving objects in each frame, *(ii)* identification of the detected objects across consecutive frames, and *(iii)* trajectory estimation. The first task deals with finding an appropriate representation of the object that is being tracked. Among the methods proposed in the literature, we can find shape and feature representation [135], e.g. points [152], contours [160]. The target object can also be described by its appearance features such as colour [48], colour histogram [102, 142, 159], or texture descriptors [23, 31, 151]. Templates (e.g. raw pixel template [161]) and models (e.g. active appearance models [43], articulated shape models [164]) can also be applied. The second task associates detected objects between consecutive frames using a similarity measure, which depends on the selected object representation, e.g. for the raw pixel template the normalised cross correlation is used in [161], or the Battacharyya distance for comparing colour histograms in [159]. Finding the correspondence between frames is used to derive and estimate the motion trajectory by connecting an object's position over time. This step is also referred to as *object to track assignment* and can be achieved by deterministic or statistical methods. The former perform combinatorial optimisation of the correspondence cost depending on the constraints and assumptions about the motion, e.g. proximity, maximum velocity and local smoothness [152]. Methods such as bipartite graph matching [23], dynamic programming [9], or min-cost max-flow net [170, 31] are used to solve the association between objects and their estimated trajectories. The other group of tracking methods perform statistical or probabilistic inference about the object's motion, where apart from the measured position of the detected objects, also the chosen uncertainty model is taken into account, i.e the measured position is corrected by the prediction, which is based on the previous steps. For this purpose, probabilistic filters can be applied, e.g. discrete Kalman filter [24, 35], extended Kalman filter [102], or particle filters [172, 23].

Furthermore, several approaches have been designed specifically to address occlusions. Hue et al. [62] proposed a *part-to-whole* approach in which the appearance model is constructed in a way that allows occlusion detection. In feature-based tracking, partial occlusion can be handled by feature clustering, based on the assumption that points which belong to the same object should be characterised by similar motion parameters [142, 25]. Zhang et al. [170] generate occlusion hypotheses based on pairs of observations which are then fed to a cost-flow network to derive the most likely solution. Mitzel and Leibe [102] apply a *buffer-and-recover* strategy in which each occluded object's track is being kept for a number of frames. Once it reappears, the track and identity of the object are recovered and the trajectory under occlusion can be extrapolated from the last seen position.

For a more extensive reading about conventional tracking approaches, we refer the reader to existing reviews of different scope, such as human motion analysis [49], appearance models [81], or tracking in general, e.g. [163, 139, 89].

Event-based Tracking

Dynamic vision sensors can overcome some of the limitations of conventional camera vision, especially in the aspects of high-speed motion analysis. For this reason, event-based tracking has been used in specific applications where high-speed tracking at low computational cost is a requirement and conventional imaging performance is simply not sufficient. For instance, DVS has been used for tracking microspheres for optically based measurements in fluid flows [42], or tracking helium-filled soap bubbles for visualisation of flows in wind tunnel tests [17]. Another type of application could be microrobotics, where precision and high speed are of critical importance to perform automated vision-based micromanipulation, as presented in [105]. In the literature, we can also find more general works on tracking features for optical flow estimation. Lagorce et al. [77] present a pool of different filters, including Gaussian or Gabor functions, to track features in the event stream. A particle filter for tracking with ego motion has been implemented in [55] for robotic applications, such as ball tracking and gaze following. Brändli et al. [22] proposed parametrizing the event stream as a set of line segments, whereas Zhu et al. [173] consider all events in optical flow estimation based on association probabilities. Furthermore, there are also hybrid approaches that perform a combination of tracking using greyscale images and stream of events, e.g. tracking features (edges) using an iterative, geometric registration approach [148], or support image-based tracking by identification of regions of interest indicated by clusters of events [85]. The above-mentioned feature tracking algorithms are used as a basis for more complex tasks, e.g. ego-motion estimation or simultaneous localisation and mapping (SLAM).

Tracking as Clustering

A very intuitive approach for event-based tracking is to perform clustering of events directly as they come from the sensors. Each cluster represents a moving object and the evolution of the cluster over time describes its trajectory. Such an approach was used in the work by Litzenberger et al. [84] and Schraml et al. [130]. The former presents an embedded vision system for tracking vehicles using a single DVS. Events generated by the sensor are assigned to circular clusters by a Euclidean distance where the position of the event is evaluated by the value of the cluster's seek radius. The algorithm was inspired by a mean-shift method, as the centre of the cluster moves toward the occurrence of the majority of the most recent events. The algorithm benefits from the stream-based processing; the events do not have to be buffered, therefore the low-memory constraint, that is important for embedded implementation, is satisfied. Schraml et al. [130] proposed an algorithm for people tracking in crowded areas using a stereo DVS mounted in an overhead position. Incoming events are assigned to the clusters by the Manhattan distance in space and time. Additionally, depth information together with local density of address events are used for noise suppression. The cluster's form and size are determined by the radial dilation factor, which depends on the spatial density of events. The clustering algorithms are well suited for embedded vision systems due to low memory usage and stream-based events processing. They are, however, dependent on experimentally adjusted parameters (e.g. object size limits) which are specific for a particular application. Moreover, both algorithms assume non-overlapping object trajectories, therefore the problem of occlusions in multiple object tracking has not been addressed by them.

7.2 Event Clustering with Gaussian Mixture Models

Building on the above-mentioned clustering methods, we propose to perform multiple object tracking through clustering events in three-dimensional space using Gaussian Mixture Models (GMMs). Our motivation towards applying the Gaussian Mixture Model is based on two main features of GMM clustering. Firstly, our algorithm intends to perform multiple object tracking in the presence of occlusions. Most of the distance-based clustering methods usually assume hard boundaries of the clusters. In the occurrence of pronounced occlusions, however, the resulting motion paths are overlapping and the strict boundaries between objects are difficult to define. GMMs offer the possibility to capture the uncertainty in the model and better tackle the return of the object after being occluded. Secondly, using a distance similarity measure to assign events to clusters presumes circularity of a cluster's shape. This assumption, however, may not hold in most of the real world applications. Describing clusters with a Gaussian distribution is less restrictive. The modelled clusters have a centre of gravity specified by the mean vector whereas the size and shape is loosely represented by the covariance matrix.

Having only one object moving in the scene, the task of tracking is trivial while all generated events represent motion of this object (not counting additional noise). However, multiple objects tracking is rather challenging, as events need to be associated with particular objects' motion paths. The dynamic vision sensor encodes only the relative light intensity change, thus we cannot use appearance features such as colour or texture to differentiate objects among each other. To this end, we propose to use the depth information estimated by the stereo matching algorithm. Adding another dimension helps with tackling the occlusions, since the objects are usually in different distances from the sensor. While estimating the object's motion path, we incorporate two assumptions about: *(i)* the spatial consistency, i.e. events generated by motion of the object occupy a coherent region in 3D space, and *(ii)* the temporal smoothness, which ensures that the position as well as the size of the object do not change drastically over time.

7.2.1 Object Representation

Figure 7.2a presents the events generated by four people walking in the scene. Each person can be characterised by a cluster of events in close spatial proximity, also referred to as the cluster's density. Furthermore, the motion of the object is also characterised by the amount of events generated at the particular address, as depicted in Figure 7.2b.

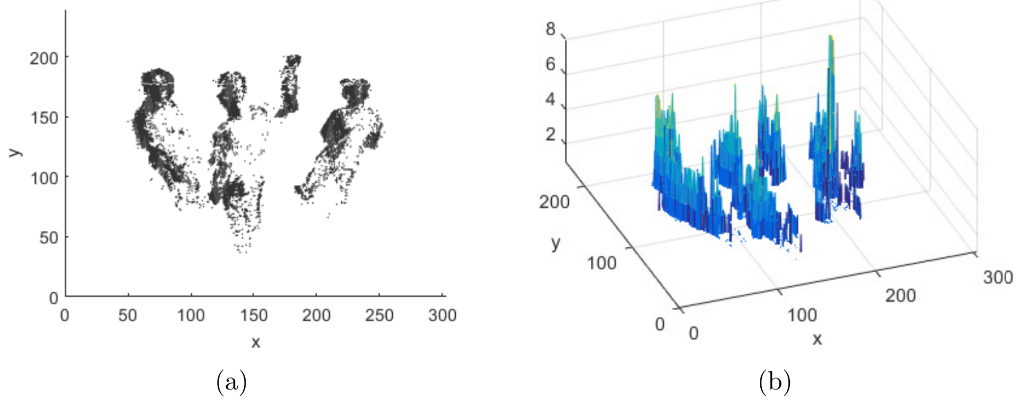


Figure 7.2: Events generated upon movement of four people presented as a projection onto XY plane (a). Amount of events generated at each (x,y) location (b).

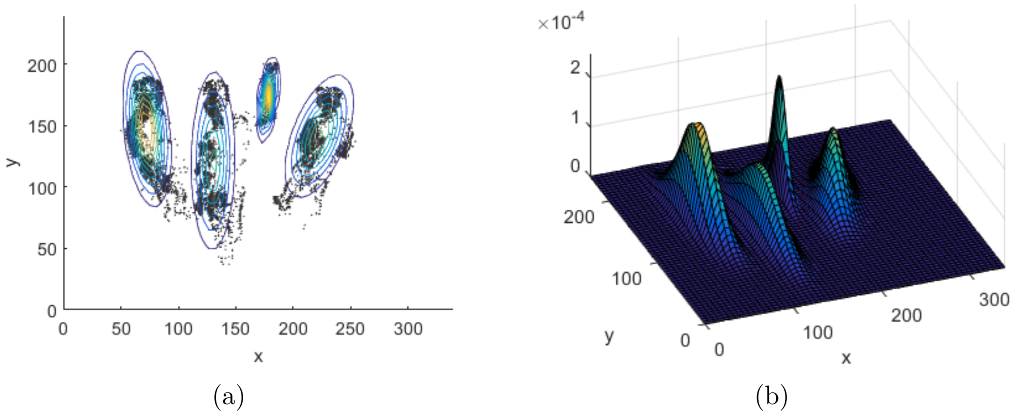


Figure 7.3: Gaussian Mixture model, comprising four components, describes the distribution of all events in space and time. The model is presented in (a) two-dimensional (contour) plot and (b) three-dimensional (surface) plot.

Let $e = (e_x, e_y, e_d, e_t)$ represent the event by its location (x, y, d), and time t:

$$e = (e_x, e_y, e_d, e_t) \mid e_x, e_y, e_d \in \mathbb{N}, e_t \in \mathbb{R} \tag{7.1}$$

The event disparity e_d is computed by the stereo algorithm. We assume that the events generated by a moving object belong to a Gaussian distribution. Then, GMM could be used to describe the events distribution of all K moving objects in the scene. A Gaussian mixture is defined as weighted sum of the mixture components, as given in Equation 7.2.

$$p(e|\lambda) = \sum_{i=1}^{K} w_i g(e|\mu_i, \Sigma_i), \tag{7.2}$$

where $g(e|\mu_i, \Sigma_i)$ is a component defined as d-variate Gaussian density function with mean μ_i and covariance matrix Σ_i:

$$g(e|\mu_i, \Sigma_i) = \frac{1}{(2\Pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}}\ exp\{-\frac{1}{2}(e - \mu_i)'\ \Sigma_i^{-1}\ (e - \mu_i)\} \tag{7.3}$$

In our case, we use a trivariate Gaussian density function ($d = 3$) as the events are clustered in three-dimensional space, including the address (e_x, e_y) and the estimated disparity e_d. As depicted in Figure 7.3, the fitted model captures the density in space and time. Please note that for visualisation purposes we only present the GMM over two dimensions (XY), however the underlying calculations are always done in 3D space. The complete GMM is defined by the parameters of each Gaussian component: the weight, mean vector and covariance matrix:

$$\lambda = \{w_i, \mu_i, \Sigma_i\}_{i=1\cdots K} \tag{7.4}$$

7.2.2 Model Parameters Estimation and Adaptation

Given a set of events $E = \{e_1, e_2, \ldots, e_M\}$ generated by moving objects over a particular period of time, the goal of tracking is to find such a GMM that best matches the distribution of E. In other words, to estimate the model parameters λ is to maximise the likelihood p of the GMM given the set of events E, as defined in Equation 7.5.

$$p(E|\lambda) = \prod_{n=1}^{M} p(e_n|\lambda) \tag{7.5}$$

An efficient approximation of the model parameters can be obtained by the Expectation Maximization (EM) algorithm (see Appendix A). Once the optimal mixture model $\lambda = \{w_i, \mu_i, \Sigma_i\}_{(i=1 \ldots K)}$ is found, each incoming event e_n is assigned to the i^{th} cluster by the value of the maximum a *posteriori* probability calculated as given in Equation 7.6.

$$P(i|e_n, \lambda) = \frac{w_i g(e_n|\mu_i, \Sigma_i)}{\sum_{k=1}^{K} w_k g(e_n|\mu_k, \Sigma_k)} \tag{7.6}$$

Clusters evolve in time due to the objects' motion, as illustrated in Figure 7.4. The optimal parameters for events generated in one time window will inherently need to be changed once the new events are generated.

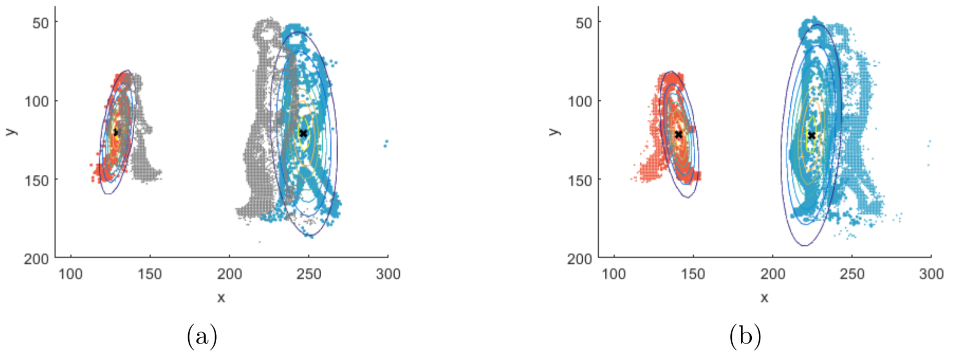


Figure 7.4: The clusters evolve in time, hence the model parameters need to be changed to fit the new events. The scene before (a) and after (b) the clusters' adaptation.

Although EM accurately estimates the Gaussian mixture parameters, it is very sensitive to new data, thus, not well suited for the task of tracking. Instead, we use the algorithm proposed by Reynolds [120], which was successfully used in speech analysis to derive a speaker model from a generic background model. We find the proposed method especially applicable to our case, as the new model parameters are to a high extent based on the previous (*prior*) model, which is nicely aligned with our assumption about the temporal smoothness of the motion. The model parameters are obtained through maximisation of the *a posteriori* having some initial *prior* model, in our case, the model from the previous time step.

Given *prior* model parameters λ_{prior} and new events (training vectors) $\{e_1, e_2, ..., e_M\}$, we first determine the probabilistic alignment of new events to the *prior* model, by computing the *a posteriori* as defined in Equation 7.6. Subsequently, the so-called *sufficient statistics* on the event data are calculated for each i^{th} component in the *prior* model. These are basic statistics required to estimate the desired adapted model parameters, namely the count c_i to compute the mixture weight:

$$c_i = \sum_{n=1}^{M} P(i|e_n, \lambda_{prior}), \tag{7.7}$$

the first moment S_i for the mean estimation:

$$S_i(e) = \frac{1}{c_i} \sum_{n=1}^{M} P(i|e_n, \lambda_{prior}) e_n, \tag{7.8}$$

and, finally, the second moment $S_i(e^2)$ used to calculate the variance:

$$S_i(e^2) = \frac{1}{c_i} \sum_{n=1}^{M} P(i|e_n, \lambda_{prior}) e_n^2 \tag{7.9}$$

Next, these new statistics are combined with the *prior* model parameters to obtain the adapted parameters for the i^{th} mixture component, that is for the adapted weight:

$$\widehat{w_i} = [\alpha_i^w c_i + (1 - \alpha_i^w) w_i)]\gamma, \tag{7.10}$$

mean:

$$\widehat{\mu_i} = \alpha_i^m S_i(e) + (1 - \alpha_i^m)\mu_i, \tag{7.11}$$

and variance:

$$\widehat{\sigma_i} = \alpha_i^s S_i(e^2) + (1 - \alpha_i^s)(\sigma_i^s + \mu_i^2) - \widehat{\mu_i^2} \tag{7.12}$$

The coefficients $\alpha_i^w, \alpha_i^m, \alpha_i^s$ control the balance between the old and new model parameters and their values depend on the expected weights of the mixtures. The γ coefficient in Equation 7.10 is used to ensure that weights sum to unity. The higher the amount of new events assigned to the cluster, the more it will be changed in the adaptation step. The values of adaptation coefficients are dependent on the state of the cluster, e.g. hidden or occluded clusters rely mostly on the last position (*prior* model), thus they require smaller values of adaptation coefficients.

7.3 Object Detection for Model Initialisation

The accuracy of the clustering algorithm to a high extent depends on the initial model parameters. Starting with the wrong assumptions about the number of moving objects, their position and size may result in error propagation and incorrect tracking, since the estimated model is inferred from the previous steps. In order to provide reliable initial model parameters, we propose an object detection algorithm which is based on the events histogram analysis. In Figure 7.5, we illustrate how the algorithm works.

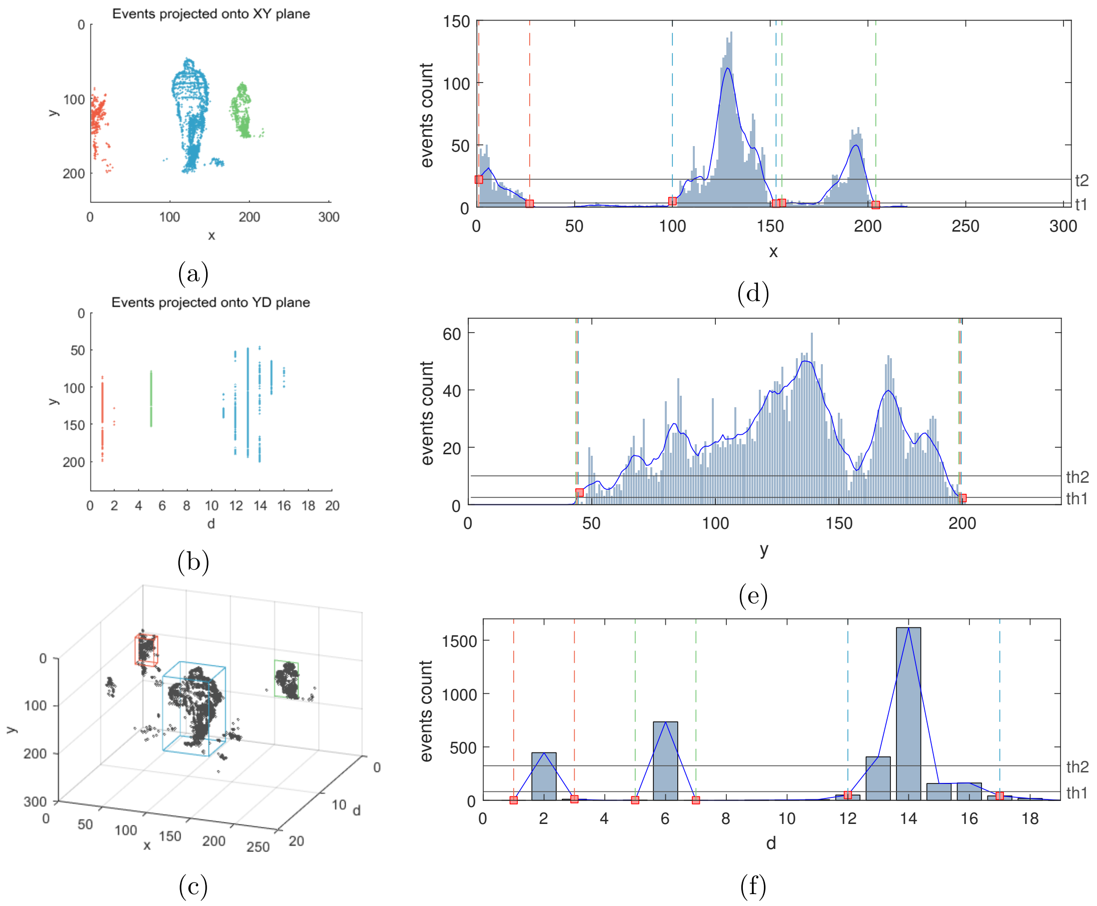


Figure 7.5: The event sequence projected onto XY (a) and XD (b) plane. The corresponding histograms of events are presented along the x, y, and d dimension in (d), (e) and (f), respectively. The peaks in histograms are detected and the boundaries of each object (person) are marked with dashed lines, in corresponding colours. The estimates of detected objects' bounding boxes are shown in (c).

As mentioned before, moving objects are represented by a group of events in close proximity in space and time. Looking at the histogram of events for a particular period of time, e.g. 10ms, along a given spatial dimension (see Figure 7.5d–7.5f), we can observe that peaks in the histograms correspond to objects' locations in this spatial dimension. The algorithm recursively searches for such peaks and analyses each combination of coordinates, to finally derive a list of objects' bounding box estimates, as marked in Figure 7.5c.

7.4 Tracking Algorithm

We have designed a tracking algorithm that can be applied to multiple object tracking, assuming that the objects are represented by quite dense event clouds. The algorithm automatically starts tracking after the initial model parameters are derived, as described in Section 7.3. The events are clustered continuously until a control function is triggered by exceeding either the time or model quality threshold. For example, the model is adapted periodically by the control function, every 10ms. However, if meanwhile there are many new events generated or the number of outliers is too high, the control function is triggered earlier. The control and adaptation part of the algorithm is depicted in the flowchart of Figure 7.6. In each control function iteration, the events that arrived in the time since the last update are clustered with the previous GMM (*prior* model).

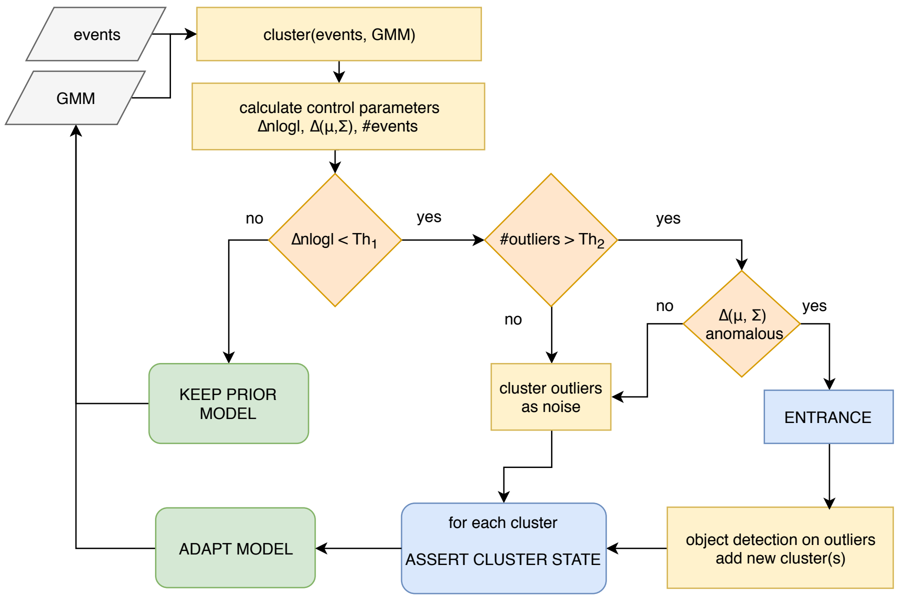


Figure 7.6: Flowchart of the tracking algorithm.

Subsequently, the control function parameters are computed, starting with the variation of the negative log-likelihood, denoted by $\Delta nlogl$. Negative log-likelihood estimates how well the model fits the current data. Once the control is triggered, the *nlogl* is calculated for both the *prior* model and the *posterior* model on the current event set. The *posterior* model is derived using the parameter update as defined in the EM algorithm, Equation A.4 - A.6. The difference $\Delta nlogl$ between *nlogl* of the *prior* and *posterior* model shows how much innovation the new data brings with respect to the previous time step. If $\Delta nlogl$ is close to zero that means the *prior* model fits the data well and no or very little adaptation is required. Otherwise, especially if the difference drops below threshold $Th_1 = -0.1$, an adaptation is necessary. In addition, we have observed that $\Delta nlogl$ lower than -0.6 usually indicates a significant configuration change, such as an entrance of new objects. Generally, in the task of multiple object tracking we need to handle special situations, such as the entrance of new objects, when the object gets occluded, stops moving or exits the scene. The algorithm treats the entrance of a new object separately, the other situations are considered a change of an existing cluster's state. We distinguish three cluster's states, namely *active*, *occluded*, and *hidden*, and transitions *T0-T6* between them as depicted in Figure 7.7.

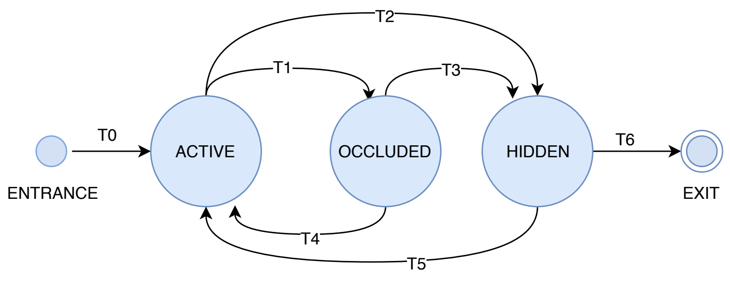


Figure 7.7: Diagram of cluster states and state transitions.

Once a new object is detected, it is by default considered *active*. From this state, it can change to being *occluded* or *hidden*, as indicated by transitions *T1* and *T2*, respectively. The *hidden* state reflects the situation where, for a particular cluster, there are no events generated that indicate the object's presence. This could mean that the objects has left the scene, stopped moving or is not visible to the sensor. The object's disappearance is described by transitions *T2* and *T3*. If an object is absent, i.e. *hidden*, for a longer period of time, it is assumed to have exited the scene (*T6*) and the object's tracker is terminated. However, if the object starts moving again after a short absence or is fully visible after being partially occluded, the tracker recognises the situation as the object's return (*T5* and *T4*, respectively). This is similar to the *buffer and recover* strategy [102] and particularly useful in handling full, transient occlusions. In what follows, we describe in more detail how the algorithm tackles the *entrance*, *exit*, and *occlusion*.

7.4.1 Entrance

A new cluster for an object is created when a significant amount of events (more than threshold Th_2) do not fit to any existing cluster in the *prior* model. The outlier detection is based on Mahalanobis distance calculated for each event. Another two parameters are used to differentiate noise events from the entrance of new objects, namely the change in mean and variance of the mixture components, denoted $\Delta\mu$ and $\Delta\Sigma$, respectively. Following the assumption about the temporal smoothness, we expect that neither position nor size of the object should change drastically over time. Therefore, any anomalous change in these parameters is assumed to indicate that clusters are being adapted to events of another (new) distribution. In Figure 7.8a we can observe the outliers corresponding to two new people (depicted in black), which could not be fitted to the *prior* model. We can clearly see the drastic drop in the value of parameter $\Delta nlogl$ (Figure 7.8d) and increase in the outliers' rate (Figure 7.8e). After calculating the *posterior* model for one (pre-existing) object (Figure 7.8b), the model changes in size to include the outliers, as indicated by the covariance change parameter (Figure 7.8f). Anomalies in clusters size or position indicate a possible configuration change and need for new clusters to allocate the outliers. To this end, the object detection is performed on the outliers to derive initial parameters for new objects' trackers. The model with two additional Gaussian components is then fitted to the data. The adapted model is depicted in Figure 7.8c.

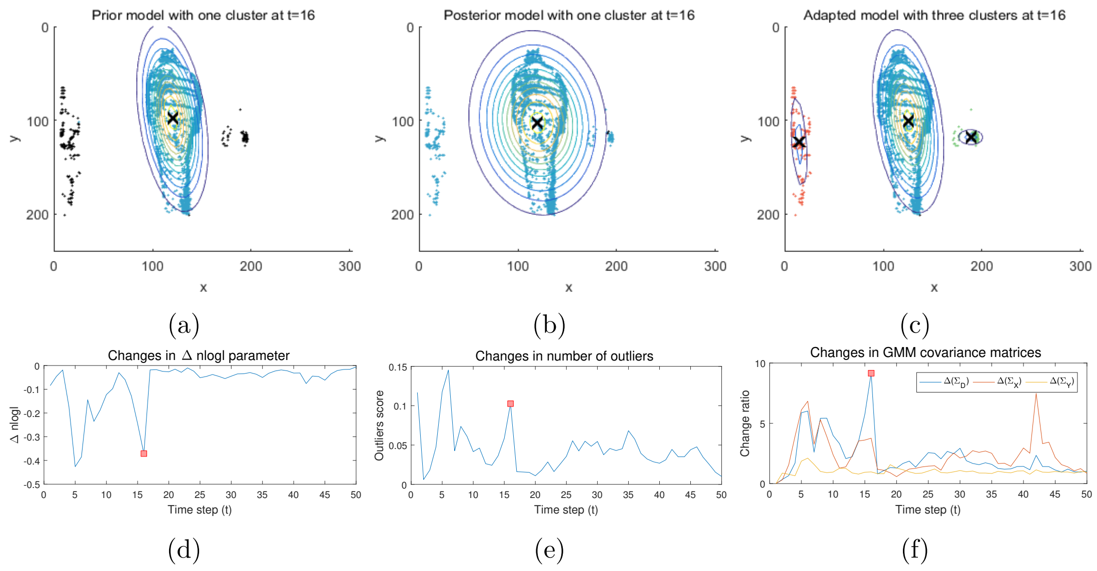


Figure 7.8: Entrance of two people (at time step t = 16): events clustered with *prior* model (a), estimated *a posteriori* with one mixture component (single person) (b) and adapted model with two additional clusters (c). Below, the control parameters: $\Delta nlogl$ (d), outliers' count (e), and $\Delta\Sigma$ (f), are plotted over the whole data sequence (annotations given for current time step).

7.4.2 Exit

As mentioned before, there are three possible reasons why an object disappears, either it exits the scene, is not visible or stops moving (*hidden*). In the proposed algorithm, the *exit* can only be reached through the *hidden* state. Figure 7.9 shows the gradual disappearance of a person. In the event sequence two people, interchangeably, are approaching and moving away from the camera and a third one (on the left) is sitting and slightly moving. The events at selected time steps are shown to illustrate the observed exit of the third person (Figures 7.9a–7.9c). The cluster corresponding to the third person first changes

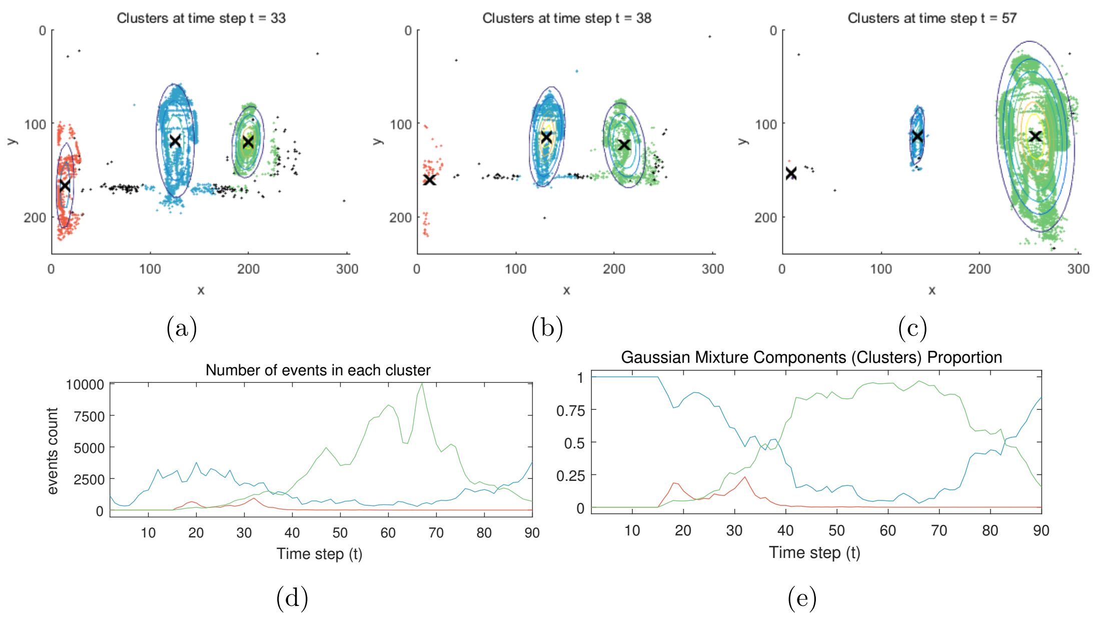


Figure 7.9: Events sequence demonstrating the exit of a person (depicted in red) from the scene presented by selected time steps (a) t = 33 (b) t = 38 and (c) t = 57. Control parameters (d) event count in each cluster, and (e) each cluster's weight are plotted over the whole sequence.

its state from *active* to *hidden* (*T2*), and then after a longer time of inactivity it is considered to exit the scene (*T3*). To detect the disappearance of the object, we look at two control parameters: the GMM components' weight and absolute amount of events in each cluster. Figure 7.9d and 7.9e show how these parameters' values change over time. We can clearly see that blue and green clusters change in size whilst they are farther or closer to the sensor, whereas the red cluster gradually disappears; starting from the step t = 33 both control parameters associated with the red cluster (third person) decrease to finally reach zero around step t = 40.

7.4.3 Occlusion

The occlusion handling is explicitly included in the design of our tracking algorithm. Analogically to the disappearance (exit) detection, when dealing with occlusion we look for anomalies in the amount of events assigned to the cluster. The control parameter used here, however, is not as previously the absolute amount of events in the cluster, but the ratio of its change over time. This allows for detection of partial occlusions, when a cluster changes its state from *active* to *occluded* (*T1*). Figures 7.10a - 7.10c depict two people walking in parallel to the sensor's image plane. One person gets occluded by another, who is walking closer to the camera. We can see that the person gets fully occluded and then returns after a short time.

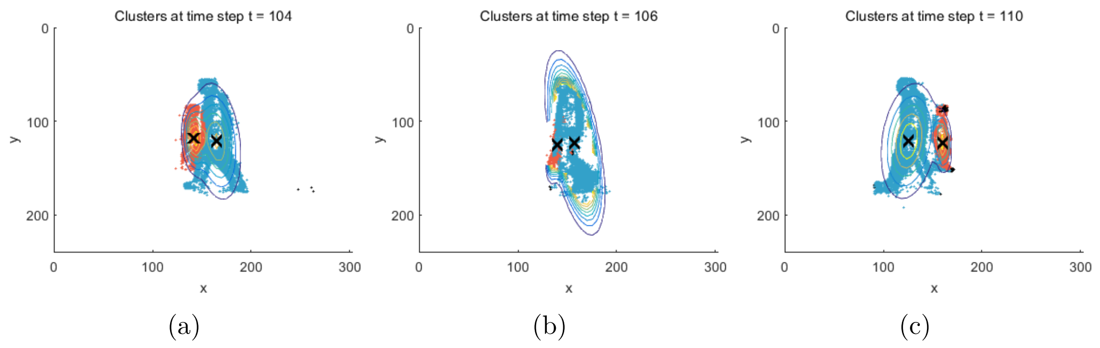


Figure 7.10: Event sequence showing one person occluding another. Selected key time steps demonstrate the scene shortly before (a), during (b), and after (c) occlusion.

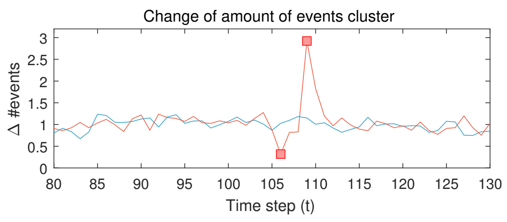


Figure 7.11: The change in amount of events in each cluster plotted for the whole sequence.

In Figure 7.11 the change in amount of events in the cluster is depicted. The red curve, corresponding to the occluded cluster, firstly shows a clear drop below 0.5 caused by occlusion and then jumps to above 2.5, which indicates the return of the cluster to *active* state.

7.5 Experimental Results

We evaluate our tracking algorithm on datasets described earlier in the context of stereo evaluation (Chapter 6), which contain both synthetic and recorded event streams. To provide a thorough evaluation of the proposed tracking algorithms, we have selected data sequences which include different tracking challenges such as partial and full occlusions, object stop or exit, as well as entrances of new objects into the scene. In order to facilitate the quantitative evaluation of the tracking algorithm, we needed to obtain ground-truth information about trajectories of objects moving in the scene, i.e. associate events to corresponding objects. For synthetic data, we render the ground-truth labels from the model. In the sensor recordings, however, the events have been labelled manually.

According to [140] there are three key properties of multiple object tracking that need to be evaluated, namely: configuration, identification and speed. The configuration refers to the correct estimation of the location and number of objects in the scene. The second property measures the accuracy of object identification, which in case of event-based tracking refers to assigning events to appropriate clusters. Finally, the performance of the algorithm in terms of speed measures how much time is required to process a video frame (image). Preferably, tracking should allow for *close-to-real-time* processing, e.g., assuming the frame rate of 25fps to enable processing of one frame in less than 40ms.

In the context of the first two properties, in order to check how well the objects are detected and located in comparison to ground truth, we performed a coverage test that includes two metrics: recall (ρ) and precision (ν), as explained in Equation 7.13.

$$\rho = \frac{|E \cap GT|}{|GT|} \qquad\qquad \nu = \frac{|E \cap GT|}{|E|} \tag{7.13}$$

The recall (ρ), also referred to as hit rate, calculates how many estimates E were correctly assigned to the cluster out of all ground-truth events GT, whereas the precision (ν) measures how many of the events assigned to the cluster are ground-truth events. Moreover, the algorithm's performance in terms of speed is assessed by two further metrics: *(i)* how many events the algorithm is capable of processing in one second, and *(ii)* time taken to process one second of data. The latter depends on the density of events in space and time, which is directly related to scene characteristics such as the number and speed of the moving objects.

7.5.1 Object Tracking

We start with tests on synthetic data. The first sequence, *synth-Scene1*, depicts six objects of different shapes and sizes moving around the scene. The selected time steps of tracking results are presented in Figure 7.12. The clusters are denoted by different colours and the estimated mixture models, represented by the cluster centre μ and contours of the computed Gaussian function. The algorithm has been applied to stereo estimates calculated by the *Coopv1* algorithm (see Section 6.2.1). The tracking algorithm successfully tackled configuration changes, such as an object's stop, return, entrance and occlusion, as can be observed in Figure 7.12, at time steps t_2, t_3, t_4, and t_{10}, respectively. In addition, in order to visualise the tracking results over the whole sequence, in Figure 7.13 the trajectories of each clusters' centre are plotted in top-down view (x and d coordinates). The ground truth trajectory is given for reference in Figure 7.13a, and two further plots present results of tracking applied to stereo estimates calculated by *Coopv1* (Figure 7.13b) and *Coopv2* (Figure 7.13c). The colours of the clusters are kept consistent with Figure 7.12 to facilitate the understanding of the trajectory plots. As can be observed, the tracking applied to both stereo algorithms' results are very similar to ground truth, except for the red cluster (*obj-2*) which shows some inconsistencies at the end of the sequence (at time steps t_9 - t_{11}). This is mainly caused by the fact that the object moves very slowly, thus is represented by a small number of events. Due to that, the estimated cluster's position starts adapting to events that are either noise or events generated by other clusters but with wrong stereo estimates.

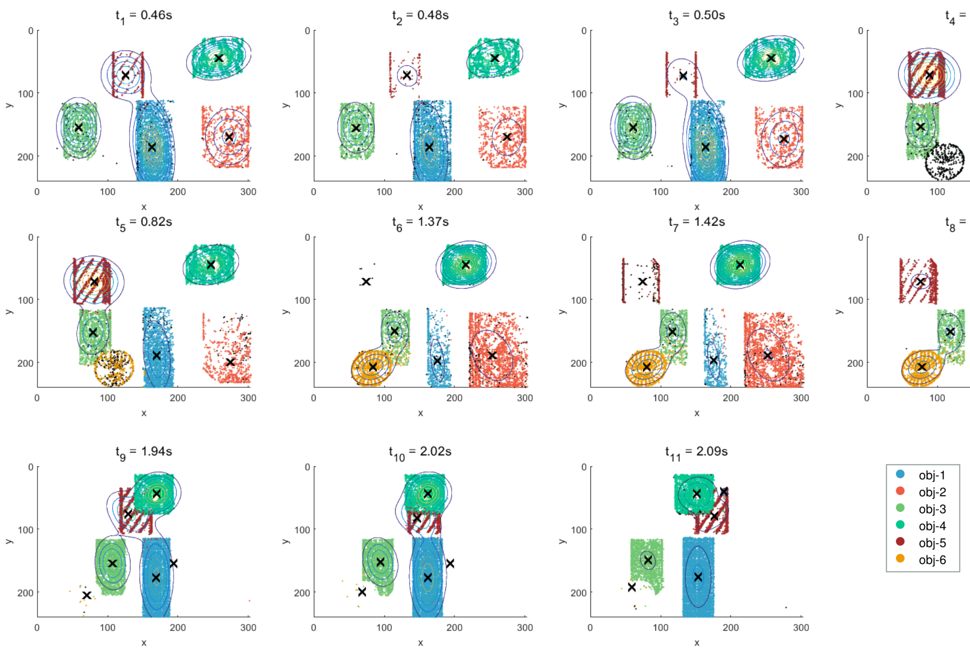


Figure 7.12: Selected time steps from the *synth-Scene1* sequence. Tracking has been applied to results of the *Coopv1* stereo matching algorithm.

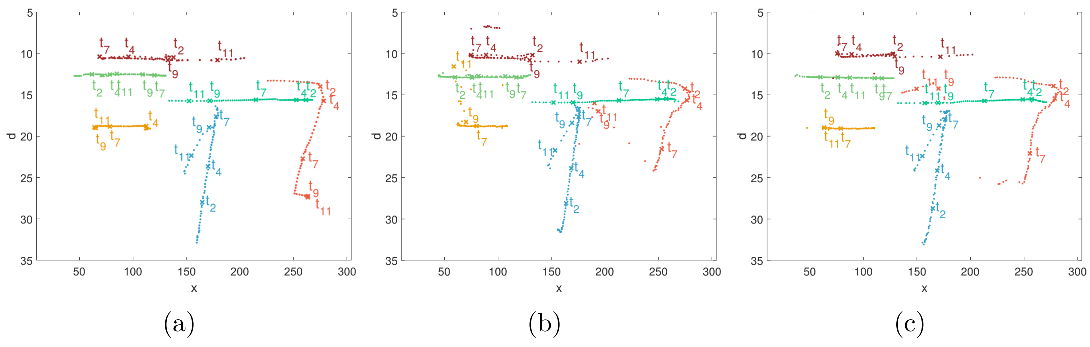


Figure 7.13: Trajectories of *synth-Scene1* clusters' centres for ground truth (a) and results of tracking applied to *Coopv1* (b), and *Coopv2* (c) stereo results.

The second synthetic event sequence, *synth-Scene2*, depicts three balls moving at different speeds. It has been designed for testing the asynchronous aspect of the algorithms, since there is no optimal time window that would fit for processing of all three objects. The tracking algorithm, in this regard, allows for asynchronous processing, since the adaptation function is called whenever the mixture model no longer fits the incoming event data. Figure 7.14 presents results of the tracking algorithm applied to *Coopv2* stereo estimates. In order to give an overview of the tracking results, the ground-truth

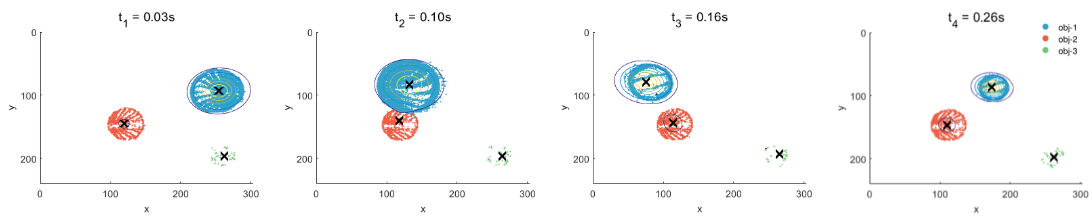


Figure 7.14: Selected time steps from the *synth-Scene2* sequence. Tracking has been applied to results of the *Coopv2* stereo algorithm.

trajectory plots are shown in Figure 7.15a. The most prominent trajectory is that of the blue cluster (*obj-1*), which represents the ball which moves at the highest speed, whereas the other objects move at relatively lower speed and, hence, have shorter trajectories. We can observe that the results of tracking applied to disparity estimates calculated by the *Coopv2* algorithm (Figure 7.15c) are almost the same as the ground truth. However, in the tracking results obtained using *Coopv1* (Figure 7.15b), we can observe that *obj-1* loses track of the fastest ball (blue), which is then picked up by the red cluster (*obj-2*), and finally a new cluster (depicted in green) is created to accommodate the events.

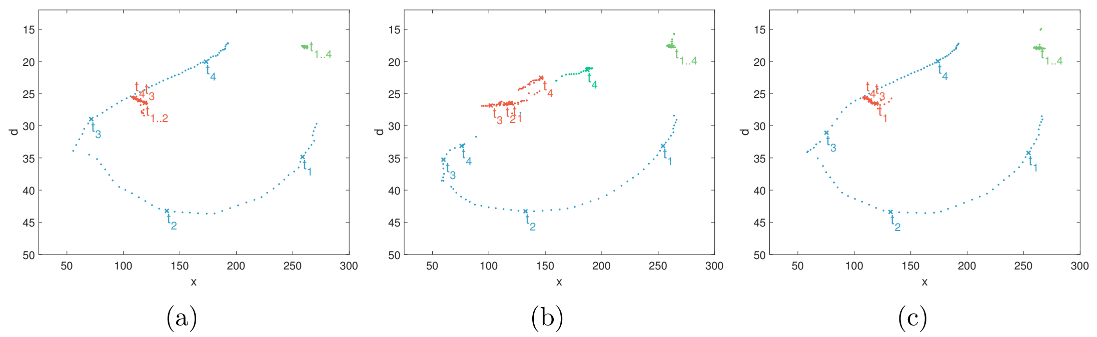
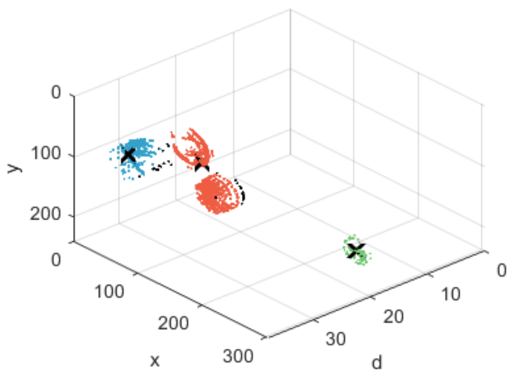


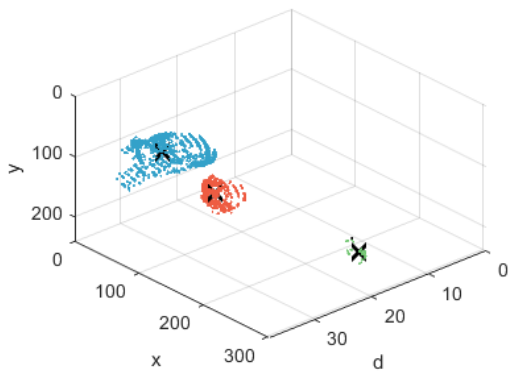
Figure 7.15: Trajectories of *synth-Scene2* clusters' centres for (a) ground-truth and results of tracking applied to *Coopv1* (b), and *Coopv2* (c) stereo results.

The reason for this tracking failure is explained in Figure 7.16, where clustered events are depicted in three-dimensional space (x,y,d). In Figure 7.16a we can see that the disparity estimates obtained from the *Coopv1* algorithm are incorrect for some parts of the moving ball, hence causing the clustering error. For comparison, the results of *Coopv2* are shown in Figure 7.16b, where the disparity estimates are changing smoothly over time. The correct behaviour of the tracking algorithm heavily depends on the quality of the disparity estimates.

The numerical results of the tracking algorithm are summarised in Table 7.1. For *synth-Scene1*, tracking on both stereo estimates gives comparable results. In *synth-Scene2*, due to the clustering failure described above, the precision and recall are lower for *Coopv1* than for *Coopv2*. In general, dealing with both stereo algorithms, the scores for recall (hit



(a)



(b)

Figure 7.16: Comparison of tracking results obtained from *Coopv1* (a) and *Coopv2* (b) stereo estimates. In (a) the tracking algorithm fails to correctly cluster some parts of the object, because the object's trajectory is not continuous in space and time.

Measure	*synth-Scene1*		*synth-Scene2*	
	Coopv1	*Coopv2*	*Coopv1*	*Coopv2*
precision	0,989	0,984	0,850	0,994
recall	0,924	0,907	0,841	0,988

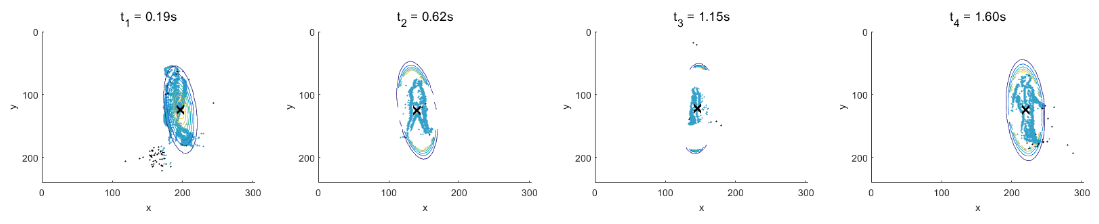
Table 7.1: Results of the tracking algorithm on synthetic datasets.

rate) tend to be lower than for precision. This means that the algorithm can accurately distinguish between clusters, however, still misses assignments of some events.

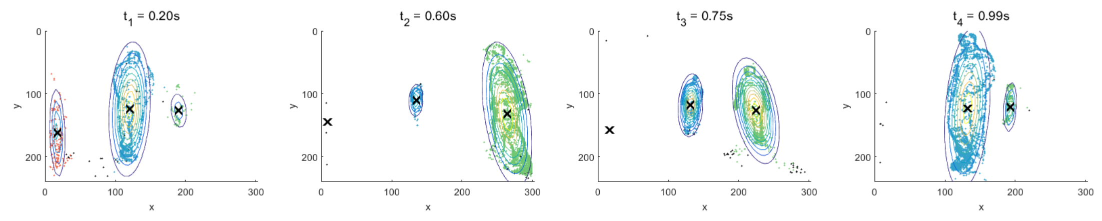
In addition, we calculated the configuration accuracy as the amount of time steps where the number of objects is correctly recognised divided by the overall amount of time steps in the sequence. The configuration accuracy drops below 1 (equals 0.62) only in the example presented above, i.e. the *synth-Scene2* tracking applied to the *Coopv1* stereo results. Otherwise, the tracking correctly recognises the number of objects throughout the duration of the sequences. In terms of performance, the algorithm processes approximately 235 kEv/s (i.e. kilo events per second), and takes on average 13 seconds to process one second of the data sequence. The latter score is relatively high, because the synthetic data sequences are quite high in spatiotemporal density of the events.

7.5.2 People Tracking

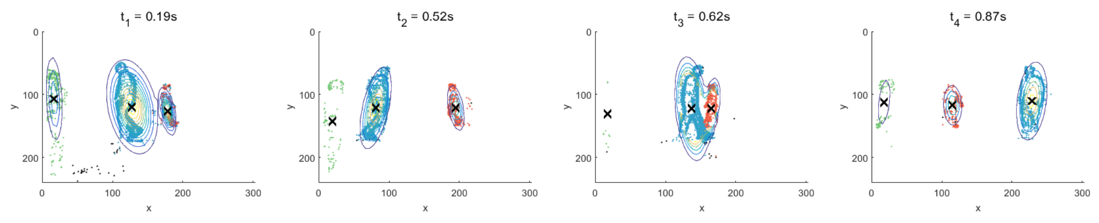
After the initial experiments on the synthetic datasets, we perform further tests on the event sequences recorded with stereo DVS. The test data depict people walking in a room, in the depth range between 1.5m and 3m. Figure 7.17a shows selected time steps of the *people1* sequence with one person walking. This is the easiest case, where the only task is to correctly distinguish events generated by the person from noise events. The next sequences depict three people, two of them moving back and forth (*people2*) and in parallel to the sensor's image plane at two different distances (*people3*), and a third person sitting and slightly moving. An overview of the *people2* sequence is presented in Figure 7.17b, and *people3* in Figure 7.17c. It is worth to note that both stereo algorithms (*Coopv1* and *Coopv2*) cannot provide a correct depth estimates for the third person in the *people2* and *people3* sequences, due to the half-occlusion, i.e. parts of the person are only seen in one (left) camera. Nevertheless, each variant of cooperative stereo expresses different tendencies in coping with such ambiguity in matching, which in turn might have an impact on the accuracy of the tracking algorithm.



(a) *people1*



(b) *people2*



(c) *people3*

Figure 7.17: Selected time steps from the *people1* (a), *people2* (b), and *people3* (c) test sequence. Tracking has been applied to results of the *Coopv1* stereo algorithm.

Figure 7.18 presents the comparison of trajectories estimated for results of *Coopv1* and *Coopv2*. The overall performance of both stereo variants is quite similar. In case of *Coopv2*, the disparity estimates tend to be more aligned to front-to-parallel planes, causing discontinuities in cluster representation across the disparity domain. This in turn may result in higher configuration error, i.e. there are several trackers assigned to one object, as can be observed in Figure 7.18f. The cooperative stereo algorithms, both *Coopv1* and *Coopv2*, work in continuous manner and calculate depth based on the history of

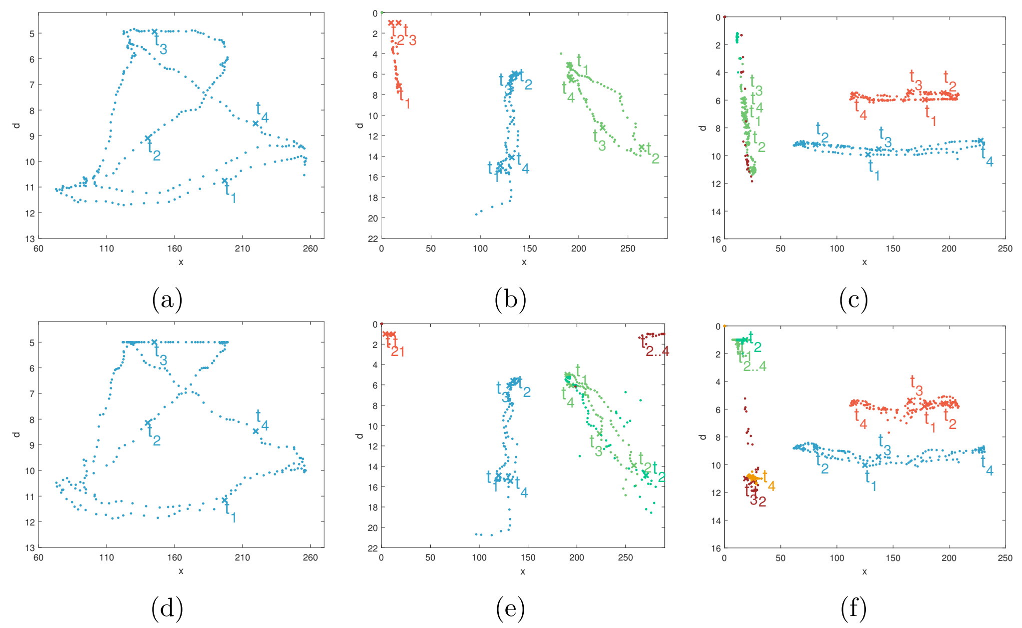


Figure 7.18: Trajectories estimated for *people1* (a,d), *people2* (b,e) and *people3* (c,f) event sequences. The results are compared for stereo estimates obtained by *Coopv1* (a,b,c) and *Coopv2* (d,e,f).

depth estimates stored in the cooperative network. For this reason, sometimes, when one object occludes the other, there still will be some events assigned to the occluded area based on the historical events available in the cooperative network. This results in the 'see through' effect, as presented in Figure 7.19. In consequence, the cooperative stereo algorithms support the task of handling the transient occlusions.

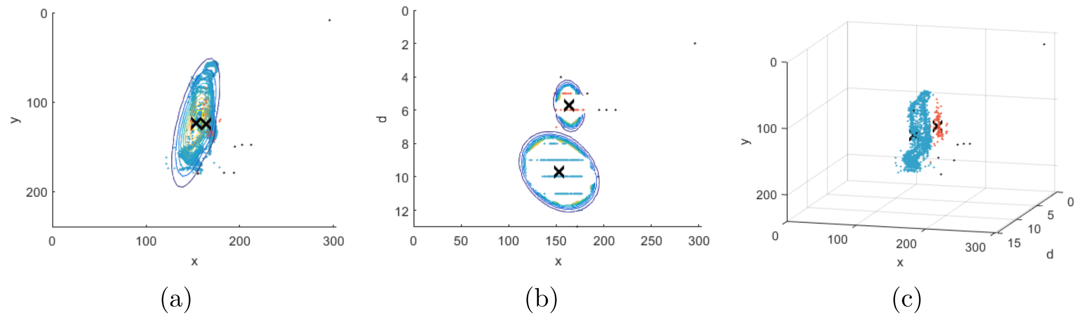


Figure 7.19: Part of the *people3* sequence demonstrating the full occlusion of one person by another, projected onto XY plane (a), XD plane (b), and shown in 3D space in (c). Some of the events from the foreground person are assigned a disparity of the background person based on historical estimates in the cooperative network. This appears as a 'see through' effect.

There are some limitations of the event-based tracking, which can be observed in the results. To begin with, the sensors generate events upon any perceived motion, including also shadows. In Figure 7.20, we can observe that the shadow (lower right corner) of the person (in green) is represented by event clouds of density sufficient to be recognised as two additional objects.

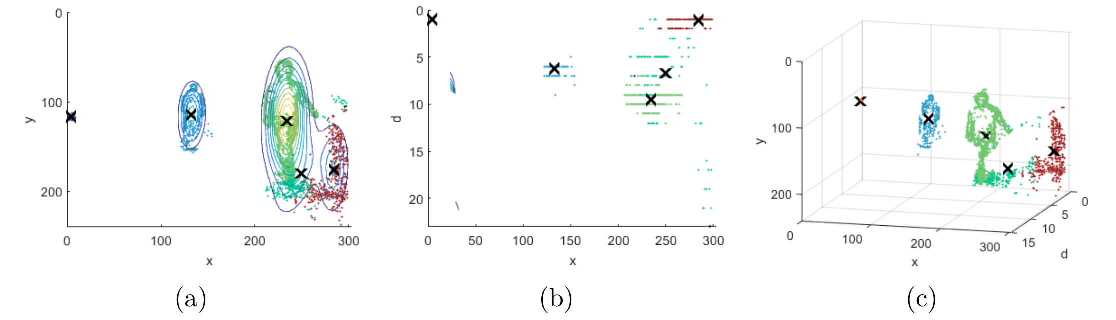


Figure 7.20: Clustering events generated by shadow. The events are presented in three ways, in two dimensions: front view (XY plane) (a), top-down view (XD plane) (b), and in three dimensions (XYD) (c).

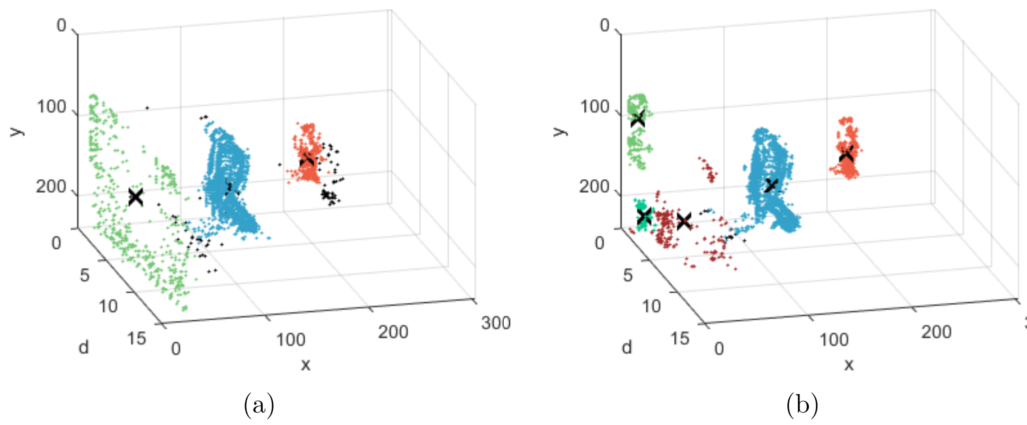


Figure 7.21: Comparison of results of the tracking applied to *Coopv1* (a) and *Coopv2* (b) tackling the entrance of a new person. The figure demonstrates that with de-fragmented clusters the tracking assigns more trackers to one object.

As mentioned previously, since the tracking algorithm is based on density-based clustering, it gives better results when events of the same object are represented by a continuous cluster of events. Otherwise, the tracking may recognise disjoint elements of the object as separate objects, which results in the assignment of several clusters to one object. In Figure 7.21 we compare how the entrance of a half-occluded person is handled by tracking applied to *Coopv1* and *Coovp2* stereo estimates. In case of *Coopv1*, the entering person is correctly clustered as one object, because the events are evenly distributed across disparity levels, as can be seen in Figure 7.21a. On the contrary, dealing with depth estimates from *Coopv2* (Figure 7.21b), the tracking algorithm erroneously recognises three clusters which are assumed to be separate objects.

The numerical results of the tracking algorithm on *people* datasets are presented in Table 7.2. Both precision and recall are quite high and consistent across all datasets and two cooperative stereo algorithms. The tracking limitations presented above usually manifest themselves in configuration errors.

	people1		*people2*		*people3*		*people4*
	Coopv1	*Coopv2*	*Coopv1*	*Coopv2*	*Coopv1*	*Coopv2*	[133]
precision	0,992	0,990	0,964	0,982	0,961	0,959	0,968
recall	0,988	0,991	0,959	0,959	0,950	0,953	0.919

Table 7.2: The numerical results of tracking algorithm on *people* datasets.

In the *people2* and *people3* sequence, the configuration errors are mainly caused by a half-occluded person. We measured the fraction of the whole sequence duration when our tracking algorithm correctly identified the number of objects in the scene. For the *people2* sequence, *Coopv1* achieves 0.86 configuration accuracy, whereas *Coopv2* performs significantly worse, i.e. the number of objects is correct only within 0.29 time steps of the sequence. For the *people3* sequence, both *Coopv1* and *Coopv2* result in similar configuration accuracy, around 0.78. In terms of performance, the tracking algorithm achieves a speed of approximately 152 kEv/s. The average time taken to process one second of the *people* sequences is 0.2s. Therefore, in real-world recordings of moving people, even an unoptimised Matlab implementation of the algorithm still allows for real-time tracking.

In addition, we present the event sequence *people4*, which has been used in the initial validation of the tracking algorithm, presented in [112]. The sequence depicts four people walking in the room, partially or fully occluding one another. This particular data sequence has been chosen to demonstrate the ability of the tracking algorithm to tackle multiple person tracking in the presence of occlusions. The tracking has been applied to depth estimates calculated using the algorithm from [133]. We have measured the precision and recall based on the manually labelled ground truth. The algorithm achieves 0.968 recall and a slightly smaller precision, namely 0.919. The visual results of the tracking on *people4* are presented in Figure 7.22 and 7.23. The first one demonstrates correctly identified people entering the room, one after another. Figure 7.23 shows the results of tracking dealing with occlusions. We can see that the algorithm can successfully differentiate events across clusters, even in a quite challenging setup (crowded scenes).

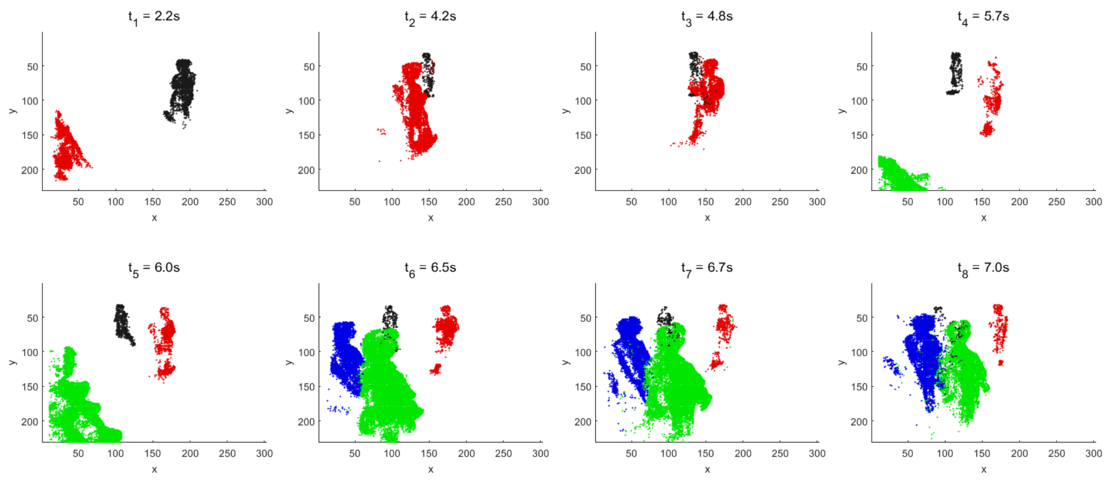


Figure 7.22: Part of the *people4* sequence where people sequentially (one after another) enter the scene. The identified people can be differentiated by colours.

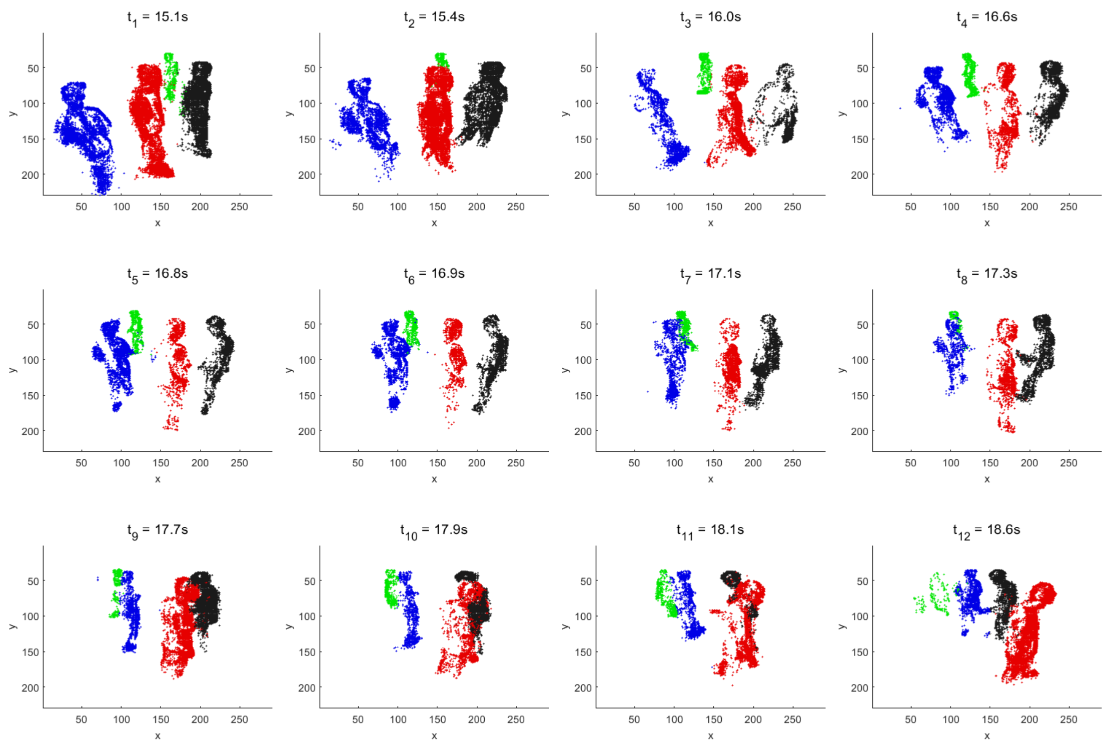


Figure 7.23: Results of the tracking on the *people4* sequence. The algorithm correctly identifies moving people in a crowded scene, and successfully tackles the partial occlusions.

7.6 Summary

In this chapter, we have presented an algorithm for multiple objects tracking using dynamic vision sensors. We have formulated the task of tracking as clustering events in three-dimensional space, using as input information about the event's spatial coordinates (x,y) and disparity estimated by the stereo matching algorithm. We have investigated the possibility to use Gaussian Mixture models, not only to detect objects but also to track their evolution in time.

The proposed tracking algorithm successfully deals with multiple moving objects of different size and speed. The clusters are continuously adapted to the new incoming events. The algorithm facilitates the asynchronous clustering of events, since the control function for cluster adaptation is called not only in predefined time intervals, but also upon major changes in the scene characteristics. As demonstrated by experiments, the tracking algorithm is able to tackle both partial and full occlusions, as well as changes in scene configuration, such as the entrance of new objects, when objects stop moving or exit the scene. An advantage of the algorithm is that objects are clustered in three dimensions (x,y and disparity), which makes the tracking more accurate and robust than its application to two-dimensional data. Nevertheless, the results are very dependent on the quality of the input disparity estimates. If parts of the object are assigned with incorrect disparities, the tracking algorithm could either assign them to a wrong cluster or erroneously create a new one, assuming the entrance of a new object. Furthermore, the clustering with Gaussian Mixture Models requires a careful choice of the initial model parameters. To this end, we proposed an efficient and accurate object detection algorithm, which finds the object locations by analysis of event histograms. The object detection is used in the first step of the algorithm to provide an initial *prior* model of the scene. In addition, it is also used to tackle the entrance of new objects, which was found especially useful in cases where several objects enter the scene at the same time.

We have demonstrated that the proposed method is well-suited to the nature of the event data. The overall performance of the algorithm is satisfying, reaching both a precision and recall on average higher than 0,96. Nevertheless, one of its limitation is the need of adjusting the algorithm thresholds (outlier and anomalies detection) using some assumptions about the scene and objects characteristics (e.g. average size, estimated depth range, granularity of clusters, etc.). Moreover, we observed that there are some tracking challenges, which might be very difficult to tackle algorithmically. For example, if objects are moving close to each other in the same direction, they form a coherent cluster of events, making it impossible to distinguish one object from the other. Another challenge are events generated by shadows, which are either recognised as part of the object, or a new one moving next to it.

CHAPTER 8

Conclusions

In this thesis, we explore feasible approaches to asynchronous processing of event data with a special focus on depth and motion estimation from a stationary stereo dynamic vision sensor (DVS) system. In the following, we summarise the main conclusions of this PhD research.

8.1 Summary

A major part of this thesis deals with event-based stereo correspondence. We have identified two main challenges of event-based stereo matching: the first one is associated with ambiguity in event-to-event matching, whereas the second is related to asynchronous event data processing. According to our experimental analysis, the commonly used time-based event-to-event matching is very prone to errors and cannot be reliably used in complex scenes. Furthermore, the response from the left and right sensor may vary due to a number of factors, such as viewing angle, relativity of objects' motion velocity, or specifics of DVS hardware platforms. Our experiments show that, exposed to the same visual stimuli and under the same conditions, there could be around 8 percent difference between the amount of left and right events. We derive that additional assumptions may be useful to resolve the matching ambiguity, e.g. a local smoothness constraint. Therefore, we introduce a smoothness constraint that is inherently associated with the aggregation of local context over time. We further demonstrate that fixed-time event aggregation limits the temporal resolution of the sensor, hence the second challenge we face is to find an appropriate way of event-based processing that preserves the asynchronous aspect of event data and thus allows exploiting the full potential of dynamic vision sensors. We propose two novel asynchronous event-based stereo matching algorithms that are tailored to the advantages and peculiarities of dynamic vision sensors.

The first algorithm, *Coopv1*, tackles the challenge of asynchronous stereo matching. We leverage an early model of biological depth computation proposed by Marr and Poggio [93] to model the stereo problem with a dynamic cooperative network where the nodes of the highest activation denote the correct disparity. We extend the basic cooperative algorithm into the spatiotemporal domain by introducing a dynamic cooperative network that stores the history of recent activity in the scene. The cooperative aspect can be considered as a refinement of temporal event-based matching as it implements the smoothness constraint by local neighbourhood operations. The network operates in continuous time and thus preserves both spatial and temporal smoothness. In our experiments, we find that the *Coopv1* algorithm improves the accuracy of single time-based correlation by approximately 40 percent.

The second stereo matching algorithm, *Coopv2*, is focused on improving the accuracy of event-based matching. Building on the *Coopv1* algorithm, we propose an enhanced cooperative stereo matching technique which calculates the similarity over a local neighbourhood of each event pair to compute the initial matching weights for the cooperative optimisation. A ground-truth based evaluation shows that using the local neighbourhood in event matching (*Coopv2*) reduces the measured mean error by over 50 percent. A quantitative comparison with a competing image-based stereo algorithm confirms the favourable performance of the *Coopv2* algorithm with respect to the current state-of-the-art.

In addition to ground-truth evaluation, we have tested both of the proposed cooperative stereo matching algorithms using a variety of datasets, including self-recorded sequences of non-rigid human motion, synthetic data of complex scenes with challenging occlusion cases, and recordings captured by a moving DVS. Our experiments demonstrate the robustness of the cooperative algorithms against varying scene complexity and simultaneous motion of contrasting speed. Furthermore, we show the applicability of our algorithms to a dynamic stereo setup, which has recently gained attention in the context of dynamic vision sensors.

An intuitive application of dynamic vision sensors is motion analysis. In this context, we propose an event-based multiple object tracking algorithm that leverages the disparity estimated by the cooperative algorithms to tackle the challenges of occlusions. The algorithm performs clustering of events in three-dimensional space by describing the scene activity with Gaussian Mixture Model. The clusters' dynamics are accounted for by constant adaptation of the model to the incoming events. In our experiments, we show that the algorithm gives satisfying results and is able to tackle different tracking challenges, such as a changing number of moving objects, or partial and full occlusions. In tests on tracking people, the algorithm achieves both precision and recall above 0.96, and is capable of real-time performance.

8.2 Future Work

We have demonstrated that the proposed adaptive cooperative stereo algorithms can successfully cope with the asynchrony of events and provide good quality results, regardless of the speed of moving objects. However, there are still open areas that could be addressed in future research:

Adaptive support Incorporating an adaptive local neighbourhood directly into the cooperative network might further improve the accuracy by adaptation of the support weights to better reflect the underlying objects' structure in space and time.

Geometrical constraints Another possibility in matching that appears especially applicable for sparse event data is to employ geometrical constraints on incoming events. An option would be to refine the initial event-based matching by line or surface fitting in 3D space.

Postprocessing One direction to explore would be to shift the focus from the initial matching step to more advanced post-processing techniques, e.g. filtering and smoothing.

Combined stereo and tracking The tasks of stereo and motion computation are correlated. We demonstrated in Chapter 7 that using stereo information in tracking is beneficial and helps handling occlusions. Further research could investigate how information provided by the tracking algorithm, such as the clusters' position, size or speed, could also be used to improve the stereo matching accuracy.

APPENDIX A

Expectation Maximization

The Expectation Maximization (EM) [40, 120] algorithm is used for the estimation of parameters of a statistical model with unknown parameters and known observations. The algorithm performs an iterative search for a (local) maximum likelihood or Maximum A Posteriori (MAP). In this thesis, the Expectation Maximization is used in the context of the proposed tracking algorithm, i.e. clustering events with Gaussian Mixture Models (GMM).

A Gaussian Mixture Model is defined by the parameters of its n components, as given in Equation A.1:

$$\lambda = \{w_i, \mu_i, \Sigma_i\}_{i=1,\cdots,n}, \tag{A.1}$$

where w_i, μ_i, Σ_i denote, respectively, the weight, mean vector and covariance matrix of the i^{th} Gaussian component.

Given a T-element set of data $X = \{x_1, x_2, \cdots, x_T\}$, the goal is to find GMM λ that best matches the distribution of X. The EM algorithm iteratively changes the model parameters to maximise the likelihood $p(X|\lambda)$ between the data X and the corresponding Gaussian distribution λ:

$$p(X|\lambda) = \prod_{t=1}^{T} p(x_t|\lambda) \tag{A.2}$$

In each iteration, the model parameters are evaluated for each d-dimensional sample from the dataset X by calculating the *a posteriori* probability for the i^{th} component by formula:

$$P(i|x_t, \lambda) = \frac{w_i\, g(x_t|\mu_i, \Sigma_i)}{\sum_{k=1}^{n} w_k\, g(x_t|\mu_k, \Sigma_k)}\ , \tag{A.3}$$

where $g(x_t|\mu_i, \Sigma_i)$ is a component defined as d-variate Gaussian density function with mean μ_i and covariance matrix Σ_i. The dimensionality d is determined by the dimensionality of elements of the underlying dataset X.

Next, the parameters are re-estimated according to the *a posteriori* probability derived in a previous step. The new GMM parameters are calculated as follows:

$$\bar{w_i} = \frac{1}{T}\sum_{t=1}^{T} P(i|x_t,\lambda) \tag{A.4}$$

$$\bar{\mu_i} = \frac{\sum_{t=1}^{T} P(i|x_t,\lambda)x_t}{\sum_{t=1}^{T} P(i|x_t,\lambda)} \tag{A.5}$$

$$\bar{\sigma_i} = \frac{\sum_{t=1}^{T} P(i|x_t,\lambda)x_t^2}{\sum_{t=1}^{T} P(i|x_t,\lambda)} - \bar{\mu_i}^2 \tag{A.6}$$

The steps are repeated until reaching convergence, which could be controlled by a likelihood improvement threshold or maximum number of iterations.

APPENDIX B

Synthetic Data Generation

For the purpose of extensive evaluation of the presented algorithms, beside the recorded sensor data, we have also used several synthetic test scenarios. In the following, we briefly describe the procedure of generating synthetic stereo datasets. Firstly, the synthetic scenes, including objects, their animation and stereo camera are modelled using Blender[1]. We have used the same camera settings as ATIS with respect to pixel size, sensor resolution and baseline between the left and right camera. The animation is rendered, i.e. projected separately into the views of both the left and right camera, at high frame rate (1000 fps). The resulting rendered images are converted to greyscale and used for the reconstruction of event streams separately for each view. The sequence of images is combined into a spatiotemporal cube of X×Y×F dimensions, where X×Y is the spatial resolution of the sensor (virtual camera) and F is the number of rendered frames. The cube serves as an input to the dynamic vision sensor simulator, which converts greyscale images into a sequence of events. In addition, the simulator behaviour can be adjusted by a set of parameters, such as the preprocessing (smoothing filter size), magnitude of simulated noise and sensor's bias settings, such as pixel sensitivity thresholds (ON, OFF), refractory period and timestamp resolution. Analogically to the real sensor, each pixel is independent and its behaviour is simulated by the *Event Generator*[2], that implements a pixel's function as described in Section 2.2. The *Event Generator* generates events based on the contrast changes for a single pixel over a whole image sequence duration (all frames).

Figure B.1 demonstrates the results of the event generator function called for one pixel. The top graph shows the continuous contrast change of a pixel, interpolated to the required timestamp resolution from the input sequence frame number (F) and rendering frame rate. Below, the output of the simulated differentiating circuit is given, showing

[1] https://www.blender.org/

[2] Functions for sensor simulation and event generation were written and made available by courtesy of Stephan Schraml, AIT Austrian Institute of Technology.

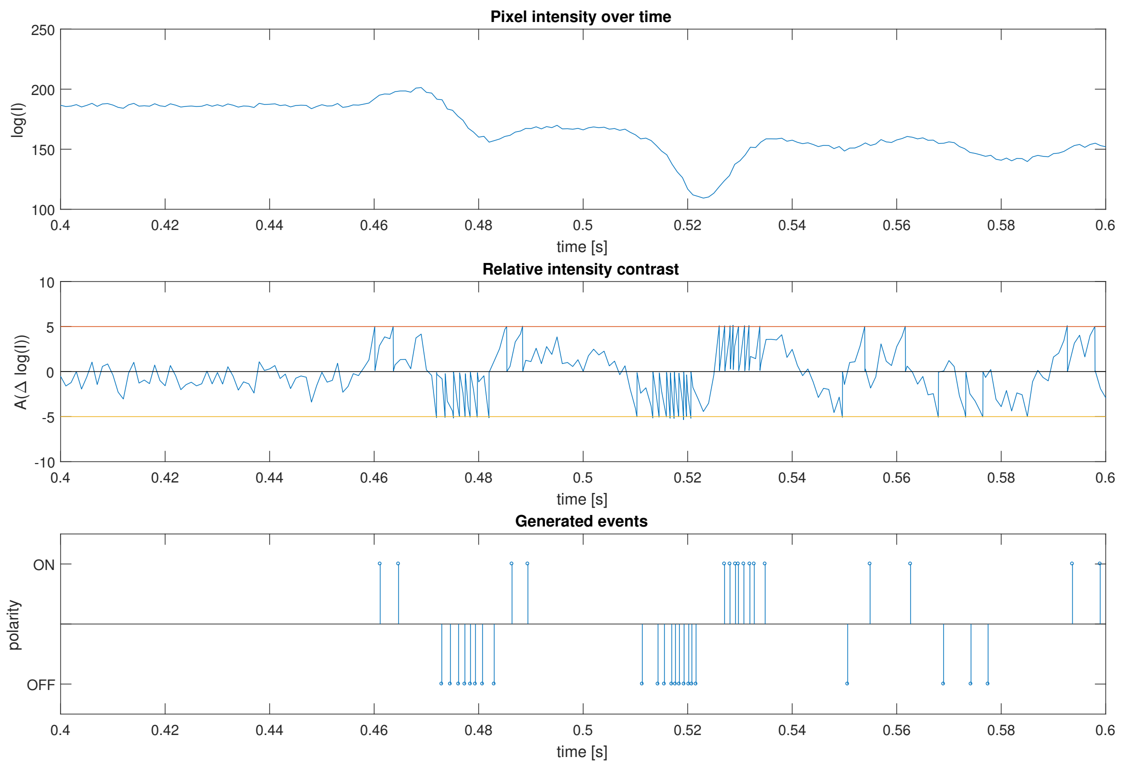


Figure B.1: Demonstration of the event generation for a single pixel.

the difference in relative intensity contrast. Events are generated when the contrast exceeds a predetermined sensitivity threshold, depicted in red and yellow for ON and OFF thresholds, respectively. The bottom graph of Figure B.1 shows the generated ON and OFF events. As we can see, the density of the events is higher in places where the contrast curve slope is steeper, whereas with slow changes, the events are sparser. In addition, thanks to the interpolation of the contrast curve, it is also possible to more realistically assign the timestamps beyond the resolution given by the frame rate of the rendered image sequence.

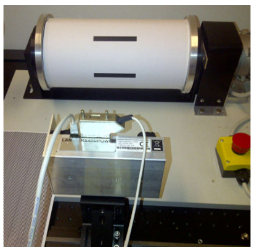
APPENDIX C

Sensor Bias Settings Tests

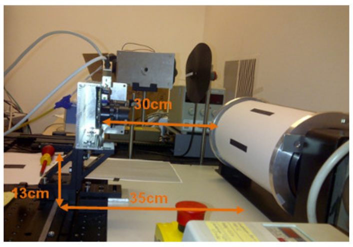
It has been observed that the amount of events generated by the left and right dynamic vision sensor can differ considerably. One reason for such dissimilarity between the outputs of the sensors could be the sensors' electronics. We investigate if by more precise bias parameter settings (individually for each sensor) we can reduce the dissimilarity of the left and right events distribution. A summary of our findings is given in Section 4.2.1. In the following we provide a detailed description of the experiments and a complete list of results.

Experiment Setup

We recorded data from the left and right sensor at exactly the same conditions, such as position, angle of view, distance, bias settings, lighting, and speed of the moving object. For the experiments, we have used the UCOS [125] stereo sensor. As a test pattern, we used two horizontal black bars on white background, placed on a drum rotating at a speed of 2.25 m/s, as shown in Figure C.1.



(a)



(b)

Figure C.1: The setup for the experiment. (a) The rotating drum with the test pattern; the sensor is placed on the sliding panel to adjust exactly the same position for the left and right sensor. (b) Side-view with marked distances.

	description	default (V)
Q	contrast baseline	0.38
Q_{ON}	contrast sensitivity (relative to Q)	0.55
Q_{OFF}		0.25
B_{Refr}	pixel's refractory period (influences data rate)	2.9
B_{Fo}	bandwidth control for noise reduction	2.13
B_{Pr}		2.78

Table C.1: Sensor parameters (bias settings) considered in experiments.

An overview of *DVS* parameters that have been used in our experiments is listed in Table C.1. We have started with recording several tests with default bias settings. We have also performed the *noise test*, where an idle scene was recorded to establish how much noise is generated by the sensor. Next, we have designed several tests to analyse sensor's behaviour under varying bias settings, each focusing on different parameter. Four test series were recorded for a duration of approx. 15s, specified as follows:

test1 changing contrast baseline $Q = 0.3, 0.35, 0.4, 0.45, 0.5$, while keeping the contrast sensitivity constant, i.e. $|Q_{\text{ON}} - Q| = |Q_{\text{OFF}} - Q| = 0.5$.

test2 changing contrast sensitivity $|Q_{\text{ON}} - Q| = |Q_{\text{OFF}} - Q| = \{2.5, 2, 1.5, 1, 0.5\}$, while keeping the constant baseline $Q = 0.4$.

test3 variation of the refractory period, i.e. $B_{\text{Refr}} = \{2.7, 2.75, 2.8, 2.85, 2.9\}$.

test4 variation of bandwidth parameters, i.e. B_{Fo} and B_{Pr} ranging from 2.1 to $2.9V$.

Experiment Results

We measure the total number of events, given in kilo events (*kEv*), the events ratio, i.e. the percentage of the left (or right) events to the total amount of events. The difference between the left and right response is measured for all events, and separately for ON and OFF events. The difference is expressed in the amount of events (*kEv*) and as percentage (ratio).

In Table C.2, the results of four bias tests are listed. We can observe that poorly chosen bias settings can lead to almost 25% of difference between the left and right events. The parameters that showed the most influence on the symmetry between left and right are Q, Q_{ON}, and Q_{OFF}. Furthermore, Table C.3 lists the results of three *noise tests*. The results of the bias tests are analysed in the context of ambiguities in event-based stereo matching described in Section 4.2.1.

	# events	*# events ratio*		Difference between left and right sensor					
		ON	OFF	all events		ON		OFF	
	(kEv)	(%)	(%)	(kEv)	(%)	(kEv)	(%)	(kEv)	(%)
test1	3137.3	27.56	72.44	382.1	12.18	171.2	19.80	553.3	24.34
	3555.5	42.46	57.54	89.3	2.51	67.2	4.45	22.1	1.08
	3806.5	49.27	50.73	75.3	1.98	31.2	1.66	44.2	2.29
	3616.7	50.10	49.90	43.2	1.20	8.0	0.44	35.3	1.95
	3794.1	49.08	50.92	126.4	3.33	73.1	3.92	53.3	2.76
test2	1507.4	43.49	56.51	362.8	24.07	155.3	23.69	207.5	24.36
	2951.7	47.52	52.48	248.0	8.40	84.7	6.04	163.3	10.54
	2263.2	49.34	50.66	35.7	1.58	17.1	1.53	18.6	1.62
	4789.5	38.41	61.59	262.9	5.49	40.9	2.23	221.9	7.52
	5890.6	16.58	83.42	7.2	0.12	3.8	0.39	10.9	0.22
test3	4160.2	28.02	71.98	25.7	0.62	15.3	1.31	10.4	0.35
	3807.7	32.82	67.18	135.7	3.56	100.9	8.07	34.9	1.36
	3610.4	41.91	58.09	75.8	2.10	144.8	9.57	69.0	3.29
	3607.9	50.58	49.42	230.6	6.39	82.3	4.51	312.9	17.55
	3877.6	50.28	49.72	198.1	5.11	14.9	0.76	213.0	11.05
test4	3301.2	59.61	40.39	421.2	12.76	176.9	8.99	244.3	18.32
	3270.7	59.48	40.52	489.6	14.97	225.7	11.60	263.9	19.92
	3066.6	59.60	40.40	492.3	16.05	241.9	13.23	250.4	20.21
	3030.4	59.99	40.01	486.8	16.06	254.2	13.98	232.6	19.19
	2780.4	59.85	40.15	443.3	15.94	244.3	14.68	199.1	17.83
	2645.4	59.30	40.70	406.3	15.36	241.9	15.42	164.5	15.28
	2633.6	59.42	40.58	388.7	14.76	244.2	15.61	144.4	13.51
	2542.0	58.81	41.19	358.0	14.08	237.6	15.89	120.4	11.50
	2420.8	58.60	41.40	330.8	13.66	228.9	16.14	101.9	10.16
	2498.6	58.96	41.04	339.4	13.58	237.4	16.12	102.0	9.95
	2511.0	59.31	40.69	335.6	13.36	238.2	16.00	97.3	9.53
	2432.4	59.70	40.30	303.4	12.47	239.1	16.47	64.3	6.56
	2618.7	56.40	43.60	229.1	8.75	221.3	14.98	7.8	0.68
	2549.4	56.37	43.63	241.9	9.49	227.8	15.85	14.0	1.26
	2546.8	56.37	43.63	225.6	8.86	220.9	15.39	4.7	0.42
	2524.3	56.14	43.86	237.1	9.39	224.5	15.84	12.6	1.14

Table C.2: Results of a series of tests, given in terms of left and right pixel response differences, for all events and with respect to ON and OFF events.

	# events	*# events ratio*		Difference between left and right sensor					
		ON	OFF	all events		ON		OFF	
	(kEv)	(%)	(%)	(kEv)	(%)	(kEv)	(%)	(kEv)	(%)
noise-test-1	410.423	14.15	85.85	25.31	0.62	14.719	25.35	12.188	3.46
noise-test-2	453.882	3.58	96.42	182.52	4.02	6.279	38.66	24.531	5.61
noise-test-3	434.999	3.47	96.53	175.27	4.03	5.823	38.54	23.35	5.56

Table C.3: Results of the *noise-test*, given in terms of left and right pixel response differences, for all events and with respect to ON and OFF events.

List of Figures

List of Tables

Acronyms

2SF	Two-Stage Filter
AD	Absolute Difference
AE	Address Event
AER	Address Event Representation
ATIS	Asynchronous Time-based Image Sensor
aVLSI	analog Very Large Scale Integration
BP	Belief Propagation
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DLS	Dual Line Sensor
DSI	Disparity Space Image
DSP	Digital Signal Processor
DVS	Dynamic Vision Sensor
EM	Expectation Maximization
FPGA	Field Programmable Gate Array
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
LED	Light-Emitting Diode
LGN	Lateral Geniculate Nuclei
MAP	Maximum A Posteriori
MRF	Markov Random Field
NSAD	Normalized Sum of Absolute Differences
ROLLS	Reconfigurable On-Line Learning Spiking neuromorphic processor
SAD	Sum of Absolute Differences
SD	Squared Difference
SLAM	Simultaneous Localisation and Mapping
SpiNNaker	Spiking Neural Network Architecture
SSD	Sum of Squared Differences
UAV	Unmanned Aerial Vehicle
UCOS	Universal COunting Sensor
VLSI	Very Large Scale Integration
WTA	Winner-Takes-All

Bibliography

[1] A. Andreopoulos, H. J. Kashyap, T. K. Nayak, A. Amir, and M. D. Flickner. A low power, high throughput, fully event-based stereo system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7532–7542, 2018.

[2] M. Azadmehr, J. P. Abrahamsen, and P. Hafliger. A foveated AER imager chip [address event representation]. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2751–2754, 2005.

[3] M. Barbaro, P. Y. Burgi, A. Mortara, P. Nussbaum, and F. Heitger. A 100$\times$100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding. *IEEE Journal of Solid-State Circuits*, 37(2):160–172, 2002.

[4] P. A. Bardow, A. J. Davison, and S. Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] S. T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, 3(1):17–32, 1989.

[6] A. N. Belbachir, M. Litzenberger, S. Schraml, M. Hofstätter, D. Bauer, P. Schön, M. Humenberger, C. Sulzbachner, T. Lunden, and M. Merne. CARE: A dynamic stereo vision sensor system for fall detection. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 731–734, 2012.

[7] A. N. Belbachir, S. Schraml, M. Mayerhofer, and M. Hofstätter. A novel HDR depth camera for real-time 3D 360° panoramic vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 425–432, 2014.

[8] R. Benosman, S. H. Ieng, P. Rogister, and C. Posch. Asynchronous event-based hebbian epipolar geometry. *IEEE Transactions on Neural Networks*, 22(11):1723–1734, 2011.

[9] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[10] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck. DDD17: end-to-end DAVIS driving dataset. *arXiv preprints*, arXiv:1711.01458, 2017.

[11] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999.

[12] M. Bleyer. *Segmentation-based stereo and motion with occlusions*. PhD thesis, Vienna University of Technology, 2006.

[13] M. Bleyer and M. Gelautz. Simple but effective tree structures for dynamic programming - based stereo matching. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 415–422, 2008.

[14] K. Boahen. Neuromorphic microchips. *Scientific American*, 292(5):56–63, 2005.

[15] K. A. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5):416–434, 2000.

[16] A. F. Bobick and S. S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[17] D. Borer, T. Delbruck, and T. Rösgen. Three-dimensional particle tracking velocimetry using dynamic vision sensors. *Experiments in Fluids*, 58(12):165, 2017.

[18] J. Y. Bouguet. Camera calibration toolbox for MATLAB. Published in the Internet, 2008. Computer Vision Research Group/Department of Electrical Engineering/California Institute of Technology - `www.vision.caltech.edu/bouguetj/calib_doc/index.html` (Accessed 17 May 2017).

[19] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[20] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck. A 240×180 130dB $3\mu s$ latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[21] C. Brandli, T. Mantel, M. Hutter, M. Höpflinger, R. Berner, R. Siegwart, and T. Delbruck. Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor. *Frontiers in Neuroscience*, 7:275.1–9, 2014.

[22] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck. ELiSeD–an Event-Based Line Segment Detector. In *Proceedings of the International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–7, 2016.

[23] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. J. V. Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1515–1522, 2009.

[24] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):90–99, 1986.

[25] G. J. Brostow and R. Cipolla. Unsupervised Bayesian detection of independent motion in crowds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 594–601, 2006.

[26] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008, 2003.

[27] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S. H. Ieng, R. Benosman, and B. Linares-Barranco. Event-driven stereo visual tracking algorithm to solve object occlusion. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2017.

[28] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco. Event-driven sensing and processing for high-speed robotic vision. In *Proceedings of the Biomedical Circuits and Systems Conference (BioCAS)*, pages 516–519, 2014.

[29] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, B. Linares-Barranco, S. Ieng, and R. Benosman. Event-driven stereo vision with orientation filters. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 257–260, 2014.

[30] J. Carneiro, S.-h. Ieng, C. Posch, and R. Benosman. Event-based 3D reconstruction from neuromorphic retinas. *Neural Networks*, 45:27–38, 2013.

[31] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 215–230, 2012.

[32] S. D. Cochran and G. Medioni. 3-D surface description from binocular stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):981–994, 1992.

[33] G. Cohen, S. Afshar, A. van Schaik, A. Wabnitz, T. Bessell, M. Rutten, and B. Morreale. Event-based sensing for space situational awareness. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, 2017.

[34] R. T. Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 358–363, 1996.

[35] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[36] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck. A pencil balancing robot using a pair of AER dynamic vision sensors. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 781–784, 2009.

[37] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger. Interacting maps for fast visual interpretation. In *International Joint Conference on Neural Networks (IJCNN)*, pages 770–776, 2011.

[38] E. Culurciello and A. G. Andreou. A comparative study of access topologies for chip-level address-event communication channels. *IEEE Transactions on Neural Networks*, 14(5):1266–1277, 2003.

[39] T. Delbruck and P. Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 845–848, 2007.

[40] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[41] G. Dikov, M. Firouzi, F. Röhrbein, J. Conradt, and C. Richter. Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware. In *Proceedings of the International Conference on Biomimetic and Biohybrid Systems, Living Machines*, Lecture Notes in Computer Science, pages 119–137, 2017.

[42] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen. Toward real-time particle tracking using an event-based dynamic vision sensor. *Experiments in Fluids*, 51(5):1465, 2011.

[43] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 300–305, 1998.

[44] F. Eibensteiner, A. Gschwandtner, and M. Hofstätter. A high-performance system-on-a-chip architecture for silicon-retina-based stereo vision systems. In *Proceedings of the International Congress on Computer Application and Computational Science (CACS)*, pages 976–979, 2010.

[45] F. Eibensteiner, J. Kogler, and J. Scharinger. A high-performance hardware architecture for a frameless stereo vision algorithm implemented on a FPGA platform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 623–630, 2014.

[46] F. Eibensteiner, J. Kogler, M. Schörghuber, and J. Scharinger. Automated stereo calibration for event-based silicon retina imagers. In *Proceedings of the 6th International Conference from Scientific Computing to Computational Engineering (IC-SCCE)*, 2014.

[47] F. Eibensteiner, J. Kogler, C. Sulzbachner, and J. Scharinger. Stereo-vision algorithm based on bio-inspired silicon retinas for implementation in hardware. In *Proceedings of the International Conference on Computer Aided Systems Theory (EUROCAST)*, volume 6927 of *Lecture Notes in Computer Science*, pages 624–631, 2011.

[48] P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21–27, 1997.

[49] D. A. Forsyth, O. Arikan, L. Ikemoto, J. F. O'Brien, and D. Ramanan. Computational studies of human motion: part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3), 2005.

[50] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1):35–49, 1993.

[51] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The SpiNNaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[52] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza. Event-based, 6-DoF camera tracking for high-speed applications. *arXiv preprints*, arXiv:1607.03468, 2016.

[53] G. Gallego, H. Rebecq, and D. Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3867–3876, 2018.

[54] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[55] A. Glover and C. Bartolozzi. Event-driven ball detection and gaze fixation in clutter. In *Proceedings on the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2203–2208, 2016.

[56] M. J. Hannah. *Computer matching of areas in stereo images.* PhD thesis, Stanford University, 1974.

[57] P. Hess. Low-level stereo matching using event-based silicon retinas. Semester's thesis, Eidgenössischen Technischen Hochschule (ETH) Zürich, 2006.

[58] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[59] H. Hirschmuller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1582–1599, 2009.

[60] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann. Local stereo matching using geodesic support weights. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 2093–2096, 2009.

[61] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz, and C. Rother. Real-time local stereo matching using guided image filtering. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2011.

[62] W. Hu, X. Li, W. Luo, X. Zhang, S. J. Maybank, and Z. Zhang. Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2420–2440, 2012.

[63] Y. Hu, H. Liu, M. Pfeiffer, and T. Delbruck. DVS benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in Neuroscience*, 10:405.1–5, 2016.

[64] S. Kameda and T. Yagi. A silicon retina system that calculates direction of motion. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 792–795, 2003.

[65] T. Kanade. Development of a video-rate stereo machine. In *Proceedings of the 1994 ARPA Image Understanding Workshop (IUW'94)*, pages 549–558, 1994.

[66] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, 1994.

[67] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous mosaicing and tracking with an event camera. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12, 2014.

[68] H. Kim, S. Leutenegger, and A. J. Davison. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 349–364, 2016.

[69] J. Kogler. *Design and evaluation of stereo matching techniques for silicon retina cameras.* PhD thesis, Vienna University of Technology, 2016.

[70] J. Kogler, F. Eibensteiner, M. Humenberger, M. Gelautz, and J. Scharinger. Ground truth evaluation for event-based silicon retina stereo data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 649–656, 2013.

[71] J. Kogler, F. Eibensteiner, M. Humenberger, C. Sulzbachner, M. Gelautz, and J. Scharinger. Enhancement of sparse silicon retina-based stereo matching using belief propagation and two-stage postfiltering. *Journal of Electronic Imaging*, 23(4):043011.1–15, 2014.

[72] J. Kogler, M. Humenberger, and C. Sulzbachner. Event-based stereo matching approaches for frameless address event stereo data. In *Advances in Visual Computing*, Lecture Notes in Computer Science, pages 674–685, 2011.

[73] J. Kogler, C. Sulzbachner, F. Eibensteiner, and M. Humenberger. Address-event matching for a silicon retina based stereo vision system. In *Proceedings of the 4th International Conference from Scientific Computing to Computational Engineering (IC-SCCE)*, pages 17–24, 2010.

[74] H. Kolb. How the retina works. *American Scientist*, 91(1):28–35, 2003.

[75] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 189–196. Springer, 1994.

[76] J. Kramer. An integrated optical transient sensor. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 49(9):612–628, 2002.

[77] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1710–1720, 2015.

[78] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie. Silicon auditory processors as computer peripherals. *IEEE Transactions on Neural Networks*, 4(3):523–528, 1993.

[79] J. H. Lee, T. Delbruck, M. Pfeiffer, P. K. J. Park, C. Shin, H. Ryu, and B. C. Kang. Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE Transactions on Neural Networks and Learning Systems*, 25(12):2250–2263, 2014.

[80] C. Li, C. Brandli, R. Berner, H. Liu, M. Yang, S. Liu, and T. Delbruck. Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 718–721, 2015.

[81] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. van den Hengel. A survey of appearance models in visual object tracking. *ACM Transactions on Intelligent Systems and Technology*, 4(4):1–48, 2013.

[82] P. Lichtsteiner. *An AER temporal contrast vision sensor*. PhD thesis, Eidgenössischen Technischen Hochschule (ETH) Zürich, 2006.

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

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

[85] H. Liu, D. P. Moeys, G. Das, D. Neil, S.-C. Liu, and T. Delbrück. Combined frame- and event-based detection and tracking. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2511–2514, 2016.

[86] W.-L. Lu, J.-A. Ting, J. J. Little, and K. P. Murphy. Learning to track and identify players from broadcast sports videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1704–1716, 2013.

[87] Q. Luo and J. G. Harris. A time-based CMOS image sensor. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 840–843, 2004.

[88] W. Luo, T.-K. Kim, B. Stenger, X. Zhao, and R. Cipolla. Bi-label propagation for generic multiple object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1290–1297, 2014.

[89] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim. Multiple object tracking: A literature review. *arXiv preprints*, arXiv:1409.7618, 2014.

[90] M. Mahowald. *VLSI analogs of neuronal visual processing: a synthesis of form and function*. PhD thesis, California Institute of Technology, 1992.

[91] M. A. Mahowald and T. Delbruck. Cooperative stereo matching using static and dynamic image features. In C. Mead and M. Ismail, editors, *Analog VLSI Implementation of Neural Systems*, number 80 in The Kluwer International Series in Engineering and Computer Science, pages 213–238. Springer US, 1989.

[92] U. Mallik, M. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings. Temporal change threshold detection imager. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, volume 1, pages 362–603, 2005.

[93] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976.

[94] D. Marr and T. Poggio. Cooperative computation of stereo disparity. In L. Vaina, editor, *From the Retina to the Neocortex*, pages 239–243. Birkhäuser Boston, 1976.

[95] J. L. Marroquin. Design of cooperative networks. Technical report, MIT Artificial Intelligence Laboratory, 1983.

[96] N. Matsuda, O. Cossairt, and M. Gupta. MC3D: Motion Contrast 3D scanning. In *Proceedings of the IEEE International Conference on Computational Photography (ICCP)*, pages 1–10, 2015.

[97] L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

[98] J. E. W. Mayhew and J. P. Frisby. The Computation of Binocular Edges. *Perception*, 9(1):69–86, 1980.

[99] C. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, Jan. 1989.

[100] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha. Artificial Brains: A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345:668–673, 2014.

[101] A. Mitrokhin, C. Fermuller, C. Parameshwara, and Y. Aloimonos. Event-based moving object detection and tracking. *arXiv preprints*, arXiv:1803.04523, 2018.

[102] D. Mitzel and B. Leibe. Real-time multi-person tracking with detector assisted structure propagation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 974–981, 2011.

[103] E. Mueggler, G. Gallego, and D. Scaramuzza. Continuous-time trajectory estimation for event-based vision sensors. In *Proceedings of Robotics: Science and Systems*, pages 1–9, 2015.

[104] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: event-based data for pose estimation, visual odometry, and SLAM. *International Journal of Robotics Research*, 36(2):142–149, 2017.

[105] Z. Ni, C. Pacoret, R. Benosman, S. Ieng, and S. Régnier. Asynchronous event-based high speed vision for microparticle tracking. *Journal of Microscopy*, 245(3):236–244, 2012.

[106] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[107] M. Okutomi and T. Kanade. A locally adaptive window for signal matching. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 190–199, 1990.

[108] M. Osswald, S.-H. Ieng, R. Benosman, and G. Indiveri. A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems. *Scientific Reports*, 7:40703.1–11, 2017.

[109] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1913–1921, 2015.

[110] E. Piatkowska, A. Belbachir, and M. Gelautz. Asynchronous stereo vision for event-driven dynamic stereo sensor using an adaptive cooperative approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 45–50, 2013.

[111] E. Piatkowska, A. N. Belbachir, and M. Gelautz. Cooperative and asynchronous stereo vision for dynamic vision sensors. *Measurement Science and Technology*, 25(5):055108.1–8, 2014.

[112] E. Piątkowska, A. N. Belbachir, S. Schraml, and M. Gelautz. Spatiotemporal multiple persons tracking using dynamic vision sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 35–40, 2012.

[113] E. Piatkowska, J. Kogler, N. Belbachir, and M. Gelautz. Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 370–377, 2017.

[114] C. Posch, M. Hofstatter, D. Matolin, G. Vanstraelen, P. Schon, N. Donath, and M. Litzenberger. A Dual-Line Optical Transient Sensor with On-Chip Precision Time-Stamp Generation. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, pages 500–618, San Francisco, CA, USA, 2007.

[115] C. Posch, D. Matolin, and R. Wohlgenannt. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, 2011.

[116] X. Qi, X. Guo, and J. G. Harris. A time-to-first spike CMOS imager. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 824–827, 2004.

[117] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri. A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9:141.1–18, 2015.

[118] H. Rebecq, G. Gallego, and D. Scaramuzza. EMVS: Event-based Multi-View Stereo. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 63.1–11, 2016.

[119] H. Reingruber. An asynchronous data interface for event-based stereo matching. Master's thesis, Vienna University of Technology, 2011.

[120] D. Reynolds. Gaussian Mixture Models. In S. Z. Li and A. Jain, editors, *Encyclopedia of Biometrics*, pages 659–663. Springer US, Boston, MA, 2009.

[121] P. Rogister, R. Benosman, S. H. Ieng, P. Lichtsteiner, and T. Delbruck. Asynchronous event-based binocular stereo matching. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):347–353, 2012.

[122] B. Rueckauer and T. Delbruck. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in Neuroscience*, 10:176.1–17, 2016.

[123] P. F. Ruedi, P. Heim, F. Kaess, E. Grenet, F. Heitger, P. Y. Burgi, S. Gyger, and P. Nussbaum. A 128×128 pixel 120 dB dynamic range vision sensor chip for image contrast and orientation extraction. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, volume 1, pages 226–490, 2003.

[124] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):312–322, 1980.

[125] smart-eye UCOS: universal counting sensor. AIT Austrian Institute of Technology. `https://www.ait.ac.at/fileadmin/mc/digital_safety_security/downloads/People_Counting_Sensor_UCOS.pdf` (Accessed 19 August 2018).

[126] D. Scharstein. Matching images by comparing their gradient fields. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 1, pages 572–575, 1994.

[127] D. Scharstein. *View Synthesis Using Stereo Vision*. Springer-Verlag, Berlin, Heidelberg, 1999.

[128] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174, 1998.

[129] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[130] S. Schraml and A. N. Belbachir. A spatio-temporal clustering method using real-time motion analysis on event-based 3D vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 57–63, 2010.

[131] S. Schraml, A. N. Belbachir, and H. Bischof. Event-driven stereo matching for real-time 3D panoramic vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 466–474, 2015.

[132] S. Schraml, A. N. Belbachir, and H. Bischof. An event-driven stereo system for real-time 3-D 360° panoramic vision. *IEEE Transactions on Industrial Electronics*, 63(1):418–428, 2016.

[133] S. Schraml, A. N. Belbachir, N. Milosevic, and P. Schon. Dynamic stereo vision system for real-time tracking. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1409–1412, 2010.

[134] S. Schraml, P. Schön, and N. Milosevic. Smartcam for real-time stereo vision - address-event based embedded system. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 2, pages 466–471, 2007.

[135] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[136] S. N. Sinha, D. Scharstein, and R. Szeliski. Efficient high-resolution stereo matching using local plane sweeps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1582–1589, 2014.

[137] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1881–1888, 2009.

[138] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. HATS: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1731–1740, 2018.

[139] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[140] K. Smith, D. Gatica-Perez, J. Odobez, and S. Ba. Evaluating Multi-Object Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 36–36. IEEE, 2005.

[141] C. Strecha and L. V. Gool. Motion — Stereo Integration for Depth Estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pages 170–185, 2002.

[142] D. Sugimura, K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto. Using individuality to track individuals: clustering individual trajectories in crowds using local appearance and frequency trait. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1467–1474, 2009.

[143] C. Sulzbachner, C. Zinner, and J. Kogler. An optimized Silicon Retina stereo matching algorithm using time-space correlation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1–7, 2011.

[144] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[145] R. Szeliski. *Computer Vision - Algorithms and Applications.* Springer London, 2011.

[146] R. Szeliski and D. Scharstein. Sampling the disparity space image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):419–425, 2004.

[147] H. Tao, H. S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 532–539, 2001.

[148] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza. Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS). In *Proceedings of the International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–7. IEEE, 2016.

[149] S. J. Thorpe. Image processing by the human visual system. In I. H. G. Garcia, editor, *Advances in Computer Graphics VI, Images: Synthesis, Analysis, and Interaction*, pages 309–341. Springer-Verlag, 1991.

[150] E. Tola, V. Lepetit, and P. Fua. DAISY: an efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010.

[151] O. Tuzel, F. Porikli, and P. Meer. Region covariance: a fast descriptor for detection and classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 3952 of *Lecture Notes in Computer Science*, pages 589–600, 2006.

[152] C. J. Veenman, M. J. T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, 2001.

[153] O. Veksler. Stereo correspondence by dynamic programming on a tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 384–390, 2005.

[154] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, 34(1):3–19, 2013.

[155] Z.-F. Wang and Z.-G. Zheng. A region based stereo matching algorithm using cooperative optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[156] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt. Event-based 3D SLAM with a depth-augmented dynamic vision sensor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 359–364, 2014.

[157] O. Williams, M. Isard, and J. MacCormick. Estimating disparity and occlusions in stereo video sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 250–257, 2005.

[158] Z. Xie, S. Chen, and G. Orchard. Event-based stereo depth estimation using belief propagation. *Frontiers in Neuroscience*, 11:535, 2017.

[159] J. Xing, H. Ai, and S. Lao. Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1200–1207, 2009.

[160] N. Xu and N. Ahuja. Object contour tracking using graph cuts based active contours. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 3, pages 277–280, 2002.

[161] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1345–1352, 2011.

[162] B. Yang, C. Huang, and R. Nevatia. Learning affinities and dependencies for multi-target tracking using a CRF model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1233–1240, 2011.

[163] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, 2006.

[164] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[165] K.-J. Yoon and I.-S. Kweon. Locally adaptive support-weight approach for visual correspondence search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 924–931, 2005.

[166] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 151–158, 1994.

[167] K. A. Zaghloul and K. Boahen. Optic nerve signals in a neuromorphic chip I : Outer and inner retina models. *IEEE Transactions on Biomedical Engineering*, 51(4):657–666, 2004.

[168] K. A. Zaghloul and K. Boahen. Optic nerve signals in a neuromorphic chip II: Testing and results. *IEEE Transactions on Biomedical Engineering*, 51(4):667–675, 2004.

[169] L. Zhang, B. Curless, and S. M. Seitz. Spacetime stereo: shape recovery for dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 367–74, 2003.

[170] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[171] Z. Zhang and Y. Shan. A progressive scheme for stereo matching. In *3D Structure from Images, Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, Lecture Notes in Computer Science, pages 68–85, 2000.

[172] S. K. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, 13(11):1491–1506, 2004.

[173] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based feature tracking with probabilistic data association. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4465–4470, 2017.

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

[175] A. Z. Zhu, Y. Chen, and K. Daniilidis. Realtime time synchronized event-based stereo. *arXiv preprints*, arXiv:1803.09025, 2018.

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

[177] C. L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000.

[178] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 600–608, 2004.

[179] D. Zou, P. Guo, Q. Wang, X. Wang, G. Shao, F. Shi, J. Li, and P. K. J. Park. Context-aware event-driven stereo matching. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 1076–1080, 2016.

[180] D. Zou, F. Shi, W. Liu, J. Li, Q. Wang, P.-K. J. Park, and E. R. Hyunsurk. Robust dense depth maps generations from sparse DVS stereos. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–11, 2017.

Curriculum Vitae

Ewa Piątkowska

Address:	Strudlhofgasse 14/6, 1090 Vienna, Austria
Phone:	(+43) 664 8251258
Email:	`epiatkow@gmail.com`

Education

2011 – now	Vienna University of Technology, Faculty of Informatics, Vienna, Austria. Doctoral programme in Technical Sciences Thesis: *Asynchronous Stereo Vision. Event-based Stereo Matching and Tracking for Dynamic Vision Sensors.*
2009 – 2011	Jagiellonian University, Faculty of Physics, Astronomy and Applied Computer Science, Kraków, Poland Master of Science (MSc) in Applied Computer Science Thesis: *Automated Facial Expressions Analysis*
2008 – 2010	DePaul University, College of Computing and Digital Media, Chicago, USA Master of Science (MSc) in Computer Science Thesis: *Facial Expressions Recognition System*
2007 – 2008	Letterkenny Institute of Technology, Letterkenny, Ireland Bachelor of (Honours) Science with First Class Honours in Applied Computing Thesis: *Design and Implementation of Web-based Message Board Application*
2005 – 2009	Nowy Sącz School of Business – National Louis University, Faculty of Informatics, Nowy Sącz, Poland Bachelor of Arts in Computer Science Thesis: *Image Recognition with use of Artificial Neural Networks*

Work Experience

2015 – now	AIT Austrian Institute of Technology, Vienna, Austria Scientist
2011 – 2014	AIT Austrian Institute of Technology, Vienna, Austria Junior Scientist
2008 – 2009	European Institute of Technology in Education, Nowy Sącz, Poland Software Developer

Selected Publications

Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz: *Spatiotemporal Multiple Persons Tracking Using Dynamic Vision Sensor*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 35–40, 2012.

Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz: *Asynchronous Stereo Vision for Event-Driven Dynamic Stereo Sensor Using an Adaptive Cooperative Approach*, IEEE International Conference on Computer Vision (ICCV) Workshops, 45–50, 2013.

Ewa Piatkowska, Ahmed Nabil Belbachir, and Margrit Gelautz: *Cooperative and Asynchronous Stereo Vision for Dynamic Vision Sensors*, Measurement Science and Technology, Volume 25, Number 5, pp. 1-8, 2014.

Ewa Piatkowska, Jürgen Kogler, Ahmed Nabil Belbachir, and Margrit Gelautz: *Improved Cooperative Stereo Matching for Dynamic Vision Sensors with Ground Truth Evaluation*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 370–377, 2017.

Research interests

Computer Vision	stereo vision, 3D reconstruction, motion analysis and estimation, face detection, facial expressions recognition
Machine Learning	conventional and modern (deep) learning concepts and applications
Cyber-Security	intrusion detection and alert correlation in cyber-physical systems