Dynamic Vision Sensors in Long-Term Monitoring Systems

Object Detection and Segmentation in Real-World Outdoor Scenarios

DISSERTATION

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Tobias Bolten, M.Sc.

geb. am 29.03.1988 in Mönchengladbach

Gutachterinnen/Gutachter

Prof. Dr. Klaus-Dietz Tönnies
Prof. Dr. Regina Pohle-Fröhlich
Prof. Dr. Alexander Binder
Assoc. Prof. Dr. Camille Simon Chane

Magdeburg, den 19.12.2024

Zusammenfassung

Die Weltbevölkerung wächst und mit ihr, durch eine einhergehende Urbanisierung, der Anteil der Menschen, die in städtischen Gebieten leben. Bei der Planung des öffentlichen Raumes wird die tatsächliche Nutzung jedoch bisher nur rudimentär berücksichtigt. Um hier Abhilfe zu schaffen, ist ein Wandel zu einem nutzerorientierten Gestaltungs- und Planungsansatz unabdingbar. Die Kenntnis über die Anzahl der Nutzer und deren Nutzung des Raumes ist für einen solchen Übergang entscheidend. Derzeit werden diese Daten hauptsächlich durch manuelle Beobachtungen erhoben. Die Durchführung derartiger Beobachtungen ist jedoch mit einem hohen Zeit- und Arbeitsaufwand verbunden und erfolgt daher in der Regel nur für kurze Zeiträume.

Sensorbasierte Monitoringsysteme können diese Situation verbessern, indem sie Langzeitbeobachtungen und automatisierte Analysen ermöglichen. Das Ziel dieser Arbeit ist es daher, ein technisches Monitoringsystem als Proof-of-Concept zu entwickeln, das eine evidenzbasierte Datenbasis über die Flächennutzung für stadtplanerische Prozesse liefern kann. Der Einsatz solcher Systeme ist jedoch, insbesondere in Deutschland, durch gesetzliche Vorgaben stark eingeschränkt. Diese erschweren den Einsatz herkömmlicher bildgebender Sensoren in Form von Videoüberwachungsanlagen im öffentlichen Raum.

In Anbetracht dieser Einschränkungen wird ein System auf der Grundlage von Dynamic Vision Sensors (DVS) entwickelt. Diese optischen Sensoren unterscheiden sich in ihrem Funktionsparadigma grundlegend von herkömmlichen bildgebenden Sensoren. Die Pixel dieser Sensoren werden durch lokale Helligkeitsänderungen getriggert, anstatt eine feste Verschlusszeit oder Bildrate zu verwenden. Änderungen der logarithmischen Pixelhelligkeit werden unabhängig voneinander erkannt und asynchron übertragen. Das Resultat ist ein räumlich dünn besetzter, mehrdimensionaler Ausgabestrom mit hoher zeitlicher Auflösung und variabler Datenrate anstelle einer Sequenz konventioneller 2D-Bilder. Dies ermöglicht eine Verarbeitung ohne direkte Berücksichtigung von Grauwerten oder Farbwerten.

Die Sensorausgabe erfolgt in Form von separaten *Events*, die die detektierten Helligkeitsänderungen beschreiben. Dies stellt im Sinne der klassischen Computer Vision eine unkonventionelle Form dar. Eine Vielzahl von Repräsentationsformen, einschließlich 2D-Frames, Voxel-Gitter und 3D Space-Time Event Clouds, eignen sich für deren Verarbeitung. Derzeit existiert keine etablierte Standardrepräsentation, die für verschiedene Anwendungen verwendet wird. Daher werden diese verschiedenen Repräsentationen in Verbindung mit unterschiedlichen Deep Learning basierten Verarbeitungsansätzen eingehend miteinander verglichen. Der Schwerpunkt liegt dabei auf der Repräsentation in Form von Event Clouds, da diese Eventrepräsentation nahezu nativ aus der Sensorausgabe erstellt werden kann, während die volle Auflösung und "Sparsity" des Signals erhalten bleibt.

Es wird ein DVS-basiertes Monitoringsystem entwickelt. Dabei werden verschiedene Herausforderungen bei der Verarbeitung berücksichtigt, insbesondere die im Signal enthaltenen Umwelteinflüsse, die aus dem realen Messaufbau im Freien resultieren. Im Rahmen einer Langzeitbeobachtung eines öffentlichen Freiraums erfolgt ein Vergleich der Repräsentationen für die Aufgabenstellung der semantischen Segmentierung, sowie der Instanzsegmentierung. Das entwickelte System umfasst dabei die gesamte Verarbeitungspipeline von der Datenerfassung und Filterung über die Objektsegmentierung bis hin zur Visualisierung der Ergebnisse.

Anhand von Aufzeichnungen wird gezeigt, dass eine entwickelte Auswahl von Zeitfenstern mit Events, die durch eine vordefinierte Menge von Objektklassen ausgelöst werden, eine hohe Qualität bei gleichzeitig geringem Bedarf an Rechenressourcen aufweist. Darüber hinaus wird gezeigt, dass das System in der Lage ist, die Anzahl der Nutzer mit hoher Genauigkeit zu bestimmen, indem die ermittelten Ergebnisse mit Referenzzählungen verglichen werden, die von menschlichen Annotatoren erstellt wurden. Eine Heatmap-Visualisierung wird entwickelt, um die räumliche Verteilung der Nutzung zusammenzufassen. Diese Visualisierung wird getestet und erweist sich als intuitiv verständlich für die Stakeholder, an die sich das Monitoring richtet.

Abstract

The world's population is growing, and with it the proportion of people living in urban areas. Currently, the actual utilization of urban public spaces is only rudimentarily considered in their planning. To address this, a shift towards a user-oriented design approach is essential. Knowledge about the number of users and their spatial distribution within the space is crucial to enable such a transition. However, manual observation is currently the primary method used to collect such data. These observations are generally time-consuming and labor-intensive, and are therefore typically conducted only for short periods of time.

Sensor-based monitoring systems can improve this situation by enabling long-term observations and automated analysis. Thus, the objective of this work is to develop a technical monitoring system as a proof-of-concept that can create an evidence-based database on the long-term use of public space as a basis for decision-making in urban planning processes. However, the use of such systems is subject to legal requirements, which are particularly strict in Germany. These regulations make the use of traditional image sensors in the form of CCTV systems in public spaces very difficult.

Addressing these limitations, a system based on Dynamic Vision Sensors (DVS), also known as event cameras, is developed. These optical sensors are fundamentally different from conventional frame-based sensors due to their underlying operating paradigm. The sensor's pixels are triggered by changes in brightness, rather than using a fixed exposure time or frame rate. Changes in logarithmic pixel brightness are detected independently and transmitted asynchronously. The result is a spatially sparse, multidimensional output stream of high temporal resolution at a variable data rate, instead of a sequence of traditional 2D frames. This allows processing without direct consideration of any gray or color values in software.

The output, in the form of separate *events* describing the detected changes in brightness, is unconventional in terms of classical computer vision, as it is spatially sparse, unordered, and asynchronous. For processing, these events can be represented in a variety of data structures, including 2D frames, voxel grids, and 3D space-time event clouds. Currently, there is no de facto standard representation that is commonly used for different processing tasks. Therefore, these different representations, coupled with different deep learning-based processing approaches, are extensively compared. The focus of this work is on event cloud representations, since this data structure is built almost natively from the sensor output, while preserving the full resolution and sparsity of the signal.

A DVS-based monitoring system is developed and evaluated. For this application, different processing challenges are considered, including environmental influences contained in the sensor signal resulting from the real-world outdoor measurement setup. In the context of a performed long-term monitoring of a public outdoor space, a comparison is conducted with respect to the application task of creating semantic and instance segmentations. The developed monitoring system covers the entire processing pipeline from data acquisition and filtering to object segmentation and visualization of the results.

Based on recordings acquired during the conducted long-term outdoor monitoring, a selection of temporal segments containing events triggered by a predefined set of object classes is shown to be of high quality, while requiring low computational resources on-site. Furthermore, the system's ability to estimate the user volume in the recorded data is shown to be highly accurate by comparison with reference counts generated by multiple human annotators. A heat map visualization is developed and evaluated to aggregate details about the spatial distribution of usage. This visualization is tested and found to be intuitively understandable by the stakeholders for whom the monitoring is intended.

Contents

Part I

Background and Context

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The global world population is continuously growing [United Nations, 2019]. In the course of this population growth, an urbanization of the chosen habitat can be observed. In 1950, less than one-third of the world's population lived in cities. By 2010, that proportion had already risen to more than half. Recent calculations by the United Nations predict that by 2050, out of a total of approximately 10 billion people, more than 7 billion will live in urban metropolises [United Nations, 2018].

As a result, public space in cities will be used more and more. Contrary to predictions made in the seventies of the last century, more people are rediscovering public space for themselves and are using it in different ways [Selle, 2010, pp. 51 ff.]. As a result, new challenges are emerging for the design of public spaces. These challenges require detailed knowledge about the needs and behaviors of different user groups in the public realm. Currently, these diverse needs are rarely considered in urban planning processes [Selle, 2010, p. 26]. The reasons for this can be found in different planning trends [Schmid, 2015] and especially in the problem of generating knowledge about the current use of an urban space and being able to use it in design processes.

In addition to formats of citizen participation such as surveys, sociological urban research uses various methods of observation to gain more precise insights [Eckardt, 2014, p. 177] [Gehl and Svarre, 2013, pp. 21 ff.]. These observations include questions about who uses public space, at what times, and in what ways. The problem here is that in most cases these observations have to be carried out manually and are therefore time and personnel intensive, resulting in rather expensive user studies. This makes it difficult to incorporate such findings into design processes. However, the detailed description and identification of urban space usage is the fundamental basis for a user-centered design process and has a significant impact on the final design result.

Therefore, the aim of this work is to address this problem with a technical solution in the form of a sensor-based observation system. The goal is to provide a system that is able to create an evidence-based database on the long-term use of public space as a basis for decision-making in the urban planning process. However, due to the application context of public space monitoring, special aspects and requirements arise for this observation system. In particular, it is necessary to comply with local data protection laws for data collection and interpretation. For this purpose, an approach and a monitoring system are presented, which addresses and mitigates these problems already by the choice of the applied sensor technology.

1.1.1 Challenges in Public Space Monitoring

Until now, the analysis of usage and interaction in public spaces is generally based on analog methods. However, classical CCTV video recordings of public spaces could form the data basis for such a monitoring system. They can be used to objectively determine, analyze and document the behavior of the users involved. Thanks to recent advances in computer vision, this could also be achieved in an automated way. However, the use of classical image sensors has limitations that must be taken into account in a public surveillance system.

The use of conventional video surveillance systems can be subject to numerous regulatory requirements and laws. These regulations vary greatly depending on the location of the system, as national legislation differs from country to country. In the context of German legislation, images of people are considered as *"personal data"* and are therefore subject to special and very strict requirements for protection. These protection requirements, especially in the context of recording in public spaces, can prevent or significantly complicate the use of traditional video surveillance in Germany from a legal and regulatory perspective. Furthermore, the use of machine pattern recognition in public spaces may also raise ethical concerns [Gehl and Svarre, 2013].

1.1.2 Monitoring Approach

Therefore, the aim of this dissertation is the development of a technical monitoring system on the basis of an alternative sensor technology for long-term monitoring in the context of urban planning. This system utilizes an optical recording paradigm that differs from conventional frame-generating sensors. In this way, privacy and personality rights are addressed at the hardware level, minimizing potential ethical and legal concerns.

The developed system is based on the Dynamic Vision Sensor (DVS), a sensor from the field of neuromorphic engineering. The pixels of a DVS operate independently and asynchronously, encoding only detected changes in brightness. The sensor output is not a traditional frame, but rather a spatially sparse data stream with a variable rate of triggered events at high temporal resolution. This change in the fundamental output paradigm has the consequence that computer vision methods developed and widely used over the last decades cannot be directly applied to the sensor signal of a DVS. This poses the challenge of developing an adapted processing with respect to the sensor signal.

The following Section 1.2 and Section 1.3 briefly introduce the research field of neuromorphic engineering and explain the operating principle of a Dynamic Vision Sensor. The Section 1.4 summarizes the challenges of DVS-related signal processing and presents common data representations that are used in the processing.

1.2 Neuromorphic Engineering

Neuromorphic engineering is an interdisciplinary field of research that brings together biology, physics, mathematics, computer science, and electrical engineering to design and implement artificial neural systems that mimic nature. *Very Large Scale Integrated Circuits* (VLSI) attempt to reproduce the underlying biological principles and functions in electrical circuits. A major pioneer in this field of research is Carver Mead, whose work in the late 1980s laid the groundwork and influenced further work [Mead, 1989].

Misha Mahowald, one of Mead's doctoral students, developed the first "silicon retina" in the course of her dissertation [Mahowald, 1992]. This retina retains the biological signal processing idea of a retina and the idiom of transmission of action potentials in the nervous system.

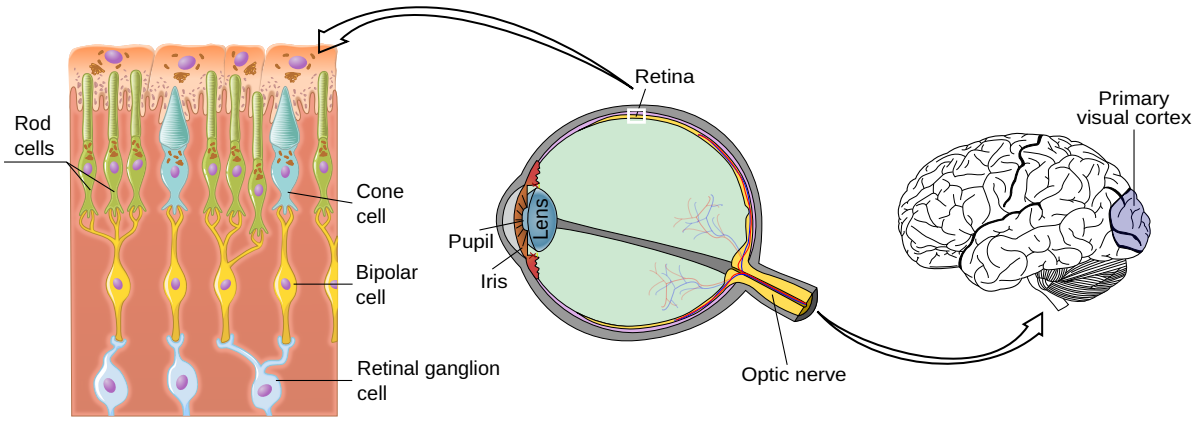


Figure 1.1: Visual pathway: Scheme of visual sensory input in humans[1].

Biological Vision

The retina in the human eye consists of several layers (see Figure 1.1) that can be grouped into four main categories[2]:

- **Photoreceptor cell**
  The lens of the eye focuses incident light on the retina and directs it to the light-sensitive photoreceptor cells (commonly known as rod and cone cells), whose action potential depends on the amount of incoming light.
- **Retina bipolar cell**
  The photoreceptors are connected by horizontal and bipolar cells. These cells collect and weigh the information from the photoreceptors and transmit it to the ganglion cells.
- **Retinal ganglion cell**
  The retinal ganglion cells generate the output of the retina. The firing rate of these cells correlates with changes in light intensity.
- **Optic nerve**
  The optic nerve transmits the information to the visual cortex, where visual perception and understanding begin.

1.3 Dynamic Vision Sensor

This biological structure is reflected in the design of each individual pixel of a Dynamic Vision Sensor.
The basic working principle is described in [Steffen et al., 2019, p. 10] as follows (see Figure 1.2):

> "[The] light information is obtained by a photodiode which thus generates the current intensity I. Photoreceptors (cones) convert I into a logarithmic voltage

[1]The left part of the figure, the "Rods and Cones" (`https://commons.wikimedia.org/wiki/File:1414_Rods_and_Cones_-_ru.svg`) visualization of the retinal cells, is adapted from a figure created by users CFCF and Kaidor found on Wikimedia Commons, licensed under CC BY-SA 4.0. The "human eye" (`https://commons.wikimedia.org/wiki/File:Schematic_diagram_of_the_human_eye_horizontal_pt.svg`) visualization is adapted from a figure created by user Rhcastilhos found on Wikimedia Commons, licensed under CC0. The visualization of the "human brain" (`https://commons.wikimedia.org/wiki/File:Gray728.svg`) is adapted from a figure created by user Mysid found on Wikimedia Commons, which is in the public domain.

[2]This description does not claim to be complete and is a simplified abstraction. The intention is to present the basic components considered in the design of the Dynamic Vision Sensor and its paradigm.

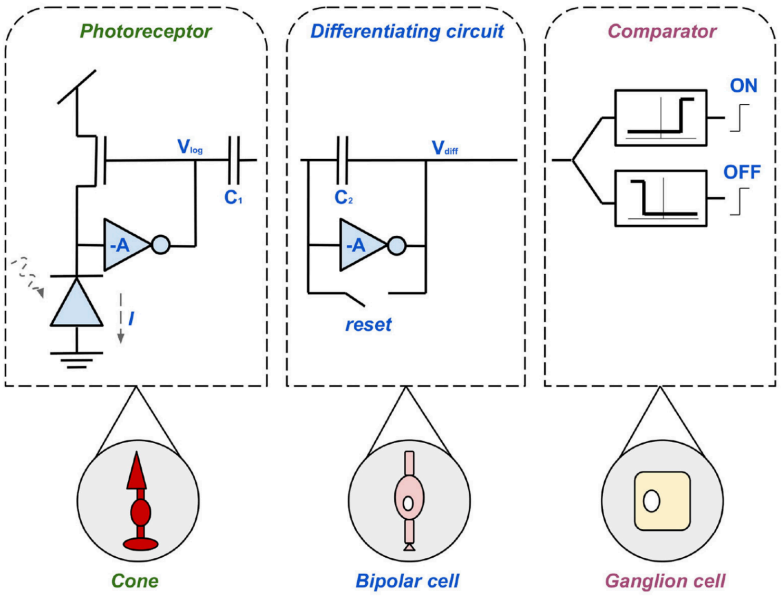


Figure 1.2: General scheme of a DVS pixel related to biology (from [Steffen et al., 2019]).

> V_p. This voltage is inversely amplified by the factor $A = C_1/C_2$. Also a positive or negative event V_{diff} is generated by the differential circuit (bipolar cell), depending on the polarity of the photocurrent. Subsequently, the pulses are collected, divided into ON- and OFF-events and forwarded by the comparator (ganglion cell)."

The pixels of a Dynamic Vision Sensor operate independently and asynchronously. Each pixel's output is based on the relative changes in light intensity it detects. An output is triggered when

$$\left|\log\left(I_{t_i}(x,y)\right) - \log\left(I_{t_{i-1}}(x,y)\right)\right| > \theta \tag{1.1}$$

a change in brightness (intensities I) is detected between two consecutive points in time (timestamps t_i, and t_{i-1}) at the pixel position (x, y) above a specified threshold value θ. This output is called an *"event"*. The term event-based vision is derived from this.

1.3.1 Address-Event-Representation

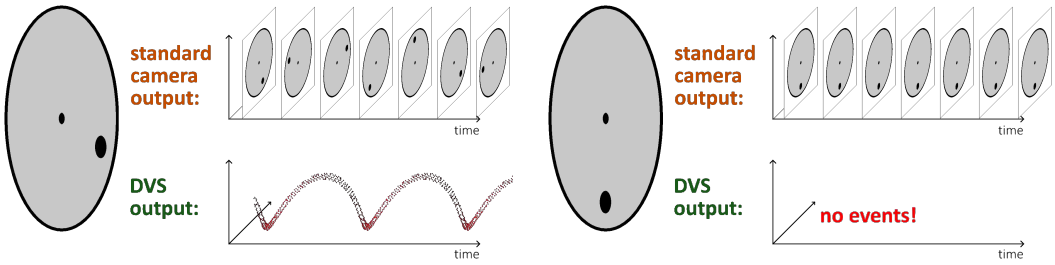
The human retina contains approximately 0.7 to 1.5 million retinal ganglion cells [Watson, 2014]. Currently, implementing in-chip wiring with a large number of separate output paths is not technically feasible. Therefore, an event-based multiplexed data protocol called Address Event Representation (AER) [Mahowald, 1992; Lichtsteiner et al., 2008] is used to bundle traffic on output circuits.

In contrast to classical image acquisition, and with further reference to the transmission of information in biology via the optic nerve, the DVS output does not result in the transmission of a full image, but in an output stream of individual, activated pixels. This data stream has a variable data rate that depends on the changes detected and can vary greatly over time.

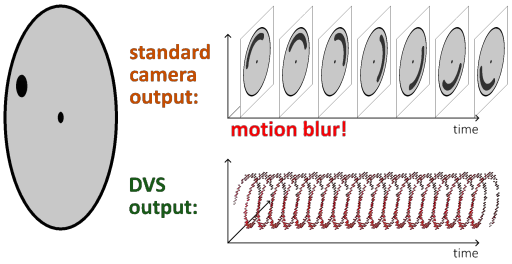
The output is asynchronously multiplexed using the Address Event Representation communication protocol. Each individual asynchronous output event $e_i = \{(x_i, y_i), t_i, p_i\}$, identified by the occurrence index i, is thus described by:

(a) Stimulus (b) Triggered DVS event stream

Figure 1.3: Visualization of the DVS output caused by a fast-moving stimulus. The DVS event polarity is color-coded with green for "on" and red for "off" (from [**Bolten** et al., 2023c]).



(a) "Moderate" motion (b) No motion



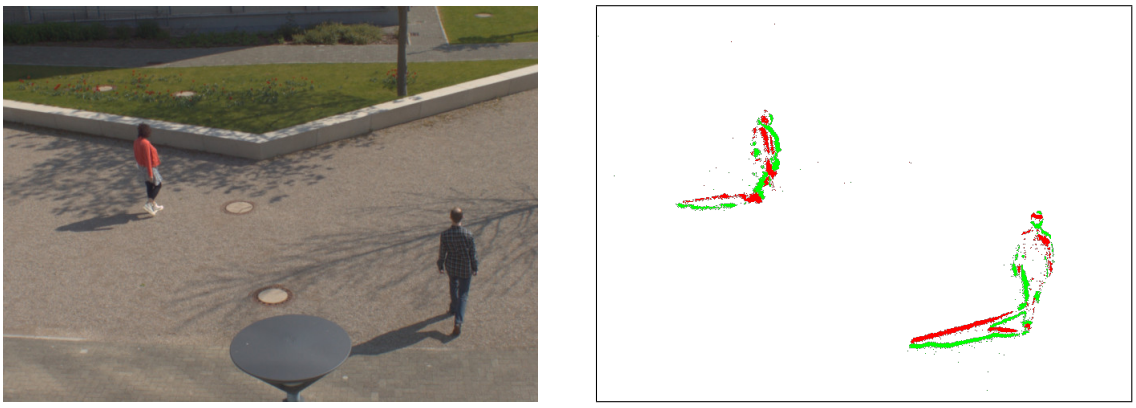
(c) "Fast" motion

Figure 1.4: Comparison of DVS and standard camera output using a rotating stimulus (adapated from [Mueggler et al., 2014]).

- **Position** (x_i, y_i)**:** Coordinates on the image plane of the DVS.
- **Timestamp** t_i**:** Time when the event was triggered (within the resolution of up to a few microseconds accuracy).
- **Polarity** p_i**:** Direction of the detected brightness change (ON: change from dark $\rightarrow$ bright; OFF: change from bright $\rightarrow$ dark)

The resulting output stream is spatially sparse but has very high temporal resolution. The ability to capture fast object motion is demonstrated in an example shown in Figure 1.3. In this example, the stimulus is assumed to be a rotating disk with a dot on it.

As shown in the Figure 1.4, no DVS output is generated if there is no movement and therefore no change in the scene. Compared to frame-based cameras, which are prone to motion blur during fast movements, the DVS stream output is much more robust in this regard.



(a) Classic RGB frame that exposes a high level of personal information

(b) AER output from a stationary DVS converted to an image in which people are abstracted by edges caused by their movements

Figure 1.5: Benefits of the Dynamic Vision Sensor from a privacy perspective.

1.3.2 Technical Advantages

Compared to their frame-based counterparts, Dynamic Vision Sensors offer several technical advantages:

High Temporal Resolution:
Dynamic Vision Sensors are able to record very fast movements. This is because the detection of changes in brightness is performed directly in analog circuits. With fast readout logic, events can be time-stamped at microsecond resolution. This makes event cameras less prone to effects such as motion blur than frame-based cameras.

Low Latency:
The pixels of a Dynamic Vision Sensor operate independently of each other. As soon as a change in the brightness is detected at a pixel, the output of that pixel is requested. As a result, there is no need to wait for an exposure time and synchronous readout for output, as is the case for frame-based cameras.

Low Data Redundancy and Power Consumption
Dynamic Vision Sensors ideally provide no sensor output in static scenes or static subareas. Compared to frame-based cameras (see Figure 1.4b), this can significantly reduce the amount of data that needs to be transmitted and subsequently processed.

Accordingly, in a Dynamic Vision Sensor, power is only consumed for processing the changing pixels. This usually results in a low power consumption of the sensor.

High Dynamic Range (HDR):
In traditional frame-based cameras, the image sensor is exposed for a pre-selected period of time known as the exposure time. Under certain lighting conditions, where very dark and very bright areas are present at the same time, it is often impossible to avoid over- or underexposure in a single shot.

The dynamic range of a sensor is the ratio of the highest to the lowest level of illumination at which the sensor is able to operate. Dynamic Vision Sensors have a much higher dynamic range, typically more than 120 dB compared to approximately 60 dB for professional frame-based systems. While the human eye has a maximum intrascene dynamic range of about 40 dB, the biological dark-adaptation process allows humans to achieve a

Event-based vision	Frame-based vision
asynchronous	synchronous
spatial sparse	spatial dense
unordered	ordered

Table 1.1: Output modality of event-based versus frame-based vision.

total dynamic range across multiple scenes of about 120 dB (compare to [Darmont, 2013, p. 37]).

The high dynamic range of DVS is due to the fact that the pixels respond independently to changes in logarithmic intensity. As a result, these sensors can be used in a wide range of scenarios, from dark to bright environments. The high dynamic range is also particularly advantageous for outdoor applications.

Privacy Aspects:

In its purest form, the output of a Dynamic Vision Sensor does not contain any intensity information for the triggered events. This means that although a DVS is an optical sensor, no gray or color values need to be considered in software components for further processing. The DVS thus directly addresses potential privacy issues by being limited to the components of the AER data stream.

The sensor's output stream contains an accurate representation of changes within the scene while minimizing the amount of personal information captured (compare to Figure 1.5, where each image pixel encodes the polarity of the last event that occurred within a 60 ms time window. An increase in brightness is shown in green and a decrease in red.).

There are approaches to reconstruct conventional intensity images from a DVS data stream (e.g., [Bardow et al., 2016; Reinbacher et al., 2016; Rebecq et al., 2019; Scheerlinck et al., 2019; Rebecq et al., 2021; Paredes-Valles and de Croon, 2021; Liu and Dragotti, 2023]) that may threaten this privacy advantage. However, direct preprocessing of the data prior to storage can significantly complicate this reconstruction task. A restrictive spatio-temporal filtering of the events, the subsampling or the aggregation of the events into other representations are possible processing steps that are suitable for this purpose.

There are also approaches to encrypt the event stream to defeat such reconstructions. For example, in [Du et al., 2021] a chaotic mapping is proposed to scramble the positions of events and to flip their polarities. While in [Zhang et al., 2024], carefully generated noise is introduced into the event stream to obfuscate the given data.

Thus, potential privacy concerns can be avoided.

1.4 Signal Processing

The paradigm shift in the operating logic of Dynamic Vision Sensors requires a rethinking of how their signals are processed. This concerns both the general challenges of dealing with this sensor technology and the various data representations introduced for it.

1.4.1 Challenges

When using Dynamic Vision Sensors, the way in which they operate presents a number of challenges that need to be considered during processing. These include (compare also to [Zhu, 2019, Section 1.5]):

Output Paradigm:

Because of the operating paradigm described above, the output of a Dynamic Vision Sensor is fundamentally different from that of a standard camera (see Table 1.1). A standard camera takes synchronous measurements based on a defined exposure and acquisition speed (fps, frames per second). This results in an ordered, synchronous, and spatially dense sampling: the classical image or frame.

The output of a Dynamic Vision Sensor is asynchronous, unordered, and spatially sparse. This, means that classical image processing methods cannot be applied natively.

Photometry:

Frame-based cameras provide absolute intensity information, such as grayscale values, for each pixel. The basic DVS event stream contains only information about the presence of brightness changes. In addition, only the polarity of the brightness change is known.

Since these changes also depend on past and present motion in the scene, or on sensor motion, and not just on the brightness within the scene, there is no such thing as classic photographic consistency. Photo consistency is the assumption that a point, under constant illumination and Lambertian reflection, will produce an equivalent result for all cameras that can see it.

Spatial and Temporal Separability:

For frame-based standard cameras, the interpretation of spatial and temporal aspects can be separated. For example, the determination of edges in an image as a spatial analysis can be separated from the search for temporal movements of an object.

For an event in the output stream of a DVS, the spatial and temporal information are joined. Without prior information, it is not possible to determine whether a single event has a new origin or belongs to an older point that has already generated events and moved to another position.

Noise and Sensor Resolution:

As stated in [Gallego et al., 2022], "[a]ll vision sensors are noisy because of the inherent shot noise in photons and from transistor circuit noise, and they also have nonidealities. This situation is especially true for event cameras, where the process of quantizing temporal contrast is complex and has not been completely characterized." The processing of DVS data is therefore challenging due to the presence of sensor noise.

In addition, the spatial resolutions of available DVS models are significantly lower than those of available standard cameras, resulting in significantly smaller object image scales.

Availability of Training Data:

Another obstacle to the development of event-based algorithms is the lack of publicly available and appropriately annotated datasets. This is due to the fact that Dynamic Vision Sensors have only recently become commercially available.

1.4.2 Event-Stream Representations

In the context of classical image processing, the output stream of a Dynamic Vision Sensor can be considered "unconventional". In order to process this type of output, different event representations have been developed. The boundary between pure event representation and basic feature extraction for a given application is often ambiguous.

Basic categories and widely used event representations are summarized here in order to categorize the following work:

Event-by-Event:

The native representation of events as $e_i = \{(x_i, y_i), t_i, p_i\}$, as provided in the AER stream, is preserved. Events are processed one at a time and in the order they are transmitted. Since a single event contains only limited information, knowledge of previous events must be included in the processing.

Pro:

- Very low latency processing is possible.
- Event sparsity and high time resolution properties are preserved and exploited.

Con:

- Application-specific development of custom computer vision algorithms is required to use this form of representation, as it is currently considered to be an "unconventional" signal without a broad base of existing solutions.

3D Space-Time Event Cloud:

The Dynamic Vision Sensor output is interpreted as a point cloud, which is formed from the unsorted set of events $\{x_i, y_i, t_i\} \in \mathbb{R}^3$. The combined spatial information (x_i, y_i) and temporal information t_i is transformed into a geometric description.

Pro:

- Event sparsity property is preserved.
- Event timestamp resolution is fully preserved.

Con:

- A "Bag-of-Events" is required. For the creation of the point cloud, it is necessary to divide the event stream into time windows. This leads to a latency in processing.
- Depending on the amount of triggered events, subsampling of the input data may be required for processing.

Event-Graph:

The spatio-temporal relationships between events are transformed into a neighborhood graph. The events $e_i = \{x_i, y_i, t_i\}$ are interpreted as vertices of the graph. The vertices are interconnected by edges based on their distance in x, y, and t.

Pro:

- Event sparsity property is preserved.
- Event timestamp resolution is fully preserved.

Con:

- Graph generation requires high computational resources, i.e. for the neighborhood calculations involved in defining edges between events.
- High memory consumption for data structure management.
- Depending on the amount of triggered events, subsampling of the input data may be required for processing.

Event-Voxelization:

The three-dimensional event stream is divided into a set of discretized regions. For example, at each spatial position (x, y) of the sensor array, voxels are sampled at short time intervals. The number of events contained in these voxels is counted. This results in a 3D voxel event histogram.

Pro:

- Classical computer vision methods, such as convolutional neural networks, are directly applicable. This allows reuse of methods and insights.
- Depending on the selected voxel size, compromises are possible in preserving the spatial and temporal properties between the aforementioned 3D representations and 2D frame conversions.

Con:

- Loss of event sparsity, as many voxels may not contain events.
- Voxel size is a freely configurable parameter that quantifies the DVS resolution. It must be fine-tuned for individual applications.
- High memory requirements for fine resolution of dense 3D voxel grids.

Frame-Conversion:

A set of events, typically based on (a) the selection of a fixed length time segment of the DVS stream, or (b) the selection of a fixed set of sequentially transmitted events, is converted into a 2D frame representation [Liu and Delbruck, 2018]. For this purpose, the selected events are typically projected onto the xy-plane. Since the events are triggered by changes, the resulting images are similar to edge images from standard cameras (compare to Figure 1.5). In classical image processing, edges are considered to have a high information content.

For the actual calculation of the pixel values, several encoding rules have been proposed. These include:

- Simple binary pixel encodings, which only consider the presence or absence of an event.
- Color-based encodings, which use color to represent certain characteristics, such as event polarization, as shown in Figure 1.5b.
- Time-surfaces, which encode event timestamps in an image.

Pro:

- It is an intuitive, long-established, and well-known form of representation.
- Classical computer vision methods, such as convolutional neural networks, are directly applicable. This allows reuse of methods and insights.

Con:

- The event-based processing paradigm is largely abandoned.
- The property of sparsity and the high temporal resolution are (mostly) lost.
- The selected frame encoding rule strongly affects later processing/results.

Chapter 2

Research and Development Context

The development goals for a proof-of-concept DVS-based monitoring system are summarized, including the monitoring metrics to be derived. The main aspects of the research are presented. Based on the developed processing pipeline, own contributions are described, and the work is outlined.

2.1 Scope and Research Topics

The main objective of this work is the development of an overall system based on Dynamic Vision Sensors for public space monitoring. The goal is to provide a technical foundation of a monitoring system that allows the integration of observed usage into urban planning processes. The observations of the public space should be carried out in the long-term and not on a day-to-day basis, as this is much more representative.

This raises the question of what kind of processing is appropriate and can be used for this purpose. The monitoring system must cover the entire process from the acquisition of the DVS signal to the automatic processing and visualization of the obtained and summarized results.

In addition to the signal processing challenges posed by the new sensor paradigm, real-world aspects must also be considered. Outdoor recordings are influenced by the environment and contain interferences that must be taken into account.

The main parameters to be derived in the monitoring are

(a) the *number of users* in the observation area and
(b) their *spatial distribution* across the measured field.

Main Research Topics

The creation of this monitoring system required the consideration of several key tasks and aspects, including:

Contributed Datasets:

For the development of event-based public space monitoring, it is crucial to select an appropriate dataset that contains classes of interest for the application scenario. It is also crucial that these labels are sufficiently detailed.

In real-world outdoor scenarios, signals from Dynamic Vision Sensors also contain environmental influences such as airborne particles, insects, or rain. In order to develop a

system capable of processing data captured in such an outdoor scenario, it is important to account for these included environmental events as well as for sensor noise in the used datasets.

In the domain of event-based vision, no datasets meeting these requirements were available (see introduction of Part II).

As a consequence, own datasets had to be recorded and processed. To develop the system, it was necessary to implement a process for creating and automatically labeling recordings with annotations on a per-event basis.

Evaluation of Event Stream Representations for Segmentation Purposes:
The previous Section 1.4.2 introduced different categories of event representations that are used to process the "uncommon" signal of the DVS.

In the literature, events are often represented by converting the event stream into a classical 2D frame structure. This data representation format is widely used and has been used in computer vision for many years, making it a reasonable approach. In addition, many algorithms and insights from classical computer vision are directly applicable in this way.

However, this frame conversion only preserves the inherent sensor properties of the DVS to a very limited extent. The loss of the inherent sparsity of the DVS data is the result of this conversion. Additionally, depending on the encoding, the high temporal resolution of the event stream may be lost or represented in a limited way.

In selecting a data representation, it is important to consider not only the preservation of sensor properties but also the complexity of creation and storage requirements. For example, an event graph representation is usually associated with high requirements, whereas a 3D space-time event cloud is practically given natively, while fully preserving the sensor properties in terms of sparsity and time resolution.

Main Research Question:
The main research question is whether and to what extent native 3D processing is advantageous for segmentation purposes of the given sensor signal. Therefore, the first systematic evaluation of 3D space-time event cloud representations and their processing with 3D deep learning approaches has been performed in comparison to other event representations.

Real-World Application:
The objective of developing a monitoring system capable of generating an evidence-based observation database to support the urban public planning process is being evaluated in a practical application scenario. Several challenges arise in this process.

In the long-term monitoring performed, a selection of scenes to be stored for further analysis must be made. This is necessary because simple, unrestricted, and uninterrupted monitoring is not practical. With respect to the intended processing, this selection must consider and exclude unwanted signal components caused by environmental effects.

An evaluation of the final quality of the developed processing methods had to be performed in the context of the conducted real-world long-term monitoring. In order to achieve this objective, representative scenes were selected for evaluation. It was necessary to confirm the feasibility of deriving the desired monitoring parameters and to develop suitable aggregations of the derived usage parameters in order to summarize the collected usage insights for the planning processes.

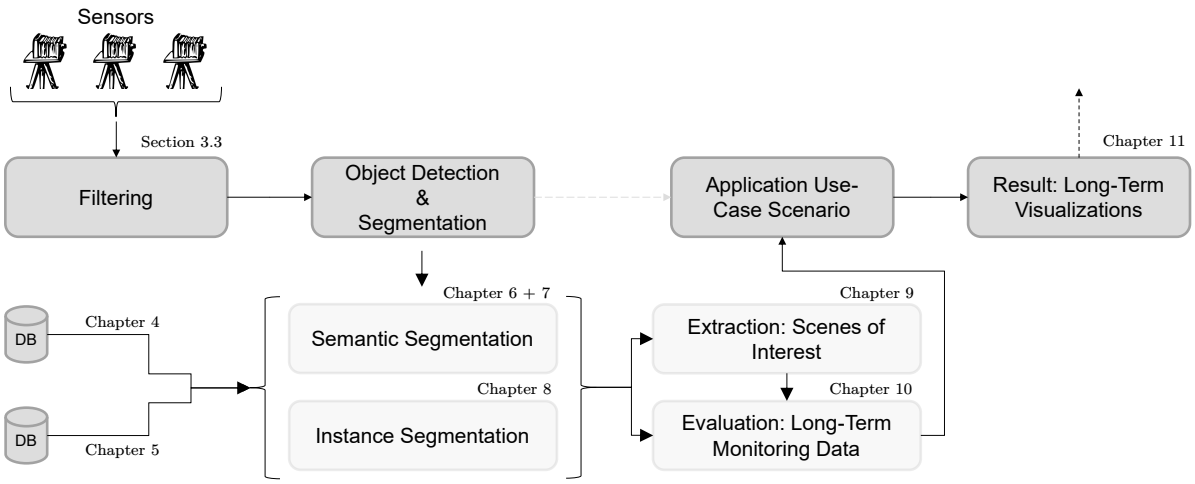


Figure 2.1: Basic concept of the implemented DVS processing chain for urban space monitoring.

2.2 Outline and Contributions

This dissertation consists of several previously published papers. I contributed most of the work, including design, implementation, evaluation, writing, and presentation, in all of the publications listed below. Parts contributed mainly by co-authors are mentioned. To acknowledge all co-authors, the first-person plural pronouns "we" or "our" are used throughout this dissertation.

Figure 2.1 shows the processing pipeline developed for the measurement system. Based on this pipeline, which covers the entire processing from signal acquisition, segmentation, application related evaluation in a real-world context, up to the appropriate result visualization, the structure and contributions of this dissertation are summarized and outlined below.

Part I: Background and Context

Chapter 3 introduces the outdoor measurement site and the Dynamic Vision Sensor used. This site was used to record the necessary long-term monitoring data for the development and evaluation. The main components of the system used on-site, as well as the monitored area, are described. Section 3.3 summarizes sensor noise reduction experiments that led to the selection of a spatio-temporal filtering procedure used throughout the work.

Part II: Contributed Datasets

First, a general overview of the state-of-the-art on event-based vision datasets is given.

We published the first DVS-based dataset in the context of real-world long-term outdoor monitoring. This dataset and the process of its creation are described in Chapter 4. The main contribution is the inclusion of environmental effects not previously accounted for and a per-event labeling for semantic segmentation purposes.

→ Published in: **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE

The lack of suitable and sufficiently discriminative and continuous label annotations for segmentation purposes is still a major limitation in event-based vision research. Chapter 5 addresses this issue by introducing a hardware setup and automated processing that enables such annotations. As a main contribution, besides providing a ready-to-use dataset, we provide a detailed description of a setup that allows the extraction of high-quality, continuous instance labels based on the fusion of an RGB color sensor and the incorporation of color features into the recording.

→ Published in: **Bolten**, T., Neumann, C., Pohle-Fröhlich, R., and Tönnies, K. (2023a). N-MuPeTS: Event Camera Dataset for Multi-Person Tracking and Instance Segmentation. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 290 – 300. INSTICC, SciTePress

I would especially like to thank Christian Neumann for his work on the physical sensor mount and for his contribution of the hue-based segmentation performed in this paper.

Part III: Segmentation

First, we introduce different levels of segmentation and review the state-of-the-art of segmentation in event-based vision. Next, we discuss the current situation where there is a lack of a priori knowledge about which event representation is best suited to solve a given processing task.

Chapter 6 lays the foundation for processing in terms of semantic segmentation in the form of 3D space-time event clouds. The optimization of different input data and network configurations for this processing based on a quantitative evaluation is the main contribution of this chapter. The developed processing pipeline, including different variants of temporal input data scaling in preprocessing, is presented and key aspects of network processing are systematically investigated and optimized.

→ Published in: **Bolten**, T., Lentzen, F., Pohle-Fröhlich, R., and Tönnies, K. (2022a). Evaluation of Deep Learning based 3D-Point-Cloud Processing Techniques for Semantic Segmentation of Neuromorphic Vision Sensor Event-streams. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 179. INSTICC, SciTePress

Thanks to Felix Lentzen for performing different training runs to evaluate the considered 3D deep learning network variants.

In the following Chapter 7, the evaluation is extended to include further 2D frame representations and 3D voxel grids, while a UNet-based processing is considered in the comparison.

→ Published in: **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2023c). Semantic Segmentation on Neuromorphic Vision Sensor Event-Streams Using PointNet++ and UNet Based Processing Approaches. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 178. INSTICC, SciTePress

Chapter 8 focuses on the extension of segmentation to the derivation of object instances. The main contribution here is the comprehensive comparison of 2D, 3D voxel and space-time event cloud representations in this processing. Adaptive preprocessing for the selection of regions of interest is introduced into the processing, and in addition to a baseline method based on the semantic segmentation approaches of the previous chapters, state-of-the-art methods for segmentation in 2D and 3D are considered.

→ Published in: **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2024). Instance Segmentation of Event Camera Streams in Outdoor Monitoring Scenarios. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 452 – 463. INSTICC, SciTePress

Part IV: Evaluation of Real-World Application

A children's playground served as a Living-Lab for a real-world, long-term monitoring. First, the database collected at this site is summarized.

The system development and monitoring was carried out over multiple years. Performing such long-term monitoring with multiple sensors results in a large amount of data. Chapter 9 describes the processing used to extract and store scenes of interest on-site. The knowledge gained from the evaluation of the semantic segmentation is used to implement a processing system that can handle real monitoring data from the actual system while taking into account limited computing resources. The evaluation demonstrates that scenes are accurately detected and stored.

→ Published in: **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2023b). Semantic Scene Filtering for Event Cameras in Long-Term Outdoor Monitoring Scenarios. In Bebis, G. et al., editors, *18th International Symposium on Visual Computing (ISVC), Advances in Visual Computing*, volume 14362 of *Lecture Notes in Computer Science*, pages 79 – 92, Cham. Springer Nature Switzerland

Thanks to Noel Dominik Hanraths Pereira for his contribution to the creation of the human reference annotations used to evaluate the developed scene selection.

Chapter 10 evaluates the overall quality of the detection and segmentation based on these stored scenes. Representative scenes are selected from the long-term monitoring performed to cover all typical on-site usage scenarios. A processing is developed based on the evaluated segmentation techniques to derive the desired monitoring parameters, and the system quality is compared to annotations provided by several human participants.

→ I would like to thank all the participating human annotators who took the time to provide their annotations, thus enabling the evaluation and comparison of the developed components.

Visualizations to aggregate and summarize results for stakeholders are created in Chapter 11. For this purpose, the obtained segmentations are used to determine the spatial distribution of the activity and are then summarized using bird's eye view heat maps.

→ Published in: **Bolten**, T., Pohle-Fröhlich, R., Volker, D., Brück, C., Beucker, N., and Hirsch, H. (2022b). Visualization of Activity Data from a Sensor-based Long-term Monitoring Study at a Playground. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAPP)*, pages 146 – 155. INSTICC, SciTePress

I would like to thank my colleagues from the Faculty of Design, Dorothee Volker and Clemens Brück for their contribution to the selection of the visualization variants created and evaluated. They also conducted and evaluated the user study.

Part V: Conclusion and Outlook

Part V concludes with a summary of the final results of the data analysis and research findings. It also critically reflects on the limitations of the work. By discussing these limitations and providing ideas for future work, the way is paved for further research.

Part VI: Appendices

To enhance the main text's readability, some elements have been moved to the appendix. This appendix provides supplementary and more detailed results for some of the evaluations discussed in the main text. It also includes a more technical description of the sensor system developed.

A method for thermal stabilization of the Dynamic Vision Sensors, which proved to be necessary in the course of the work, as well as a retrospective evaluation of the achieved stabilization is discussed in Appendix A. The technical hardware setup of the developed monitoring system, including the system components, is described in the following Appendix B.

→ For his work on the thermal stabilization system and its retrospective analysis, I would like to thank Christian Neumann. I would also like to thank Dorothee Volker and Clemens Brück from the Faculty of Design for their joint work on the design and manufacture of the used sensor enclosure and its mast mount.

Appendix C provides a basic overview of the developed software, its basic features and structure. Finally, Appendix D provides additional information and event statistics on the contributed datasets, while Appendix E provides more information on the network configurations used and extended evaluation results.

Chapter 3

Application Scenario and Sensor System

The measurement system and site are introduced. They are used in this dissertation as an example for the development of the public space monitoring system. Furthermore, the selected Dynamic Vision Sensor model is presented. Finally, an analysis of the sensor noise contained in the recordings is performed and countermeasures taken are presented.

3.1 Living-Lab: Main Characteristics

A children's outdoor playground in Mönchengladbach-Rheydt, Germany, was chosen as an example for the development of the Dynamic Vision Sensor-based monitoring system. This area served as a "Living-Lab" for the development of this system.

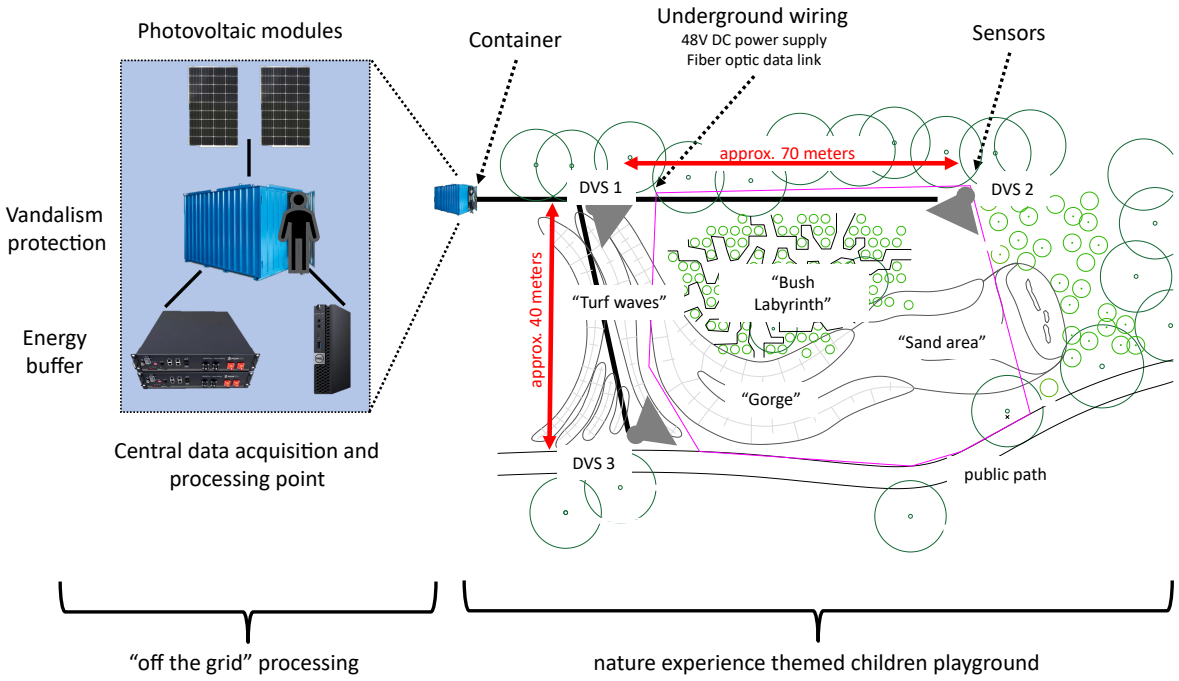
A Living-Lab is a concept that refers to real or simulated environments in which new technologies or products can be tested in a practical and efficient manner [Hossain et al., 2019]. To enable fast, efficient and realistic prototyping, it takes an *open innovation* approach that involves users and stakeholders in the development process. In the specific case, this concerns the development of the sensor system and the stakeholders involved from research, urban planning, city administration and real users.

The monitored area of this playground is approximately 2800 square meters and is covered by three fixed and elevated mounted Dynamic Vision Sensors. A schematic plan of the area is shown in Figure 3.1a.

The measurement area was redesigned as part of a construction project. As part of this redesign, underground cabling for the sensor system was incorporated. In addition to the sensor power supply, this cabling also includes a data connection between the sensor locations, which was realized with fiber optic cables due to the distances to be bridged. The central data acquisition component is located in a container on-site to protect it from weather and vandalism (see Figure 3.1b). The entire measurement and test setup is located in an urban forest area and is designed for autonomous operation using photovoltaic technology and a battery for intermediate energy storage.

Each of the three Dynamic Vision Sensors is equipped with an 8 mm wide-angle lens[3] and mounted in a weatherproof enclosure on a mast at a height of approximately 6 meters. The sensors are tilted approximately 25 degrees toward the ground (see Figure 3.1c). The positions of the sensors were chosen to minimize blind spots caused by terrain features of the playground

[3]Computar MEGAPIXEL V0814-MP, f=8mm, f/1.4 – f/16, 1 inch, C-Mount



(a) System concept of the sensor system realized in Mönchengladbach-Rheydt

(b) Central data acquisition and processing point

(c) Sensor mount and enclosure

Figure 3.1: Living-Lab: Sensor and measurement setup (adapted from [**Bolten** et al., 2021]).

(a) DVS1

(b) DVS2

(c) DVS3

Figure 3.2: Grayscale images of the fields of view of the Dynamic Vision Sensors in the Living-Lab.

Company	Name	Reference	Year	Spatial res.
iniVation	DVS128	[Lichtsteiner et al., 2008]	2008	128 px × 128 px
	DAVIS240	[Brandli et al., 2014]	2014	240 px × 180 px
	DAVIS346	-[4]	2017	346 px × 260 px
	DVXplorer	-[5]	2020	640 px × 480 px
Samsung	DVS Gen1	-[6]	2014	640 px × 480 px
	DVS Gen2	[Son et al., 2017]	2017	640 px × 480 px
	DVS Gen3	-[6]	2018	640 px × 480 px
	DVS Gen4	[Suh et al., 2020]	2020	1280 px × 960 px
Prophesee	ATIS	[Posch et al., 2011]	2011	304 px × 240 px
	Gen3 CD	-[7]	2017	640 px × 480 px
	Gen3 ATIS		2017	480 px × 360 px
	Gen4 CD	[Finateu et al., 2020]	2020	1280 px × 720 px
	GenX 320	-[8]	2023	320 px × 320 px
CelePixel	CeleX-I	[Chen et al., 2012]	2012	64 px × 64 px
	CeleX-II	[Guo et al., 2016]	2016	192 px × 160 px
	CeleX-III	[Huang et al., 2017]	2017	384 px × 320 px
	CeleX-IV	[Guo et al., 2017]	2017	768 px × 640 px
	CeleX-V	[Chen and Guo, 2019]	2019	1280 px × 800 px

Table 3.1: Major DVS manufacturers and model overview (adapted and extended from [Gallego et al., 2022, Table 1, p. 6] and [Gehrig and Scaramuzza, 2022, Table 1, p. 4]).

(trees, bushes, or hills). Figure 3.2 shows the resulting field of view (FoV) of each sensor as a reference grayscale image for better understanding.

For a more detailed description of the hardware setup, see Appendix B.

3.2 Sensor Model Selection

For many years, Dynamic Vision Sensor technology was only available in prototype form. It was only used by a few researchers. This changed at the turn of the millennium. Since then, several new sensors have been introduced. These are now being sold commercially by various companies for productive use. Table 3.1 provides an overview of the most common models of Dynamic Vision Sensors currently used in research and commercial applications. Currently, purchase prices of several thousand EUR per unit can be expected.

At the time of starting this work, the CeleX-IV Dynamic Vision Sensor manufactured by CelePixel offered the best compromise between availability, spatial resolution, and retail price. Therefore, this specific sensor model was selected and is used throughout this work. In the following, this sensor model is briefly introduced.

CeleX-IV Dynamic Vision Sensor

The CeleX-IV sensor is the fourth generation in a series of Dynamic Vision Sensors [Guo et al., 2017]. It was developed and marketed by CelePixel Technology Co., Ltd.

In the course of this work, Will Semiconductor Co., Ltd. acquired this company. As part of its subsidiary OmniVision Technologies, they continue to develop event-based cameras

[4] https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf
[5] https://inivation.com/wp-content/uploads/2023/03/DVXplorer.pdf
[6] https://rpg.ifi.uzh.ch/docs/CVPR19workshop/CVPRW19_Eric_Ryu_Samsung.pdf
[7] https://docs.prophesee.ai/stable/hw/sensors/gen31.html
[8] https://www.prophesee.ai/event-based-sensor-genx320/

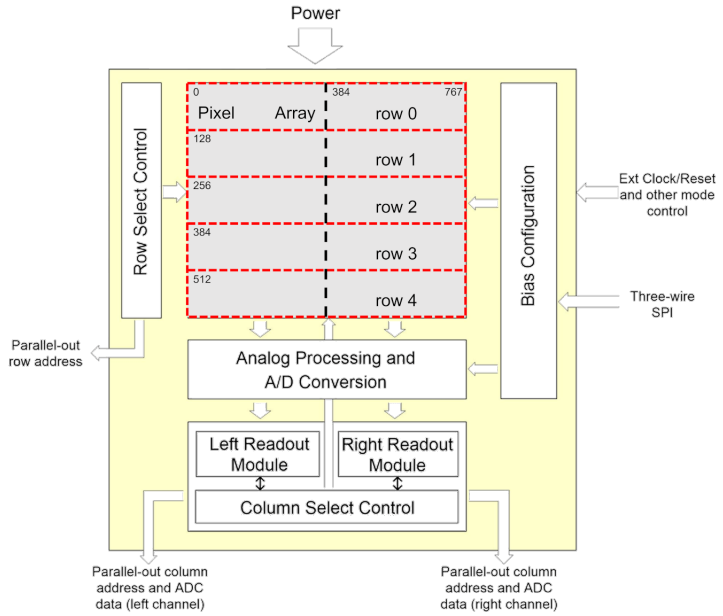


Figure 3.3: CeleX-IV logical sensor block diagram (adapted from [CelePixel, 2018]).

[Guo et al., 2023]. In this way, the rich experience of CelePixel will be part of the further development in the field of event-based vision.

The CeleX-IV is a high resolution sensor, with 768 × 640 pixels and a maximum output of 200 Meps (Mega events per second). Using TSMC's 0.18 µm 1P6M CMOS image sensor manufacturing process, each pixel occupies 18 × 18 µm with 9 % fill factor [CelePixel, 2018]. The sensor consists of the pixel array, a row/column handshaking processor, a column parallel ADC data path, and a configuration interface (see Figure 3.3).

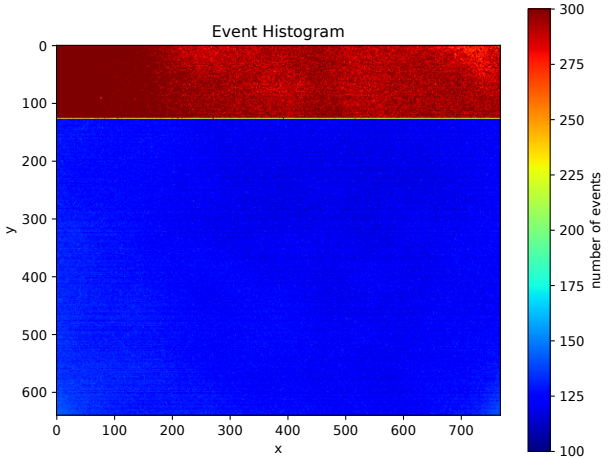
The basic sensor operation is summarized in [CelePixel, 2018] as follows (adjusted for grammar):

> "The pixels are organized into rows and columns that share the same request and acknowledgement buses. Each pixel monitors the relative voltage change on a logarithmic photodetector and triggers an event when a certain threshold is reached. When one or more pixels within a row are triggered, a request signal is sent to the row arbitration processor. The row arbitrator can receive multiple requests simultaneously. After arbitration, only one row is [randomly] acknowledged. The row address is encoded and sent out of the chip. The triggered pixels within the selected row send another request to the column processor. The column processor, which is divided into two in-parallel channels, each responsible for 384 pixels, will process the column requests one by one."

The combined row and column outputs, with timestamps added by an interface FPGA module, form the sensor's event output stream.

In addition to dividing the sensor array into two column blocks for readout, the rows are also divided into semantic blocks. The available 640 pixel rows of the sensor array are divided into five separate blocks of 128 rows each. However, initial experiments with the sensor revealed that these blocks are not internally processed in the same way. A simple experiment to illustrate this is shown in Figure 3.4. Assuming a homogeneous stroboscopic flash with uniform

(a) Homogeneous strobe flashing (paper sheet used as diffuser)



(b) Event histogram resulting from accumulation over 1.8 seconds (note the transition between rows 127 and 128)

Figure 3.4: CeleX-IV sensor readout issue. The sensor block in the top row has a higher reading frequency than the other blocks.

and random on-chip row acknowledgement, the triggered event rate should be nearly identical for the entire sensor array. However, this is not the case.

The top block has a significantly higher event rate. It turns out that this block has a four times higher readout frequency in comparison, as stated after consulting with the manufacturer. As shown, this results in different event rates. In addition, the off-chip timestamps, which are added by the interface FPGA, also show differences in their accuracy between the blocks.

According to the manufacturer, the only way to avoid this is to fully disable the upper sensor row block. Therefore, unless explicitly stated otherwise, the available sensor resolution is limited to 768×512 pixels for all recordings in the course of this work.

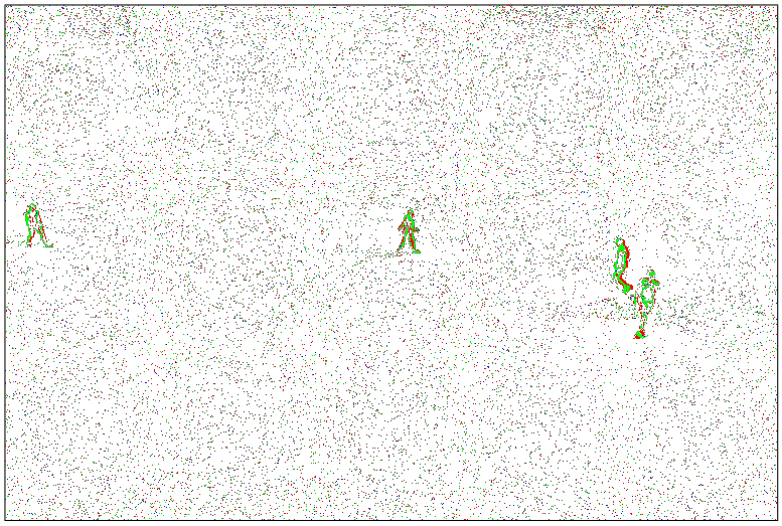
3.3 Sensor Noise

As with all image sensors, Dynamic Vision Sensors are also subject to noise. This refers to all effects that result in unwanted signals that degrade the system and imaging performance of a sensor. In particular, this includes the effects of dark currents, thermal noise, shot noise, 1/f noise and readout noise (for a more detailed and general introduction to digital sensor noise see [Nakamura, 2005, Chapter 3.3, pp. 66 ff.]).

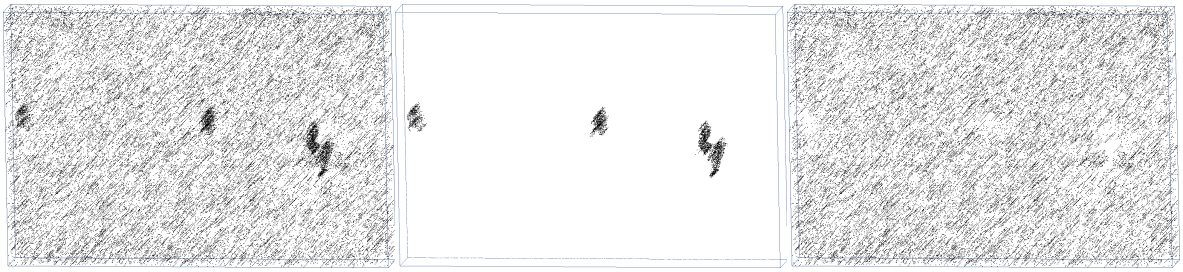
One of the major consequences of noise effects in Dynamic Vision Sensors are so-called “Background Activity” (BA) events. These are DVS events that are triggered by pixels without a change in the scene. These events are therefore not meaningful. The level of background activity depends heavily on the actual lighting of the scene, but also on the DVS bias settings [Guo and Delbruck, 2023; Graça and Delbruck, 2021].

These bias values can be used to optimize the sensors for different application requirements and conditions. They include settings for selected contrast sensitivity thresholds, pixel bandwidth, and pixel refractory period. The refractory period defines a time span during which a pixel cannot generate another event. This setting can be used to limit the maximum trigger rate of pixels so that they do not consume the entire capacity of the output bus.

Since signal events are also often suppressed when noise rates are reduced, the choice of these settings is a compromise. The optimal choice of these parameters for a particular



(a) Polarity encoded 2D projection; all events



(b) (x, y, t) view; all events

(c) Signal events; manually selected

(d) Noise events, mainly BA

Figure 3.5: CeleX-IV sensor background noise example. Recorded with a stationary sensor under natural daylight. A 25 ms event stream window is displayed.

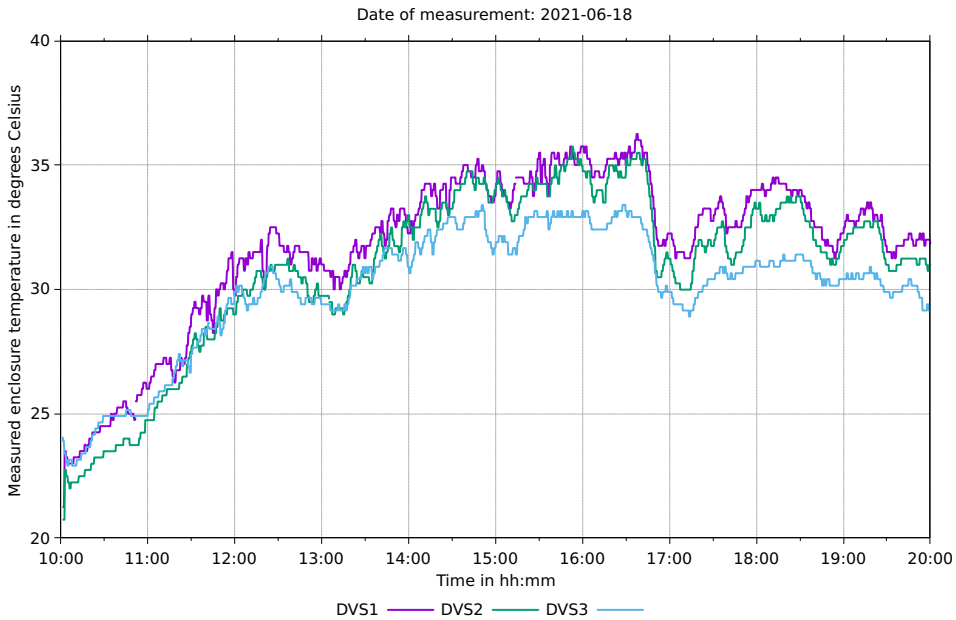


Figure 3.6: Measured internal temperatures of the DVS mast head enclosure (see Figure 3.1c) on a summer day.

application is therefore a difficult task, even for experts in the field. Recently, papers have been published that provide at least some guidance to the user [McReynolds et al., 2022; Graça et al., 2023a,b].

The CeleX-IV sensor has a high level of background activity compared to other sensors, even in well illuminated scenes. This is clearly shown in Figure 3.5.

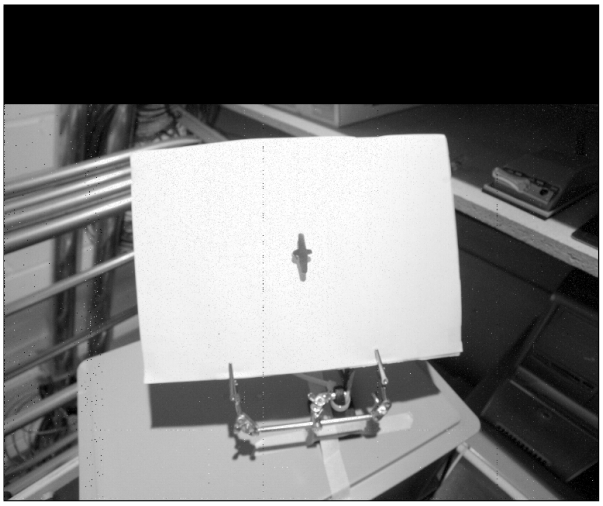
Unfortunately, the design of the CeleX-IV DVS pixel is not fully disclosed to the public. In addition, the communication interface between the sensor and the acquisition computer system is implemented using an FPGA whose configuration is also largely undocumented. The SDK for the sensor only allows the setting of a single contrast threshold. All other bias configurations are not accessible to the user and are handled internally by the FPGA and the on-chip bias configuration. This limits the ability to fine-tune the CeleX-IV sensor for desired applications and noise levels.

As a consequence, from an application point of view, the sensor configuration and the associated sensor (noise) behavior had to be considered as "given" in the context of this work.

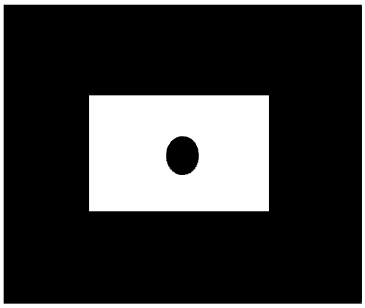
Nevertheless, the prevailing sensor noise can be controlled to some extent, as it is strongly temperature dependent. This aspect will be discussed in more detail in the following section and is used to improve the signal quality. Remaining noise components can be further reduced by signal processing in the form of (spatio-temporal) event filtering. An evaluation of different filters for the selection of a suitable signal preprocessing follows in Section 3.3.3.

3.3.1 Background Activity Temperature Correlation

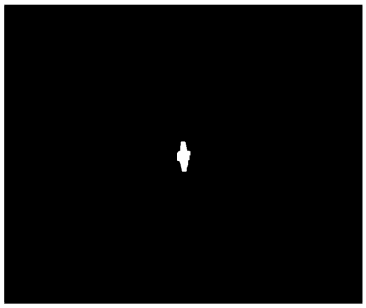
The measurement setup in the described outdoor Living-Lab (see Section 3.1) means that the sensors have to be operated at very different ambient temperatures. This includes also high temperatures caused by direct solar radiation on the sensor housings. The measured internal temperature curves of the actively ventilated sensor enclosures on a typical summer day are shown in Figure 3.6. High sensor temperatures are known to have a significant effect



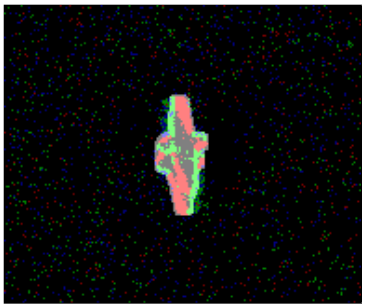
(a) Grayscale reference of the FoV of the sensor. A servo-driven moving rotor provided a constant, high-contrast motion/signal in the scene.



(b) Manually selected segmentation mask of the *homogeneous* part of the scene. Serves as an area for measuring BA events.



(c) Example of an automatically generated segmentation mask of the moving rotor. Events under this mask are counted as signal events.



(d) Cropped and enlarged rotor area from Subfigure c. The segmentation mask is highlighted on the polarity-encoded event stream projection.

Figure 3.7: Setup for temperature-dependent CeleX-IV noise experiments.

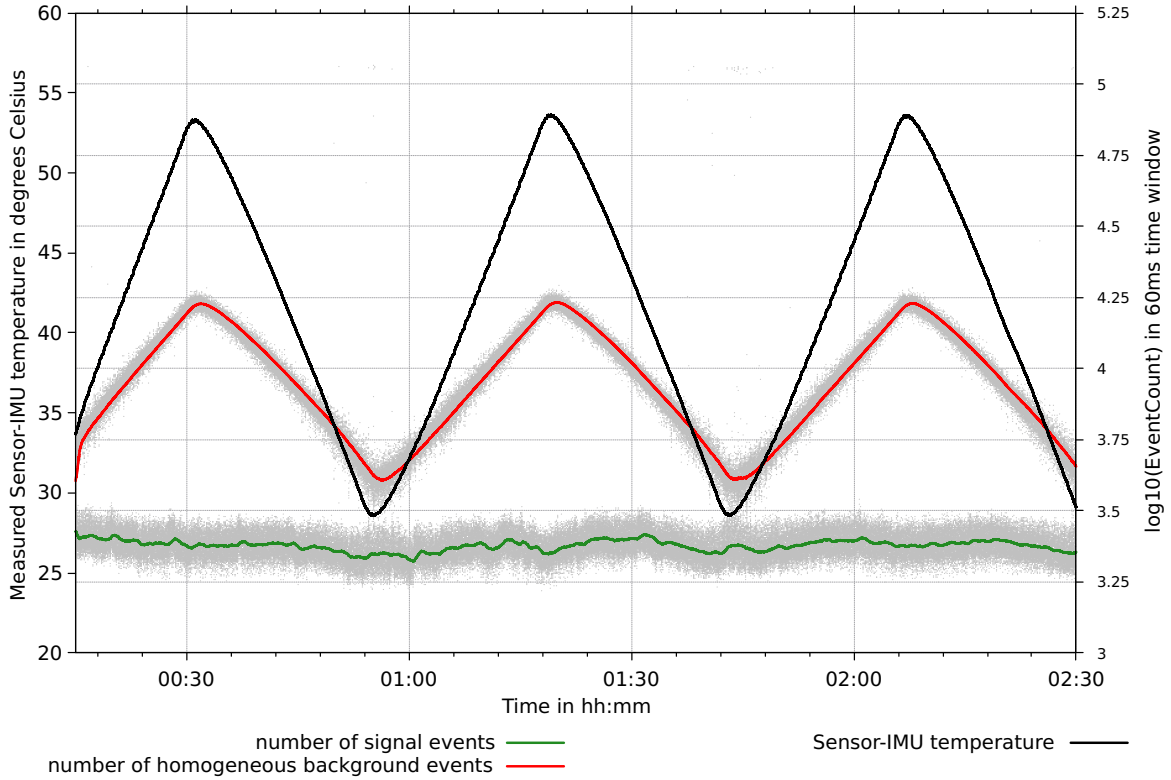


Figure 3.8: Measured signal and background activity event counts over different sensor temperatures. The experimental setup is shown in Figure 3.7.

on the noise level of Dynamic Vision Sensors [Nozaki and Delbruck, 2017; Xu et al., 2018a; Berthelon et al., 2018]. The following experiment was performed to get an insight into these temperature-related effects on the CeleX-IV sensor.

The sensor was placed in a controlled and static indoor environment illuminated by an incandescent halogen bulb, i.e. a light source that does not tend to flicker. A homogeneous white and planar surface was placed in front of the sensor. In the center of this surface, a servo-driven rotor provided a rotary motion with constant angular velocity that served as a high-contrast active stimulus. A grayscale image of the sensor's field of view is shown in Figure 3.7a for reference.

The static, white and homogeneous area excluding the rotor region was manually segmented (see Figure 3.7b). This area is used to measure the triggered background activity level as it is well-lit and does not change. Conversely, the moving rotor is automatically segmented over the duration of the recording (Figure 3.7c,d) and the matching events are considered as "real" signal events.

In this experiment, the sensor temperature was controlled to follow a triangular temperature curve. The resulting event counts for this setup are summarized in Figure 3.8. The sensor heating performed, as shown by the displayed sensor's IMU temperature, corresponds to the expected sensor temperatures on-site without any external temperature stabilization. As shown here, the background activity behavior of the sensor is highly dependent on temperature, while the signal output triggered by the moving rotor is only slightly affected by temperature.

For this reason, we have equipped the measurement system with an active temperature stabilization based on Peltier elements in order to homogenize the noise behavior to a lower level

for all recordings. For details on the temperature sensors used, their location, and further implementation details of this external temperature stabilization, see Appendix A.

3.3.2 Spatio-Temporal Filtering

Parts of the following filter descriptions have been previously published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE

In addition to the influence of sensor temperature on the overall noise level, background activity can be counteracted by filtering. Event filters do this by assuming that signal events are more dense and correlated than noise events, both spatially and temporally.

The "Background Activity Filter" [Delbruck, 2008] or variants of it [Czech and Orchard, 2016] are widely used in applications. Here, a filtering decision is made for each event directly based on the number of events in its spatio-temporal neighborhood.

Filters with minimized computational requirements can also be considered. For instance, for each event, only a limited number of previous events are checked [Guo et al., 2020b], or the spatio-temporal neighborhood is computed using hash functions [Guo et al., 2020a]. These approaches are therefore very memory efficient. This can be important if a hardware-related implementation is desired [Khodamoradi and Kastner, 2018].

It is also possible to learn the structure in the spatio-temporal neighborhood and use it for filter decisions by using deep learning-based approaches [Baldwin et al., 2020, "EDnCNN"] or [Guo and Delbruck, 2023, "Multilayer Perceptron Denoising Filter"].

Filter-Logic

In the described long-term monitoring scenario, a fast and computationally efficient approach for noise filtering is required. Based on the aforementioned assumption that signal events are more correlated in space and time, the following spatio-temporal filters were further examined:

Neighborhood-Filter:

The event stream is divided into non-overlapping time windows of several milliseconds. For each event e, spatially adjacent events in its time window tw_i and its preceding window tw_{i-1} are evaluated, and the number of populated spatial neighbor cells containing at least one other event is counted.

An 8-neighborhood is used for tw_i and a 4-neighborhood in the preceding tw_{i-1}. The event e is discarded if there are less than α populated neighboring cells for tw_i or less than β neighboring cells for tw_{i-1}.

In the following experiments, we set $\alpha = 4$ and $\beta = 2$, using a time window length of 60 ms.

Time-Filter:

For each event e it is checked if there was another event at the same spatial position in the previous x milliseconds. If there was no other event in that time span, the event e is considered to be noise.

In the following experiments, we calculated different results for the time thresholds x=3 ms, x=6.5 ms, and x=10 ms.

SeqX-Filter:

For each event e, the spatial distances to a few immediately preceding events in the sensor stream are calculated. If the smallest spatial distance is below a defined threshold σ, the event is kept. This filter thus uses the row-column-ordered readout logic of the events used by the DVS to select the events to be considered in the neighborhood formation. See [Guo et al., 2020b] for more details.

In the following experiments, we set the number of previous events considered to 10 and the threshold $\sigma = 0.01$.

EDnCNN:

The EDnCNN filter is based on deep learning. This approach was included to allow comparison with potential improvements that more complex and computationally intensive approaches might allow.

It consists of three 3×3 convolutional layers (using ReLU, batch normalization, and dropout) and two fully connected layers [Baldwin et al., 2020]. For each event, the network makes a binary classification based on a feature vector generated from the spatio-temporal neighborhood. An Adam optimizer with a decay rate of 0.1 and a learning rate of $2 \cdot 10^{-4}$ was used for training.

3.3.3 Filter Evaluation Results

Comparison for Different Sensor Temperatures

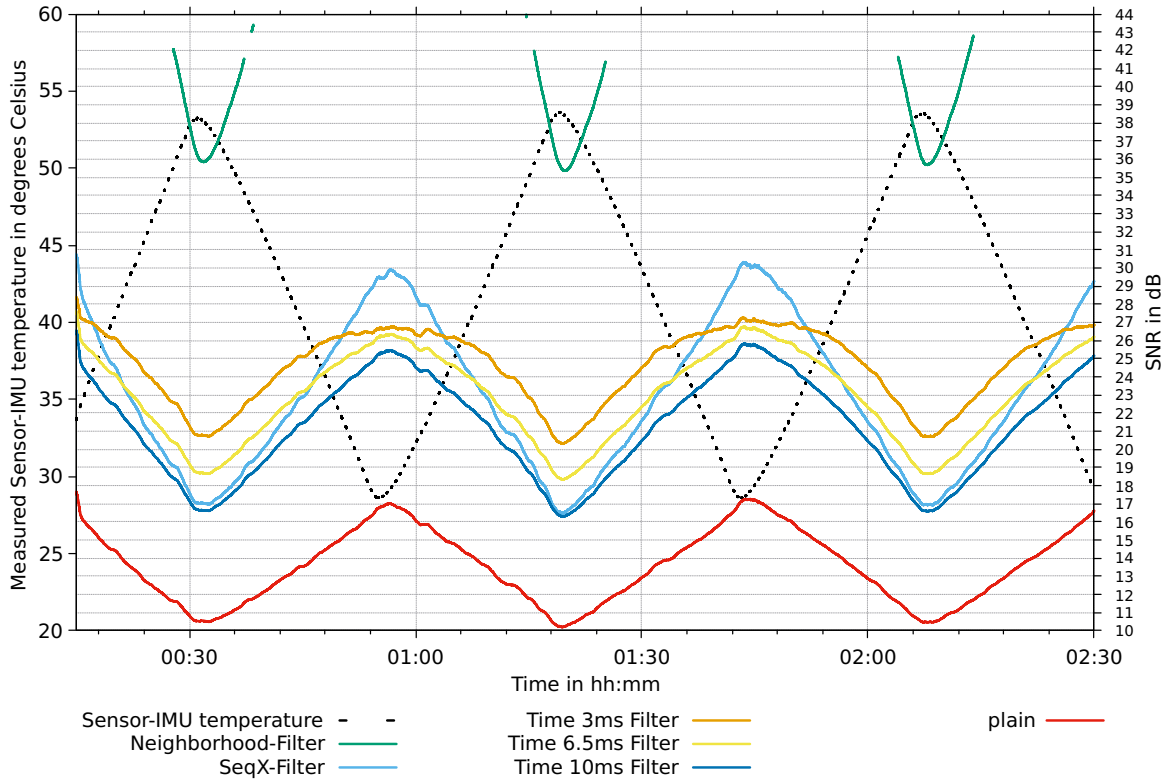
The experimental setup shown in Figure 3.7 and 3.8, which focuses on the different levels of transistor circuit noise that occur at different sensor temperatures, is continued here. The effects of varying sensor temperatures and spatio-temporal filtering are evaluated by classical signal-to-noise (SNR) measurements:

$$\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}} \tag{3.1}$$

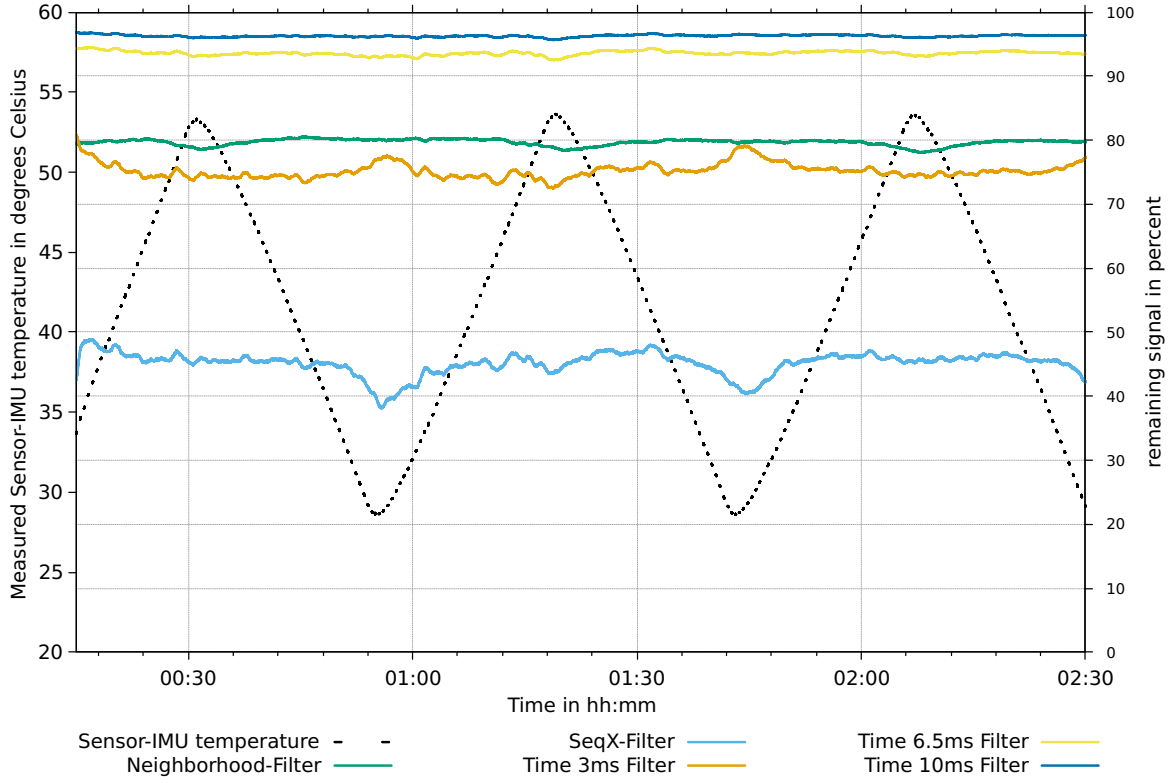
The counts of triggered and segmented events, normalized by the number of pixels in the corresponding segmentation masks, are considered as powers P of the signal or noise, respectively. The resulting SNR values, with and without applying the presented spatio-temporal filtering as postprocessing, are shown in Figure 3.9a.

As expected, the resulting SNR is temperature dependent and can be improved by filtering. At the cost of removing signal components, it is possible to improve the SNR. Therefore, the remaining percentage of signal events must also be considered and is shown in Figure 3.9b.

For this experiment, it must be noted that the *Neighborhood-Filter* is often able to remove all triggered background activity events, which results in an undefined SNR and an incomplete curve in the figures. On the other hand, this filter also eliminates $\approx 20\,\%$ of the rotor-triggered signal events. The used *Time-Filter* preserves most of the signal events depending on the threshold. Even with a threshold value of x=10 ms, an improvement in SNR of about 8.5 dB is achieved, while preserving about $\approx 97\,\%$ of the signal events. Since the *EDnCNN* filter is computationally expensive and requires a learning phase, this filter was omitted for these temperature-correlated evaluations and is only considered in the next section.



(a) Signal-to-noise ratio



(b) Remaining *signal* events after spatio-temporal filtering

Figure 3.9: Effects of spatio-temporal filtering at different sensor temperatures. The underlying experimental setup is shown in Figure 3.7.

Comparison in Real-World Application Scenario

The results of this section have previously been published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE

In addition to the experiments with an artificially induced signal at different sensor temperatures, further experiments were performed. These were performed in the real application context at the measurement site. They focused on the investigation of different signal events triggered by different classes of objects at approximately constant sensor temperature.

Based on these data, we also compare the performance that can be achieved with the presented spatio-temporal filters. For a fair comparison, the EDnCNN approach was trained on the corresponding data. Similar to [Padala et al., 2018], here we report the **P**ercentage of **R**emaining **E**vents (PRE) after filtering. However, we additionally consider the effect of filtering on individual object classes:

$$\mathrm{PRE}_{\mathrm{c}}^{\mathrm{f}} = \frac{\mathrm{E}_{\mathrm{c}}^{\mathrm{f}}}{\mathrm{E}_{\mathrm{c}}} \cdot 100 \tag{3.2}$$

where c denotes the considered object class, f denotes to the applied filter method, and E denotes the number of events corresponding to the selection.

Figure 3.11 summarizes the resulting distribution of PRE filter scores for different object classes. In general, these spatio-temporal filters can significantly reduce background activity within this application setup. However, the proposed filters differ in their results with respect to individual object classes.

The *EDnCNN* achieves good denoising results on both background activity and induced global effects such as shadows and ego motion. This approach also preserves a relatively high proportion of events caused by classes related to the outdoor measurement, such as rain or included tree crowns. The *Neighborhood Filter*, on the other hand, achieves a higher level of reduction on these classes such as `RAIN`, but also removes a higher proportion of events caused by other object classes.

After the *EDnCNN* filter, the *Time-Filter* retains most of the events caused by objects of interest such as `PERSON`, `DOG`, `BICYCLE` or `SPORTSBALL`, depending on its threshold. At the same time, about half or more of the other events caused by background activity or other "unwanted" classes are removed. The *SeqX-Filter* performs worst in comparison on almost all of these object classes of interest.

The average PRE results per filter remain comparable to the overall view when different object sizes resulting from different perspective distances are considered separately. Only the variation within the results decreases with increasing object size. Example results for the class `PERSON` are given in Figure 3.10, where the sensor rows shown correspond to Figure 3.3.

Summary

Background activity was stabilized to a constant level, while SNR was improved with an external active sensor temperature control system to support subsequent high-level processing approaches (see Appendix A). Different spatio-temporal filters were evaluated to further improve the signal-to-noise ratio.

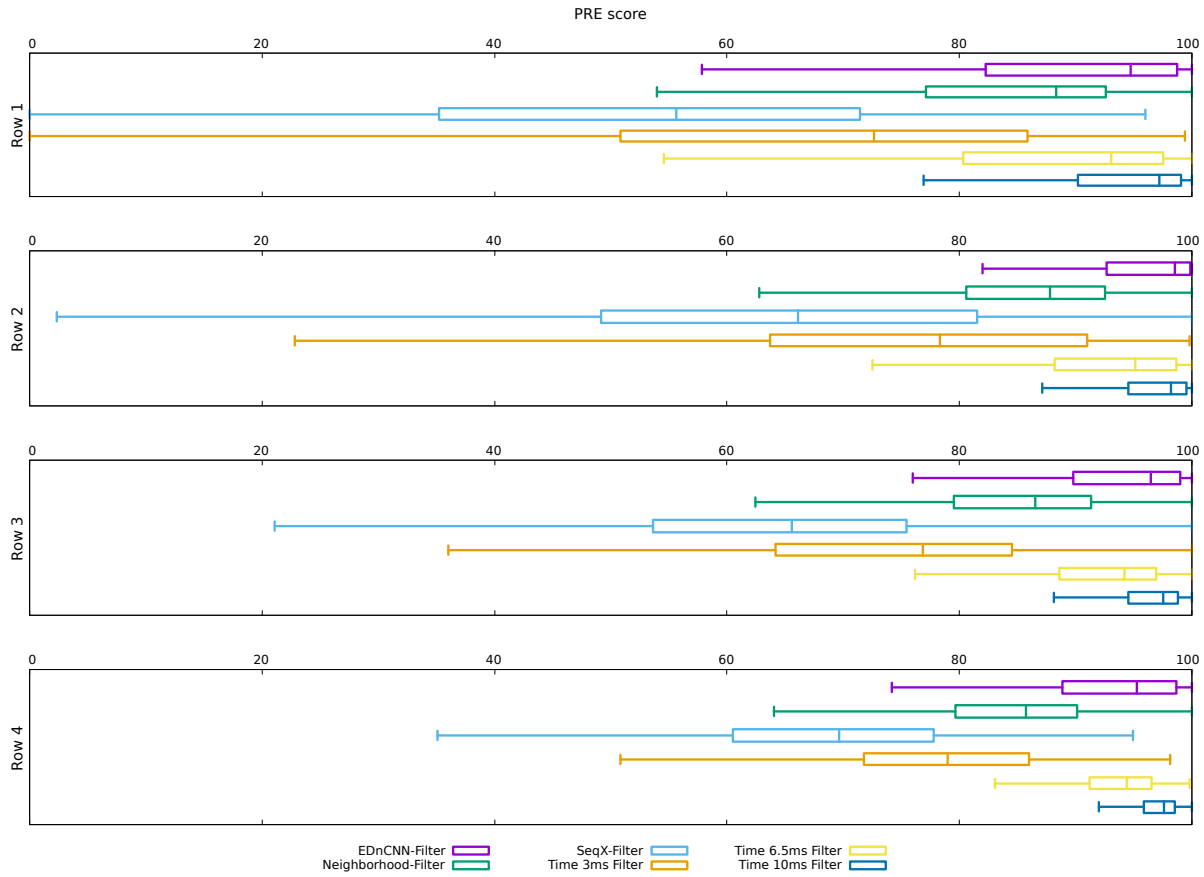


Figure 3.10: PRE scores for object class `PERSON`. Results are split by sensor row blocks (see Figure 3.3) to distinguish different object sizes created by perspective (adapted from [**Bolten** et al., 2021]).

In the measurement scenario of this work, due to the off-grid system setup and the central processing of three DVS streams on a single computer system, event streams must be filtered as simply and efficiently as possible for noise reduction and downstream system optimization. Therefore, a trade-off between filter performance and runtime requirements must be made. In general, a filter should be able to

- remove as many background activity events as possible, and
- preserve as many events caused by moving objects as possible.

The *Neighborhood-Filter* removes a high percentage of desired events through a restrictive filtering. The runtime requirements of *EDnCNN* are high. Because it is too computationally expensive, it is not suitable for the desired application.

Therefore, the *Time-Filter* was selected for further preprocessing in this work, as it preserves a high proportion of desired object events while achieving a significant reduction in background activity. The remaining noise components are further addressed in subsequent processing steps.

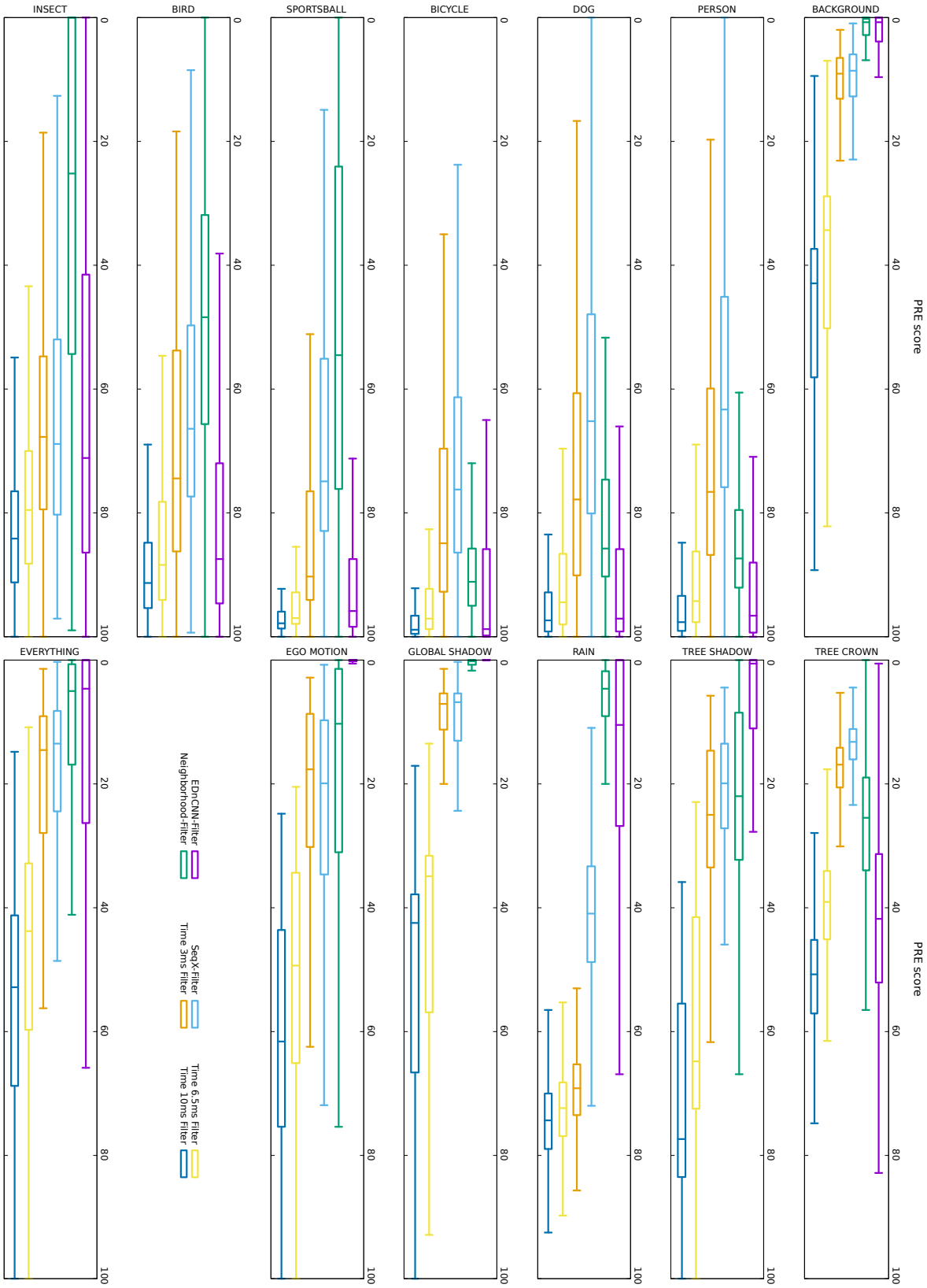


Figure 3.11: PRE scores for different object classes (adapted from [**Bolten** et al., 2021]).

Part II

Contributed Datasets

Contents

List of Figures

List of Tables

Datasets: State-of-the-Art

The state-of-the-art description in this dissertation chapter is an extended version of the previously published summary of related work given in the following papers:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE

Bolten, T., Neumann, C., Pohle-Fröhlich, R., and Tönnies, K. (2023a). N-MuPeTS: Event Camera Dataset for Multi-Person Tracking and Instance Segmentation. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 290 – 300. INSTICC, SciTePress

Datasets

Compared to frame-based image processing, the amount of freely available DVS-based datasets is considerably smaller. One reason for this is that this type of technology is relatively new compared to traditional image sensors. However, a steady development and publication of various datasets is taking place due to ongoing work and research in the field of neuromorphic computer vision.

Conversion and Simulation

Several earlier developments are based on the straightforward adaptation of image-based datasets to the event domain.

In [Orchard et al., 2015], the event-based pendants N-MNIST and N-Caltech101 of the well-known frame-based datasets ([Lecun et al., 1998; Fei-Fei et al., 2004]) were generated by simulating small eye movements called saccades. For this simulation, individual images of the original datasets were displayed on an LCD monitor and a DVS pointed at the monitor was moved accordingly. In [Serrano-Gotarredona and Linares-Barranco, 2015], a similar approach was taken. The only difference is that the displayed image was moved instead of the DVS. In order to easily take advantage of the large frame-based datasets available, such as ImageNet [Russakovsky et al., 2015], even more recent developments [Kim et al., 2021] are still based on such recording setups. Video sequences have also been converted to event datasets in similar setups [Hu et al., 2016].

The output of these conversion techniques is technically limited by the source material. For example, the high dynamic range of a Dynamic Vision Sensor cannot be fully exploited because it is not included in the input frame data. The frame rate of the source material is another limiting factor when considering video conversion in this setup. The temporal resolution of a

Dynamic Vision Sensor is significantly higher than the sampling frequency of commonly used conventional camera systems. As a result, DVS recordings converted in this way are limited in their properties.

An algorithmic conversion of frame-based data into DVS events is one way to address the limitations. Several methods and software frameworks have been developed for this purpose [García et al., 2016; Bi and Andreopoulos, 2017]. In [Hu et al., 2021], a framework called "v2e" (video to event) was presented, which incorporates a DVS pixel model including temporal noise, leak events, finite bandwidth as well as a Gaussian threshold distribution into the conversion. This approach uses a deep learning-based synthetic super-slow motion frame interpolation technique on the input video material to address the aforementioned time resolution limitation.

Another basic approach to the creation of DVS datasets is the full simulation of the scene [Rebecq et al., 2018]. A simulated DVS event stream with high spatial and temporal resolution can be derived based on a 3D modeled scene and defined object/camera motions in combination with an ideally photorealistic image rendering pipeline.

However, sensor-specific properties must be taken into account during both the conversion and the full simulation. These include the background activity behavior or the pixel refractory period. For some sensor models, such as the CeleX-IV sensor, this information is not publicly disclosed by the sensor manufacturer. Therefore, these parameters would have to be determined experimentally.

Native DVS-Recordings

Numerous neuromorphic datasets that are available to the public are composed of only short sequences of specific actions or patterns. In addition, these recordings were often made with well-aligned sensors in well-constrained and controlled laboratory environments.

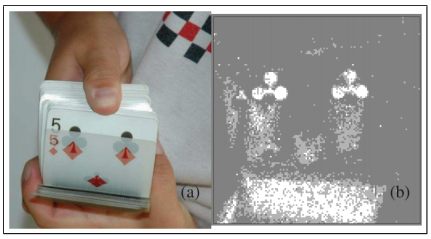
The following examples are provided to illustrate these characteristics:

POKER-DVS:
This dataset [Pérez-Carrasco et al., 2013] consists of recordings of a card deck that is fully browsed in front of the sensor. The deck contains ten numbered pip cards and could be fully scanned in less than 1 second. This recording setup is illustrated by the Figure II.1a.

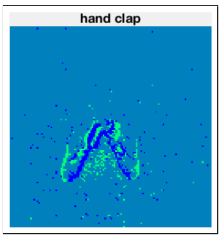
DVS128 Gesture:
This dataset [Amir et al., 2017] contains recordings of eleven different hand gestures performed by different people. Each gesture lasts about 6 seconds. An example is given in Figure II.1b.

Sign Language:
The "ASL-DVS" dataset [Bi et al., 2019] contains 24 letters from the American Sign Language alphabet and was recorded in an environment with low environmental noise and constant illumination. Each included sample has a duration of approximately 100 ms. The "SL-ANIMALS-DVS" dataset [Vasudevan et al., 2020] includes recordings of 19 different sign language gestures in isolation, recorded from a distance of 2 to 2.5 meters, so that the entire upper body is included. Each sign has an average duration between 3 and 6 seconds. An example is shown in Figure II.1c.

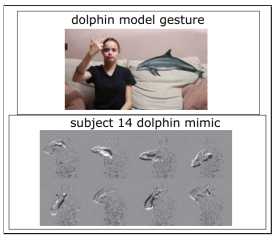
ATIS-Plane:
This dataset [Afshar et al., 2019] contains recordings of four different airplane models that were dropped in front of the sensor and crossed its field of view. In this scenario,



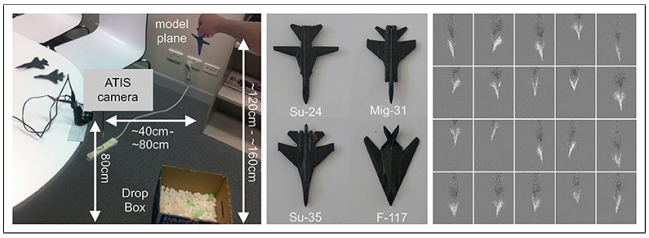
(a) POKER-DVS (from [Pérez-Carrasco et al., 2013])



(b) DVS-Gesture (from [Amir et al., 2017])



(c) SL-ANIMALS-DVS (from [Vasudevan et al., 2020])



(d) ATIS-Plane (from [Afshar et al., 2019])

Figure II.1: Examples taken from well-constrained native DVS datasets.

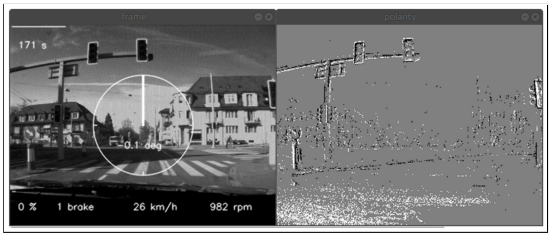


Figure II.2: Example of a DVS dataset recorded in an outdoor driving scenario (from [Binas et al., 2017]).

the duration of each sample is approximately 250 ms in length. Figure II.1d illustrates this dataset.

As mentioned above, these datasets are characterized by well-constrained scenarios. In addition, there are a number of datasets available that were created under more complex recording conditions.

Application Scenario: Monitoring of Public Outdoor Spaces

Existing datasets created under these more complex recording conditions often originate from the application areas of robotics (e.g., [Delmerico et al., 2019]) or autonomous driving. The datasets from the field of autonomous driving are the most relevant, considering the monitoring scenario of this work. This is because they consist of longer recordings taken under complex and real-world outdoor conditions, often in urban environments.

One example is the "N-CARS" dataset [Sironi et al., 2018]. This is a large, real-world, event-based dataset for vehicle classification. It consists of about 12,000 vehicle samples and almost as many background samples. The "GEN1" dataset [De Tournemire et al., 2020] also contains recordings from this context, but additionally provides annotations for the object classes `PERSON` and `CAR`. While the "DET" dataset [Cheng et al., 2019] only provides annotations for lane markings, "DDD17" and its successor "DDD20" [Binas et al., 2017; Hu et al., 2020] do not provide annotations at the level of objects, but only at the level of parameters of the

corresponding vehicle. The "DSEC" dataset [Gehrig et al., 2021], another large automotive dataset, provides synchronized stereo event camera recordings and includes ground truth depth information.

Although these datasets provide several hours of real-world outdoor footage, they are not readily applicable due to annotations that are unusable in our application context. However, due to the nature of recording from a moving vehicle, there is also a fundamental difference in the sensor signal due to the included ego motion compared to the static setup of the measurement system in this work (see Figure II.2). This makes it difficult to use these datasets in the desired application context.

In addition, there are some prior works and datasets that are directly related to parts of this dissertation. These works deal with the tasks of person detection and tracking.

In [Jiang et al., 2019], an approach for person detection in traffic monitoring applications was presented. For this purpose, a custom dataset was used which has a total length of only ≈ 14 seconds. In a very similar application context, a person detection was also implemented in [Chen et al., 2019a]. This was followed by the publication of a dataset in [Miao et al., 2019]. The detection part of this dataset consists of only twelve short sequences, with an average length of about 30 seconds, and provides only bounding box labels for the two classes of `BACKGROUND` and `PERSON`. Unfortunately, such small datasets are not suitable for current deep learning-based approaches. In [Ojeda et al., 2020] and [Bisulco et al., 2020] two approaches for filtering the event stream were presented with the goal of implementing person detection close to hardware. For this, a DVS dataset consisting of several hundred sequences was used. However, this dataset is not publicly available.

The task of multi-person tracking was addressed in [Piątkowska et al., 2012]. Due to the lack of available datasets with appropriate ground truth annotations, their approach was only tested on a very limited set of manually annotated data. With the work of [Hu et al., 2016], DVS benchmark datasets have been published. These benchmark datasets include parts that address both object detection and tracking. Person detection was also addressed [Boretti et al., 2023]. However, these cases only consider single object or single class scenarios, which is not sufficient for monitoring public areas.

In summary, there is currently only a very limited set of event-based datasets available that are suitable for the desired application scenario. Furthermore, the label annotations provided by these datasets are not sufficiently discriminative. Object labels are often given only as bounding boxes. This applies even to large datasets like the GEN1 automobile dataset. In the following parts of this dissertation, the tasks of semantic and instance segmentation are considered. Therefore, databases containing these annotations are required. Unfortunately, providing annotations at this level of detail is not very common at the moment.

An exception is the work of [Alonso and Murillo, 2019], which provides an extension for parts of the DDD17 dataset in the form of automatically derived semantic labels. Due to processing limitations, the provided labels are of variable quality (compare with Section E.3 in the Appendix). An extension of the DSEC dataset in the form of semantic labels is given in [Sun et al., 2022], while [Chaney et al., 2023] provides a multi-sensor based dataset including semantic labels. However, these recordings were made with a moving sensor, resulting in the limitations described above.

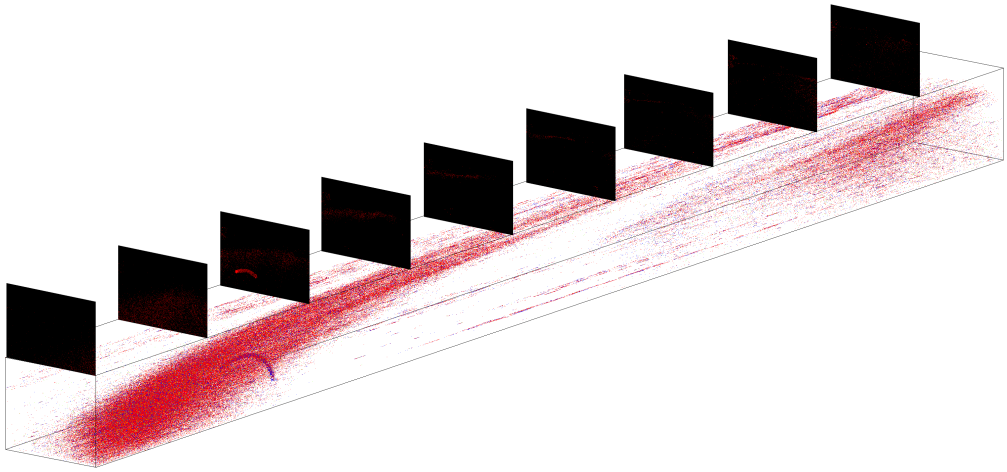
A dataset with manually annotated instance-level masks is given in [Huang et al., 2024]. This dataset was recorded in a cluttered indoor environment, while the sensor was mounted on a moving robotic arm. Currently, there are no publicly available datasets with instance-level annotation for monitoring applications.

One way to address this problem is to convert frame-based datasets to the event domain, as

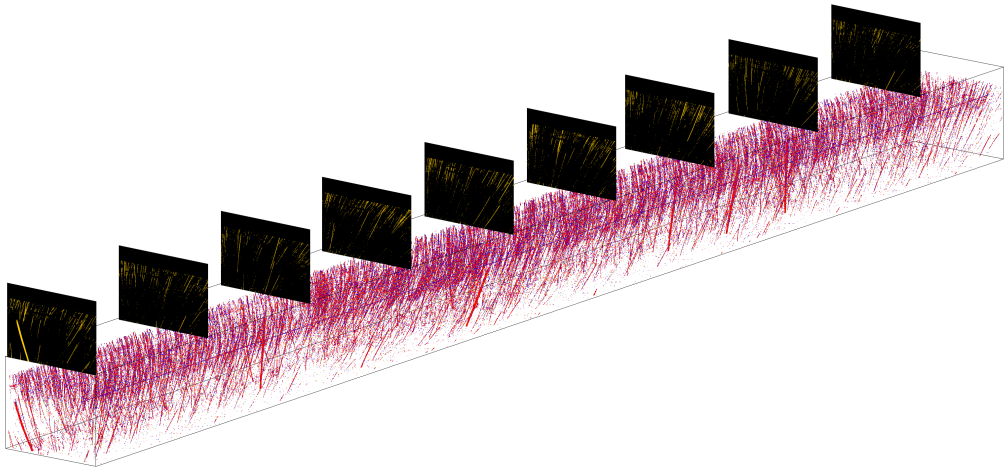
mentioned above. In conventional frame-based computer vision, datasets exist that provide annotations beyond bounding boxes for tracking and segmentation [Voigtlaender et al., 2019]. In the context of CCTV video surveillance, frame-based datasets typically consist of recordings taken at low frame rates ([Pranav et al., 2020] at 3 fps, [Jin et al., 2018] at 15 fps, or [Oh et al., 2011] at 30 fps). This low temporal resolution in the footage presents a hurdle in terms of the temporal resolution of a DVS during a conversion, even when slow motion interpolation is applied.

In addition, current DVS databases fail to consider the environmental influences embedded in the real sensor signal of outdoor recordings. Due to the high sensitivity in detecting brightness changes per sensor pixel and the temporal resolution of the event stream, the Dynamic Vision Sensor is able to detect fine details. This includes small airborne particles or objects such as raindrops or insects. In addition, global brightness changes, e.g. due to clouds are also detected. Figure II.3 shows an example of these included artifacts. These artifacts can also only be considered to a limited extent in a conversion or simulation.

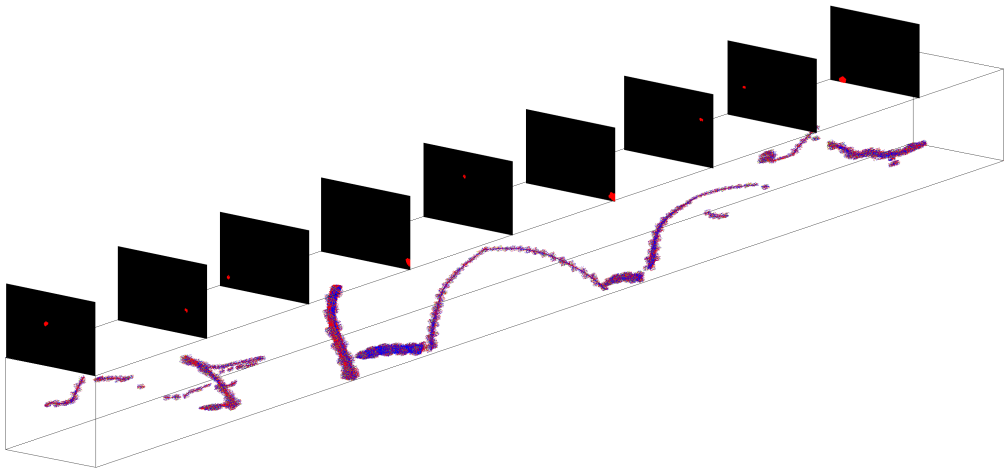
Therefore, in the course of this work, two own datasets were created and made available to the scientific community. These datasets, called DVS-OUTLAB and N-MuPeTS, address the use case of outdoor monitoring. They provide annotations on semantic or instance level for each DVS event. These datasets are introduced in detail in the following chapters.



(a) Global brightness changes induced by cloud drift



(b) DVS events caused by raindrops falling in front of the sensor



(c) DVS events caused by flying insects in front of the sensor

Figure II.3: Visualization of environmental influences captured in outdoor Dynamic Vision Sensor recordings (filtered for display).

Chapter 4

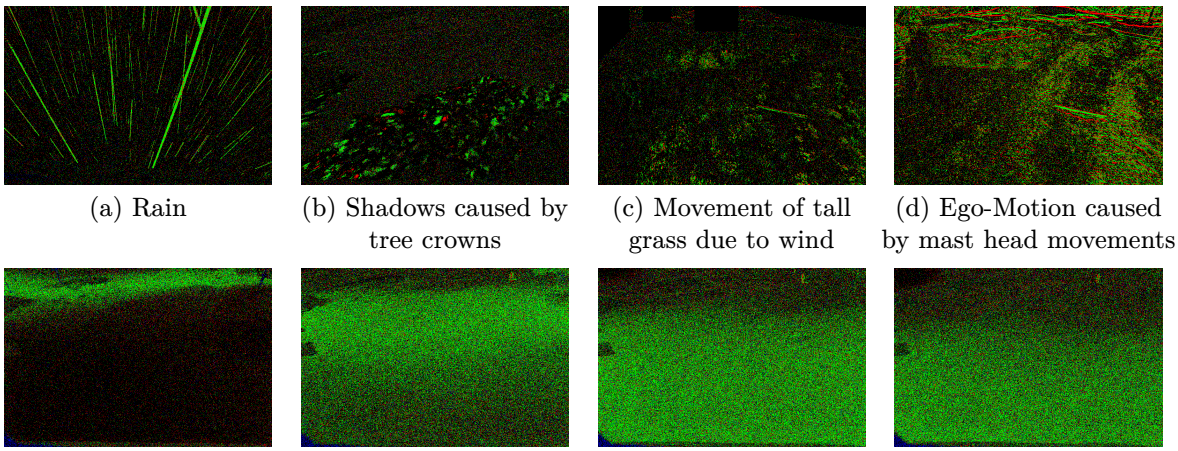
DVS-OUTLAB: Long-Term Monitoring in Outdoor Scenarios

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE

The lack of available labeled datasets is one of the main obstacles for the development of DVS-based applications. For outdoor applications, the operating scheme of a Dynamic Vision Sensor offers particularly advantageous characteristics. While there are a number of annotated datasets for tasks such as action or anomaly detection (e.g., [Jin et al., 2018; Oh et al., 2011; Pranav et al., 2020]) or traffic flow analysis (e.g., [Luo et al., 2018; Snyder and Do, 2019]) in frame-based video surveillance, there is currently a lack of event-based datasets for outdoor surveillance.

The DVS-OUTLAB dataset addresses this shortcoming. It is a dataset processed directly from real-world DVS-based recordings of a public open space. In this way, the behavior of real sensors as well as environmental influences are directly taken into account. For the generation of semantic segmentation annotations, a semi-automated annotation toolchain was developed. The resulting selection of nearly 50,000 labeled regions of interest from nearly seven hours of real-world event data is described in the following.



(a) Rain (b) Shadows caused by tree crowns (c) Movement of tall grass due to wind (d) Ego-Motion caused by mast head movements

(e) Global illumination changes caused by clouds shown at four different timestamps

Figure 4.1: Visualization of environmental influences caused by the outdoor scenario. Each pixel encodes the polarity of the last occurred event within a time window of 60 ms. An increase in brightness is displayed in green and a decrease in red (adapted from [**Bolten** et al., 2021]).

4.1 Dataset Composition

In the context of a long-term stationary monitoring scenario, a dataset is required

(a) that was recorded with a fixed-mounted sensor,

(b) includes challenging illuminations, sensor noise, and environmental influences (compare to Figure II.3 or Figure 4.1) as they occur in real-world outdoor recordings, and

(c) provides object labels for classes of interest in the monitoring scenario at a level of detail sufficient for semantic segmentation.

In addition, large datasets are often required in machine learning, especially when using deep learning techniques. Only these datasets allow the development of reliable and comparable algorithms or complete processing pipelines. Therefore, an additional requirement for the dataset is

(d) that it is large enough to support the training of deep learning approaches.

4.1.1 Recording Setup

The dataset consists of selected and further processed recordings from multiple Dynamic Vision Sensors mounted at a public children's playground as part of the Living-Lab introduced in Section 3.1. A schematic plan of the monitored playground and the system components used to create the dataset is shown in Figure 3.1 on page 22. An illustration of the scene and typical projected object sizes caused by different object distances to the sensor due to perspective are shown in Figure 4.2.

It was not possible to use additional sensors in the setup due to very strict privacy laws regarding the surveillance of public spaces and the high expectation of privacy of potential users. This is especially true for the use case of a children's playground. Therefore, the measurement system used to create this dataset was not extended with classic RGB frame-based CCTV components.

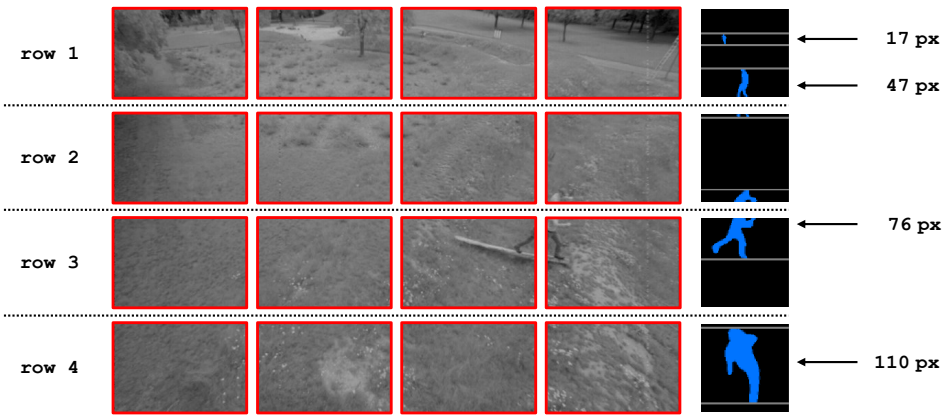


Figure 4.2: Grayscale image of the field of view of DVS1. Typical projected object heights for an adult person are indicated. Sensor row blocks are shown and later processed regions are marked in red (adapted from [**Bolten** et al., 2021]).

4.1.2 Included Scenes

The dataset contains data from two semantically different recording sessions of the same area. The first session contains only staged scenes. The second session is composed of environmental influences captured in the real long-term monitoring.

Staged Scenes

In order to generate a large and annotated dataset, typical application scenarios were staged at the site. Several actors, including children, actively participated in this process. Their explicit consent allowed the activation of a *special* sensor operation mode.

Unlike many other event cameras, the CeleX-IV is able to output the absolute brightness value for each triggered DVS event in the output stream, depending on the selected sensor operation mode. This allows the accumulation of classic 2D grayscale images. Due to the aforementioned privacy concerns, the use of this sensor mode and the acquisition of these grayscale values was only possible while the area was closed to the public and with the explicit consent of the actors involved. It was ensured that this specific sensor operation mode was not used during the long-term monitoring.

In this way, several hours of material with activity were recorded. This material was then semi-automatically annotated with object labels based on the captured grayscale information. This will be described in more detail later.

Environmental Influences

In addition, environmental artifacts are also included in the dataset. This was done by manually extracting parts of real long-term recordings from the same location. In this way, different lighting and weather conditions are incorporated in the data.

4.1.3 Included Challenges

Although this is a stationary sensor application, the real-world scenario and outdoor environment introduce data characteristics that can challenge the development of computer vision solutions. The following included challenges can be highlighted:

Background Noise:

The output of currently available Dynamic Vision Sensors contains noise (see also Sec-

tion 3.3). The CeleX-IV sensor used has a relatively high noise level, which makes processing a challenge.

Object Sizes and Classes:
The low spatial resolution of the CeleX-IV DVS sensor compared to state-of-the-art frame-based systems, in combination with the wide-angle lens used and the size of the monitored area, results in small object sizes. Figure 4.2 illustrates this using the example of an adult person. The size of the object varies greatly depending on the distance from the sensor.

The dataset also covers a multi-class labeling scenario. This multi-class scenario, combined with the small projected object sizes and the embedded noise, poses a serious challenge for automated processing.

Environmental Influences:
Especially in outdoor scenarios, Dynamic Vision Sensors tend to capture various types of environmental influences due to their high sensitivity and temporal resolution (see Figure 4.1). These influences occur both locally (e.g., shadows cast by tree canopies) and globally in the scene (e.g., rain). Subsequent processing is challenging due to these artifacts.

4.2 Provided Dataset

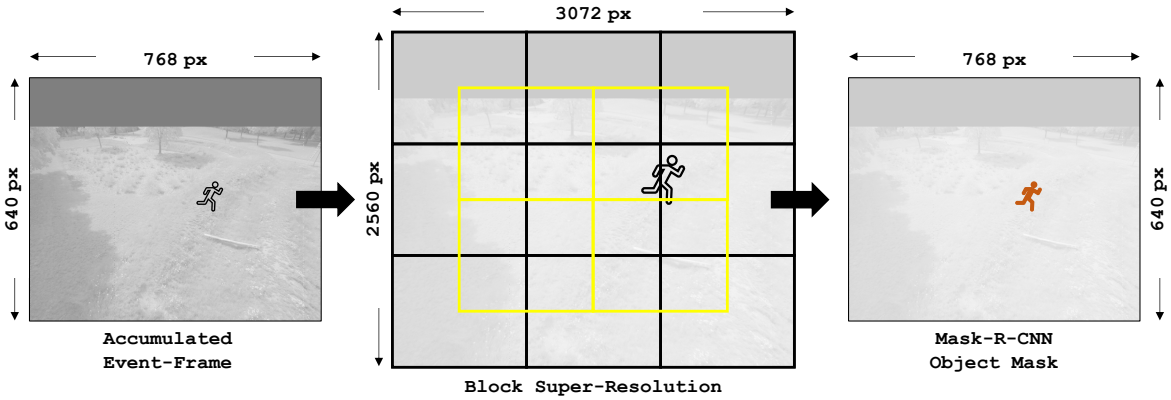
As stated before, the published dataset contains data from two semantically different recording scenarios of the same area. The labeling process for both parts, the composition performed, and the resulting statistics for the published data are described below.

4.2.1 Label Generation

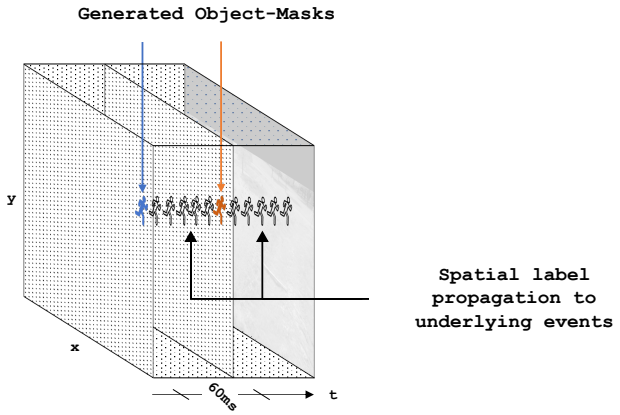
In the context of the specially staged scenes, it was possible to calculate grayscale images from the brightness values of the CeleX-IV sensor data. Using these images, generated at 60 ms time intervals, well-known frame-based convolutional neural network (CNN) methods were used to generate semantic object label proposals. For this purpose, an implementation [Abdulla, 2017] of the Mask R-CNN object detector [He et al., 2017] was used. This detector was pre-trained on the COCO database [Lin et al., 2014], which is a large-scale RGB dataset for object detection and segmentation. The classes `PERSON`, `DOG`, `BICYCLE`, and `SPORTSBALL` were selected from the set of predefined COCO classes as objects of interest in the context of the application scenario.

To acquire the object labels, a deep learning super-resolution network [Lim et al., 2017] was first applied to scale the input images by a factor of four. This was done to mitigate the effects of the sensor's low spatial resolution while increasing the scale of the objects in the captured grayscale images. The scaled images were divided into 13 blocks so that the width of each block corresponded to the expected input size of 1024 pixels of the Mask R-CNN network used. This procedure is shown in Figure 4.3a. The middle blocks, highlighted in yellow in this figure, are used to improve the detection of objects split into multiple parts by processing overlapping input regions in the center of the image. For each block, the result of the Mask R-CNN inference was calculated. The detections were combined into a single mask image of the size of the original DVS image. Different labels at the same position were resolved based on a priority list.

Based on these object masks, the generated labels were propagated back to the events of the corresponding 60 ms time segment of the original event stream. In this step, objects that did not move in the corresponding time windows were filtered out, since they are visible in the



(a) Mask R-CNN-based label generation on accumulated grayscale images



(b) Label propagation from 2D object masks to DVS event stream

Figure 4.3: Visualization of label generation and propagation on staged dataset scenes (adapted from [**Bolten** et al., 2021]).

Sensor	Number of time windows recorded	Number of time windows containing label	Percentage of time windows containing label
DVS1	131,492	98,853	75.18
DVS2	131,488	52,520	39.94
DVS3	131,488	76,337	58.06
Total	394,468 ≈ 6 h 30 min	227,710 ≈ 3 h 45 min	57.73

Table 4.1: Number of time windows and segments with automatic label suggestions for staged scenes (TimeWindow $\hat{=}$ 60 ms).

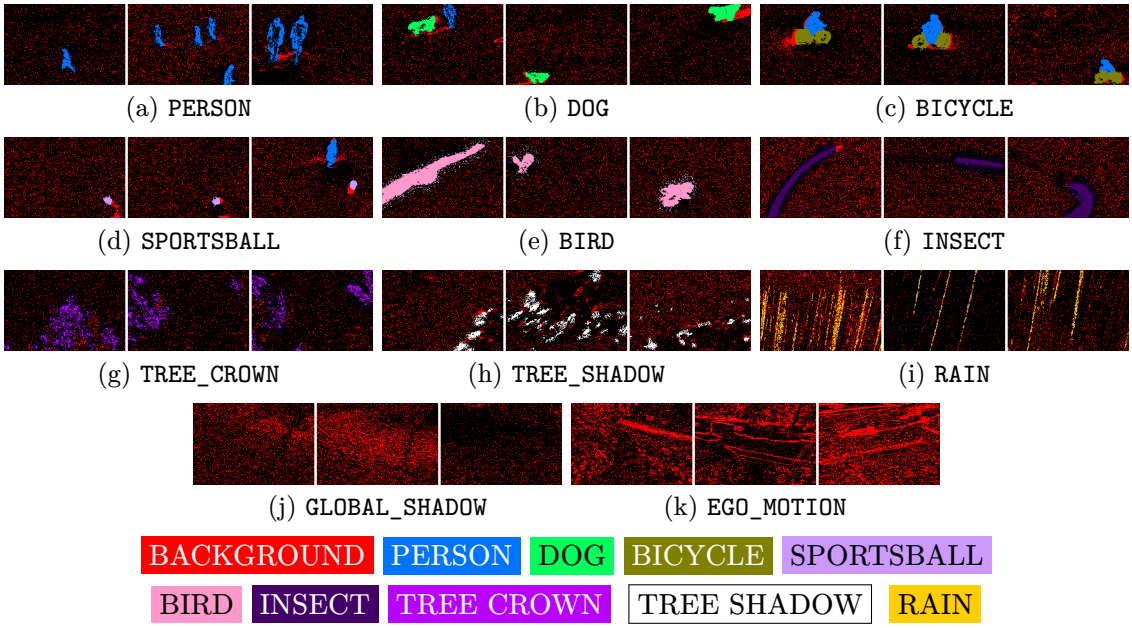


Figure 4.4: Sample snippets of the labeled 192 × 128 pixel regions of interest in DVS-OUTLAB (adapted from [**Bolten** et al., 2021]).

accumulated grayscale images but do not have a significant number of events associated with them in the DVS stream.

In addition, segments containing environmental effects were manually selected, annotated, and included from the performed real-world and long-term recordings. As a result, the dataset contains realistic effects of different environmental influences and weather conditions.

4.2.2 Dataset Statistics

The total length and number of temporal segments containing automatically generated label proposals from the Mask R-CNN pipeline are summarized in Table 4.1. Since in almost all staged scenes the actors entered the measurement field near DVS1, there is a bias in the ratio of detections between the different sensors. Also, as almost all staged scenes involve a person, there is a very large imbalance in the frequency of class occurrences.

For this reason, only a subset of the available data was used. In the process of this subselection, the following two aspects were taken into account.

	Class label	Number of time windows containing label	Number of time windows containing label in row 1	row 2	row 3	row 4
Objects of interest	PERSON	7399	2833	2678	1255	633
	DOG	709	351	259	54	45
	BICYCLE	4378	2023	1834	478	43
	SPORTSBALL	500	147	244	74	35
Environmental interferences	BIRD	3807	825	1353	1173	456
	INSECT	5939	842	1329	1532	2236
	TREE_CROWN	6375	1731	2511	1576	557
	TREE_SHADOW	6901	50	1660	2387	2804
	RAIN	7052	1776	1776	1750	1750
	GLOBAL_SHADOW	4800	1200	1200	1200	1200
	EGO_MOTION	4800	1200	1200	1200	1200

Table 4.2: Performed subselection from all labeled event data. Each row represents a 128 pixel sensor block on the Y-axis, starting to count at the top (see Figure 4.2). (TimeWindow $\hat{=}$ 60 ms).

Object Sizes:
In order to equalize the distribution of object sizes resulting from different object-to-sensor distances, a sensor row-block guided subselection was performed. These row blocks are illustrated in Figure 4.2 and correspond directly to the logical CeleX-IV sensor row blocks used for data readout (see Figure 3.3 on page 24). An equal number of examples per class and sensor row block were randomly selected if available in the original data.

Label Quality:
In addition, the label quality of the Mask R-CNN predictions for the selected data was taken into account. The labeling was manually reviewed to ensure that only acceptable predictions were included in the final selection.

The number of resulting and labeled event time windows is shown in Table 4.2. As mentioned above, a subselection based on CeleX-IV row blocks was performed to harmonize the distribution of the included object sizes. Therefore, the annotated data is also divided into regions with a height of 128 pixels. This is due to the fact that the available sensor resolution of 512 pixels is divided into four rows.

During the manual quality check of the Mask R-CNN labels, individual regions were checked. These regions are each 192 pixels wide with respect to the labels propagated to the DVS. This results in final annotations that are patched into regions of 192×128 pixels each. The spatial boundaries of these patches are highlighted by the red boxes in Figure 4.2. Fully processed and selected regions are shown in Figure 4.4.

A 70/15/15 % split is also provided with the dataset, which can be used for training, validation, and testing of further applications. This split also takes into account the distribution across the sensor row blocks and thus different object sizes.

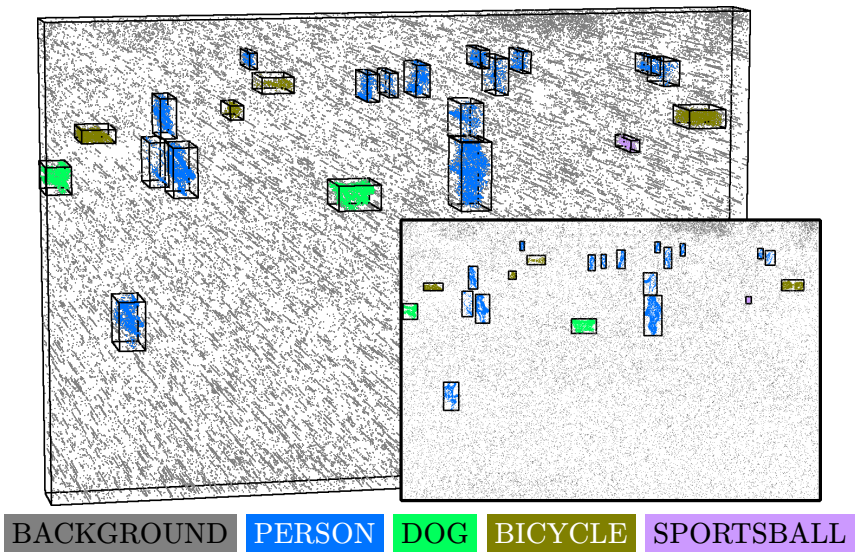


Figure 4.5: DVS-iOUTLAB dataset example scene
(a 60 ms time window with 95,726 events is displayed; from [**Bolten** et al., 2024]).

4.3 Instance Augmentation: DVS-iOUTLAB

In order to derive a further semantic segmentation, the entire raw recordings of the staged scenes were also processed using a point-based segmentation method, which is described in more detail in Chapter 6.

Instances were then manually selected, checked, and extracted from the semantic segmentations. Both the newly obtained segmentation results and the already labeled sections of DVS-OUTLAB were used. In this manual process, a total of 52,293 instances of `PERSON`, 3,649 instances of `DOG`, 7,024 instances of `BICYCLE`, and 3,134 instances of `SPORTSBALL` were extracted, covering all sensor row blocks. Instances of objects that were split at the input data patch boundaries were excluded.

From this pool, randomly selected instances were used to artificially populate new challenging scenes, while directly containing *instance* annotations. The selected objects were spatially moved during the scene creation process to perform data augmentation. The selected objects were randomly moved along the x-axis of the sensor's field of view. The y-coordinates of the events were not changed in order to preserve the projected object sizes, which would otherwise have had to be changed due to perspective. In this way, new scenes were created with objects in close proximity to each other, while avoiding real object occlusions based on the convex hulls of the objects.

In this augmentation process, 10,000 scenes were created, divided into 8,000 for training and 1,000 each for test and validation. Real background noise from the CeleX-IV sensor, recorded from an empty scene, served as the basis for the empty scenes to be filled. Each of the created scenes includes between three and 32 objects, with a range of 3–24 persons (with an average of 8.76), as well as a maximum of 2 dogs, 4 bicycles, and 2 sportsballs. Instead of smaller patches, as used in DVS-OUTLAB itself, the scenes created here cover the *entire* spatial sensor resolution of 768×512 pixels.

A sample scene of the newly created instance dataset named DVS-iOUTLAB is shown in Figure 4.5. In summary, this augmented version of the previous dataset contains challenges from a multi-class, multi-instance segmentation scenario combined with real sensor noise.

Chapter 5

N-MuPeTS: Instance Segmentation and Tracking

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Neumann, C., Pohle-Fröhlich, R., and Tönnies, K. (2023a). N-MuPeTS: Event Camera Dataset for Multi-Person Tracking and Instance Segmentation. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 290 – 300. INSTICC, SciTePress

In order to take full advantage of event-based sensors, it is essential to have access to datasets with high quality and discriminative labels for research and development purposes. The N-MuPeTS dataset addresses this need by providing high-quality instance segmentation labels for each individual DVS event. The dataset has a total duration of over 85 minutes and covers several movement scenarios with one to four people, addressing the challenges of multi-person segmentation and tracking scenarios.

A detailed description of the hardware setup and the corresponding software processing pipeline used to create the dataset is presented. This setup allowed the acquisition of multi-person DVS recordings with high-quality per-event instance labels. Statistics on the created ready-to-use multi-person DVS dataset N-MuPeTS are also provided.

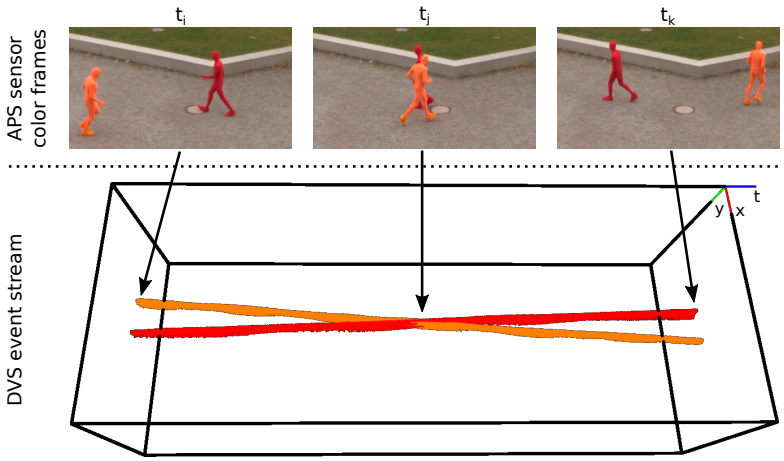


Figure 5.1: Concept visualization for segmentation-based DVS tracking.
The high time resolution supports continuous tracking approaches by using high-quality segmentations (from [**Bolten** et al., 2023a]).

Dataset Requirements

In the context of multi-object tracking (MOT), a dataset should include the following common challenges [Islam et al., 2015; Xu et al., 2019; Luo et al., 2021]:

1. object occlusions (through infrastructure as well as by other persons in the scene)
2. similar appearance and body shape of recorded persons
3. changes in pose and movement patterns (e.g., kneeling, standing, walking, and running)
4. interactions among multiple persons (including abrupt changes in movement direction and speed)
5. objects in different sizes

5.1 Label Extraction

A meaningful dataset requires accurate labels. In the case of instance labeling of event data, manual annotation is not efficient. This problem was solved by introducing additional information in a way that minimally affects the original data.

Dynamic Vision Sensors with color filter matrices must usually be considered as prototypes (e.g., [Berner and Delbruck, 2011, "cDVS"], [Li et al., 2015, "C-DAVIS"], [Moeys et al., 2018, "SDAVIS192"]). They are not commercially available and do not offer the higher spatial resolutions of newer DVS models. Commercially available is the "DAVIS346C" sensor[9], which only offers a spatial resolution of 346 × 260 pixels [Taverni et al., 2018].

Thus, for practical purposes Dynamic Vision Sensors are not capable of recording color information. This circumstance was exploited to encode the information of which event belongs to a particular person in the color of the person's clothing itself. In order to record the color information, a second frame-based color camera, referred to as an active pixel sensor (APS) camera, is required.

Assuming proper recordings, label data with high accuracy can be extracted from the color frames. A detailed description of the used hardware setup is given in Section 5.2.

[9] `https://shop.inivation.com/collections/davis346`

5.1.1 Color Features

The color features were generated by single-colored full-body skin-tight garments. The following specifications are important for this:

1. A single homogeneous color per suit and different suit colors per actor are required because person instances are separated by color hue.

2. The suit must cover the entire body. This ensures that color information is available for all event data triggered by a person.

3. The suit should be skin-tight so that the silhouette of a person is not larger than normal.

Dynamic Vision Sensors are commonly based on CMOS technology. CMOS image sensors are also usually sensitive to near-infrared light (NIR).

Synthetic fabrics tend to be strongly NIR-reflective. The spectral response of the garment is dominated by the reflected NIR light, regardless of its visible color. As a result, DVS events can be triggered for any color selected. The single-color garments also provide enough contrast under natural lighting to trigger events normally.

Initial experiments showed that fabrics can be distinguished by color hue even when the separation in hue is small (see Figure 5.5 and the description in Section 5.3.2). This enables the use of many colors and therefore many person instances can be distinguished simultaneously.

However, there are multiples sources of color artifacts that must be considered:

1. The lens introduces chromatic aberrations, i.e. phantom colors towards the edges of the image area, that can only be partially corrected.

2. The camera records color information through a color filter matrix. Only the three colors, red, green, and blue, are recorded. The real source color is a mixture of these. The colors must be reconstructed during demosaicing. During this process errors are introduced near borders because neighboring color signals could be combined while the sources were separated in reality. This also introduces new colors that are not part of the real scene. Since the proposed approach is based on the separation of people by color hue, erroneous color information hinders the processing.

3. A third source of errors is the on-chip *binning* process of the camera sensor itself. The effect that *binning* has on resulting colors is similar to the previous point. By aggregating 2×2 pixels it is possible to get phantom colors not included in the real scene.

It is also important to avoid overexposure in each color channel. Single-colored suits can quickly lead to overexposure in a single channel. It is recommended to deliberately underexpose the scene so that no required information is lost.

5.1.2 Environmental Influences

Data acquisition in an outdoor environment is naturally influenced by the environment itself. One important aspect is the lighting of the scene. Direct sunlight is not preferred because the color of the direct sunlight itself is different from the color of diffusely reflected light. In consequence, the parts in direct sunlight will need a different white balance than parts in the shadows. The information about which parts are exposed to direct sunlight is not known precisely, so the necessary distinction cannot be made. Overcast is therefore preferred for its diffuse scene lightning effect.

Airborne particles are another practical issue. Acquisition during precipitation is not useful because the sensitivity in detecting changes in brightness and the temporal resolution of a DVS

are high enough to image the raindrops themselves. Wet scenes should be avoided due to the possibility of unwanted reflections and apparent color shift of wet surfaces of some materials. Therefore, dry weather without large airborne particles is preferred. This includes also insects and pollen, as these environmental influences would also be included in the DVS signal.

5.2 Hardware Setup

There are several aspects and requirements for hardware setup to accurately process event and color information. This includes the selection of the sensors used, as well as their mounting in a stereoscopic setup to optimize the perspective mapping. Furthermore, the colored garments used to incorporate the color features must also be taken into account.

5.2.1 Sensors

Like [Marcireau et al., 2018], the proposed approach is also based on exploiting color features within the scene. Compared to their hardware setup, which consists of three Dynamic Vision Sensors and an optical beam splitter, a simpler setup was used here, which does not impose any constraints on the recorded spectrum.

A single CeleX-IV DVS was directly connected to a notebook via USB 3, which acquired the plain event stream and stored it on an external solid-state drive. Experiments have shown that event noise is strongly dependent on sensor temperature (see also Section 3.3). To reduce the included noise, the sensor operating temperature was stabilized at a low level using a cooling system based on Peltier elements. The cooling is based on the system discussed in Appendix A, although the measurement setup used here omits the enclosure, as it was used only temporarily.

In common video codecs, such as H.264 [Richardson, 2010], color information is compressed. Initial experiments have shown that these values are insufficient for the intended hue segmentation. Therefore, we aim to collect the APS color data in a lossless manner. The processing requirements are minimal frame loss, maximum image resolution, full color information, and maximum frame rate.

The Sony IMX477 CMOS image sensor [Sony, 2018] formed the basis of the APS camera used in the hardware setup. It is possible to capture a lossless *raw* video stream from the IMX477 sensor. The video stream features a resolution of 4056×3040 pixels with 12-bit per pixel. The resulting data rates are a challenge for data acquisition applications.

There were several limitations here:

- The camera interface on the APS sensor board is not capable of transmitting the full resolution in uncompressed form at the desired frame rate. Thus, *binning* was enabled, which in turn halves the image dimensions to 2028×1520 pixels.
- The USB bus capacity of a single computer is exceeded by the combined data rates of the APS, DVS, and storage on external USB media. Therefore, a dedicated Raspberry Pi 4 was used to acquire the APS frames and store them on another external solid-state drive.

The lens selection was made to match the field of view of the sensors as closely as possible. The IMX477 was combined with a 4 mm wide-angle lens from Edmund Optics[10]

Both sensors together form a stereoscopic camera. The cameras were stacked vertically as parallax is less apparent along the vertical axis with cameras looking at the scene from

[10]Edmund Optics TECHSPEC® UC Series #33-300, f=4 mm, f/1.8 – f/11, 1/2 inch, C-Mount.

(a) CeleX-IV with cooler (bottom) and IMX477 with Raspberry Pi (top)

(b) Full-body garment and cotton shoe cover

Figure 5.2: Sensor and color garment setup (from [**Bolten** et al., 2023a]).

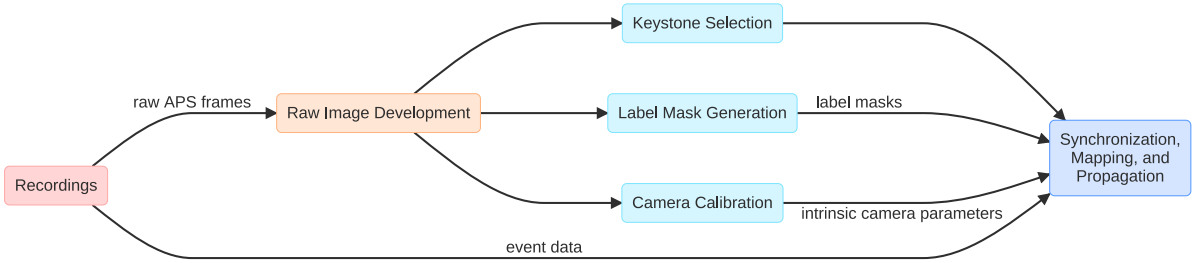


Figure 5.3: Overview of software pipeline used for N-MuPeTS dataset generation (from [**Bolten** et al., 2023a]).

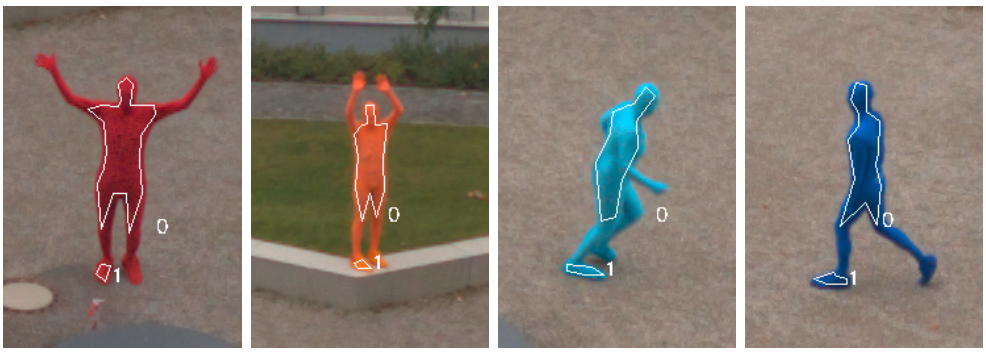
an elevated position. Also, the lenses were mounted as close to each other as possible. The physical mount was milled from aluminum alloy and placed on a solid surface so that the likelihood of tripod shake triggering unwanted events was minimized. The optical axes were adjusted towards the same point in the center of the scene so that lateral overlap of the resulting views is maximal. The resulting sensor hardware setup is shown in Figure 5.2a.

5.2.2 Color Garments

The full-body garment was made up of two parts. The entire body, including the head, hands, and feet, was covered by a "Morphsuit" – a commercially available Spandex-based suit. In order to guarantee a natural walking pattern, all persons were allowed to wear everyday shoes. The shoes were covered using separate covers sewed from colored cotton fabric. An example of a complete garment is depicted in Figure 5.2b.

5.3 Software Pipeline

The software processing pipeline for generating the label masks and propagating them to the DVS data consists of several steps, which are explained in the following. An overview of the process is shown in Figure 5.3.



(a) RED (b) ORANGE (c) CYAN (d) BLUE

Figure 5.4: Example of one-time manually annotated colored regions for clustering (from [**Bolten** et al., 2023a]).

5.3.1 Raw Image Development

The frame-based APS recordings from the IMX477 sensor were captured as lossless as possible to minimize color artifacts and inaccuracies, especially in hue. This means that a full digital photo development process had to be applied to obtain proper color images from the raw data.

In order to minimize the required data bandwidth of the sensor, the acquisition was performed in a packed data format. For every two pixels, their 12-bit intensities were combined into three bytes. After unpacking the individual pixels from this structure, a black level correction was performed and a color image was generated by using an edge-aware demosaicing algorithm on the Bayer-filtered data. After clipping and normalizing the data, a color correction matrix was applied to transform the sensor-specific color values into a standardized, device-independent color space. Finally, a gamma correction was applied. The resulting image was cropped to a resolution of 2000×1500 pixels to remove non-informative stride pixels and color artifacts from the edges of the image.

In a second step, chromatic aberrations were corrected. Finally, the tone curve and white balance were optimized. This was done by adjusting the image using an open source photo editor[11].

5.3.2 Label Mask Generation

The label masks were automatically generated on the basis of the developed APS frames by means of color hue segmentation. Each recorded image was processed independently.

First, regions of interest were marked by hand (see Figure 5.4). Areas of different illumination should be included. This is a countermeasure against the influences described in Section 5.1.2. Conversion to HSV color space results in multiple hues per garment color selection. These clusters can be visualized in a polar pixel histogram with 360 bins (see Figure 5.5). These clusters were further processed to give a single centroid in hue for each garment color.

For this purpose, an agglomerative clustering was computed on the hue values. Clusters with differences in hue below a given threshold were merged. This threshold was increased as long as the number of resulting clusters was lower than the number of garment colors used during recording. It is assumed that a normal distribution is suited to model potential color variations, e.g. due to lighting. A normal distribution was fit to each cluster. The calculated mean hue of each cluster was used for the hue segmentation in the next step. The hue range

[11] `https://www.darktable.org/`

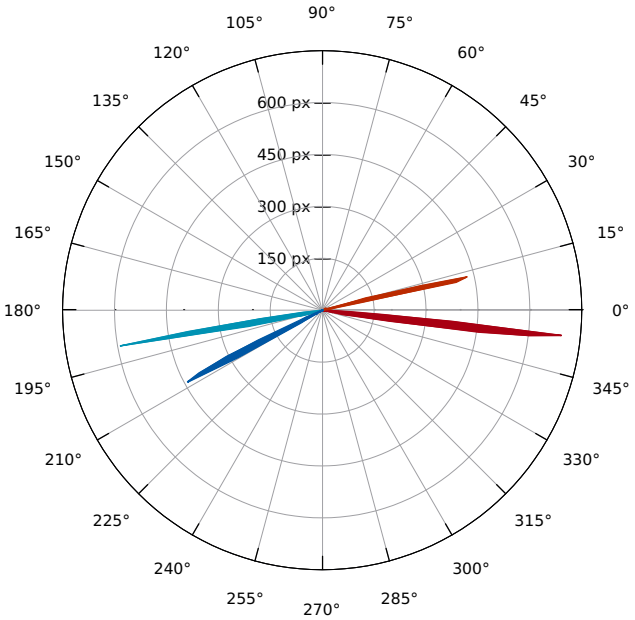


Figure 5.5: Polar histogram of garment colors extracted from annotations on developed APS frames shown in Figure 5.4 (adapted from [**Bolten** et al., 2023a]).

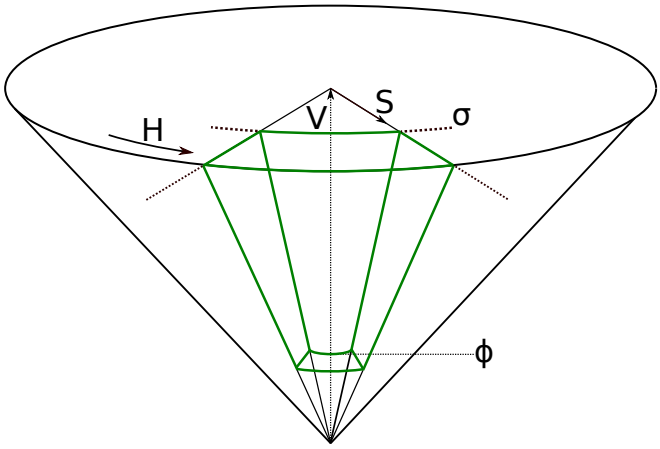
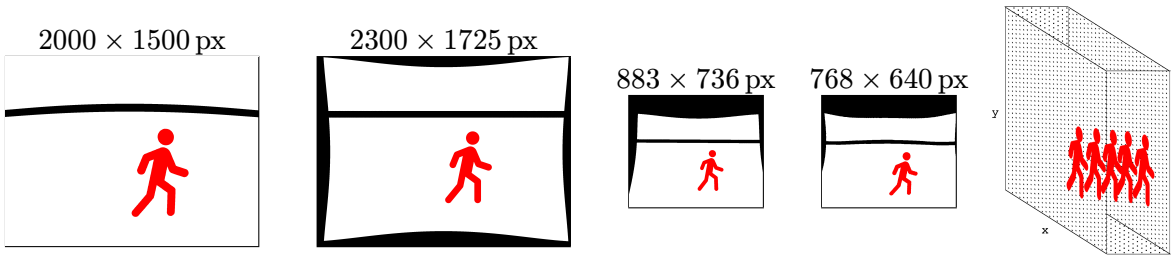


Figure 5.6: HSV-dependent decision region for segmentation.

in HSV color space is circular. Care must be taken to use circular mean for all computations involving hue.

The hue segmentation was computed by calculating the differences to the centroids in hue of each cluster and then applying a threshold α. At this point, a circular sector is selected for each color in the HSV color space. We can further narrow the decision regions by incorporating saturation and value. The idea is to exclude all colors that cannot origin from the color garment in the given setup. We removed all pixels with a saturation $\leq \sigma$ and a value $\leq \varphi$. The resulting decision region is exemplarily shown in Figure 5.6. The thresholds must be selected so that as little as possible from the regions of interest is cut away. At last, small holes were closed by a morphological operation. This process resulted in a separate binary mask for each garment color.

Additional care was taken to prevent multiple labels per pixel and to remove false positives. When the resulting label mask was set for multiple colors at one position only one color was kept. Centroids' hue values were used in ascending order to define priorities. Most



(a) Generated label mask with lens distortion

(b) Label mask with lens correction applied

(c) Undistorted label mask mapped to DVS

(d) Redistorted label to match plain DVS data

(e) Propagated DVS event labels

Figure 5.7: Processing steps to map and propagate label masks to DVS event stream (from [**Bolten** et al., 2023a]).

false positives were removed by applying a connected component analysis and suppressing all connected components with a size of less than τ.

5.3.3 Synchronization, Mapping, and Propagation

Several steps are necessary to project the obtained hue-based label masks onto the DVS event stream. A priori, intrinsic camera parameters must be estimated for both sensors of the setup. By using a slowly moving chessboard and applying corner detection, the points needed for calibration were extracted.

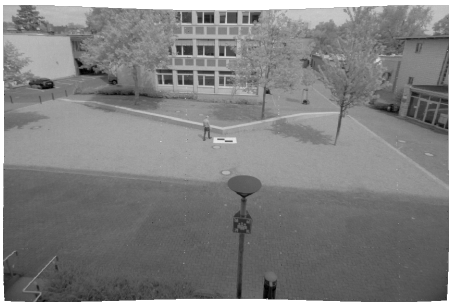
It is important to temporally synchronize the views so that the label masks derived from the APS camera match the DVS view. The recording of raw data from the APS color sensor is technically limited to 40 fps due to the hardware setup that was used[12]. For further processing, the continuous data stream of the DVS was therefore also divided into sections of the length of 25 ms. The resulting APS frames and the time windows of the DVS were then synchronized based on system clock timestamps corresponding to their acquisition time. The clocks of the used systems were adjusted via NTP[13]. This limits the error over time to a small value. In addition, an initial synchronization was performed manually at the beginning of each recording using a visual cue. This visual cue was used to compensate for most of the absolute error between the system timestamps.

For mapping, the generated label mask was first undistorted with the determined camera parameters of the APS. This process step is illustrated exemplarily in Figure 5.7a and b. The mask was then projected onto the DVS, changing the field of view and image resolution (see Figure 5.7c). For this purpose, a homography was determined based on manually selected points on the ground plane of the scene (compare to Figure 5.8). This homography was then used to warp the mask in perspective.

Finally, the redistorted instance labels (see Figure 5.7d) were propagated to the plain DVS event stream. All events within a synchronized time window were assigned the label provided by the mask at each corresponding spatial position. The end result is a per-event instance label annotation on the DVS output stream (see Figure 5.7e).

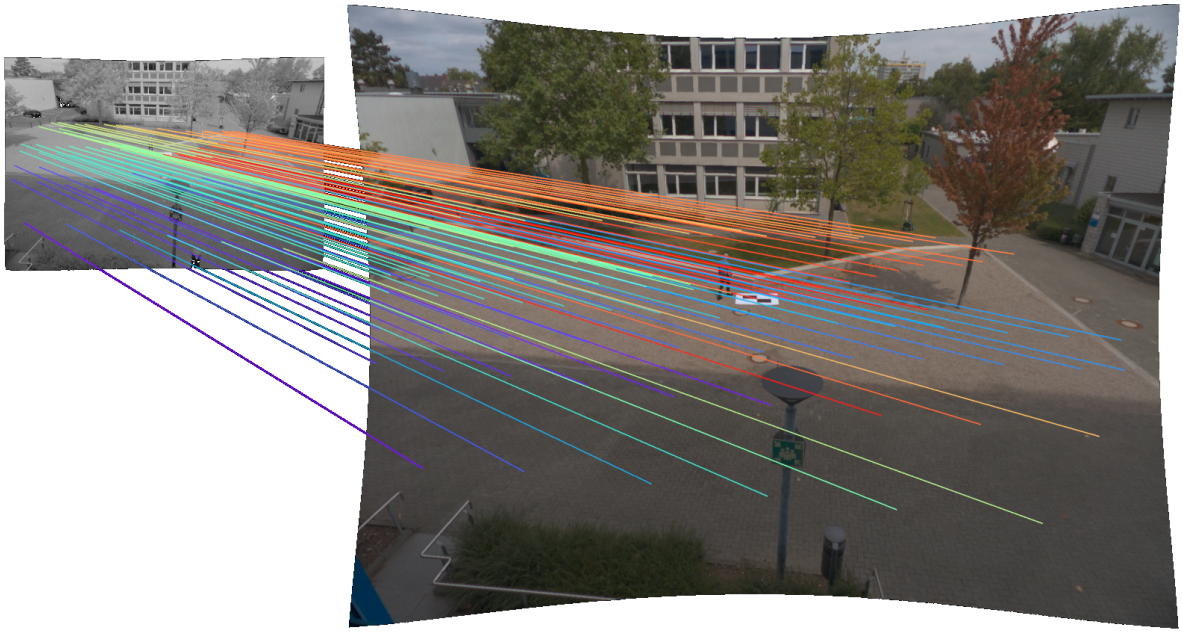
[12]Using the 12-bit packed data format at a sampling rate of 40 fps results in an approximate data rate of 178 MB/s which must be continuously transferred and stored.

[13]`https://tools.ietf.org/html/rfc5905`



(a) DVS field of view

(b) APS field of view



(c) Selection of corresponding ground key points between the DVS (left) and APS (right) fields of view by means of a large visual cue

(d) Perspective warp (blended)

Figure 5.8: Mapping of sensor outputs.

Limitations

There is one notable limitation. Label mask generation based on the APS frames works for both, moving and standing persons. In contrast, a DVS ideally generates event data only for regions with movement. In consequence, noise of the DVS is labeled as belonging to a person when the person is not moving.

The label is wrong when looking at the event data itself, because the events were not triggered by the persons themselves. But from the point of view of a tracking algorithm, the label is correct because a person cannot disappear in a scene. Therefore, we decided not to postprocess the label masks in this dataset, but to annotate whether a person is standing still.

5.4 Provided Dataset: N-MuPeTS

Neuromorphic-Multi-Person Tracking and Segmentation Dataset

First, an overview of what is included in the published dataset is given, and each annotation used is briefly explained. Then, several statistics derived from the dataset are discussed.

During recording in an unconstrained outdoor environment, interference can not be completely avoided. Any deviation from a perfect sequence of actions must be edited during postprocessing. We decided to mark the sequences with lower quality and split the recording at the point where the quality changes. This process was done manually to ensure high accuracy. We observed a number of issues, including both, recording artifacts and postprocessing errors. Based on the expected impact of these issues on the usability of segmentation and tracking applications, three classes of quality are distinguished. From worst to best, the quality classes are:

Quality 3 corresponds to these major issues:

- uninvolved persons in scene
- cars moving in scene
- wildlife in scene, i.e. birds

Quality 2 corresponds to these minor issues:

- unintended occlusion, i.e. tree trunks
- one or more persons are outside the static mask
- incomplete masks, i.e. intersection with static mask (see Section 5.4.3)

Quality 1 includes all remaining sequences, i.e. sequences without any of the aforementioned issues.

Quality 2 and 3 can be used to obtain longer sequences. In the following, quality 1 is presented exclusively. A sample scene of the dataset is shown in Figure 5.9.

5.4.1 Scenarios

We defined a series of scenarios as a protocol for recording and as a guide for the actors. Where possible, we repeated the scenario for each person and all possible numbers of persons. The scenarios are listed in Table 5.1. The order is mostly chronological and corresponding to sequence indices. For the majority of the scenarios, we introduced annotations with a corresponding name. The annotations are not exclusive for certain sequences, e.g. CROSSING will occur frequently during "crossing paths" but also anytime else. Thus, the scenarios increase the frequency of their corresponding annotations for their duration.

For the generation of label masks $\alpha = 10°$, $\sigma = 50\,\%$, $\varphi = 10\,\%$, and $\tau = 10\,\text{px}$ were used.

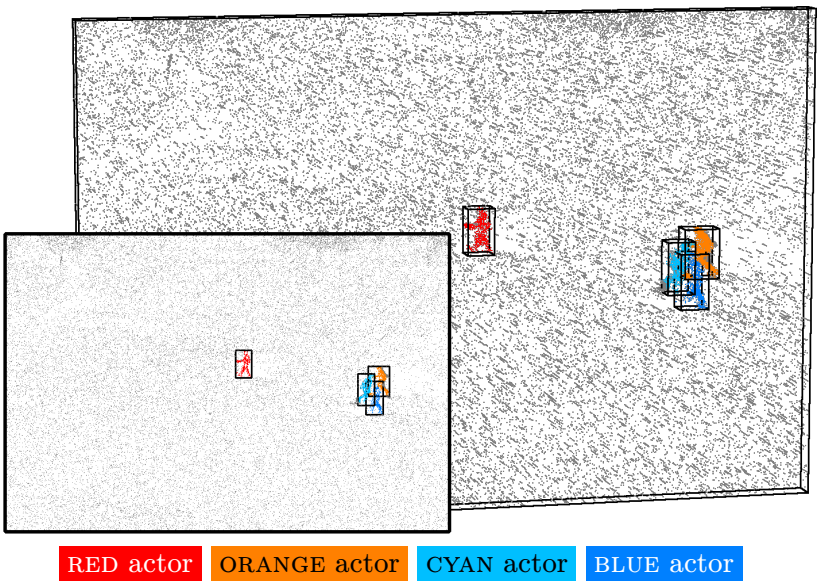
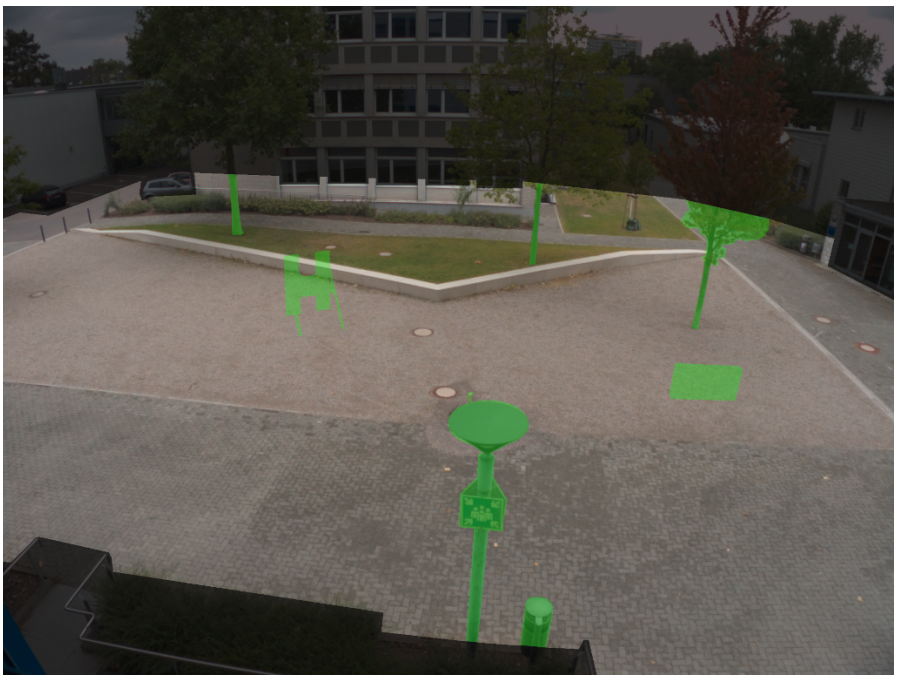


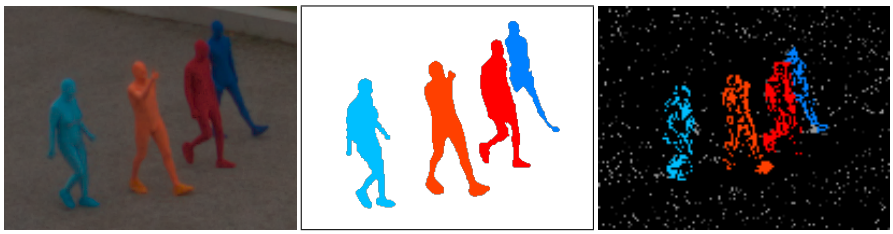
Figure 5.9: Example scene from N-MuPeTS dataset (55,072 events; adapated from [**Bolten et al., 2024**]).

Scenario	Primary parameter
background (empty scene)	–
single person	speed
crossing paths	angles of paths, speed
parallel paths (same directions)	speed, distance
parallel paths (opposing directions)	speed, distance
occlusions	speed, pose
meeting & parting	direction
random path	speed
helical path	–

Table 5.1: Considered scenarios used as guide to actors in dataset creation.



(a) Masked scene
(static mask blended in black, possible occlusion areas due to infrastructure are blended in green)



(b) All four actors (c) Generated label mask (d) Propagated DVS labels

Figure 5.10: Setup during N-MuPeTS dataset recording and corresponding labels (adapted from [**Bolten** et al., 2023a]).

5.4.2 Persons and Garment Colors

The recorded persons are pseudonymized in the data provided by using the color of the suit they were wearing (see Table 5.2 and Figure 5.10b). Some colors are less practical than others in a given environment. In our case, vegetation tended to be green-yellowish. Red and blue are well separated. The choice of orange and cyan, which are close in hue (see Figure 5.5), is due to the unavailability of different colored suits at the time the dataset was created. Figure 5.10c shows an automatically generated label mask in the used setup, while Figure 5.10d shows the result of corresponding mapping and propagation to the DVS view.

5.4.3 Static Mask

We used a binary mask to prevent vegetation as well as uninvolved persons and objects from causing false positives during label mask generation, as described under “Quality 3”. Some sequences had to be split and labeled as quality 2 instead of quality 1, because one or more person were obscured by the static mask.

This static mask was applied only to the color frames from the APS camera. In the labeled

event dataset, all event data is unmasked. Figure 5.10a shows the scene used to create the dataset. The areas outside the applied static mask are blended in black.

5.4.4 Annotations

Annotations were made manually and are available per color.

The scenario description, used to brief the actors, and the annotations, describing all the included activities, are directly related to the main challenges of MOT as defined in the introduction of this chapter. The mapping and annotations are discussed below.

Background:

One annotation not resembling a problem in itself is BACKGROUND, i.e. an empty scene. This provides separate recordings only including sensor noise. In quality 2, some foreign activity can occur.

Annotation key:

BACKGROUND no person with colored garment is in the scene.

Object Occlusion:

Object occlusion can be further divided into occlusion with infrastructure and occlusion between persons. In addition to occlusion with static infrastructure, such as tree trunks in the scene, occlusion is also possible with portable infrastructure that was temporarily placed in the scene during parts of the dataset creation. Example areas of possible occlusion due to infrastructure are highlighted in green in Figure 5.10a.

Occlusion between persons can happen whenever two or more persons are in the scene. It is very frequent during CROSSING and SIDEBYSIDE.

Annotation key:

OCCLUSION one or more persons are behind an obstacle.

Similar Shape:

All actors are comparable in size and weight. The person representing CYAN naturally has a different silhouette while the remaining persons have a very similar figure. This makes distinction by size impracticable and provides a greater challenge for tracking algorithms.

Annotation keys:

RED, ORANGE, CYAN, BLUE person with specified garment color is in scene.

Pose and Movement:

Most of the time the actors were upright and walking. For short durations they were in differing poses. EXCERCISING is a collection of movement patterns found in sports. This includes push-ups and burpees (including jumping). These represent movements that are rarely observed in pedestrian monitoring scenes with high rates of change in motion speed, and thus can be challenging for tracking algorithms.

STANDING, WALKING, and RUNNING correspond to three intervals of speed during bipedalism.

RANDOM is a special case where an actor tries to move as unpredictably as possible, while including changes in speed and course. This aims at stressing physical motion models included in many tracking algorithms.

Annotation keys:

EXERCISING one or more persons are doing exercise like activity.

KNEELING, STOOPED, WAVING one or more persons are in the specified pose.

STANDING, WALKING, RUNNING one or more persons are moving the specified manner.

RANDOM one or more persons are abruptly changing direction and speed, including backwards movement.

Interaction:

Crossing paths require two persons, but the number of possible variations is limited. The following patterns are considered relevant:

1. person A and person B walk past each other (CROSSING)
2. person A and person B walk in the same direction (SIDEBYSIDE)
3. person A or person B change their direction at the point of intersection (MEET)

The parameters are the angle to the sensor, the angle between the paths, and the speed. With three persons, the number of possible variations increases drastically. For this dataset we managed to include four people.

Annotation keys:

CROSSING the paths of two or more persons are crossing in temporal proximity.

MEET two or more persons are meeting and parting again, paths are converging then diverging.

SIDEBYSIDE two or more persons are moving side by side at the same speed.

HELIX two persons try to circle around each other so that in a (x, y, t)-diagram, i.e. a 3D plot of the resulting event point cloud, their paths resemble a helical path.

Different Sizes:

As mentioned before, the actors are of similar size. Still, the projected size varies greatly with distance from the camera due to perspective. While a person in the center of the scene has a projected size of approximately 52 pixels in the DVS event stream, a person at the far end of the scene has a projected size of only 29 pixels. Sequences with persons at the far end of the scene are marked with FAR.

In conclusion, persons are included in the dataset in a variety of sizes. Of course, the variation in size corresponds directly to the position along the vertical axis in the stream.

Annotation key:

FAR one or more persons are near the distant footway in the scene (compare with Figure 5.10a).

5.4.5 Statistics

The dataset subset that constitutes quality class 1 is summarized in Tables 5.3, 5.4, and 5.5. In these tables, all durations are given in seconds and rounded to the nearest integer. The Appendix (Section D.2) contains detailed information for all quality classes and annotations.

Garment color	Size	Sex	Weight
RED	1.84 m	male	70 kg
ORANGE	1.82 m	male	82 kg
CYAN	1.74 m	female	80 kg
BLUE	1.80 m	male	65 kg

Table 5.2: Technical specifications of participating persons.

RED	ORANGE	CYAN	BLUE	Cumulative duration per color combination	Cumulative duration sum per person count
					300
●				313	882
	●			158	
		●		215	
			●	195	
●	●			213	701
●		●		41	
●			●	125	
	●	●		236	
	●		●	76	
		●	●	9	
●	●	●		496	687
●	●		●	123	
●		●	●	51	
	●	●	●	17	
●	●	●	●	350	350

Table 5.3: Cumulative durations per color combination in quality class 1.

Annotation	Number of sequences	Mean duration	Cumulative duration
RED	137	12.5	1713
ORANGE	123	13.6	1671
CYAN	101	14.0	1416
BLUE	77	12.3	946
BACKGROUND	46	6.5	300
STANDING	96	3.3	312
WALKING	441	5.1	2259
RUNNING	107	4.7	504
RANDOM	18	8.9	160
HELIX	9	7.2	64
FAR	35	4.5	157

Table 5.4: Duration statistics per annotation in quality class 1.

Annotation	Number of occurences
OCCLUSION	136
EXERCISING	9
KNEELING	13
STOOPED	18
WAVING	9
CROSSING	212
MEET	48
SIDE BY SIDE	94

Table 5.5: Occurrence statistics per annotation in quality class 1.

Annotation	Train	Test	Validation
(a) Object occlusion			
OCCLUSION	57.40	31.54	11.06
(b) Pose & movement			
EXERCISING	61.96	30.05	7.99
WAVING	33.97	22.54	43.49
WALKING	67.57	18.87	13.55
RUNNING	69.41	17.14	13.45
RANDOM	85.13	9.62	5.25
(c) Object interaction			
CROSSING	64.37	22.48	13.15
MEET	66.40	23.26	10.34
SIDEBYSIDE	64.66	18.83	16.50
HELIX	62.15	10.80	27.05
(d) Object size			
FAR	60.87	20.72	18.41
(e) Over-all			
mean	63.08	20.53	16.39

Table 5.6: Percentage of dataset annotation labels in N-MuPeTS split (based on labels per actor given in 25 ms dataset time windows).

Overall, the cumulative duration of sequences in quality 1 is about 56 % of the total dataset, which equates to a duration of appropriately 2920 seconds. Table 5.3 summarizes the recorded durations for each color combination. It can be noted that all possible color combinations are included. The last column contains the cumulated durations for 0, 1, 2, 3, and 4 simultaneously active actors. Recordings with up to two or three persons each contribute 24 % to the total duration of the dataset. Scenes with one person ($\approx 30\,\%$) and with all four persons ($\approx 12\,\%$), along with empty background scenes, form the remaining components.

Table 5.4 gives an overview of the number of sequences and their duration in relation to the assigned annotations. The cumulative duration is included in a separate column for convenience. Due to the aforementioned rounding, small discrepancies can occur. Annotations that describe short events in time are counted instead, see Table 5.5.

Considering the cumulative duration each actor is present, RED, ORANGE and CYAN are included in equal shares. Actor BLUE is slightly less frequent. According to the natural movement pattern of humans, the annotation WALKING is included significantly more often than others.

5.4.6 Dataset Split

To develop segmentation-related applications, the following dataset splitting logic has been proposed.

For training and evaluation of segmentation purposes, all recordings with the best quality should be used, except for time windows in which at least one person is standing (dataset

annotations KNEELING, STOOPED or STANDING). This leads to the exclusion of ≈ 7.6 minutes of recording and is therefore negligible. This exclusion is necessary for segmentation applications, as standing persons are indistinguishable from background noise in the DVS signal.

The following procedure was used to generate a dataset split. The remaining recordings were divided into consecutive ten second segments. Based on these segments, the data was divided into training, validation, and test sets. This windowing of the data was done in order to achieve a higher variability between the splits compared to randomly sampling and selecting time windows of only a few milliseconds.

In selecting a 60/20/20 % split for training, validation and test, care was taken to ensure that the selected activity annotations of the dataset are approximately equally represented in the generated splits. The detailed distribution of annotations for each generated set is given in Table 5.6.

Summary of Part II

An overview of existing DVS datasets in the literature was given. This included native DVS datasets recorded with real sensors as well as datasets based on conversion or full simulation. The available datasets often have limitations

- in terms of their duration,
- regarding the orientation and distance of the included objects with respect to the sensor used,
- or do not address outdoor applications and thus do not consider environmental influences in the sensor signal.

Furthermore, they are often accompanied by label annotations that are inappropriate (or insufficiently discriminative) for outdoor monitoring applications. Therefore, two own datasets were recorded, processed and published.

The DVS-OUTLAB dataset provides several hours of raw event data from an actual long-term monitoring setup that was performed in a real-world outdoor scenario. The labels created for this dataset address various classical object classes, such as persons, as well as environmental influences (such as rain or shadows). The dataset consists of specially staged scenes and an automated process to derive labels on the basis of these recordings was described. It also includes manually selected recordings from long-term observations that contain environmental artifacts. In this way, about 47,000 regions of interest were processed, containing semantic segmentation labels for each individual DVS event.

The dataset includes challenges such as real sensor background noise, very different object sizes (including very small objects), and the aforementioned environmental effects that occur under different light and weather conditions.

The N-MuPeTS dataset provides longer, continuous and native DVS-based recordings covering multiple scenarios within a multi-person tracking and segmentation use case. The recordings for this dataset were also performed in an outdoor environment. For the actors recorded within these scenarios, high-quality instance labels were derived. In addition, the actors' poses, movements, and interactions within the sequences were also annotated. The scenarios played by the actors are directly derived from common challenges encountered in the context of multi-person tracking applications.

By exploiting color features within the scene and combining a Dynamic Vision Sensor with a classical RGB imager, highly accurate instance labels were generated. This is even the case in error-prone scenarios including intersecting movements and occlusions. The sensor hardware setup used and the complete software processing pipeline for automated derivation and propagation of multi-person instance labels on DVS recordings based on hue segmentation were described.

These datasets provide segmentation labels at the level of each individual DVS output event. Therefore, there are no event representation and encoding constraints to consider when us-

ing these datasets. This opens the possibility to compare any event encoding strategy for subsequent computer vision tasks. This includes approaches that operate directly on the three-dimensional (x, y, t) data of the DVS stream. The datasets were made freely available to support and accelerate the development and deployment of fully DVS-based processing pipelines in real-world usage scenarios.

Part III

Segmentation: It's About Event Representation

Contents

List of Figures

List of Tables

Segmentation: State-of-the-Art

The state-of-the-art description in this dissertation chapter is an extended version of the previously published summary of related work given in the following papers:

Bolten, T., Lentzen, F., Pohle-Fröhlich, R., and Tönnies, K. (2022a). Evaluation of Deep Learning based 3D-Point-Cloud Processing Techniques for Semantic Segmentation of Neuromorphic Vision Sensor Event-streams. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 179. INSTICC, SciTePress

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2023c). Semantic Segmentation on Neuromorphic Vision Sensor Event-Streams Using PointNet++ and UNet Based Processing Approaches. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 178. INSTICC, SciTePress

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2024). Instance Segmentation of Event Camera Streams in Outdoor Monitoring Scenarios. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 452 – 463. INSTICC, SciTePress

Levels of Detail in Segmentation

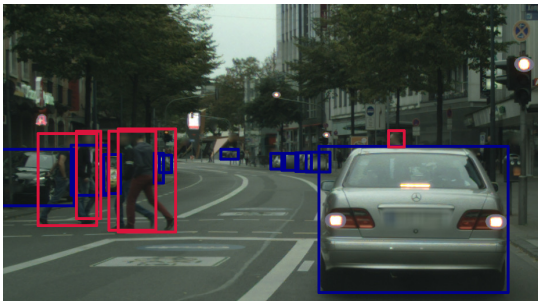
Segmentation is an important part of computer vision and is needed for a variety of scene understanding tasks. Different levels of detail can be distinguished for this task (see Figure III.1):

Object Detection:
The object detection task deals with the localization of objects and the assignment of class labels. Each detected object is usually represented by a bounding box.

Motion Segmentation:
Motion segmentation is the process of identifying and grouping regions of coherent motion. The main objective is to distinguish and separate areas or objects that are moving independently from the rest of the scene or from each other. Grouping in complex scenes may involve camera ego-motion, and therefore, separating foreground and background motion.

Semantic Segmentation:
In semantic segmentation, in addition to object detection, each input point or pixel is assigned a class label, making it a per-pixel classification task. By assigning labels, the entire input is divided into segments.

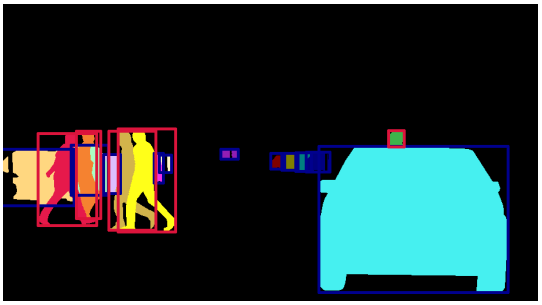
(a) Input image



(b) Detection (per-object bounding box and class label)

(c) Motion Segmentation

(d) Semantic Segmentation
(per-pixel class label)



(e) Instance Segmentation
(per-object mask and class label)

(f) Panoptic Segmentation
(per-pixel class and instance labels)

Figure III.1: Different levels of segmentation. The input image is taken from the "Cityscapes" dataset [Cordts et al., 2016] and has been cropped.

Instance Segmentation:

Instance segmentation combines semantic segmentation with object detection, with the goal to detect all objects and assign per-pixel class labels to them. In addition, a unique identifier is assigned to each of the input points of the detected objects. In contrast to semantic segmentation, this makes it possible to distinguish between different objects of the same class.

Panoptic Segmentation:

Panoptic segmentation, as defined by [Kirillov et al., 2019], combines semantic and instance segmentation and distinguishes between two categories: *stuff* and *things*. *Stuff* refers to amorphous regions such as the sky that cannot be counted, while *things* are countable objects such as people and cars.

Each input point is given a class label, while *things* are also given a unique identifier to make them distinguishable. Since *stuff* is amorphous and uncountable, the identifier is omitted. The result for *stuff* is therefore analogous to semantic segmentation, while *things* are instance-segmented.

Event-Stream Segmentation: Related Work

Research on event-based segmentation is not as extensive as its frame-based counterpart. This is due to the novelty of the sensor technology itself. However, it is an active area of research with steady progress. The following sections summarize related work that addresses these segmentation tasks within the event-based vision domain.

Event Clustering and Object Detection

The clustering of events captured by Dynamic Vision Sensors can be used as a method for object detection. The high temporal resolution of the output also facilitates tracking the identified clusters.

In [Litzenberger et al., 2006], an embedded system for cluster building and tracking was implemented, inspired by the mean-shift clustering approach. Circular event regions are formed by assigning each new event to a cluster based on a distance criterion. This updates the cluster weight and center position. [Schraml and Belbachir, 2010] utilized a head-mounted stereo setup of Dynamic Vision Sensors to capture and represent moving objects in real-time from a top-down view. The clustering process employed density and distance metrics. The method was able to assign events to individual objects in a real-world scenario test setup. A stereo DVS setup was also used in [Piątkowska et al., 2012] to address the problems associated with high object occlusion. Extending the clustering task by using Gaussian Mixture Models and stochastic prediction of object states in particular time steps, people were detected and tracked.

In [Linares-Barranco et al., 2015], a low-power FPGA-based solution for noise reduction and object tracking has been proposed. Their clustering approach operates with predefined positions, where each module expects an object at a specific initial position in the visual field. This architecture assumes a fixed maximum number of objects, which are processed in parallel on the FPGA. [Lagorce et al., 2015] utilized the event-driven properties by integrating spatial and temporal correlations of events in an asynchronous iterative framework. They combined multiple kernels to extract and compute different features and grouped them for object tracking. The process involved the use of various kernels, including Gaussian, Gabor, and arbitrary user-defined kernels.

In [Barranco et al., 2018] a real-time mean-shift clustering algorithm has been proposed. Their approach avoids any preselection or a priori knowledge about the number of clusters or their shape. Tests have been performed on a dataset with simple shapes [Mueggler et al., 2017], as well as in a robotic setup. [Rodríguez-Gomez et al., 2020] integrated feature tracking and clustering into an event-by-event processing approach. Corners are detected and objects are clustered, while background regions and object regions with higher event rates are distinguished by building an attention priority map.

Another common approach to object detection, in addition to direct clustering at the event level, is based on converting data into classical frames and processing them using well-known deep learning methods.

In [Chen et al., 2019a], three different frame encodings have been proposed. Each is designed to focus on a different feature of the data stream, such as event frequency, timestamp information, or temporal aspects. These frames, as well as composite multichannel combinations of them, are then processed. In [Wan et al., 2021], a frame encoding called neighborhood suppression time-surface was introduced to address issues related to areas with highly varying event densities. While [Jiang et al., 2019] exploited the specificity of the used event camera to capture and incorporate classical intensity images. These approaches are subsequently based on the application of variants of the well-known 2D object detector framework family YOLO [Redmon et al., 2016]. Similarly, in [Damien et al., 2019], three frames were generated by accumulating the events in three different time windows to account for objects of different speeds. These frames are then processed using a single-shot detector (SSD) [Liu et al., 2016] and Faster R-CNN approach [Ren et al., 2017].

Motion Segmentation

Clustering typically relies on spatial distance or density metrics. As a result, objects that are spatially close to each other and similar are difficult to distinguish using clustering approaches. Identifying and separating objects that have independent motion is therefore an alternative method.

[Barranco et al., 2015] used motion estimation in a segmentation scenario. Their approach learns the location of contours and their boundary ownership, i.e., which side of the boundary belongs to the foreground and which to the background. Using different extracted features and a structured random forest, they demonstrated a layer segmentation of the given scenes. In [Vasco et al., 2017] the processing was performed in a robotic setup. This means that the sensors are typically non-stationary as they are mounted on a moving robot. They used the knowledge of the robot's own joint kinematics to detect and track corners in the event stream and learned their statistics. Independently moving objects in the scene are then identified by large discrepancies between the predicted corner velocities induced by ego-motion and the measured corner statistics. In [Mishra et al., 2017], events can be distinguished as static background or dynamic foreground based on microsaccades, which are minimal movements of the sensor. They assigned the events to "spike groups", which are clusters of events in space-time, and extracted robust temporal statistics that are a consequence of the introduced micro-movements. [Mitrokhin et al., 2020] worked with time-surfaces in (x, y, t) to perform a foreground-background segmentation of moving objects. They converted the event stream into a 3D graph and learned motion segmentation using a graph convolutional network. Due to the high memory requirements, the large number of events, and the resulting edge connections between graph nodes, the processing is limited in input time length.

Another group of motion segmentation algorithms were inspired by and are closely related to [Gallego and Scaramuzza, 2017; Gallego et al., 2018]. These works introduced the con-

cept of motion warping to event-based data in order to maximize the contrast of an image through motion compensation. Additionally, inspired by 3D point cloud processing techniques, [Mitrokhin et al., 2018] globally estimate the ego motion from the event stream and iteratively detect (multiple) objects that do not conform to the estimated model. They formulated a time-image representation on which this stabilization is performed. By estimating image plane velocities instead of sensor angular velocities, [Stoffregen and Kleeman, 2018] calculated the optical flow and simultaneously segmented the data into objects moving at the same velocity. [Stoffregen et al., 2019] proposed a per-event segmentation method for independently moving objects. Their method jointly estimates the segmentation and object motion parameters by maximizing an objective function, exploiting the idea of motion-compensated event images. The probability that an event belongs to a cluster is determined by maximizing the sharpness of warped event clusters in these images. Similarly, [Zhou et al., 2021] solved the problem of identifying moving objects as an energy minimization problem involving the fitting of multiple motion models. They used Markov random fields and an unstructured spatio-temporal graph representation, which allowed them to solve the problem by graph cuts.

[Wang et al., 2022] introduced a cross-domain motion segmentation method that fuses classical frames and events to improve segmentation performance. By fusing the visual cues from both events and frames, the complementary domain attributes were exploited, such as addressing slow object motion and highly textured scenes.

Semantic Segmentation

Motion segmentation approaches do not distinguish or identify semantic classes. Therefore, the found objects have no class assignment. Semantic segmentation fills this shortcoming.

In [Sekikawa et al., 2019], the so-called EventNet was introduced, which is designed for real-time processing of asynchronous event streams in a recursive and event-based manner. Its processing approach is inspired by PointNet [Qi et al., 2017a], but the sparse input signal is directly processed recursively rather than in a cloud-based manner. Two separate processing modules allowed immediate per-event updating of global features as well as for a final output that is computed only on demand. In combination with multi-layer perceptrons, which were replaced by a lookup tables for inference, real-time processing was achieved. By design, however, it is not capable of extracting hierarchical features from the data.

[Alonso and Murillo, 2019] proposed a semantic segmentation processing based on a classical encoder-decoder convolutional neural network architecture combined with a novel representation for DVS data. They introduced a dense 6-channel image representation that encodes both the histogram and the temporal distribution of events. In their test scenario, the comparison showed that it outperformed other previously used event representations. The method proposed in [Gehrig et al., 2020] used existing video datasets, exploiting their given labeling, to convert them into labeled synthetic event data. These events and copied labels were used to train a network architecture according to [Alonso and Murillo, 2019], while real event data was used at test time.

In [Wang et al., 2021a], a knowledge distillation approach called EvDistill was proposed. It was designed to train a smaller neural network ("student") with unlabeled and unpaired event data by distilling knowledge from a larger network ("teacher") trained with labeled image data. This was mainly achieved by introducing a bidirectional modality reconstruction module to bridge the modalities. While their method is effective, it should be noted that the use case and type of event data should be similar to the type of labeled source data for optimal performance. A similar idea was pursued in [Sun et al., 2022]. They proposed a method called ESS, which aims to transfer a semantic segmentation task from labeled image datasets to unlabeled events through unsupervised domain adaptation. ESS creates motion-invariant

event embeddings that are aligned with image embeddings without the need for video data or hallucinated motion by using pre-trained encoders. The process involves encoding events into motion-invariant embeddings, reconstructing them into still images, aligning these images with events to form pseudo pairs in the source and target domains, and predicting the final output from the image and event embeddings.

The combination of event-based and frame-based vision modalities is also used for semantic segmentation. In [Biswas et al., 2024], a hybrid approach was proposed that includes a simple and efficient cross-domain learning scheme to extract complementary spatio-temporal embeddings from both frames and events. It uses a spiking neural network for temporal feature extraction from events and a classical artificial neural network for spatial feature extraction. In a subsequent step, the temporal and spatial embeddings are mixed for robust prediction. As part of the ablation study, event-only processing was considered. [Ghasemzadeh and Shouraki, 2023] proposed an event-frame based semantic segmentation network that also uses both, frames and events. Their main goal is to combine the complementary advantages of both modalities by using an encoder-decoder architecture with two parallel encoders and a fusion of the resulting feature maps. They introduced a blurring module for data augmentation and evaluated their method in terms of robustness. In contrast to these modality-specific methods, a unified fusion framework, called CMX, has been proposed in [Zhang et al., 2023]. CMX also incorporates event-based data.

Instance and Panoptic Segmentation

Using semantic segmentation approaches, it is impossible to distinguish between spatially close or even occluded objects of the same class. This is especially important in the context of surveillance applications, such as monitoring a group of people.

In this case, instance or panoptic segmentation is an appropriate approach, but the resulting challenges are largely unaddressed for event-based data.

A method based on Spiking Convolutional Neural Networks called HULK-SMASH has been published in [Kirkland et al., 2022]. It emphasizes a departure from traditional deep learning models by employing a time- and feature-based similarity metric. The latent spiking neurons in the layers are resolved back to pixel space.

A processing approach called Bimodal SegNet has been proposed in [Kachole et al., 2024], which fuses event-based data and RGB frame data for instance segmentation. It includes two separate encoders that capture and combine information at different resolutions using a Cross-Domain Contextual Attention layer. Leveraging elements of UNet and transformer architectures, the segmentation process is enhanced. An event-based panoptic segmentation approach based on a Graph Mixer Neural Network has been proposed in [Kachole et al., 2023]. It operates on 3D event graphs and progressively reduces the number of nodes by downsampling in the encoder while upsampling in the decoder. Unlike conventional graph network processing, where graphs are constructed prior to network operations and predictions, a novel subgraph construction strategy and parallel processing scheme were introduced.

Event Representations and Processing

Dynamic Vision Sensors are a relatively new type of sensor technology. However, they are already being used in a variety of research areas. These include, for example, industrial vibration measurement and control applications [Dorn et al., 2017], autonomous driving scenarios [Chen et al., 2020a], and even space surveillance applications [McMahon-Crabtree and Monet, 2021].

However, as stated by

> "there is not yet a clearly adopted way of representing the stream of events to feed a CNN", (Alonso and Murillo, 2019)

and

> "[e]fficient low-level encoding of event data is still an open research problem". (Biswas et al., 2024)

There are significant differences in the way in which events are represented for processing. In many cases, they are transformed into alternative representations in order to solve a given task. The boundary between pure representation and feature extraction is often blurred in the context of the intended application.

The main categories of event representations currently in use are outlined in Section 1.4.2 on page 12. However, their use in the proposed approaches is not always completely distinct.

Processing based on individual events is often used in filtering, where decisions are made for each event based on the previous events in the spatio-temporal neighborhood. Moreover, it is also used in clustering approaches, where the events serve natively as a basis [Litzenberger et al., 2006; Schraml and Belbachir, 2010; Piątkowska et al., 2012; Linares-Barranco et al., 2015; Barranco et al., 2018; Rodríguez-Gomez et al., 2020].

In most of the high-level approaches mentioned in the related work above, the event stream is converted into dense 2D grid frame representations for processing [Mitrokhin et al., 2018; Alonso and Murillo, 2019; Chen et al., 2019a; Jiang et al., 2019; Gehrig et al., 2020; Wan et al., 2021; Wang et al., 2021a, 2022; Ghasemzadeh and Shouraki, 2023; Kachole et al., 2024]. However, this results in a loss of signal sparsity and, in general, a loss of information in the representation (e.g., the dense time resolution).

Alternative representations that address these concerns have been less widely used. Event-Graph-based approaches can preserve both sparsity and dense temporal resolution [Zhou et al., 2021; Kachole et al., 2023]. However, their initialization can be complex and comes with high memory requirements.

Compared to frame-based representations, event voxelization aims at a better representation of the spatio-temporal relationships involved, as in [Biswas et al., 2024; Zhang et al., 2023].

Space-time event clouds are rarely used, although they can be created natively and directly from the event stream of the Dynamic Vision Sensors. For segmentation, point clouds are used only for training in [Sekikawa et al., 2019] and as a basis for comparison in [Mitrokhin et al., 2020]. While [Wang et al., 2019] leveraged point clouds to classify hand gestures in event-based data.

Event-Stream Representations

As mentioned earlier, segmentation is an important part of computer vision. Event-based research in this area is not as extensive as its frame-based counterpart, or to quote Sun et al.:

> "[...] semantic segmentation with event cameras is still in its infancy".
>
> (Sun et al., 2022)

Therefore, this work addresses the research question of which type of representation (see Figure III.2a), combined with which processing approaches, yields the best results.

For this purpose, 3D space-time event clouds, voxel- and frame-based representations were intensively compared. Several existing deep learning-based segmentation approaches were evaluated in order to examine their existing potential in the field of event-based vision. In the specific case of this work, the challenges arising from the large-area measurement scenario under real-world outdoor conditions were addressed (see Figure III.2b).

Studied Representations and Encodings

The event stream generated by Dynamic Vision Sensors is often converted into alternative representations for processing. In this work, the following representations were considered for segmentation purposes:

Space-Time Event Cloud Representation

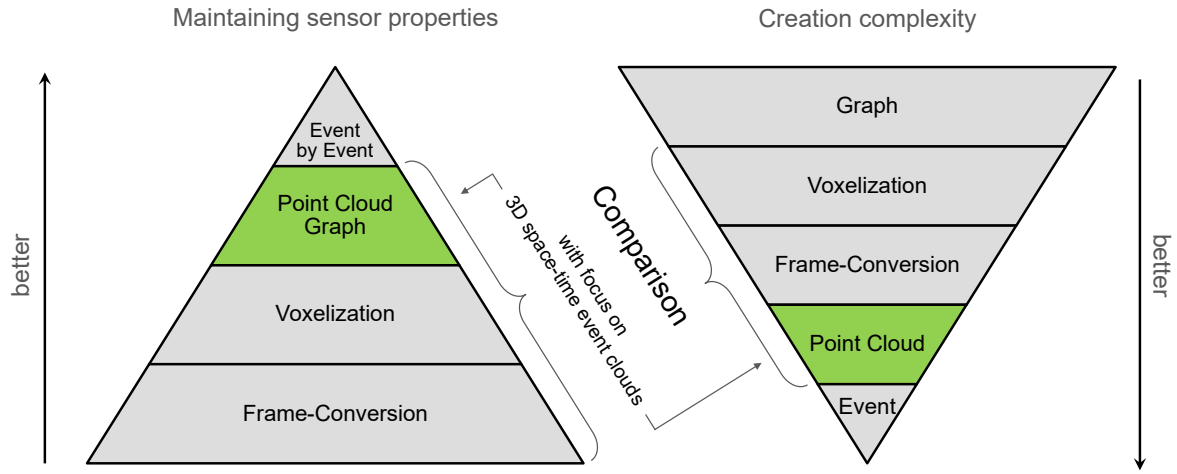
Temporal windows of the continuous event stream are directly represented as a point cloud. Each DVS event e_i defines a 3D point formed by its (x_i, y_i, t_i) coordinates, resulting in a representation as unordered set of $\{x_i, y_i, t_i\} \in \mathbb{R}^3$. This preserves the sparsity and high temporal resolution of the DVS signal and transforms it into a geometric description. This concept is illustrated in Figure III.4a.

Spatio-temporal scaling is important for cloud creation and processing. In the following chapter, several variations will be introduced and tested.

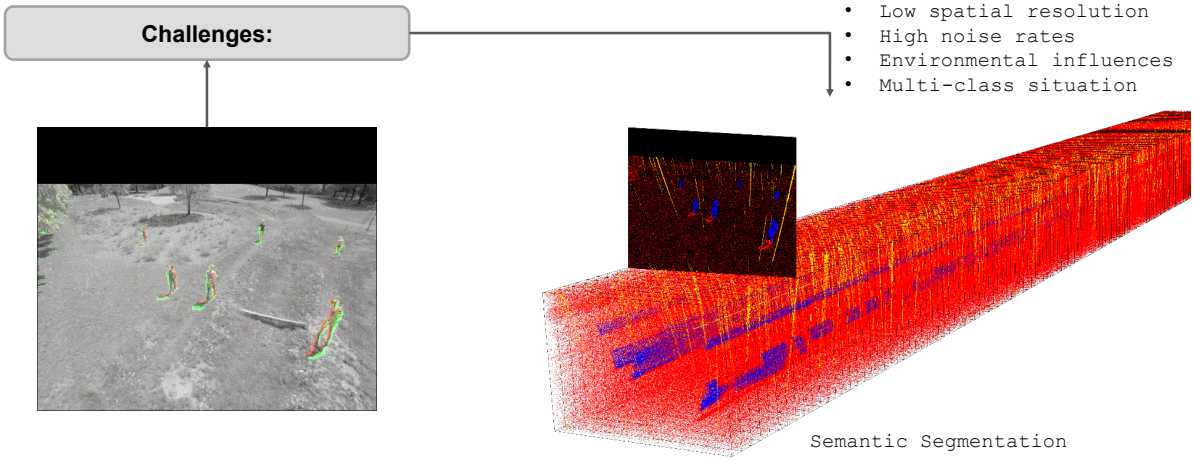
Event-Voxelization Representation

As a point cloud structure, space-time event clouds are irregular, unstructured, and unordered. This means that the event points are not scattered homogeneously and with the same density in different regions, are not placed in a regular grid, and since they are represented as a set, the ordering of the points does not alter the represented scene.

These characteristics are changed by means of a voxelization. This process maps and aggregates the data into a regular 3D grid. The sparsity of the signal is lost in this transformation. This concept is illustrated in Figure III.4b.



(a) Event representations and comparison focus



(b) Segmentation challenges

Figure III.2: Event representation, comparison focus and processing challenges.

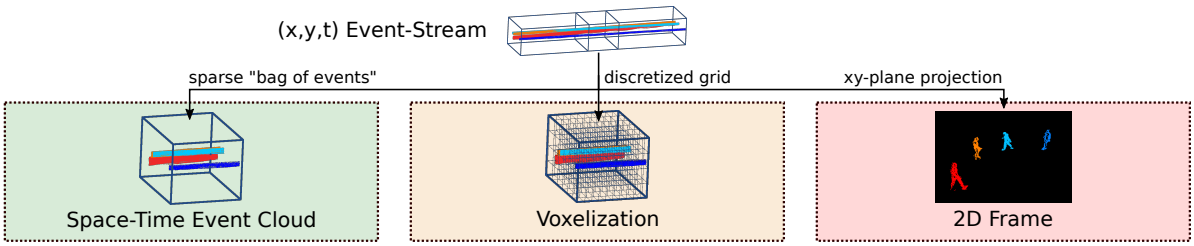


Figure III.3: Overview of event-stream representations under study (from [**Bolten** et al., 2024]).

In this work, unless otherwise stated, voxels were formed per DVS pixel to account for fine spatial structures, while the time dimension was discretized into t_{bin} bins. This discretization results in a data representation of the shape ($\text{x} \times \text{y} \times t_{\text{bin}}$). By counting the triggered events per voxel, a 3D voxel histogram is constructed that represents the distribution of events in the spatial-temporal domain.

Frame Representation

Classic 2D frames are created by projecting events onto the xy-plane. This results in a dense 2D grid of fixed size defined by the pixel resolution of the sensor. It allows direct processing using classical computer vision approaches. Since events are triggered by changes, the resulting images visually resemble edge images.

In this work, the following event projection variants were considered.

Single-Channel Encodings:

Simple single-channel frames can be created within this projection. This concept is illustrated in Figure III.4c. They are included in this work as a baseline for comparison with more complex encodings.

Binary encoding:

The frame pixel at (x, y) is set to white if a corresponding event with the same spatial coordinate exists in the DVS data.

Frequency encoding:

The frame pixel at (x, y) encodes the normalized count of event occurrences at that spatial position as proposed in [Chen et al., 2019a]:

$$\sigma(n) = 255 \cdot 2 \cdot \left(\frac{1}{1 + e^{-n}} - 0.5 \right) \tag{III.1}$$

where n denotes the number of occurred events within the considered time window at the evaluated pixel coordinate.

The frequency of occurrence of events within a time interval represents an indication of whether the event is noise or a valid event. Assuming that moving objects trigger a larger number of events, this encoding leads to a higher weighting of the edges whereas noise is reduced assuming advantages for subsequent signal-processing.

Three-Channel Encodings:

Classical three-channel RGB images could also be generated within the projection of the DVS stream. This concept is illustrated in Figure III.4d.

Polarity encoding:

The frame pixel at (x, y) encodes the polarity of the latest event that occurred at that spatial position. The direction of brightness change is encoded by the following colors: (a) decrease in red, (b) no change in blue and (c) increase in green[14].

Merged-Three-Channel (MTC) encoding:

This encoding was proposed in [Chen et al., 2019a]. It incorporates three different single-channel encodings, each addressing different features of the underlying event stream, to create an RGB false color image.

[14]The utilized CeleX-IV sensor determines the polarity of an event based on its gray value, this can also lead to events with "no brightness" changes.

Red Channel: The *Leaky-Integrate-and-Fire* neuron model includes information about temporal continuity.

Each pixel is interpreted as a neuron with its own membrane potential with a fixed decay rate to realize a kind of memory. The triggered events will increase the corresponding membrane potential and will cause a firing of the neuron if a threshold is exceeded. The firing rate determines the pixel value in the frame encoding.

Green Channel: The *Surface-Of-Active-Events* includes time-surface information about the direction and speed of object motion through its gradient.

An advantage of the DVS technology is the high time resolution of the event stream. The goal of this channel is to incorporate this resolution into the frame representation. Here, the pixel values are directly dependent on the corresponding timestamp of the events.

Blue Channel: The *Triggering Frequency* contributes to distinguishing between noise and valid events.

This is the inclusion of the previously described single-channel frequency encoding as one channel of the merged representation.

Multichannel Encodings:

A highly multichannel frame representation was also considered. The occurrence of an event is encoded per time channel, resulting in a representation of the form ($x \times y \times t_{\text{channel}}$). This way the temporal context can be better preserved and exploited when converting the DVS stream.

In this case, the time component of the signal is separated and stored in many channels during the projection process. This concept is illustrated in Figure III.4e.

Example frame encodings are shown in Figure III.5.

These encodings were selected because they attempt to represent and preserve different levels of information (e.g., the sparsity and time resolution of the sensor).

Alternative Representations and Encodings

A variety of other encoding rules have also been described in the literature.

Frame Representation

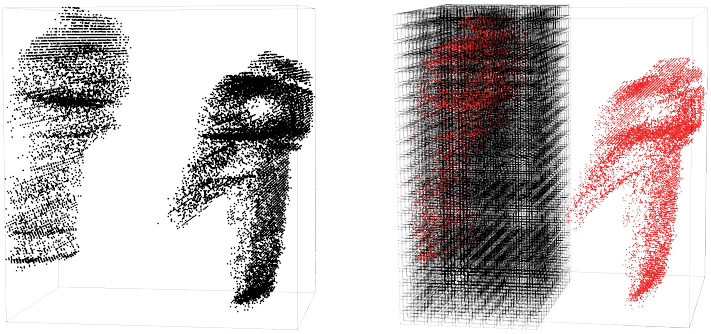
For the event projection step into 2D grids, some of the most common other encodings are:

Event Frames Encodings:

A widely used visualization method in various SDKs[15] encodes events in a similar way as described by Kogler in [Kogler, 2016, Section 5.2.1 "Event to Event-Image Converter"]. Assuming an 8-bit grayscale image, the background is initialized with the mean gray value of 128. Each event updates only the gray value at its own spatial position.

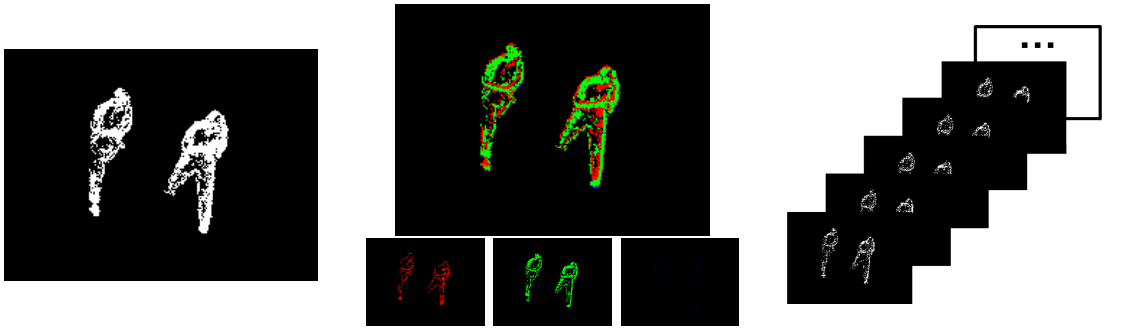
Kogler described different update rules, including the "Event event-image", where positive event polarities update the corresponding image value to 0 and negative ones to 255. The "Binary event-image" sets 255 for both polarities. This results in grayscale images

[15] such as "jAER" [Delbruck, 2008], "DV Software" by iniVation, and "Metavision SDK" by Prophesee



(a) Plain 3D (x, y, t) space-time event cloud

(b) $(x \times y \times t_{\text{bin}})$ 3D voxelization (half shown, for better clarity)

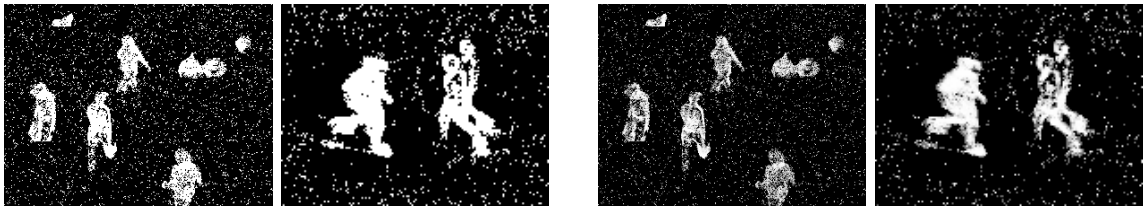


(c) $(x \times y \times 1)$ single-channel 2D frame representation

(d) $(x \times y \times 3)$ RGB 2D frame representation

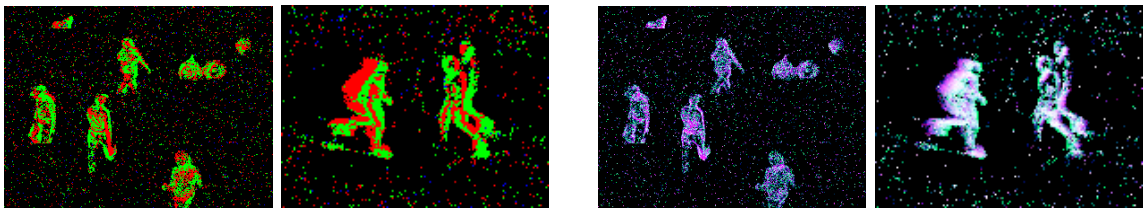
(e) $(x \times y \times t_{\text{channel}})$ highly multichannel 2D frame build by time-splitting

Figure III.4: Graphical rendering of the basic ideas for event representation (extended from [**Bolten** et al., 2023c]).



(a) Binary encoding

(b) Frequency encoding



(c) Polarity encoding

(d) MTC encoding

Figure III.5: Snippets of frame-encoded events (extended from [**Bolten** et al., 2024]).

composed of a total of three or two different grayscales, respectively. He also introduced the "Grayscale event-image", where each updated pixel is incremented or decremented by one, depending on the polarity of the event. In this way, events at the same spatial position can cancel each other out.

In general, shades of color can also be used instead of grayscales in these conversions. For example, Prophesee uses shades of blue by default in its SDK.

Since the basic ideas of these conversion approaches are covered within the considered *binary* and *polarity* frame representations, further conversion variants were not included in the following experiments.

Histogram Encodings:

A 2D event histogram encodes the spatial distribution of triggered events. Events are accumulated over time into histogram bins, where each bin represents a certain spatial region. The histogram values therefore correspond to the frequency of events occurring in these regions.

There are different ways to construct these histograms, such as ignoring the event polarity [Liu and Delbruck, 2018] or constructing separate histograms for positive and negative event polarities [Maqueda et al., 2018].

Since the event frequencies are already considered in the *frequency* representation, event histograms were not considered separately.

Time-Surface Encodings:

Time-surfaces can be used to minimize the loss of time resolution associated with the above representations. In this representation, the pixel values are used to store the timestamp of the last received event that occurred at that location. This represents the most recent temporal activity history within a local spatial neighborhood [Delbruck, 2008; Lagorce et al., 2017].

In addition to the simplest case, where only positions where events have been triggered are taken into account, different timestamp update mechanisms are possible. In the case of linear time-surfaces, the time-surface is reset for each new time window. An exponential decay can also be used to decay all stored timestamps.

As one type of time-surfaces is included in the considered *MTC* representation as the *Surface-Of-Active-Events* channel, event time-surfaces were not considered separately.

Event-Cube Encoding:

Event cubes were developed to combine the frequency information from histograms with the time information contained in time-surfaces. In event cubes, each time window is further divided into three micro time windows. Events, separated by polarity, are counted weighted by their temporal distance from the center of the neighboring micro-time bins[16].

Event cubes were not included in the further experiments, as they were not yet widely used at the time of the work and are only now gaining attention in the community.

Event-Graph Representation

When constructing event graphs, the graph nodes are formed from the triggered DVS events. The edges between these nodes are often created based on the k-nearest neighbors in the spatio-temporal event neighborhood.

[16]For further details, see `https://docs.prophesee.ai/stable/tutorials/ml/data_processing/event_preprocessing.html#event-cube`

The selection of this number of k neighbors must be done carefully [Mondal et al., 2021]. In addition, as for example in [Kachole et al., 2023], the graph is often limited to an arbitrary number of events, to account for the high computational demands and memory constraints.

In [Bi et al., 2019] and [Bi et al., 2020], a non-uniform grid subsampling procedure is used to reduce the number of events to be considered in the graph. In [Wang et al., 2021b], an octree grid filtering algorithm is applied, while [Chen et al., 2020b] performs random sampling. For saving computation time, the complete event stream is often divided into short temporal sections so that only few events are considered for the respective graph construction.

In the long-term monitoring considered in this work, the recorded object sizes vary greatly due to the perspective. Thus, the number of events per object varies significantly. In addition, both local and global environmental effects are included. Overall, this makes it difficult to select appropriate parameters for the construction of such graphs. Therefore, graphs were not included in the evaluations performed.

Learned End-to-End Encoding

Instead of handcrafting encoding rules, a fundamentally different approach can be applied by integrating the encoding and creation of data representations end-to-end into the learning process of the subsequent application.

By designing the derivation of raw sensor data into a representation in a fully differentiable manner, it is possible to unify specific application tasks, such as segmentation, with the creation of the representation used for processing. This can be achieved by applying gradient-based learning to the entire system.

Gehrig et al. presented the “Event Spike Tensor” as the first end-to-end learnable event representation [Gehrig et al., 2019]. This representation has also been adopted in other works [Kong et al., 2022; Acin et al., 2023].

The further inclusion of end-to-end learned event representations was not considered in this work, as their inclusion in the learning process may complicate the convergence of the training process. Furthermore, it increases the complexity and computational cost. This work focuses on comparing the general suitability of the proposed representations. Further refinement, including the integration of end-to-end approaches, is an interesting extension for future work based on the knowledge gained.

Chapter 6

Semantic Segmentation: Sparse 3D Space-Time Event Clouds

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Lentzen, F., Pohle-Fröhlich, R., and Tönnies, K. (2022a). Evaluation of Deep Learning based 3D-Point-Cloud Processing Techniques for Semantic Segmentation of Neuromorphic Vision Sensor Event-streams. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 179. INSTICC, SciTePress

The challenges of generating event-wise semantic segmentation for DVS sensor data in an outdoor scenario with objects of different scales and artifacts due to sensor noise and environmental influences are addressed. Many event-based processing applications transform the event stream into alternative representations. To avoid the conversion of these streams into conventional 2D frames, the feasibility of using 3D point cloud approaches was evaluated, analogous to [Wang et al., 2019]. These 3D processing approaches were selected based on the assumption that frame representations do not sufficiently preserve the inherent sensor properties.

Extending previous work, the main contributions presented in this chapter are as follows:

- The evaluation of PointNet++, the pioneering model and foundation for three-dimensional point-based deep learning processing. The first analytical comparison, optimization, and quantitative evaluation of various input and network configuration parameters in the application domain was conducted. This resulted in an optimized and recommended configuration for processing.
- Extension of the performed evaluation to three enhanced point cloud-based network structures, which aim to improve the vanilla PointNet++.
- Comparison of the results obtained using these 3D point cloud deep neural network structures with an event-based 2D CNN processing baseline in terms of quality and runtime.

6.1 Proposed 3D Processing

The basic characteristics of the 3D processing performed for semantic segmentation are outlined below.

6.1.1 Methods

PointNet++ was selected for its pioneering role in the direct application of neural networks to point clouds, serving as the foundation for many subsequent methods (see [Guo et al., 2021] for a comprehensive review of methods).

PointNet++ [Qi et al., 2017b]: Hierarchical Feature Learning

PointNet++ is based on the basic PointNet [Qi et al., 2017a] from the same authors, which was the first neural network to be widely and successfully applied directly to point clouds. It uses max-pooling as a symmetric function to make the network invariant to the order of the input points. Multilayer Perceptrons (MLPs) used for feature extraction are shared, resulting in a relatively small network. After extracting geometric features from the input cloud, an MLP is trained for segmentation.

PointNet++ learns a spatial encoding of point cloud data. For this purpose, the input data is hierarchically divided and summarized. By repeatedly applying a simple PointNet as a feature extractor, local and global features are built and combined. The result is a representation of the entire point cloud that captures not only global geometric features, but also local neighborhood information. This general process is illustrated in the Figure Figure 6.1.

The hierarchical processing is realized by the so-called Set Abstraction (SA) layers of the network. First, representative centroids of local regions are selected by Farthest Point Sampling (FPS; see Figure 6.2a). Then, local neighboring points are selected around these centroids. By default, this is done using a ball query with a defined radius (see Figure 6.2b). A maximum number of points is then selected within this ball.

The extracted feature vectors of these local regions are geometrically represented by the centroid coordinates for the following step. This is illustrated in Figure 6.2c and d for the first and subsequent second layer. This process of creating common structural partitions allows the weights of the feature extractors to be shared per network layer.

However, due to the hierarchical processing, interpolation is required to create point-wise features. Therefore, for semantic segmentation, the resulting features are finally interpolated by a Feature Propagation (FP) layer.

The following methods were additionally selected because (a) they all aim at an improvement over vanilla PointNet++ (e.g., by making better use of relations, distances and directions) and (b) their reference implementations are also freely available.

A-CNN [Komarichev et al., 2019]: Annularly Convolutional Neural Network

The main concept of A-CNN is to apply convolution to the point clouds, leveraging their relationships. Therefore, a method for ordering the points is needed. The authors project every point inside a local neighborhood into an approximated tangential plane at the representative point. In the plane, angles can be computed and used for ordering. Following this, an annular convolution can be applied.

LSANet [Chen et al., 2019b]: Feature Learning by Local Spatial Aware Layers

In LSANet, the process of abstraction and local feature learning is accompanied by a parallel branch that learns so-called *spatial distribution weights* from the coordinates of

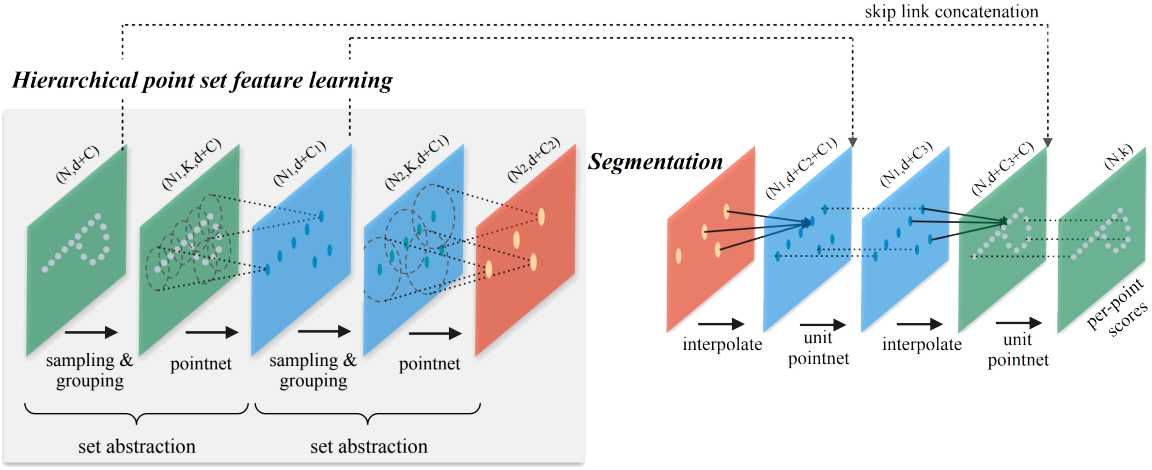
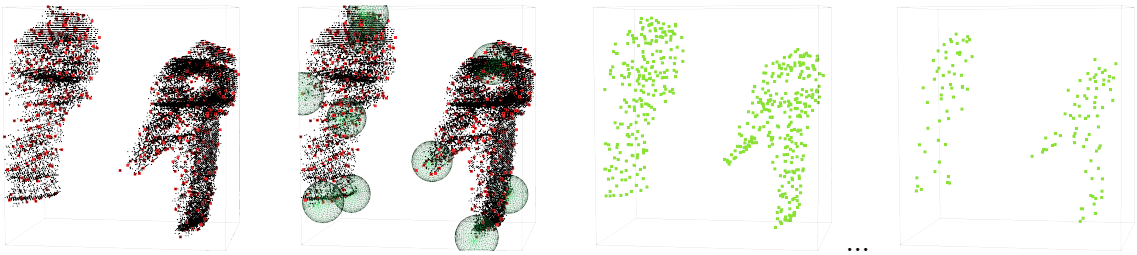


Figure 6.1: PointNet++ structure for segmentation (adapted from [Qi et al., 2017b]).



(a) Farthest Point Sampling (input data from Figure III.4a)

(b) Subset of ball query neighborhood selections

(c) Abstracted event set of layer 1

...

(d) Abstracted event set layer 2

Figure 6.2: Summary of PointNet++ processing concept (from [**Bolten** et al., 2023c]).

all points in a local neighborhood. These weights implement the concept of attention in neural networks, which means that some parts of the input are considered more important than others. The attention mask is multiplied element-wise with the features. In LSANet, the attention weights are based on the coordinates, i.e., the geometric properties of a local region, which should allow the network to better learn details of the structures.

SpiderCNN [Xu et al., 2018b]: Parameterized Convolutional Filters
The authors of SpiderCNN do not attempt to order the point cloud as in A-CNN. Instead, they adapt the convolutional filters to support unordered input.

In classical discrete convolution, only discrete weights for each position, defined by the kernel size, are needed. In point clouds, the positions of neighboring points are not known, so instead of defining discrete weights, a function is used that calculates a weight based on the coordinates of a point. During the learning phase, parameters of the function that generates the weights are learned instead of the weights themselves. The structure differs from PointNet++ in that the point cloud is not sampled and therefore no interpolation is required.

6.1.2 Processing Pipeline

Previous work applying 3D point cloud processing techniques to DVS event data, such as [Wang et al., 2019], process temporal windows of the stream that include the full spatial resolution of the sensor.

Event Count Requirements

The *DVS128* sensor [Lichtsteiner et al., 2008] used by Wang et al. has a spatial resolution of 128×128 pixels. In contrast, the CeleX-IV sensor used in this work has a usable spatial resolution of 768×512 pixels (see Section 3.2). This considerably higher resolution results in a significantly higher number of events that can be triggered in a given time window of the same length.

Considering the entire underlying dataset [Amir et al., 2017], the average number of events per 60 ms time window in the work of Wang et al. is about 3,175 events. In comparison, the average number of events per 60 ms time window for the DVS-OUTLAB data (see Chapter 4) is about 30 times higher with about 97,000 events. Section D.1 in the Appendix provides additional statistics on the number of events for this data.

Therefore, we decided to use spatial patching in addition to temporal windowing. As shown in Figure 6.3, the provided data is divided into 16 equally sized patches of $192\,\text{px} \times 128\,\text{px} \times 60\,\text{ms}$, referred to as Patch-of-Interest (PoI). This patching reduces the average number of raw events per patch to approximately 7,500 events. The divide-and-conquer approach allows smaller objects and finer structures to be preserved for subsequent processing. This is important because further filtering and sampling steps are required for the generation of the final point clouds.

Preprocessing Steps

To generate the input point cloud for processing, the following preprocessing steps are performed per patch (compare with Figure 6.4 for clarification):

1. **Spatio-temporal Filtering:** The CeleX-IV sensor data contains a high amount of sensor noise as described in Section 3.3. As described there, the event stream is spatio-temporally filtered at this point.

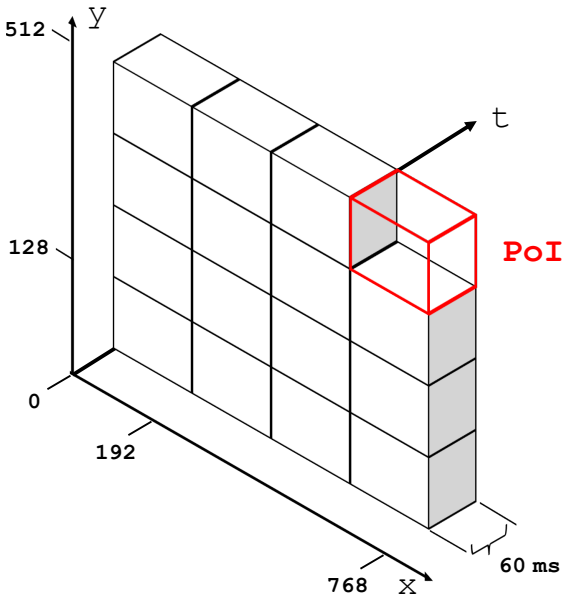


Figure 6.3: PoI as divide-and-conquer data selection method (from [**Bolten** et al., 2022a]).

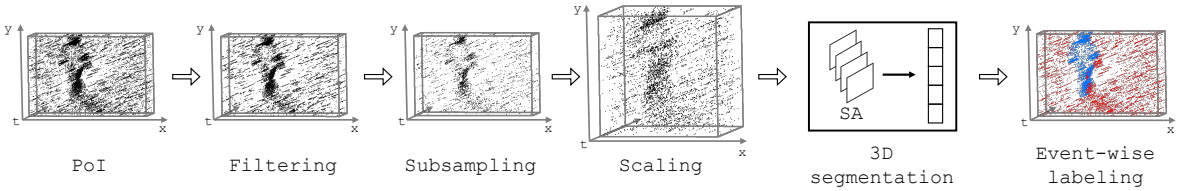


Figure 6.4: Overview of 3D-based processing steps per PoI (from [**Bolten** et al., 2022a]).

The event stream is filtered by removing any event that is not supported by at least one other event at the same spatial (x, y) coordinate within the preceding 10 ms (referred to as Time-Filter). The selection of this filter is based on the performed filter analysis. It reduced the background activity by about 50 % while preserving the highest proportion of other class events.

2. **Subsampling:** A prerequisite for the use of PointNet++ and the other 3D network variants is that the number of events used for training must be constant. Therefore, a uniform random subsampling of the time-filtered events is performed in such a way that each event has the same probability of occurrence.

   The average event count per patch is approximately 4,800 events after applying the previous time-filtering step. See Appendix Section D.1 for details on the number of patched events before and after filtering. A subsampling target of $n = 4096$ events was chosen as appropriate based on this evaluation of the number of events.

   For patches with fewer events, copies of randomly selected events are inserted until the required number is reached. The impact of these duplicates is negligible due to the use of max-pooling in the point cloud network processing logic.

3. **Spatio-Temporal Scaling:** After filtering, the entire patch of 192 px × 128 px × 60 ms is moved so that its origin is at $x = 0$, $y = 0$, $t = 0$. This results in the following 3D space-time point cloud definition:

$$\begin{aligned} S^T_{\text{native}} = \{e_i = (x_i, y_i, t_i) \mid & \\ & t_i \in T, i = 1, \dots, 4096, \\ & x_i \in \mathbb{N} : 0 \leq x_i < 192, \\ & y_i \in \mathbb{N} : 0 \leq y_i < 128, \\ & t_i \in \mathbb{R} : 0 \leq t_i < 60.0\} \end{aligned} \tag{6.1}$$

   where T is the current 60 ms time window.

   In addition, two other variants were considered in the following experiments. The 3D event coordinates were scaled along the time axis:

$$\begin{aligned} S^T_{\text{tScaled}} = \{e_i = (x_i, y_i, t_i) \mid & \\ & t_i \in T, i = 1, \dots, 4096, \\ & x_i \in \mathbb{N} : 0 \leq x_i < 192, \\ & y_i \in \mathbb{N} : 0 \leq y_i < 128, \\ & t_i \in \mathbb{R} : 0 \leq t_i < 1.0\} \end{aligned} \tag{6.2}$$

   Or along all three axes:

$$\begin{aligned} S^T_{\text{cube}} = \{e_i = (x_i, y_i, t_i) \mid & \\ & t_i \in T, i = 1, \dots, 4096, \\ & x_i, y_i, t_i \in \mathbb{R} : -1 \leq x_i, y_i, t_i \leq 1\} \end{aligned} \tag{6.3}$$

A semantic segmentation was then generated from these point clouds using the network structures, resulting in an event-wise labeling as shown in Figure 6.5.

6.2 Proposed 2D Processing

The basic characteristics of the 2D processing performed for comparison are outlined below.

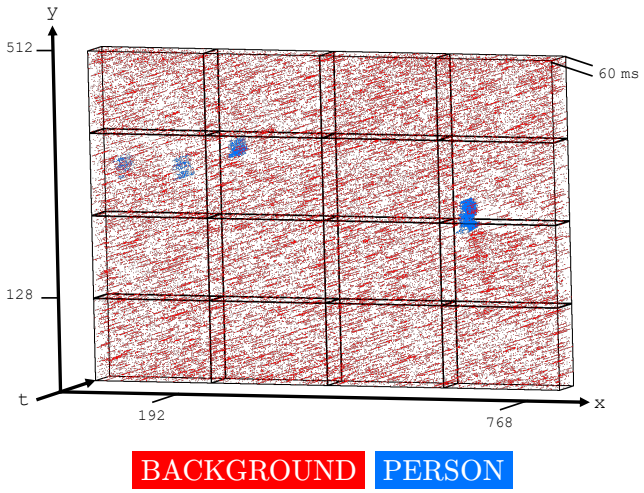


Figure 6.5: 3D space-time event cloud visualization, including PoI-boundaries and semantic segmentation (from [**Bolten** et al., 2023b]).

6.2.1 Methods

As a baseline comparison to the above 3D point cloud processing networks, traditional 2D convolutional neural network approaches were also evaluated. Pixel-accurate segmentation and labeling is required for further processing and accurate comparison.

Established state-of-the-art CNN structures such as UNet [Ronneberger et al., 2015] or Mask R-CNN [He et al., 2017] are well suited to generate these segmentations. Mask R-CNN is known to be able to learn good and robust predictions. Therefore, we used this network model for the following comparisons. For this purpose, the predicted object masks are subsequently considered at the level of a semantic segmentation. In the following Chapter 7, the processing is extended by including a UNet-based processing.

Mask R-CNN [He et al., 2017]:

Mask R-CNN is a neural network architecture focused on object recognition and segmentation. It simultaneously detects objects in an image and accurately extracts their pixel-level boundaries. The main components are summarized in the following.

The first element is a backbone network, such as ResNet [He et al., 2016], responsible for feature extraction. Following this, a Region Proposal Network generates object proposals at different scales and aspect ratios. A fixed size feature map is extracted from the feature pyramid generated by the backbone network, using an RoIAlign layer that ensures precise spatial alignment for Region of Interest (RoI) processing. The RoI-head consists of two sub-networks, one for classification and bounding box regression and another for instance mask prediction.

This integrated architecture facilitates object detection, accurate bounding box refinement, and detailed instance mask predictions, allowing for simultaneous recognition and segmentation of objects in an image at pixel-level boundaries.

6.2.2 Processing Pipeline

The 2D comparison was performed by converting the event stream into 2D frame representations. To ensure comparability with the 3D point cloud methods, the frame-based Mask R-CNN analysis was based on input data that has been preprocessed in the same way. This means that the generated input frame representations originate from the same temporal DVS

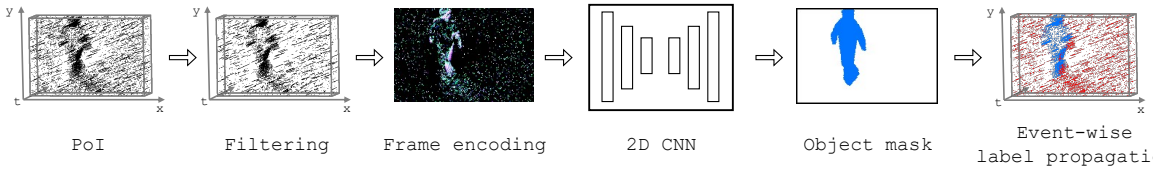


Figure 6.6: Overview of 2D-based processing steps per PoI (from [**Bolten** et al., 2022a]).

event stream patches of 192 px × 128 px × 60 ms, which were also spatio-temporally pre-filtered with the same time-filtering parameter of 10 ms.

These patches were encoded in the Binary, Frequency, Polarity and MTC frame representations (see Figure III.5 on page 93). These frame encodings were selected for comparison because they include different aspects of the event stream modality and provide different levels of abstraction, allowing for a proper comparison with the 3D methods. For each of these frame encodings, different Mask R-CNNs were trained from scratch using the implementation from [Abdulla, 2017].

The predicted object masks were then considered at the level of semantic segmentation. To compare the obtained results with the point cloud approaches, the 2D label masks generated by Mask R-CNN were propagated back to the original event stream. Each event received the label from its spatial position in the corresponding predicted object mask. This resulted in event-wise labeling (see Figure 6.6).

6.3 Experiments and Evaluation

The quantitative analysis of the proposed approaches was divided into the following parts:

(a) optimization of the basic PointNet++ network in terms of input and network parameters,

(b) the evaluation of PointNet++ successor network variants, and

(c) the comparison of the obtained results including the 2D frame-based baseline.

For all experiments, the proposed training, validation and test splits of the DVS-OUTLAB dataset were used (see Chapter 4).

6.3.1 Network Training

The hyperparameters of the networks have been left at their default values whenever possible.

3D Processing Networks:

For the 3D point cloud networks, this includes the Adam optimizer with a learning rate of 0.001 and a learning rate decay of 0.7 every 200,000 samples. The batch size was set to 16 point clouds.

The input points were randomly shuffled before training the network. No further data augmentations were applied.

PoinNet++'s local neighborhood creation process selects sample points within ball quires formed around the centroids selected by Farthest Point Sampling. In the reference implementation provided by the PointNet++ authors, only the first matching points in the ball are considered and directly selected. This FIFO order of selection, combined with the event ordering used in the underlying DVS readout logic, would result in insufficiently captured local neighborhoods. Therefore, the applied random shuffling is important for the correct functioning of the local neighborhood building in the processing.

(a) Reference scene in RGB (b) Dense 2D label (c) Corresponding events

Figure 6.7: Dense label description compared to sparse event stream.

2D Processing Network:

In the case of the Mask R-CNN training, the default learning rate was reduced by a factor of 2 to 0.0005. This was done to avoid gradient explosion, which tends to occur with smaller batch sizes. As in our case, only three images were used per batch due to memory requirements. The Keras stochastic gradient descent (SGD) optimizer was used for training with a momentum of 0.9 and no data augmentation was applied.

6.3.2 Evaluation Metrics

Evaluation of segmentation tasks is often based on dense 2D label images, even for event cameras, see e.g. [Alonso and Murillo, 2019]. For each 2D pixel in the annotation, the corresponding pixel value of the network prediction is evaluated and compared. However, this kind of evaluation ignores the fundamental property of a Dynamic Vision Sensor that the generated event stream is spatially sparse. The issue with this approach is clearly illustrated in Figure 6.7, which shows a dense 2D segmentation label with areas without corresponding triggered events.

Networks operating on a sparse representation, such as PointNet++, are unable to predict results at positions where no events have been triggered. Furthermore, this type of 2D comparison ignores the fact that multiple events may have been triggered at a single spatial position within the selected time window. It is therefore challenging to make a comparison for semantic segmentation on this dense 2D basis.

For this reason, the obtained 2D labels were propagated back to the underlying event stream. By evaluation on the basis of *individual events*, the number of triggered events for each predicted label is also directly taken into account.

In the evaluation process, a confusion matrix was created and standard metrics for segmentation were derived. These were precision, recall, and F1 score, each calculated per class. The F1 score is defined as the harmonic mean of precision and recall:

$$\text{F1 score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{6.4}$$

with TP, FP, FN for the number of true positives, false positives and false negatives. The weighted F1 score was used to account for class imbalances. Here, metrics are calculated for each label, and their average is weighted by support.

In order to effectively compare the different networks considering the total number of 10 classes, the classes were further grouped as follows:

- Background: `BACKGROUND` (sensor noise)
- Objects: `PERSON`, `DOG`, `BICYCLE` and `SPORTSBALL`
- Environmental influences: `RAIN`, `INSECT`, `BIRD`, `TREE_CROWN`, and `TREE_SHADOW`

6.3.3 PointNet++ Optimization

Key architectural aspects with high impact on the performance of the PointNet++ model were identified and ranked as follows:

1. Number of layers
2. Number of points sampled in the first Set Abstraction layer
3. Variation of spatio-temporal scaling
4. Forcing local neighborhoods to have more or less temporal context

Optimization Strategy

Each aspect was tested individually while keeping the other aspects constant. A greedy approach was used in which parameters were tested sequentially and then held constant for subsequent testing.

Number of layers:

The number of layers describes the number of SA layers and FP layers, respectively. It was chosen from $\{3, 4, 5, 6\}$. Starting point was the default model for semantic segmentation provided by the authors of PointNet++, which was adapted to a different number of layers.

The first tests utilized the S^T_{cube} scaling method, which was also mentioned by Qi et al. in [Qi et al., 2017a,b]. The conducted test showed that varying the number of layers had a negligible effect. Therefore, the following experiments employed the smallest and fastest configuration with three layers.

Number of points sampled in the first SA layer:

Next, the number of points selected by Farthest Point Sampling for the initial SA layer was modified. In addition to the default of 1024 points, 512, 2048 and 3072 points were tested. The number of points in the subsequent layers were not changed.

While increasing the number of points from 512 to 1024 to 2048 had a significant impact, the improvement from sampling 3072 points was negligible and took almost 50 % more time for inference. Therefore, 2048 points were chosen for further testing.

Variation of spatio-temporal scaling:

Next, the defined spatio-temporal scaling variations were tested (see Equation 6.1 – 6.3 in Section 6.1.2).

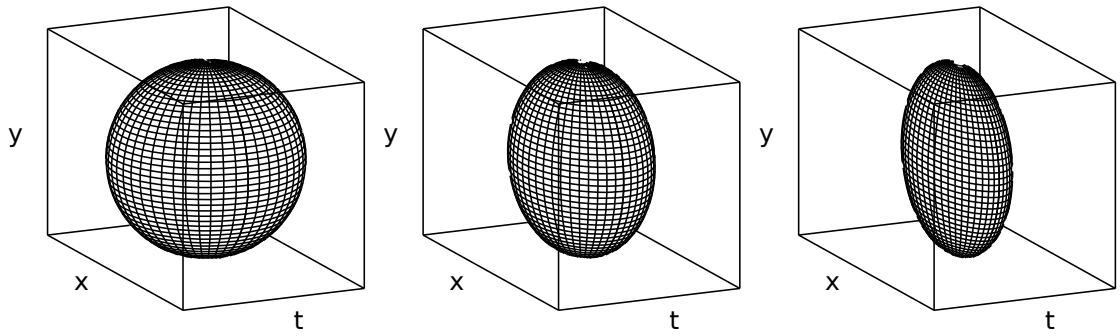
For this, the radii of the SA layers were adjusted to the axis with the largest range of values. Due to the shorter time axis, the full time interval was allowed for neighborhood building. Both alternative spatio-temporal scalings, utilizing either the S^T_{tScaled} or the S^T_{native} scaling, achieved substantially better results than the previous configuration.

Forcing local neighborhoods:

The idea of distinguishing between the spatial and temporal axes when selecting a radius based on varying resolutions was also discussed in [Bi et al., 2019]. For the considered data, the temporal axis is always much shorter than the spatial axes, but has a much higher resolution.

For the next tests, the time component was given a higher weight (γ) in the Euclidean distance measure:

$$d_{i,j} = \sqrt{\alpha \cdot (x_i - x_j)^2 + \beta \cdot (y_i - y_j)^2 + \gamma \cdot (t_i - t_j)^2} \tag{6.5}$$



(a) Equal α, β, and γ weight (b) Increased γ weight (c) Further increased γ weight

Figure 6.8: Consideration of neighborhoods through the use of higher γ in distance calculations (from [**Bolten** et al., 2022a]).

The effect is shown in Figure 6.8 for three levels. First, all axes have the same weight ($\alpha = \beta = \gamma = 1.0$), which results in a sphere covering the entire time axis. Second and third, the sphere is compressed with respect to the time axis.

Tests have shown that decreasing the time interval has minimal initial impact, e.g. when the time axis weight γ is 3.2, resulting in equivalent distributions for the space and time axes. However, when the time interval is reduced significantly, as in the case of a γ weight of 20, performance decreases.

The final network configuration is shown in Table 6.1. The configuration for each individual optimization step can be found in Section E.1.1 in the Appendix.

6.3.4 Results and Comparison

The results of the PointNet++ optimization process are summarized in Table 6.2, while the results of the 3D vs. 2D processing comparison are summarized in Table 6.3.

PointNet++ Optimization and Segmentation Quality (see Section 6.3.3)

Table 6.2 shows the F1 scores from the optimization experiments, summarized for the categorized object classes. With an overall F1 score of about 0.936, the optimized network achieves satisfactory results. For a qualitative evaluation of the results, examples segmentations are shown in Figure 6.9. Here, the processed patches are projected to 2D frames and the resulting labeling is represented by false colors. Detailed F1 score results for each class can be found in Section E.1.2 of the Appendix.

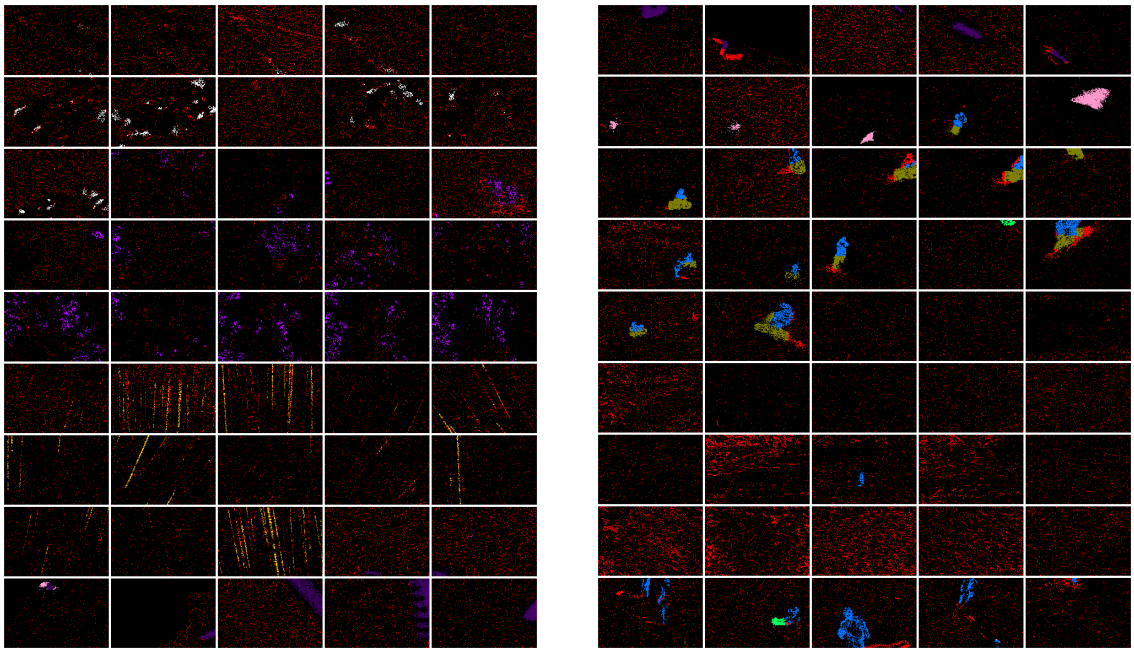
To further evaluate the ability of PointNet++ to segment, the learned features were taken into account. The basic requirement is that good features should be discriminative. This means that features of the same object category should be close together, while features of different object categories should be far apart. For this purpose, the feature vectors generated by PointNet++ before the last fully connected layer, which generates the final probability distribution vector, were considered. For the selected network configuration, this is a 128-dimensional feature vector per input point.

The feature vectors were computed for five thousand randomly sampled input points per class and projected onto a 2D plane using t-SNE [Van der Maaten and Hinton, 2008]. Figure 6.10 shows the visualization of this latent feature space. It indicates that the desired classes can be well distinguished by the trained model.

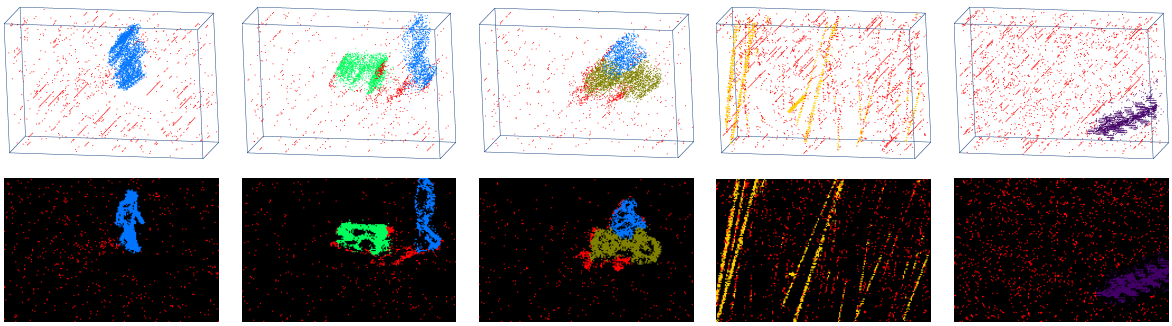
The results obtained from the optimized PointNet++ configuration were compared with the proposed 3D point cloud network variants and the 2D baseline.

Configuration	Parameter details
Number of input events	4096 events
Number of layers	3 layers
Number of points in first SA layer	2048 points
Spatio-temporal scaling	S^T_{native} variant
Local neighborhood weighting	$t_{\text{weight}} = 1$
Detailed configuration PointNet++(4096, 3L)	SA(2048, 9.6, [32, 32, 64]) → SA(256, 28.8, [64, 64, 128]) → SA(16, 76.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])

Table 6.1: Optimized PointNet++ configuration summary (compare to network description syntax used in [Qi et al., 2017b]).



(a) Segmented Patches-of-Interest projected into 2D frames (adapted from [**Bolten** et al., 2022a])



(b) Magnified results (top row: 3D space-time event clouds; bottom row: 2D false color frame projections; from [**Bolten** et al., 2023b])

Figure 6.9: False color examples for PointNet++ semantic segmentation results.

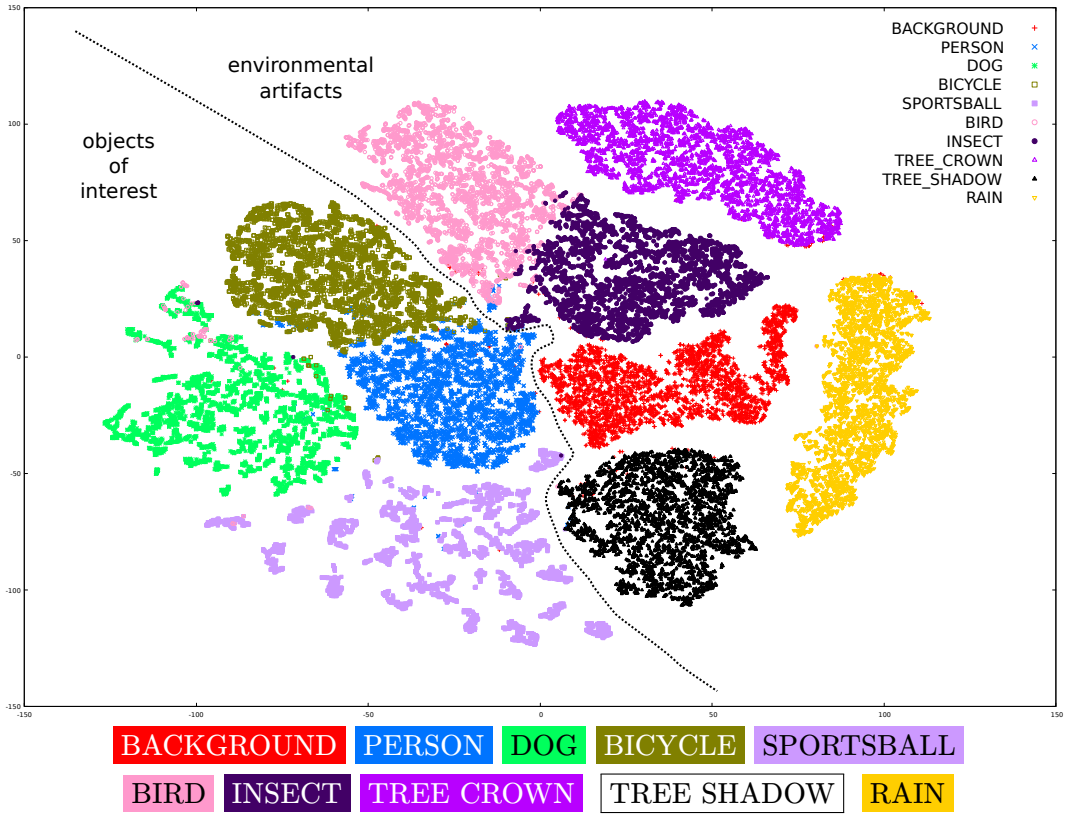


Figure 6.10: Two-dimensional t-SNE visualization of the PointNet++'s feature space used for segmentation (adapted from [**Bolten** et al., 2023b]).

3D Processing Variants: (see Section 6.1.1)

The A-CNN and LSANet network variants were able to slightly outperform PointNet++ in the important non-background categories, while the inference time remained almost unchanged. In contrast, SpiderCNN produced significantly worse results, suggesting the effectiveness of the set-abstraction framework used by the other methods. This is due to SpiderCNN's approach of avoiding sampling and interpolation.

Table 6.3a summarizes the F1 scores achieved for these network variants in comparison to vanilla PointNet++.

2D Processing Baseline: (see Section 6.2.1)

The *Binary* frame encoding yielded the worst results in the 2D Mask R-CNN comparison, regardless of whether a ResNet50 or ResNet101 was used as the feature extraction backbone. Since this encoding preserves the least amount of information, this result is not unexpected.

The three-channel color frame encodings *Polarity* and *MTC* showed significantly better results for non-background classes compared to the *Frequency* encoding. The results obtained from the *Polarity* and *MTC* encodings were comparable to each other, but they were generally inferior to the vanilla PointNet++ results.

The summary of achieved F1 scores is shown in Table 6.3b,c.

Comparing PointNet++ and Mask R-CNN, the selected PointNet++ configuration contains about 101 or 145 times less trainable weights, depending on the 2D backbone used (see Table 6.4). This results in a faster inference speed of up to a factor of ≈ 2.95 (see Table 6.5).

Configuration	Background	Objects	Environmental influences	Overall
(a) PointNet++ layer experiments				
3 layers	0.952	**0.741**	0.755	0.900
4 layers	0.953	0.721	0.763	0.900
5 layers	0.954	0.737	0.775	0.904
6 layers	**0.955**	0.723	**0.781**	**0.905**
(b) PointNet++ point count experiments				
n = 512	0.948	0.704	0.732	0.890
n = 1024	0.952	0.741	0.755	0.900
n = 2048	0.958	0.746	**0.806**	0.914
n = 3072	**0.959**	**0.750**	0.805	**0.915**
(c) PointNet++ input scaling experiments				
S^T_{cube}	0.958	0.746	0.806	0.914
S^T_{tScaled}	0.966	**0.817**	0.849	0.933
S^T_{native}	**0.968**	0.806	**0.863**	**0.936**
(d) PointNet++ spatio-temporal scaling experiments				
weight $\gamma = 1$	**0.968**	0.806	**0.863**	**0.936**
weight $\gamma = 3.2$	0.966	**0.807**	0.858	0.934
weight $\gamma = 20$	0.960	0.788	0.814	0.920

Table 6.2: PointNet++ optimization on DVS-OUTLAB dataset shown as weighted F1 scores.

Configuration	Background	Objects	Environmental influences	Overall
(a) 3D network variants				
PointNet++	**0.968**	0.806	**0.863**	0.936
A-CNN	**0.968**	0.823	0.862	**0.938**
LSANet	**0.968**	**0.826**	0.857	0.936
SpiderCNN	0.952	0.724	0.735	0.895
(b) Mask R-CNN with ResNet50 backbone				
Binary	0.949	0.821	0.843	0.911
Polarity	**0.953**	0.844	**0.873**	0.922
Frequency	0.952	0.828	0.863	0.918
MTC	**0.953**	**0.848**	0.870	**0.923**
(c) Mask R-CNN with ResNet101 backbone				
Binary	0.947	0.816	0.835	0.907
Polarity	**0.953**	0.842	**0.875**	**0.923**
Frequency	0.952	0.833	0.861	0.918
MTC	0.950	**0.844**	0.862	0.918

Table 6.3: Weighted F1 score results of 3D network variants and 2D Mask R-CNN baseline on DVS-OUTLAB dataset.

Configuration	Number of trainable parameters
(a) PointNet++	
PointNet++(*, 3L)	437,930
PointNet++(*, 4L)	967,594
PointNet++(*, 5L)	2,418,602
PointNet++(*, 6L)	7,285,162
(b) 3D network variants	
LSANet(*, 3L)	556,810
SpiderCNN(*, 3L)	1,080,798
A-CNN(*, 3L)	2,113,706
(c) Mask R-CNN	
Mask R-CNN(ResNet50)	44,646,734
Mask R-CNN(ResNet101)	63,664,974

Table 6.4: Trainable network parameter comparison.

Configuration	Average inference time in ms
(a) PointNet++	
PointNet++(512, 3L)	13.3 $\pm$ 0.5
PointNet++(1024, 3L)	23.1 $\pm$ 0.4
PointNet++(2048, 3L)	43.0 $\pm$ 0.7
PointNet++(3072, 3L)	62.7 $\pm$ 1.2
(b) 3D network variants	
LSANet(2048, 3L)	43.5 $\pm$ 1.0
SpiderCNN(*, 3L)	37.6 $\pm$ 0.9
A-CNN(2048, 3L)	41.3 $\pm$ 0.9
(c) Mask R-CNN	
Mask R-CNN(ResNet50)	114.9 $\pm$ 4.2
Mask R-CNN(ResNet101)	127.1 $\pm$ 4.1

Table 6.5: Network runtime comparison with batch size of one (Intel Xeon Gold 6226R CPU, NVIDIA Quadro RTX6000).

Chapter 7

Semantic Segmentation: Frame and Voxel Representations

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2023c). Semantic Segmentation on Neuromorphic Vision Sensor Event-Streams Using PointNet++ and UNet Based Processing Approaches. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 178. INSTICC, SciTePress

The challenges of generating event-wise semantic segmentation for Dynamic Vision Sensor data in an outdoor environment are further explored. In addition to traditional 2D frames and 3D space-time event clouds, additional event representations were evaluated. These include dense highly multichannel frames and 3D voxel grids. By applying UNet-based processing to these alternative representations, the feasibility comparison of 3D point cloud approaches for semantic segmentation was extended.

The main contributions presented in this chapter are as follows:

- Systematic comparison of single-channel and high-multichannel 2D event representation as well as 3D event stream voxelization.
- Extension of the previous analysis and evaluation by including a UNet network structure to generate a semantic segmentation based on these representations.

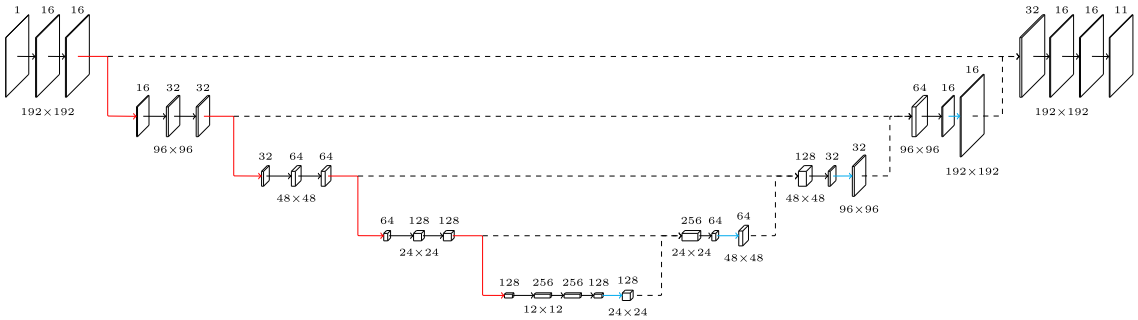


Figure 7.1: 2D UNet configuration for DVS-OUTLAB dataset (from [**Bolten** et al., 2023c]).

7.1 Proposed Processing

The work of Chapter 6 was extended with respect to further event representations and network structures for processing.

7.1.1 Methods

The 3D point cloud-based processing with PointNet++ (see Section 6.1.1) was further compared to the results of an UNet-based processing.

UNet [Ronneberger et al., 2015]: Convolutional Networks for Biomedical Image Segmentation

The UNet architecture has its origin in the segmentation of medical images, but it has been successfully applied to a wide range of applications (e.g., [Pohle-Fröhlich et al., 2019; McGlinchy et al., 2019; Liu and Qian, 2021]). It is a convolutional neural network with a U-shaped architecture that enables precise pixel-level semantic segmentation.

The architecture is divided into an encoder and a decoder part. Within the encoder, the spatial resolution is reduced by convolution and max-pooling, while the number of feature channels is increased. This extracts high resolution and deep contextual features. In the second part, the decoder, the original resolution is restored by upsampling. By increasing the resolution of the output in this way, the decoder learns to produce an output with precise localization.

The UNet architecture combines the feature channels from before downsampling and after upsampling through skip connections, allowing the network to propagate and recombine context information and reconstruct localization information.

A visualization of an example configuration is given in Figure 7.1.

7.1.2 Processing Pipeline

While the 3D representation of the (x, y, t) space-time event cloud is generated directly from the event stream, converted representations were used for comparison. For the following UNet-based experiments, 2D frame variants and 3D voxelization representations were considered.

Event Representations

As a baseline for further comparison, the binary projection of the event stream was included purely as a single-channel 2D frame. A highly multichannel frame representation was also included to better represent and exploit the temporal context of the DVS stream. Finally, the approach of interpreting the DVS event stream as a 3D spatio-temporal volume using

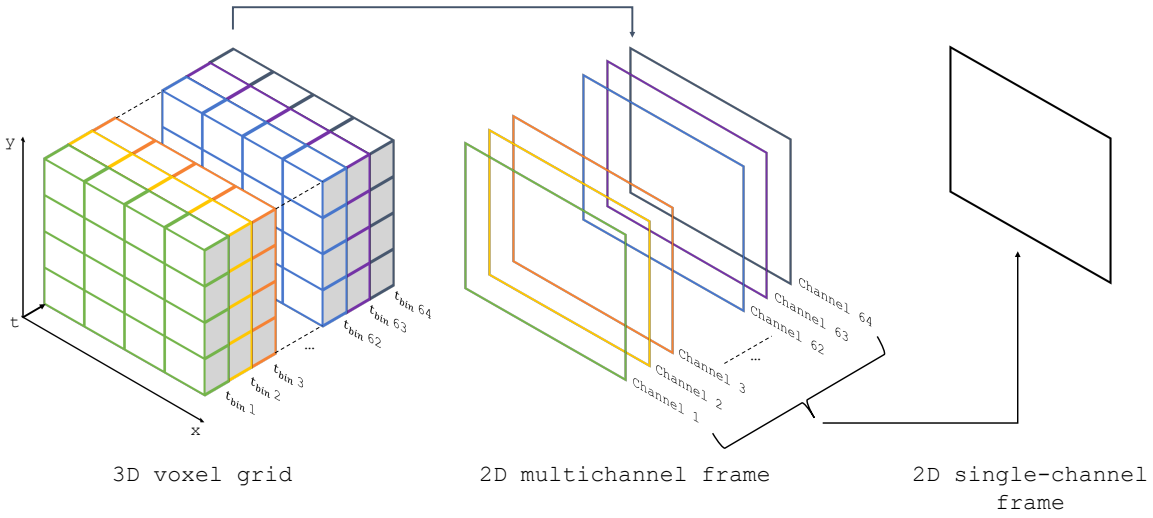


Figure 7.2: Event representations considered in UNet experiments.

voxelization was also evaluated. Further details for these representation variants are given in Figure III.4 and in Figure 7.2.

Preprocessing Steps

To ensure fair comparison, the representations were based on input data that has been pre-processed in the same way as the 3D reference point cloud. These generated representations were derived from the same temporal DVS event stream patches of 192 px × 128 px × 60 ms.

A 3D voxel histogram was created per pixel position, dividing the time-axis into 64 components (t_{bin}, as referred to in Figure 7.2), in order to construct the representations. The labels were converted to an equivalent voxel format.

The partitioning into 64-time components was chosen to ensure that the input dimension, being a power of two, always produces integer results throughout the downsampling and up-sampling processes in the subsequent UNet processing logic. Furthermore, for UNet processing, the spatial resolution was extended by zero-padding to quadratic inputs. This resulted in a resolution of 192 × 192 pixels for the DVS-OUTLAB data.

Furthermore, two versions of the data were generated and tested. One version included all events, while the other was spatio-temporally pre-filtered to reduce sensor noise and estimate the effect of noise reduction on semantic processing. For this purpose, a time-filter was applied to remove all events that were not supported by another event at the same spatial (x, y)-coordinates within the previous 10 ms.

7.2 Experiments and Evaluation

In the following, the quantitative evaluation of the proposed approaches is presented based on the DVS-OUTLAB dataset (see Chapter 4). A further comparison of the methods proposed in this chapter on a dataset from the autonomous driving application domain can be found in Section E.3 of the Appendix.

7.2.1 Network Training

The network layer configurations and training hyperparameters utilized in the experiments were as follows:

PointNet++ Processing Network:

The PointNet++ architecture was configured according to the parameters given in Table 6.1 on page 108. The network was trained with the same hyperparameters as defined in Section 6.3.1.

UNet Processing Network:

The UNet trainings were conducted using an Adam optimizer with a learning rate of 0.001 and exponential decay rate of 0.99 after each epoch. The batch size was set to six. A sparse categorical cross-entropy, weighted by the inverse frequency of class occurrence, was selected as the loss function to account for the class imbalances present in the datasets.

In all UNet experiments performed, the model was built with a depth of four blocks and a number of 16 filters in the first block, multiplied by two in each subsequent block. The 2D and 3D convolutions were configured with a kernel size of 3×3 and $3 \times 3 \times 3$, respectively. An illustration of the implemented 2D UNet configuration is given in Figure 7.1.

7.2.2 Evaluation Metrics

Unlike PointNet++, UNet-based processing methods may lead to predictions of a class for a spatial position with no DVS event triggered (referred to as the "void" background). In addition to the logic described in Section 6.3.2, the following simple postprocessing was performed before evaluation:

If a non-void class prediction is made, but there is no event present, the prediction is interpreted as void and ignored. If a void class prediction is made, but an event is present, this prediction is reinterpreted and considered as the dominant class for evaluation (class `BACKGROUND` for DVS-OUTLAB).

For a fair comparison of the considered 3D point cloud and voxel variants with the predictions resulting from the pure 2D variants, the results were further equalized for evaluation. This was necessary due to the higher number of possible errors in the predictions with a higher output dimension. The 3D predictions were projected along the t-axis by considering the most frequent prediction at each spatial position for comparison with the 2D results. Compared to the metrics given in Section 6.3.4, this postprocessing results in minor changes in the metrics for PointNet++.

7.2.3 Results and Comparison

The evaluation results of the PointNet++ processing and the proposed event representations combined with the 2D and 3D UNet processing are summarized in Table 7.1.

The PointNet++ approach achieved higher segmentation results compared to the 2D and 3D UNet approaches on the monitoring data considered in this work. In general, this is consistent with the comparison to Mask R-CNN presented in Chapter 6.

However, the further comparison of PointNet++ showed that it is sensitive to increased numbers of input events, as they occur for example in the field of autonomous driving (see Section E.3 in the Appendix). The success of PointNet++ is based on the processing of sufficiently representative events selected by Farthest Point Sampling and corresponding neighborhood formation. Since the capture of fine details is important for the processing, these capabilities decrease as the number of events to be processed increases. Adjusting the PointNet++ configuration with respect to the number of input and sampled events in the Set Abstraction layers process is limited by its computational complexity.

Configuration	Background	Objects	Environmental influences	Overall
(a) PointNet++ reference results				
PointNet++(4096, 3L)	**0.968**	0.816	**0.853**	**0.936**
(b) UNet results on unfiltered data				
UNet 2D 1-channel frame input	0.951	0.842	0.764	0.902
UNet 2D 64-channel frame input	0.958	0.847	0.780	0.912
UNet 3D voxel grid input	0.941	0.843	0.775	0.895
(c) UNet results on spatio-temporal time-filtered (10 ms) data				
UNet 2D 1-channel frame input	0.925	0.838	0.757	0.868
UNet 2D 64-channel frame input	0.938	**0.850**	0.826	0.897
UNet 3D voxel grid input	0.928	0.843	0.809	0.883

Table 7.1: Weighted F1 score results on DVS-OUTLAB dataset.

In both application domains, UNet-based processing performed slightly better on unprocessed raw data than on spatio-temporal pre-filtered data. An improvement of the UNet-based results was observed with an increase in the number of 2D frame channels used, i.e. the adaptation from a single-channel projection to the 64-multichannel frame. In comparison, there were only minor differences when using the 3D voxel grids.

The Dynamic Vision Sensor generates an output stream that is sparse. Considering the data provided by the DVS-OUTLAB dataset and the voxel representation being evaluated, less than 1 % of the voxels were populated with events. The use of 3D convolutions rapidly increases and "blurs" the set of non-zero features in the processing. Therefore, incorporating processing based on submanifold sparse convolutions [Graham and van der Maaten, 2017] and adapting the UNet network to a sparse voxel network for semantic segmentation [Graham et al., 2018; Najibi et al., 2020] is an interesting task for further work.

Chapter 8

Instance Segmentation: Comparison of Representations

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2024). Instance Segmentation of Event Camera Streams in Outdoor Monitoring Scenarios. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 452 – 463. INSTICC, SciTePress

The processing is extended to segment instances in order to be able to distinguish between objects of the same class that are close to or occluded by each other. In the spirit of the previous chapters, further consideration was given to space-time event clouds, voxel and frame-based representations. A systematic evaluation of different state-of-the-art deep learning-based instance segmentation approaches was performed, addressing challenges in the context of event-based outdoor monitoring.

The main contributions presented in this chapter are as follows:

- Introduction of a new density-based event preprocessing to extract spatially adapted regions of interest for processing.
- Inclusion of a baseline comparison method based on semantic segmentation and clustering.
- The first systematic evaluation of various state-of-the-art deep learning-based instance segmentation approaches in the context of event-based monitoring was conducted.

(a) Baseline method (b) Point-based methods (c) Voxel-based method (d) Frame-based methods

Figure 8.1: Overview of the included instance segmentation processing approaches in comparison (from [**Bolten** et al., 2024]).

Introduction

The sparse, unordered, and asynchronous output of Dynamic Vision Sensors poses challenges for processing in terms of classical computer vision approaches (see also Section 1.4.1 on page 11). In the considered use case scenario, unconstrained real-world factors, small object sizes, and occlusion pose additional processing challenges. To gain deeper insight into the use of the monitored area, instance segmentation approaches were compared.

As mentioned, research on event-based segmentation is not as extensive, but it is possible to apply methods from other domains. For example, the development of methods for traditional 2D frame-based processing is more advanced. Libraries such as *Detectron2* [Wu et al., 2019b] are available, providing state-of-the-art recognition and segmentation algorithms as well as pre-trained models. Basically, two different approaches can be distinguished here. In proposal-based approaches, objects are first detected using bounding box techniques and then segmented. A well-known example of this is *Mask R-CNN* [He et al., 2017].

On the other hand, the notable *YOLO* family [Redmon et al., 2016; Jocher et al., 2023] directly predicts bounding boxes and class probabilities for objects in a single pass. Along with the use of pixel-level grouping or clustering techniques to form instances, such as [Xie et al., 2020, 2022; Wang et al., 2020], this provides proposal-free methods.

Since the event stream can be interpreted as a point cloud, 3D processing methods are also interesting for inferring instances. Basically, 3D methods can also be distinguished into proposal-based and proposal-free approaches. The former decompose the segmentation problem into two sub-challenges: detecting objects in 3D and refining the object masks [Yang et al., 2019; Engelmann et al., 2020]. The latter typically omit the detection part and try to obtain instances by clustering after semantic segmentation (e.g., following the assumption that instances should have similar features) [Zhao and Tao, 2020; Jiang et al., 2020; Chen et al., 2021].

8.1 Proposed Processing

As before, the event data stream from Dynamic Vision Sensors is converted into alternative representations for processing. The construction of space-time event clouds, their voxelization and conversion into classical 2D frames for further processing were considered (see Figure III.3 on page 90).

In the following, the considered deep learning frameworks used for instance segmentation on top of these representations are briefly outlined and introduced.

8.1.1 Point Cloud-based Processing Methods (see Figure 8.1b)

JSNet [Zhao and Tao, 2020]: Clustering-based Processing

JSNet consists of four main components: a shared feature encoder, two parallel branch decoders, feature fusion modules for each decoder, and a joint instance and semantic

segmentation (JISS) module. High-level semantic features are learned by PointNet++ [Qi et al., 2017b] and PointConv [Wu et al., 2019a] architectures and are further combined with low-level features for more discriminative values. The JISS module transforms the semantic features into an instance embedding space, where instances are formed by applying a mean-shift clustering.

3D-BoNet [Yang et al., 2019]: Proposal-based Processing
3D-BoNet is designed for a single-stage, anchor-free instance segmentation in 3D point clouds. It uses a PointNet++ [Qi et al., 2017b] backbone to extract local and global features, followed by two branches: one for instance-level bounding box prediction and another for point-level mask prediction. The bounding box prediction branch is a key component, generating unique, unoriented rectangular bounding boxes without predefined spatial anchors or region proposals. The subsequent point-mask prediction branch uses these boxes and features to generate point-level binary masks for valid instances, distinguishing them from the background.

8.1.2 Voxel-based Processing Method (see Figure 8.1c)

SoftGroup [Vu et al., 2022]: Clustering and Proposal-based
SoftGroup attempts to combine the strengths of proposal-based and grouping-based methods while addressing their limitations. First, a bottom-up stage uses a point-wise prediction network to generate high-quality object proposals by grouping on the basis of soft semantic scores. This stage involves processing point clouds to generate semantic labels and offset vectors, which are then refined into preliminary instance proposals using a soft grouping module. Second, the top-down refinement stage improves the generated proposals by extracting corresponding features from the backbone. These features are used to predict the final results, including classes, instance masks, and mask scores.

8.1.3 Frame-based Processing Methods (see Figure 8.1d)

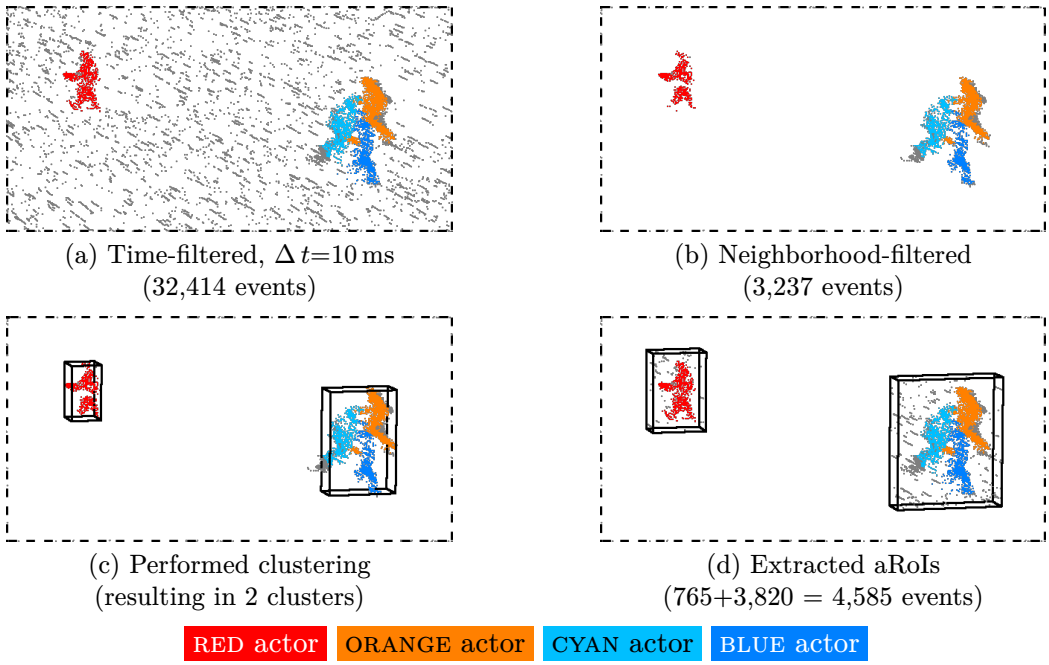
Mask R-CNN [He et al., 2017]: Proposal-based Processing
Proposal-based processing is considered to be the baseline technique for frame-based instance segmentation [Sharma et al., 2022]. Therefore, Mask R-CNN was also included in this evaluation.

For a summary of the components of Mask R-CNN, see Section 6.2.1 on page 103.

YOLO v8 [Jocher et al., 2023]: Proposal-free Processing
At the time of evaluation, YOLO v8 was the latest version of the popular single-shot detection method and aims to improve accuracy and efficiency over previous versions. A major change is that YOLO v8 is an anchor-free model, meaning that object centers are predicted directly instead of the offset from an anchor box. This typically results in fewer predictions and better, faster non-maximum suppression.

8.2 Preprocessing and Baseline

Following the spatio-temporal filter analysis performed in Section 3.3, a time-filter was applied prior to any further processing steps in order to account for the high level of background activity of the CeleX-IV sensor used. Each event that is not supported by another event at the same (x, y)-position within the preceding Δt was removed.



(a) Time-filtered, Δt=10 ms (32,414 events)

(b) Neighborhood-filtered (3,237 events)

(c) Performed clustering (resulting in 2 clusters)

(d) Extracted aRoIs (765+3,820 = 4,585 events)

Figure 8.2: Adaptive RoI generation steps. Provided false color corresponds to ground truth label of included dataset instances. Adaptive RoIs are represented by bounding boxes. Event counts shown refer to full, uncropped scene as shown in Figure 5.9 on page 65 (adapted from [**Bolten** et al., 2024]).

8.2.1 Adaptive Region-of-Interest Extraction (aRoI)

According to the functional paradigm of event cameras, the scene is already separated into foreground and background at sensor level for moving objects when using a static sensor. However, a straightforward selection of events triggered by object motion is often not possible due to the remaining high noise level. Therefore, to separate and select dense regions of events for further processing, the following size-adaptive Region-of-Interest algorithm is proposed:

1. Extended spatio-temporal filtering
   First, an additional spatio-temporal filtering stage based on the *Neighborhood-Filter* (see Section 3.3.2) is applied. As shown before, this filter achieves an almost complete removal of background activity events at the cost of events from instances (see Section 3.3.3). This processing step is shown in Figure 8.2a → b.

2. Hierarchical single-linkage clustering [Müllner, 2013]
   The remaining events are hierarchically clustered into regions based on the Euclidean distance. Clustering is controlled by a predefined cutoff distance d_{cut} that prevents spatially distant clusters from merging. Resulting clusters with a number of events less than $\min_{\#\text{events}}$ are discarded in this step. An example for this step is displayed in Figure 8.2c.

3. Bounding box expansion and filter reset
   In order to account for the events of objects that may have been filtered (cf. the feet of the red actor in Figure 8.2), the bounding box of each cluster is expanded by $\text{bbox}_{\text{offset}}$ pixels. Within each expanded bounding box, the event stream is reset to the original time-filtered event stream, reactivating the events removed by the second restrictive filtering step. Each resulting bounding box forms an aRoI. This is shown in Figure 8.2d.

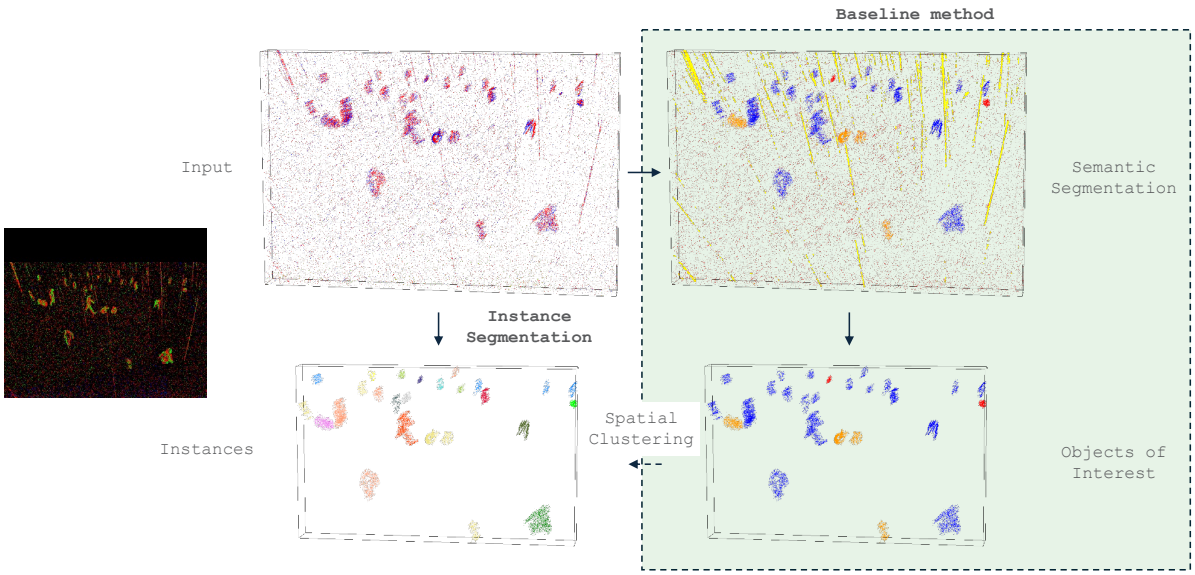


Figure 8.3: Processing visualization of the baseline segmentation clustering approach.

This processing results in Regions-of-Interest of variable spatial size, where spatially separated objects are in their own aRoI and groups of objects share an aRoI without being sliced into separate parts.

8.2.2 Baseline: Semantic Segmentation and Clustering

As a basic approach, the use of hierarchical event clustering, extended by the application of prior semantic segmentation, was included in the comparison. This process is shown in Figure 8.1a and Figure 8.3.

For this semantic segmentation step, vanilla PointNets++ [Qi et al., 2017b] were trained as described in Chapter 6, using the generated aRoIs as input. Clustering is subsequently applied separately to the events of each semantic class based on the predicted labels to group the predictions into individual instances.

The clustering cutoff distance d_{cut} in this step was individually selected per semantic class based on the maximum Euclidean distance between nearest neighbor pixels within the ground truth instances (see $\text{dist}(\text{NN}_{\text{inst}})$ in Table 8.1). This selection ensures that all events of a single instance are grouped together by this baseline approach.

8.3 Experiments and Evaluation

Next, the methodology and results of the comparative evaluation of the methods presented in Section 8.1 are described. All event representations were based on 60 ms sliding time windows of the data. The configuration for the performed preprocessing was as follows:

- Spatio-temporal time-based pre-filtering with threshold Δt=10 ms
- Region-of-Interest generation with
  - $d_{\text{cut}} = 29.0$ px
  - $\min_{\#\text{events}} = 50$ events
  - $\text{bbox}_{\text{offset}} = 10$ px

For the experiments, the proposed training, validation and tests splits of the DVS-iOUTLAB (see Section 4.3) and N-MuPeTS (Section 5.4) were used.

Annotation	$\max(\text{dist}(\text{NN}_{\text{inst}}))$	$\text{avg}(\#\text{Pixel}_{\text{inst}})$
(a) DVS-iOUTLAB		
PERSON	$\approx$ 23.09 px	332.28 px
DOG	$\approx$ 16.49 px	360.31 px
BICYCLE	$\approx$ 21.59 px	450.28 px
SPORTSBALL	$\approx$ 5.39 px	116.05 px
(b) N-MuPeTS		
PERSON	$\approx$ 22.14 px	511.80 px

Table 8.1: Object instance statistics.

The DVS-iOUTLAB data composes challenges from a multi-class, multi-instance scenario combined with real sensor noise. While the N-MuPeTS dataset contains only the single object class PERSON, it also provides processing challenges due to included object occlusions (with infrastructure and other people), similar body shapes, different body poses, and different motion/interaction patterns. Scenes with objects in close proximity (especially intersections) are particularly challenging to segment. In addition to this, there is also sensor noise in the data.

8.3.1 Network Configurations and Input

Network configurations and hyperparameters have been left at their default values where possible.

As already mentioned, for point and voxel-based methods, the temporal scaling of the input data is of particular interest, as it has a significant impact on the computation of spatio-temporal distances and neighborhoods. In the experiments performed, the time information was scaled and represented in milliseconds.

Event Representations

Space-time event clouds are naturally formed from the (x, y, t)-coordinates of the events themselves. However, while the spatial shape of the input event cloud can vary, and therefore the generated aRoI can be used directly as a basis, the deep learning processing techniques require a fixed number of events as input for training (see also Section 6.1.2). Therefore, the event clouds were sampled to a fixed number of events by random choice. The sizes of 1024 and 2048 events per aRoI served as sampling targets, since these powers of two are closest to the mean event counts of the generated aRoIs (more detailed event count statistics are given in Section E.2 in the Appendix). For aRoIs with fewer events, doublets were generated to achieve the desired number, following the original logic of PointNet++ processing that forms the basis of the point-based methods under study. The selected grouping radii and the configuration of the set abstraction layers of the networks were adapted from the results of the optimization process performed for PointNet++ in Section 6.3.3.

For voxel grids, the shape of the data is defined by the size of the voxels themselves, not by the number of events. Therefore, event subsampling per aRoI was not performed. Instead, the time-filtered aRoIs were used directly as input.

For frame-based processing, the 2D encodings were created using the full usable spatial resolution of the sensor, since the frame-based processing also requires a fixed input resolution. Examples and further details of the performed event representations are given on page 90 and the following.

8.3.2 Evaluation Metrics

Prior semantic segmentation is used by some of the processing approaches under evaluation. Therefore, semantic segmentation quality metrics are also reported. For this the F1 scores, defined as the harmonic mean of precision and recall, are reported as a weighted average using the given support per class on a per-event basis (see also Section 6.3.2).

Regarding the instance segmentation quality, the standard COCO dataset [Lin et al., 2014] and challenge metrics[17] are reported, including *mean average precision* $\text{mAP}_{0.5}^{0.95}$, which is the precision averaged over the intersection over union (IoU) threshold range from 0.5 to 0.95 with a step size of 0.05, as well as the $\text{mAP}^{0.5}$ and $\text{mAP}^{0.75}$ at fixed IoU values. For reproducibility, the metric implementations from [Detlefsen et al., 2022] were used for all reported results.

IoUs were calculated based on segmentation masks rather than bounding boxes. For comparability between the different methods, these masks were formed and evaluated in 2D.

The evaluation is based on the events included in the constructed aRoIs. Since the spatial shape can vary between the different encoding variations (aRoI size vs. fixed and full frame resolution), only the areas covered by the aRoIs were considered and included for frames in the metric calculation.

8.3.3 Application Results and Comparison

The evaluation was performed on datasets derived from a DVS-based monitoring application scenario as introduced previously. Therefore, typical application-oriented challenges such as occlusions and spatially close objects are included in the considered scenes. The presented results focus on these challenges.

For the N-MuPeTS dataset, an additional and intentionally challenging subset of the test data was constructed. This test subset restricts the scenes to a selection in which at least one actor is occluded or actors are spatially very close to each other. In terms of annotations, this means a subset of test time windows from the N-MuPeTS dataset in which at least one actor is labeled with one of the following dataset annotations (see Section 5.4):

- OCCLUSION
- CROSSING
- MEET
- SIDEBYSIDE
- HELIX
- FAR

Results

Table 8.2 shows the metric results for the DVS-iOUTLAB dataset. For the N-MuPeTS dataset, Table 8.3 reports the results on this intentionally challenging subset. The metrics for the complete N-MuPeTS test set, as well as individual results per class for DVS-iOUTLAB, are reported in Section E.2.2 in the Appendix.

Segmentation baseline:

The segmentation baseline approach depends on the quality of the semantic segmentation performed. In this aspect, it achieved very good F1 scores on both datasets.

As expected, in terms of instance segmentation and the corresponding mAP results, it often failed with merge errors, because instances of *same* classes that are very close to each other were clustered together. This is especially true for the selected challenging

[17] `https://cocodataset.org/#detection-eval`

Network	Configuration	Semantic quality weighted F1 score	Instance quality mIoU	$mAP^{0.95}_{0.5}$	$mAP^{0.5}$	$mAP^{0.75}$
(a) Baseline method: PointNet++ with spatial clustering						
PointNet++ Clustering	in 2048 events	0.94	0.80	0.57	0.71	0.62
	in 1024 events	0.93	0.82	0.58	0.71	0.61
(b) Space-time event cloud-based methods						
JSNet	4 layers in 2048 events	0.95	**0.89**	0.81	0.87	0.86
	4 layers in 1024 events	0.92	0.85	0.70	0.77	0.75
3D-BoNet	4 layers in 2048 events	0.94	0.84	0.71	0.81	0.78
	4 layers in 1024 events	0.93	0.83	0.70	0.80	0.76
(c) Voxel-based method						
SoftGroup	voxel grid	**0.97**	0.86	**0.88**	**0.98**	**0.96**
(d) Frame-based methods						
Mask R-CNN	polarity	0.92	0.78	0.62	0.96	0.72
	MTC	0.92	0.78	0.61	0.96	0.71
YOLO v8	polarity	0.92	0.79	0.60	0.93	0.66
	MTC	0.91	0.80	0.58	0.89	0.65

Table 8.2: Segmentation results on DVS-iOUTLAB test set (60 ms event time window).

Network	Configuration	Semantic quality weighted F1 score `NOISE`	`PERSON`	Instance quality `PERSON` mIoU	$AP^{0.95}_{0.5}$	$AP^{0.5}$	$AP^{0.75}$
(a) Baseline method: PointNet++ with spatial clustering							
PointNet++ Clustering	in 2048 events	0.91	**0.95**	0.74	0.25	0.42	0.25
	in 1024 events	0.91	**0.95**	0.74	0.25	0.41	0.24
(b) Space-time event cloud-based methods							
JSNet	4 layers in 2048 events	**0.92**	**0.95**	0.82	0.54	0.79	0.57
	4 layers in 1024 events	0.91	0.94	0.80	0.46	0.70	0.48
3D-BoNet	4 layers in 2048 events	0.91	**0.95**	0.80	**0.56**	0.77	0.59
	4 layers in 1024 events	0.89	0.93	0.75	0.42	0.62	0.44
(c) Voxel-based method							
SoftGroup	voxel grid	0.84	0.92	**0.83**	0.55	0.70	0.57
(d) Frame-based methods							
Mask R-CNN	polarity	0.80	0.89	0.72	0.41	0.80	0.41
	MTC	0.80	0.89	0.72	0.42	0.80	0.43
YOLO v8	polarity	0.83	0.92	0.70	0.55	**0.87**	**0.61**
	MTC	0.83	0.92	0.70	0.54	0.86	0.60

Table 8.3: Segmentation results on *challenging sequences* of N-MuPeTS test subset (60 ms event time window).

test subset of the N-MuPeTS dataset. This can be clearly seen in the metric difference of this approach between the two datasets.

Point-based processing:

Segmentation tended to fail when an aRoI was significantly larger than average. These regions occur when many objects are spatially very close to each other, so that they were clustered into a single input aRoI during preprocessing. The unsampled event count in these regions deviates strongly from the overall mean, so that the applied random event selection changes the spatio-temporal object densities and event neighborhoods substantially. For these error-prone aRoIs, JSNet-based processing mostly led to interpretation as BA event noise, while 3D-BoNet predicted better semantic values, but often proposed very large and merged object instance boundaries.

A simple postprocessing of the obtained results seems useful, since small errors in semantic segmentation often propagate in the form of small instances. Therefore, we recommend that instances consisting of only a few events be removed and ignored for further processing.

Voxel-based processing:

SoftGroup achieved very good results on the DVS-iOUTLAB dataset which includes spatially close but not intersecting objects. Considering scenes containing occlusions of objects of the same semantic class (as in N-MuPeTS, which are considered to be particularly difficult), it was observed that instances often merge in these cases.

Looking at the $\mathrm{mAP}^{0.5}$ value, the best overall result was obtained for DVS-iOUTLAB, while the value for N-MuPeTS was worse than all other high-level approaches.

Frame-based processing:

The IoU thresholding performed for mAP calculation is more punitive for frame-based mask prediction.

As described in Section 3.2, the sensor used provides 768×512 actively acquired pixels. This low spatial resolution of the DVS sensor results in small object sizes. This is also shown in Table 8.1. The $\mathrm{avg}(\#\mathrm{Pixel}_{\mathrm{inst}})$ value indicates the average projected object pixel size per instance in each dataset. Even a few mismatching pixels in the predicted masks will significantly lower the IoU score. Comparing the $\mathrm{mAP}_{0.5}^{0.95}$ and $\mathrm{mAP}^{0.5}$ (improvements up to $\approx$40 %) showed that the segmentation worked well, but was limited by the predicted accuracy of the pixel mask.

When detecting and separating occluded objects of the same semantic class, the selected Mask R-CNN tended to predict a mask containing only one object. YOLO v8 predicted better partial masks at the expense of multiple false predictions.

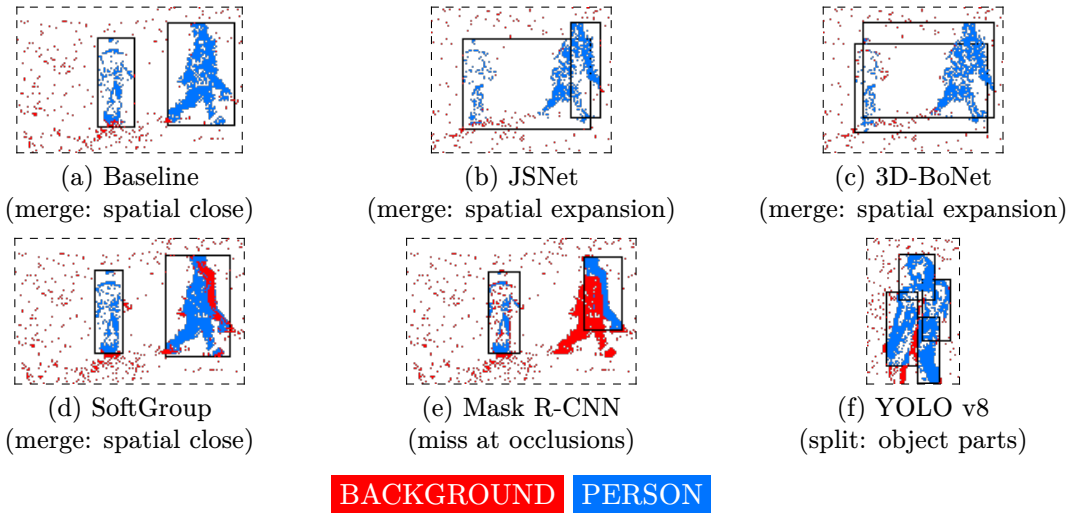
Figure 8.4 shows example segmentations in the form of false color images for the N-MuPeTS dataset. This illustrates the typical worst-case errors described above.

An instance segmentation could be effectively derived from the considered event representations and corresponding off-the-shelf processing approaches. From a practical point of view, the proposal-based point and voxel-based approaches require temporal normalization in addition to temporal scaling for training convergence. We recommend shifting the continuous event timestamps for each input aRoI between zero and the chosen sliding time window length. In Section 6.1.2 this is referred to as S_{native}^{T}.

The point-based approaches are inspired by and built on PointNet++ as a backbone. By sharing the MLPs per point, relatively small network structures are built (see Table 8.4). This

Network	Number of total parameters
Baseline PointNet++	437,285
JSNet	8,083,045
3D-BoNet	1,824,582
SoftGroup	30,836,090
Mask R-CNN	44,619,824
YOLO v8	3,264,396

Table 8.4: Number of trainable network parameters for DVS-iOUTLAB network configuration.



(a) Baseline (merge: spatial close)
(b) JSNet (merge: spatial expansion)
(c) 3D-BoNet (merge: spatial expansion)
(d) SoftGroup (merge: spatial close)
(e) Mask R-CNN (miss at occlusions)
(f) YOLO v8 (split: object parts)

Figure 8.4: Typical prediction *error cases* on N-MuPeTS displayed as false color aRoI-montage images (best viewed digitally zoomed; from [**Bolten** et al., 2024]).

feature is important when aiming for a sensor-near implementation where hardware resources are limited.

By using a submanifold sparse convolution [Graham et al., 2018], the voxel-based processing provided good results and can offer a good trade-off in terms of processing complexity. For applications where small compromises in pixel accuracy of segmentation are acceptable, classical frame-based processing provides a good starting point, while offering a wide range of well-established frameworks for processing.

Summary of Part III

A literature overview of event-based segmentation approaches was presented. The current state-of-the-art in event-based segmentation is not as advanced as in frame-based computer vision, although methods and approaches are continuously being developed. It is important to note that there is currently no ideal and universally accepted way to represent Dynamic Vision Sensor event stream data for processing. This is because no well-established format and processing has emerged.

Therefore, different event representations were tested for segmentation tasks suitable for the outdoor monitoring application use case of this work. These representations include types of different dimensionality, such as simple 2D frame projections, spatio-temporal voxel grids, and space-time event clouds.

For space-time event cloud processing, the number of events must be limited for practical reasons. A processing pipeline has been introduced that addresses this problem through a patching procedure. The complete pipeline combines the steps of filtering, subsampling, and spatio-temporal scaling for further application of deep learning-based 3D processing approaches. Using PointNet++, the pioneering work and foundation for point-based deep learning, the 3D processing network hyperparameters were optimized and the influence of spatio-temporal event cloud scaling on semantic segmentation results was investigated. This led to the proposal of a 3D data scaling variant and network parameters to be used.

The learned features of PointNet++ were analyzed and the obtained processing results were compared with results obtained using frame-based representations, including different frame encoding variants. Subsequently, the processing was compared with the results of well-known state-of-the-art CNN architectures. The obtained semantic segmentations using the 3D space-time event clouds achieved better results in terms of quality *and* runtime compared to the CNN-baseline approach.

In the following step, we extended the processing and comparison to include a voxel grid representation of the data. In addition, a high multichannel frame representation generated from the voxel grid and simple binary frame projections were included in the comparison. In the context of the application, UNet and PointNet++ were compared for their processing capabilities. The results show that PointNet++ achieved better results. Among the alternative representations, the highly multichannel frame representation proved to be the preferred option in these experiments.

The output stream generated by a Dynamic Vision Sensor is spatially sparse. As a consequence, a significant number of the voxels generated do not contain any data. The UNet-based processing was still able to achieve good results on this sparsely populated voxel representation. However, applying simple 3D convolutions quickly increases and blurs the set of non-zero features, making processing difficult. Furthermore, the utilization of 3D convolutions and their corresponding kernels leads to a larger and more complex UNet network architecture. This results in increased difficulty and time consumption during network training, as well as slower inference.

Finally, we extended the processing and comparison to segment object instances. The experiments included multiple state-of-the-art deep learning-based approaches that cover space-time event cloud, voxel-, and frame-based event input representations.

A novel preprocessing algorithm was proposed that addresses a previous limitation of the space-time event cloud processing methodology. The spatial patching procedure used, which was designed to meet the event count requirements, was modified to form patches of varying sizes. This prevents objects from being cut into multiple pieces prior to processing.

The segmentation was performed on two different datasets, including an intentionally challenging subset of the data to address the core challenges involved. The comparison performed shows differences in the results obtained by approaches using different representations. The space-time event cloud processing achieves good results. It is mostly limited in cases where the formed spatial patches deviate strongly from the mean in terms of event counts due to the inclusion of many objects. These cases occur when many objects are spatially close to each other. Voxel grid processing, on the other hand, performed worst when it comes to separating objects that are very close to each other. Frame-based methods have largely been hampered by the low spatial resolution of sensors, as even small errors in the result lead to significant metric penalties.

Overall, very good segmentation results can be achieved using standard processing approaches. These adaptations of existing algorithms for the event-based image processing domain by comparing and exploiting different representations is a promising approach to drive developments forward, since real-world outdoor applications of Dynamic Vision Sensors are still rare. The use of standard processing approaches is a suitable way to change this.

Part IV

Evaluation of Real-World Application

Living-Lab: Playground

Contents

List of Figures

List of Tables

Data Source Prolog

For the development and evaluation of the DVS-based long-term monitoring system, the Living-Lab presented in Section 3.1 was used as the measurement site. Due to the open nature of this measurement site, the general public served as test participants.

The measurement system's technical commissioning was made possible by the cooperation of several departments. The urban green space planning office of the City of Mönchengladbach and the Niederrhein University of Applied Sciences collaborated under the EFRE-funded research project "plsm"[18]. This collaboration consolidated the necessary administrative and technical actors to operate the measurement system.

In addition to staged recordings, as contained, for example, in parts of the DVS-OUTLAB dataset (see Section 4.1.2 on page 49), real-world data was recorded during a long-term observation between 2019 and 2021. The following evaluation of the system focuses primarily on the 2021 recordings due to the impact and restrictions on public life caused by the COVID-19 pandemic, which peaked in 2020. For the summer of 2021 measurement period, recordings from the three Dynamic Vision Sensors used in the Living-Lab are available for a total of 140 days, from May to October.

These types of scenario-based recordings provide more representative samples than day-to-day experiments. However, long-term, multi-sensor recordings result in a large amount of data that must be stored and processed. It is a challenge to select and store only those scenes that are of further interest in an unconstrained "real-world" setup. The final evaluation of the application is based on the data from the recordings summarized in Table IV.1, which were obtained by applying the scene selection methodology described in the next Chapter 9 directly on-site during the 2021 measurement period.

Month	DVS1	DVS2	DVS3	Cumulative
May	17 h 12 min	26 h 49 min	23 h 16 min	67 h 18 min
June	18 h 02 min	42 h 43 min	35 h 40 min	96 h 25 min
July	27 h 06 min	59 h 57 min	47 h 46 min	134 h 49 min
August	27 h 19 min	66 h 29 min	53 h 04 min	146 h 52 min
September	19 h 51 min	44 h 23 min	38 h 39 min	102 h 54 min
October	18 h 32 min	30 h 46 min	30 h 07 min	79 h 25 min
Sum	128 h 03 min	271 h 08 min	228 h 34 min	627 h 46 min

Table IV.1: Recorded durations of scenes selected on-site in the 2021 measurement period.

[18] `https://plsm-project.com/` European Regional Development Fund grant number: EFRE-0801082

Chapter 9

Extraction of Scenes of Interest

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2023b). Semantic Scene Filtering for Event Cameras in Long-Term Outdoor Monitoring Scenarios. In Bebis, G. et al., editors, *18th International Symposium on Visual Computing (ISVC), Advances in Visual Computing*, volume 14362 of *Lecture Notes in Computer Science*, pages 79 – 92, Cham. Springer Nature Switzerland

Identifying temporal segments of interest is a significant challenge in most long-term monitoring. For example, as in the application use case scenario, the observed children's playground is not constantly used.

Therefore, the goal of on-site processing was to identify and store only those segments that contain events triggered by a specific set of objects of interest. The processing must address several challenges including the impact of sensor noise, detection of small objects, environmental influences, and the limited computational capacity of the on-site computing hardware.

The main contributions presented in this chapter are as follows:

- Development of a multi-stage processing chain that addresses the aforementioned processing challenges,
- while taking into account the limited on-site computational and power capabilities of the measurement system hardware within the Living-Lab.
- An evaluation of the proposed approach on manually labeled real-world data.

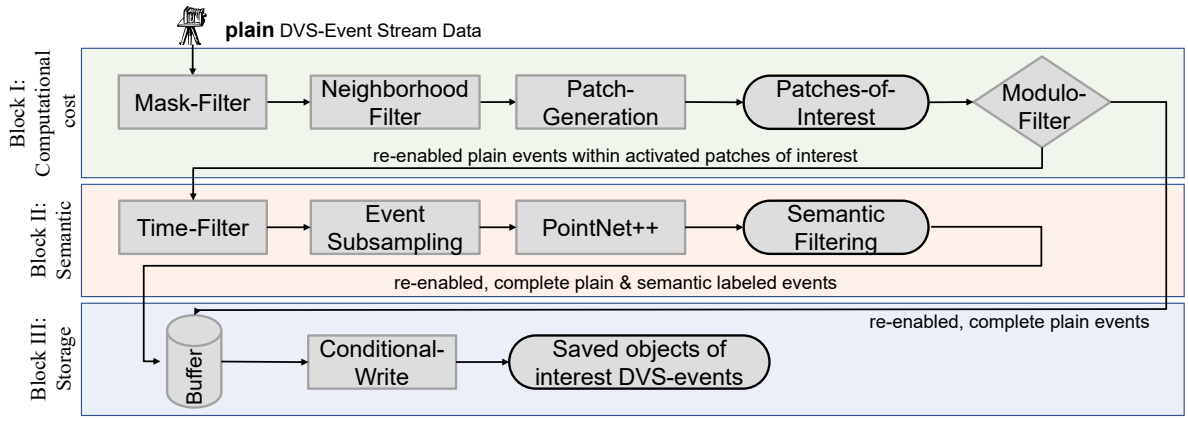


Figure 9.1: Structure of proposed conditional writing pipeline (from [**Bolten** et al., 2023b]).

9.1 Scenario and Goal

The goal of the on-site scene extraction was to identify temporal segments containing DVS events triggered by a predefined set of object classes.

The proposed processing pipeline preprocesses the data stream through multiple filters to identify Patches-of-Interest, which are then semantically segmented. This process is designed to filter out and ignore sensor noise and environmental effects. The temporal segments that were identified are stored for further and more detailed offline analysis, e.g. to extract spatial positions and movements of objects within the monitored area.

Ideally, a Dynamic Vision Sensor output event should only be triggered when an actual brightness change is observed in the scene. However, the output of available sensors contains a significant amount of noise. A substantial DVS noise effect that can be addressed by spatio-temporal filtering are background activity events.

However, outdoor measurements are likely to include environmental artifacts in the sensor signal in addition to background noise. These artifacts include rain, flying insects, clouds, shadows, and object motion from wind. Therefore, they must be considered in the processing pipeline. While spatio-temporal filters can substantially reduce background noise, larger amounts of environmental noise remain in the signal (see Section 3.3).

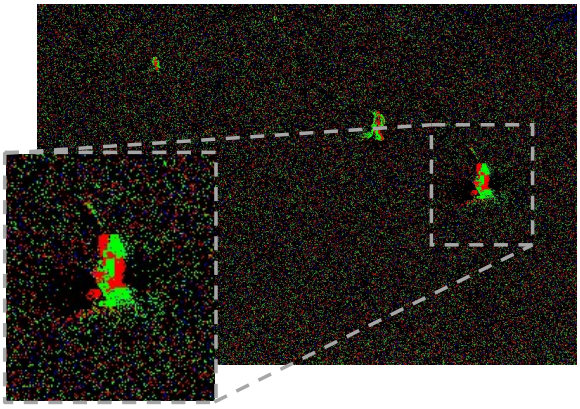
As a result, relying only on event filtering and selecting scenes based solely on the number of events will result in false positives in an outdoor environment. A higher-level semantic analysis of the event stream is necessary.

9.2 Processing Pipeline

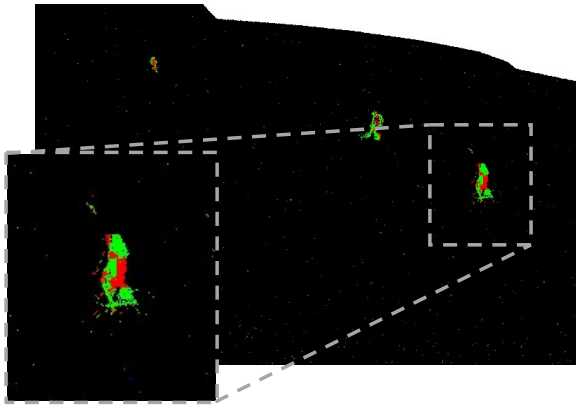
A multi-stage semantic filtering pipeline was developed, where the object recognition part is triggered only by a defined set of classes by applying semantic segmentation based on the 3D processing described in Chapter 6. To reduce the subsequent computational load, the event stream is spatio-temporally pre-filtered to identify important Patches-of-Interest within the sliding time window. A decision is made whether to save the current data based on the semantic segmentation of these selected patches. The entire process, as outlined in Figure 9.1, is described in detail below.

All processing steps have in common that they process the continuous event stream in non-overlapping 60 ms sliding time windows.

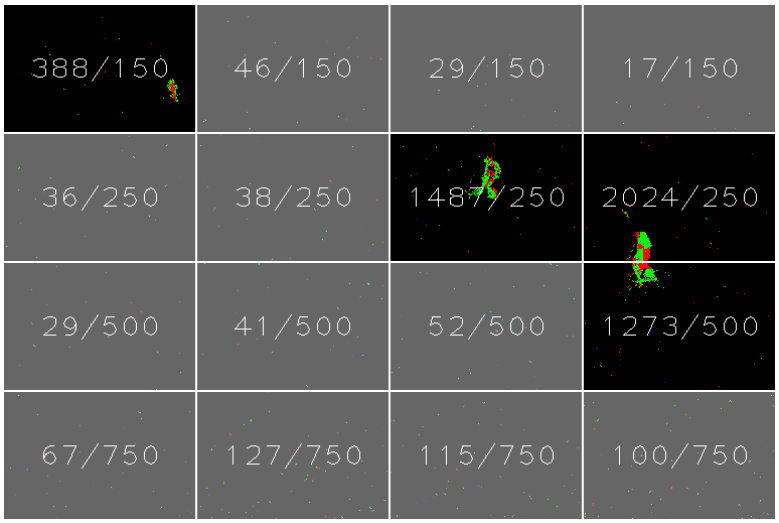
(a) Grayscale image of empty scene with blended mask of public sidewalk (provided as reference)



(b) Plain DVS input, projected as 2D frame (shown with cropped and enlarged region)



(c) Masked and Neighborhood-Filtered (shown with cropped and enlarged region)



(d) PoI extraction by thresholding, shown based on Figure 9.2c. Discarded patches are grayed out. Active PoIs are selected by applying an event threshold depending on the row of the patch to account for different perspective distances. The text shows `number of triggered events / threshold`

Figure 9.2: Step-by-step visualization of Patch-of-Interest generation (adapted from [**Bolten** et al., 2023b]).

9.2.1 Block I: Patches-of-Interest

The basic idea of this step is a classical divide-and-conquer approach. The goal is to preprocess the input stream with simple and fast operators. Thus, only Patches-of-Interest need to be processed by subsequent steps with significantly higher computational requirements. It is expected and intended that time windows with high levels of environmental noise and artifacts will pass these steps, but will be rejected later.

In contrast to the patching performed in [Sabater et al., 2022], where small 6×6 pixel patches are constructed and processed, the sensor array was divided into patches that contain enough spatial resolution to be processed independently and meaningfully using only the (x, y, t)-information of each patch itself. This formation of Patches-of-Interest is based on the preprocessing described in Section 6.1.2, where the (x, y)-sensor plane was divided into 16 equally sized patches (see Figure 6.3 on page 101). As previously mentioned, the selection of this patch size of 192×128 pixels depends directly on the event count requirements of 3D semantic processing. To mark a patch as either active or inactive for subsequent processing, spatio-temporal filtering and thresholding are used.

The process of generating Patches-of-Interest starts with removing all events outside the monitored area (mask filtering of fixed areas, such as public sidewalks). This is followed by restrictive filtering of sensor noise. According to the analysis of the spatio-temporal event filters performed in Section 3.3.2 and 3.3.3 as well as the runtime requirements for real-time filtering of three sensor signals, the *Neighborhood-Filter* logic is applied to reduce the background noise artifacts contained. This filter was selected because this analysis showed that it achieves a very restrictive filtering result while still retaining enough true object events for subsequent active area identification. These steps are illustrated in Figures 9.2b and c.

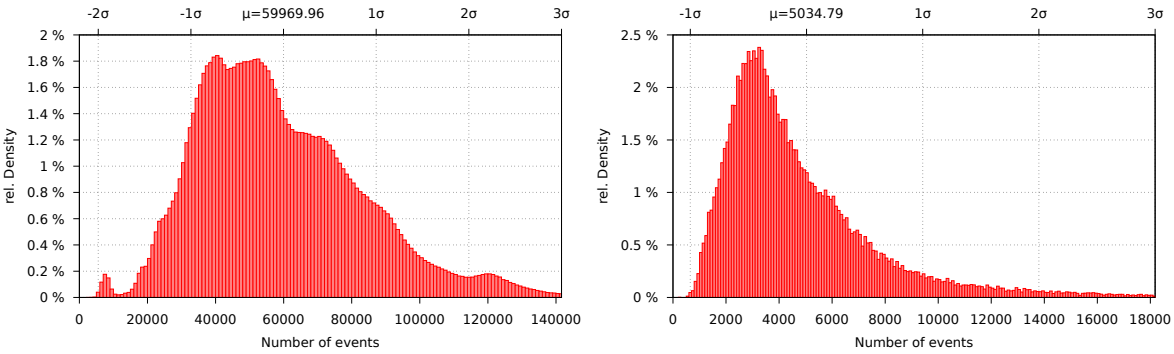
A Dynamic Vision Sensor separates the static background from the moving parts at the sensor level based on its fundamental operating principle. Therefore, the accumulated number of filtered events within these patches can be used as a simple measure of change within the scene (while ignoring their semantic origin). Finally, to flag a patch as an active Patch-of-Interest, a threshold is applied to the number of filtered events. To account for different perspective distances and the corresponding differences in the number of triggered events, different thresholds are used for each row of patches. These thresholds were empirically optimized and selected during long-term monitoring. This thresholding procedure is illustrated in Figure 9.2d.

Only active flagged patches are further processed. In some cases, the event counts in many or all patches may surpass the threshold for an extended period (e.g., during heavy rain) and become marked as active. This would lead to very high computational resource requirements in the later processing. Therefore, it is possible to bypass downstream processing for a limited number of sliding windows (shown as a modulo filter in Figure 9.1). Skipped time intervals are stored directly in the write buffer, and their final processing depends on the outcome of adjacent, fully processed segments.

9.2.2 Block II: Semantic Analysis utilizing PointNet++

A semantic segmentation for the active Patches-of-Interest into a specific set of object classes is the main component of the proposed pipeline.

The processing of 3D space-time event clouds was found to be faster and of at least equal or higher quality for the task of semantic segmentation than a UNet or a Mask R-CNN counterpart based on frame conversion. Therefore, a PointNet++ was selected as the processing



(a) Event counts in complete and unprocessed sensor view

(b) Event counts in active and time-filtered PoIs

Figure 9.3: Event count comparison on real-world data tested in Section 9.3 (adapted from [**Bolten** et al., 2023b]).

method within this pipeline. However, the number of input events forming the space-time event clouds must be limited for this processing, as described in Section 6.1.2.

Therefore, the previously created Patches-of-Interest are further processed individually. For each active Patch-of-Interest, the filtering is reset. The raw event stream is restored within these patches. This is necessary because the previous neighborhood filtering is rather restrictive and may remove events that belong to the desired object structures. Then the DVS preprocessing method for PointNet++ based processing is applied, as described in Section 6.1.2. This includes a less restrictive spatio-temporal time filter, subsampling, and scaling.

The previously selected sampling target of 4096 events for this preprocessing, which is based on the analysis of the DVS-OUTLAB dataset, was retained because the recorded event numbers were comparable to the real long-term monitoring recordings. For more details, compare Figure 9.3 with the results reported in Section D.1 in the Appendix.

After applying PointNet++ to the preprocessed (x, y, t) event clouds, each event is assigned a class label. The final write decision is made in the next processing block based on the number of events that belong to a specific set of object classes of interest.

9.2.3 Block III: Conditional Write

The sliding time windows incoming may or may not have labels within their patches, as they can skip semantic segmentation.

They are stored in a temporal first-in first-out buffer that can hold a fixed number of time windows. The size of this buffer is defined by the number of *chunks*. Each chunk contains at least one time window in which a PointNet++ based semantic segmentation must have taken place, regardless of any skipping.

Events predicted to be background or belonging to environmental noise classes are discarded. The decision to save a sliding time window of DVS data onto disk depends on the count of events predicted to belong to objects of interest classes within the fully processed and semantically segmented PoIs of each chunk. A minimum threshold is applied to these events. This step is identical to the thresholding technique described above that is used to identify active PoIs earlier in the pipeline, but the thresholding is applied only to the events that are labeled as objects of further interest.

To ensure continuity in the saved sliding window chunks, not only the chunk containing the positive object count threshold is saved, but also the chunk immediately before and after it. This process is shown in Figure 9.4.

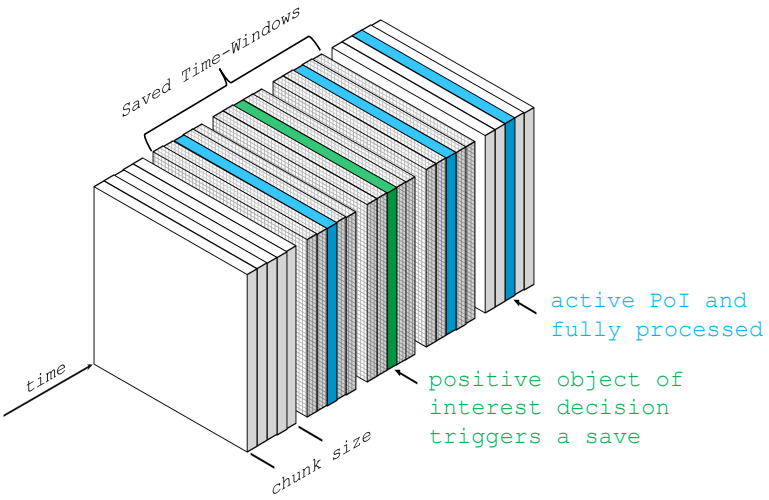


Figure 9.4: Visualization of conditional buffer write logic (sliding time windows marked in green indicate a positive filtering decision that results in writing the gray chunks; from [**Bolten** et al., 2023b]).

9.3 Practical Application and Results

The segmentation network within the pipeline is based on the optimized, trained and evaluated model of PointNet++ as described in Section 6.3.3. This model is based on the underlying training dataset of DVS-OUTLAB (see Chapter 4).

Therefore, the classes `PERSON`, `DOG`, `BICYCLE`, and `SPORTSBALL` were defined as objects of interest in this context. The classes `BACKGROUND`, `RAIN`, `TREE`, `INSECT`, `BIRD` and `TREE_SHADOW` were considered as noise and unwanted environmental artifacts.

The following evaluation of the described pipeline is based on ten days of unprocessed, complete raw data collected during the 2020 measurement period, as the 2021 recordings have already been processed and stored on-site by the use of this pipeline.

The reduced use of the playground during the 2020 period makes the processing even more challenging with respect to false negatives. Since the main goal of the processing is to filter out environmental artifacts and save only temporal segments containing a defined set of objects, this is acceptable since the selected recordings still include both positive cases and different levels of environmental influence.

9.3.1 Runtimes

The semantic segmentation of the events within the selected Patches-of-Interest by the PointNet++ network has the highest computational requirements in the entire processing chain. An advantage of this network architecture is that each point is processed by *shared* Multi-Layer Perceptrons. This results in relatively small networks, allowing an NVIDIA Jetson TX2 module to be used on-site as an energy-efficient edge AI computing device instead of a power-hungry full-scale GPU. This is very beneficial in the application scenario, as the entire on-site system is solar-powered (see Section 3.1).

Using this resource-limited module resulted in an average runtime of 159 ms ($\pm$7.14 ms, based on a measurement of 10,000 individual executions, each with a batch size of one for live processing) for PointNet++ inference. In the worst case, with all 16 Patches-of-Interest continuously active, it is possible to achieve a complete processing every $\approx$ 2.54 seconds. However, a real-world analysis of activated Patches-of-Interest on data containing various factors

Date	Rain in l/m^2	Fraction of time windows w/o any active PoI per day	Mean number of active PoIs per time window
Thu, July 02	6.4	71.34 %	1.95
Fri, July 03	0.0	38.30 %	5.97
Sat, July 04	0.9	42.63 %	2.12
Sun, July 05	0.1	43.34 %	4.09
Mon, July 06	1.7	44.31 %	4.49
Tue, July 07	1.2	44.23 %	4.66
Wed, July 08	7.3	92.63 %	0.61
Thu, July 09	0.0	47.13 %	3.16
Fri, July 10	1.0	40.15 %	5.85
Sat, July 11	0.0	9.53 %	6.83
All		46.93 %	4.57

Table 9.1: Patch-of-Interest generation statistics on real-world data.

(see Table 9.1) showed that this scenario is very unlikely to occur. During the ten-day period analyzed, approximately 47 % of the sliding time windows contained no activated PoI at all. In the remaining time, on average, less than a third of the PoIs were active. This shows that PointNet++ processing can generally be achieved at a higher frequency.

Different days of the week, usage scenarios, and weather conditions with widely varying rates of PoI activation were included in the evaluated test period. As an example, compare Wednesday, July 8 with Saturday, July 11. On that Wednesday, there was no significant activity except for some very short and heavy rain showers. This resulted in a high percentage of data being discarded in the first processing step. On the other hand, on the aforementioned Saturday, a high level of activity combined with environmental influences such as moving shadows resulted in many PoIs that had to be processed later.

Apart from semantic segmentation, most of the runtime is consumed by filtering the event stream. These operations, each with a complexity of $\mathcal{O}(\text{eventCount})$, are performed by individual threads. Therefore, to meet the real-time requirement, each step must be completed within the length of the sliding time window. Table 9.2 shows the average runtimes per filter for the typical total number of input events that occur. These tests demonstrate that each filter has the capability to process the event stream in real-time as the continuous event stream is processed in sliding windows of 60 ms.

9.3.2 Scene Filtering Quality

In order to evaluate the quality of the filtering achieved, the following two types of errors are crucial:

(a) *false negatives* (wanted but not recorded) and
(b) *false positives* (recorded but not wanted).

The error rates were determined using the complete, unprocessed recordings of a single DVS, as presented in Table 9.3. This resulted in a total of 88.95 hours of recorded time. The results were compared to manual annotations made by a human observer. This comparison was made as follows to keep the cost of annotation in an acceptable range.

Processing step	Mean runtime for one 60 ms time window		
	50 kE	170 kE	420 kE
Mask-Filter	1.026 ms ± 0.25	1.415 ms ± 0.45	3.230 ms ± 1.00
Neighborhood-Filter	6.494 ms ± 1.54	13.638 ms ± 1.96	42.946 ms ± 5.31
Time-Filter	0.379 ms ± 0.11	21.740 ms ± 4.42	46.167 ms ± 7.46

Table 9.2: Filter runtimes at different input event counts in kilo events (kE) (averaged over 150 sliding time windows of 60 ms, computed on an Intel Core i7-8700 CPU).

Date	Activity annotated by human	FP-rate "false alarm"	FN-rate "missed"	F1 score
Thu, July 02	11.27 %	0.007	0.006	0.969
Fri, July 03	18.20 %	0.012	0.038	0.954
Sat, July 04	32.16 %	0.015	0.029	0.970
Sun, July 05	23.30 %	0.019	0.048	0.944
Mon, July 06	10.21 %	0.026	0.023	0.886
Tue, July 07	10.37 %	0.014	0.054	0.916
Wed, July 08	1.50 %	0.006	0.152	0.761
Thu, July 09	13.37 %	0.015	0.021	0.945
Fri, July 10	4.63 %	0.010	0.063	0.873
Sat, July 11	28.55 %	0.024	0.043	0.948
All	15.13 %	0.015	0.038	0.941

Table 9.3: Scene extraction quality (FP $\hat{=}$ false positive, FN $\hat{=}$ false negative).

A two-dimensional frame was exported from the DVS stream for every 32nd sliding time window of the unprocessed event stream by projecting the events onto the xy-plane. This leads to one frame every $32 \cdot 60\,\text{ms}$, which is equivalent to 1.92 seconds. Considering the size of the observed area, this form of temporal subsampling is appropriate because the desired objects, such as pedestrians, would remain visible in a selected frame while crossing the area at their expected speeds. This sampling procedure results in a total of $\approx$ 166.700 exported frames which were manually checked for objects of interest.

False Negatives:

In this process, 969 detections were marked by the human observers that were not saved by the proposed processing pipeline. This results in a false negative rate of $\approx 4\,\%$. However, around 75 % (724 out of 969) of the missed detections were found in the top row of recorded patches, where the recorded content was far away from the sensor. The resulting small object sizes, one of the primary challenges in processing, make it difficult even for a human observer to distinguish between objects and noise. Therefore, these cases are also likely to be very difficult for other automated processing approaches.

False Positives:

The proposed processing chain applied to the raw recordings resulted in the storage of about $\approx$ 14.06 hours of the total input data. These saved time windows were compared to the manually annotated frames from the previous step. The export step used for these manually labeled images corresponds to the modulo 32 skipping used for automatic processing. In this way, the resulting chunks were compared to each other. This resulted in 2090 chunks stored by the automated processing that were not marked as active by the human reference, resulting in a false positive rate of $\approx 1.45\,\%$.

Due to the good results of the applied semantic segmentation, the presented pipeline overcomes the aforementioned artifacts caused by environmental influences. In the presented numerical results, this is shown by the inclusion of rain.

However, on rainy days fewer people use the outdoor area (e.g., Wednesday, July 08 with only 1.5 % annotated activity time) and the processing becomes more complicated. On particularly rainy days, the false negative rate increased and objects were missed. But the false alarm rate and specificity stayed at acceptable levels. This shows that a good robustness against rain has been achieved. However, desired objects may be (partially) occluded by rain in the sensor's field of view, resulting in lower detection sensitivity.

In general, the evaluation showed that the developed processing pipeline is able to identify relevant time segments in the context of a long-term observation and to store them for later, more detailed evaluation.

Chapter 10

Derivation of the User Volume: Evaluation on Long-Term Data

The trained segmentation approaches need to be evaluated in the context of the Living-Lab monitoring to demonstrate the capabilities of the developed application-oriented system. A multi-stage processing to estimate the user volume is evaluated to validate the feasibility of the developed proof-of-concept system.

Based on the comparison of manually and automatically extracted instance counts, as well as on the obtained qualitative segmentation results, the reliability of the system is assessed.

The main contributions presented in this chapter are as follows:

- Selection of representative scenes from the long-term monitoring and manual annotation to evaluate the developed processing.
- Application and evaluation of the trained segmentation approaches on real-world Living-Lab data recorded during the long-term monitoring to estimate the number of included users.
- Based on typical error cases, limitations, and optimizations are presented, which should be considered when reproducing such monitoring systems.

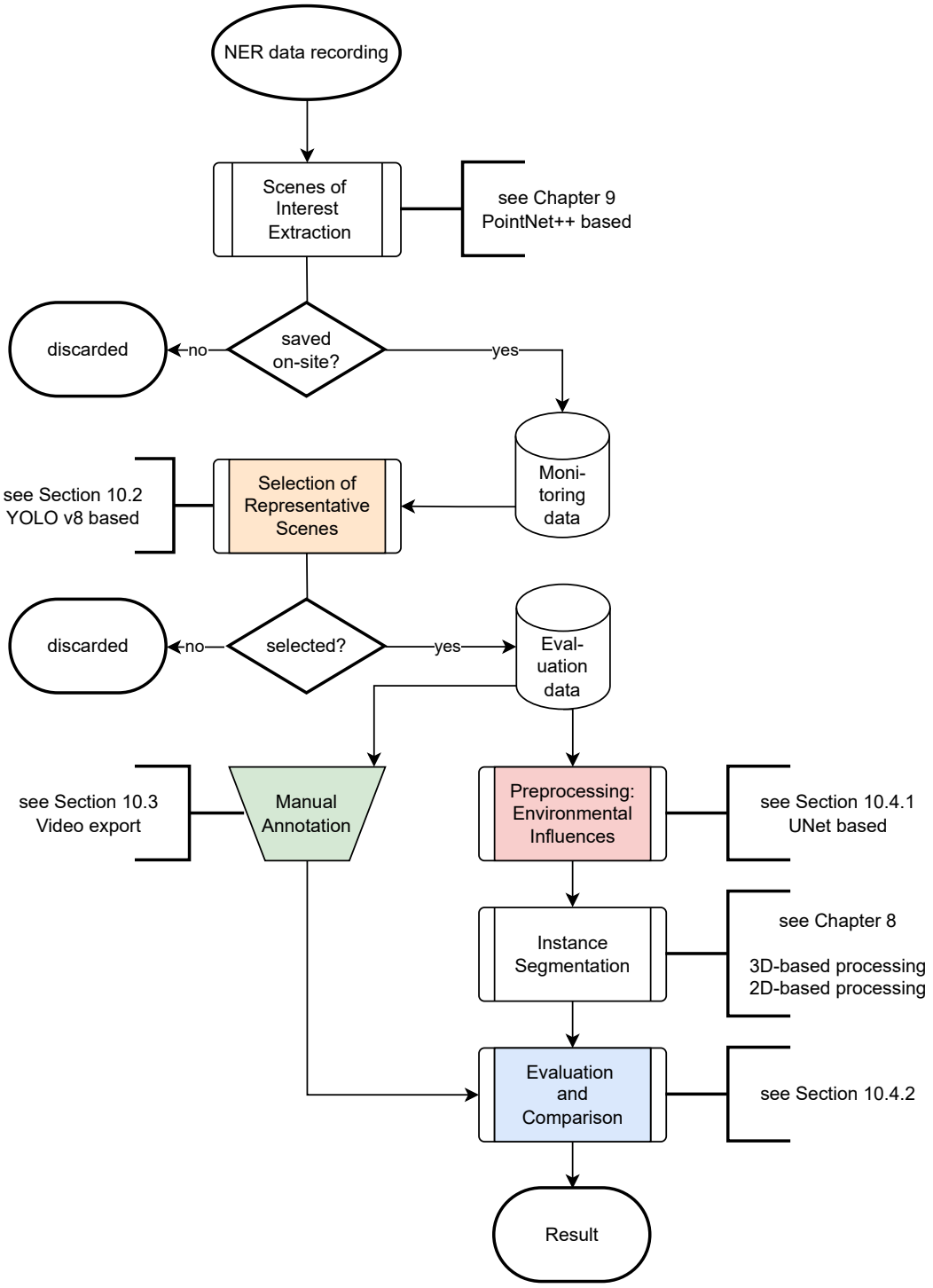


Figure 10.1: Overview of the evaluation process performed (color-highlighted blocks are discussed in more detail).

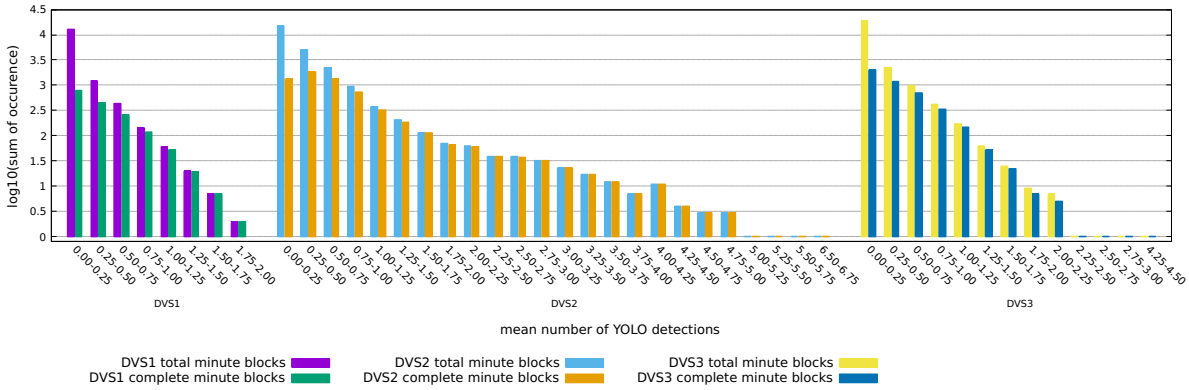


Figure 10.2: Histogram of the number of YOLO-based `PERSON` detections.

10.1 Processing Overview and Objective

One of the two primary parameters to be derived by the developed monitoring system is the user volume within the observation area. The focus here is on the *counting* of the *object instances* observed. As part of the long-term monitoring, a large amount of data containing potentially interesting objects (see Chapter 9) was stored on-site.

The segmentation methods presented in Part III were used to implement a people counting toolchain. The evaluation procedure for this processing is shown in Figure 10.1.

A manual annotation of the recorded data serves as a reference for the obtained processing results. A complete processing and subsequent manual evaluation of the processing results is not expedient due to the recorded data volume of approximately 630 hours. In order to keep the processing and especially the manual annotation effort within reasonable limits, it was necessary to select representative scenes from the long-term data to evaluate the processing.

These representative scenes were then processed, with the first step being the removal of unwanted environmental effects, followed by the prediction of object instance masks to be used for counting. Based on these automatically determined object counts, a comparison with the manual annotations was performed.

10.2 Selection of Representative Scenes

The data basis for this selection was formed by all recordings from the measurement period 2021 of the three Dynamic Vision Sensors used in the Living-Lab, which were recorded on-site by the extraction of scenes of interest processing approach introduced and evaluated in Chapter 9. A quantitative overview of these recordings is given in Table IV.1 on page 137.

To select the scenes, a 2D polarity frame-based representation was exported for each 60 ms event time window contained in these recordings. These polarity frames were then processed by YOLO v8 to obtain an estimate of the number of `PERSON` instances included in each frame.

At this point, YOLO v8 was selected as the instance segmentation approach because the previous analysis showed that it achieved the best $\mathrm{mAP}^{0.5}$ score on the challenging subset (see Table 8.3) as well as on the full dataset of N-MuPeTS (see Table E.5). The lower penalization of pixel inaccuracies in the generated object masks within the selected $\mathrm{mAP}^{0.5}$ metric is intentional, since only the number of objects found and not the pixel accuracies are of interest for the further selection of representative scenes. In addition, the preprocessing required for the

YOLO v8 inference and its runtime are low, which was of particular interest considering the number of inferences required for the entire database.

For further processing, the recordings and the corresponding inference results were divided into one-minute blocks. This division was based on the "wall" clock time, i.e. the real-world local time of the recordings. These wall clock aligned time blocks may also be shorter than one minute, as the on-site processing allows the recordings to start and end at arbitrary 60 ms time windows.

For each of these time blocks, the frame-averaged number of detected `PERSON` instances was determined. This means that the actual number of frames in that block was used to normalize the sum of the detections. These normalized instance counts were then grouped into histogram bins as shown in Figure 10.2. These histograms approximate the distribution of the number of people who used the area covered by each sensor within the analyzed measurement period.

From these histograms it is already possible to deduce differences in area usage, e.g. the area covered by DVS1 is used less by larger groups. However, it can also be seen that there is a strong imbalance between the number of time blocks recorded and the average number of people detected in these blocks. Scenes with fewer included people strongly dominate the real-life scenario in the field.

In order to obtain a subset of recordings that are representative of the real-world usage scenario, subsampling was performed according to the determined distribution. For each sensor, 90 one-minute time blocks were randomly selected. Only complete blocks, i.e. blocks covering an entire minute of wall clock-aligned data, were considered to keep the subsequent evaluation comparable.

The random selection of the one-minute blocks was based on the frequency of occurrence in the determined usage histogram. However, in order to include all usage scenarios in the evaluation, at least two one-minute blocks were forced to be selected from each usage bin in the histogram. This ensured that also scenes with a high number of users, which are challenging in terms of processing, were also included in the subselection.

10.3 Manual Annotation

As mentioned above, 90 one-minute time blocks of representative data were subsampled for each of the three Dynamic Vision Sensors used. This results in a total of 270 minutes of sampled data for evaluation. Since the implemented processing handles individual 60 ms time windows of event data, this results in a total of 270,000 exported event time windows.

Manual annotation of this amount of data, either at the level of semantic segmentation or the creation of instance masks, is not economically feasible given the time required. In order to keep the requirements for manual annotation within reasonable limits, the evaluation performed takes into account, per one-minute block, the *maximum* number of `PERSON` instances *simultaneously* present in the scene.

To facilitate manual annotation, the selected data was exported as polarity-frame encoded video files so that they could be played back with standard software. The video files were randomly sorted and renamed to hide the previously determined level of usage within the selection step. This was done to prevent any a priori assumptions from influencing the manual annotation process.

Each of the video files was viewed and annotated by the same set of ten human viewers. For this purpose, each participant was provided with an "Annotation Guide" that introduced the

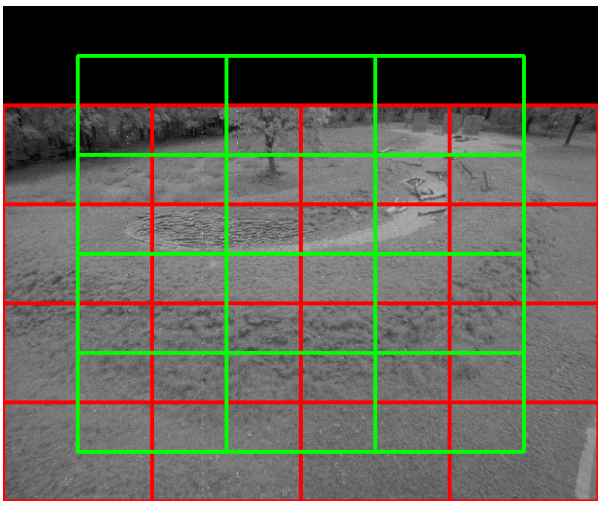


Figure 10.3: Highlighting of processed spatial patches in UNet-based semantic segmentation to remove environmental influences (grayscale background of the scene is provided for reference only).

technology of the Dynamic Vision Sensor, its output, and the data representation and encoding used for the evaluation. In addition, an example scene was shown, and the annotation scheme used was explained. The special characteristics of the recording and the annotation to be created were briefly presented by means of examples. This included an explanation of the environmental influences recorded and rules for counting people entering, leaving, or standing still.

The desired counting method per scene and thus the abstraction to a single metric value per video file was explicitly explained and illustrated.

10.4 Processing and Evaluation

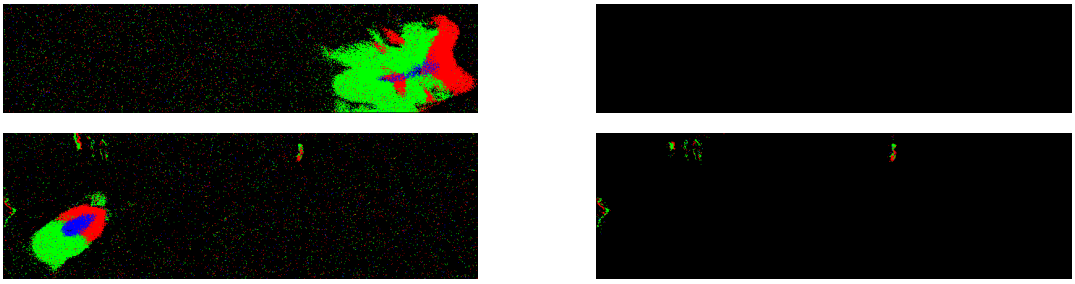
The human-generated annotations serve as the basis for benchmarking the automated processing. In the application context of the real monitoring data, both 2D frame-based and 3D space-time event cloud-based instance segmentation were considered.

10.4.1 Preprocessing: Removal of Environmental Influences

As shown in Figure 10.1, the processing of the sampled evaluation data is done in two steps. In order to reduce the number of events, which is required to support the 3D point cloud processing methods (see Section 6.1.2), and to account for the less included and less considered environmental influences within the dataset used to train the instance segmentation approaches, a semantic segmentation is performed prior to the instance extraction.

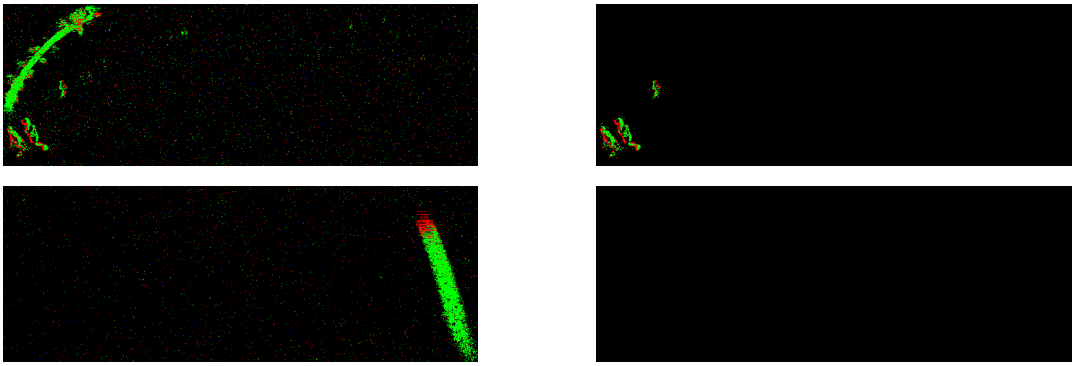
The DVS-iOUTLAB dataset and the N-MuPeTS dataset used for instance segmentation are either based on augmented object of interest classes (i.e., `PERSON`, `DOG`, `BICYCLE`, and `SPORTS-BALL`) or contain only `PERSON` instances. Despite the inclusion of sensor background noise, these instance segmentation datasets do not specifically target the influence of environmental effects.

For this reason, a divide-and-conquer approach was performed by first applying and filtering based on a semantic segmentation. The underlying dataset for the semantic segmentation, DVS-OUTLAB, covers different environmental influences, and therefore the trained processing approaches are able to remove these aspects.



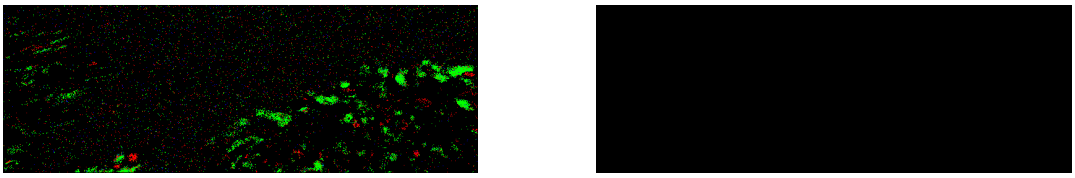
Polarity encoded frame | Filtered for `PERSON` events

(a) Crawling insects



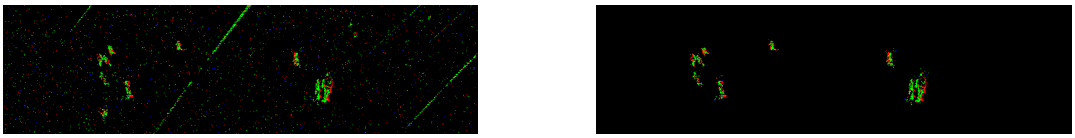
Polarity encoded frame | Filtered for `PERSON` events

(b) Flying insects



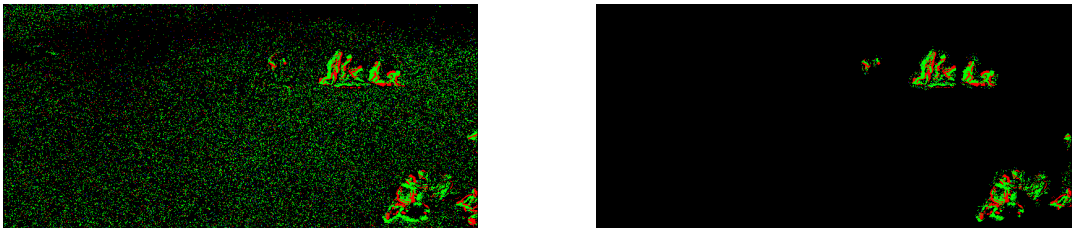
Polarity encoded frame | Filtered for `PERSON` events

(c) Tree shadows



Polarity encoded frame | Filtered for `PERSON` events

(d) Rain



Polarity encoded frame | Filtered for `PERSON` events

(e) Global illumination changes

Figure 10.4: Results of the applied UNet-based environmental influence removal.

As semantic segmentation method for this step, a UNet-based processing (see Chapter 7) was selected, since the data preprocessing effort, the export of frame representations, is low and was already performed as a step in the selection of representative scenes.

The previous experiments, and thus the trained models, are based on the processing of spatially patched data. Therefore, this was also carried out at this point. In order to support the segmentation and to reduce the errors at the boundaries of the patches, an overlapping of these patches was performed, as shown in Figure 10.3. The overlapping segmentations were combined into a single result of the size of the original DVS input data. Different label predictions at the same position were resolved based on a priority list, with `PERSON` class predictions being prioritized.

Figure 10.4 shows example results of this processing.

10.4.2 Results and Comparison

The selection of instance segmentation methods considered is based on the analysis conducted in Chapter 8.

YOLO v8 was selected as a representative of the 2D frame-based processing. The YOLO v8 model outperformed the Mask R-CNN reference on the N-MuPeTS dataset, especially on the challenging sequences that contain spatially close instances, representing an expected scenario on the playground. Both JSNet and 3D-BoNet were selected as 3D point cloud-based processing approaches and included in the comparison, as their performance on the N-MuPeTS dataset was found to be highly comparable. The voxel-based processing using SoftGroup was not evaluated on the long-term monitoring data, as it performed worse in the previous experiments on the N-MuPeTS data.

This led to a comparison between 3D space-time event cloud and frame-based processing techniques, as also stated as the main research question. A polarity frame encoding was selected for the generation of the 2D images, as this encoding scheme allowed for highly satisfactory results in the semantic and instance segmentation experiments performed. The preprocessing of the space-time event clouds is based on the generation of adaptive input regions, as described in Section 8.2.1.

The sampled representative sequences very rarely contain instances of non-`PERSON` classes, such as `DOG` or `BICYCLE`. Due to the very small number of captured instances of these classes, the evaluation, as well as the manual annotation, was limited to the evaluation of the maximum number of `PERSON` instances simultaneously present in the scene.

Metric

The obtained results of human and automatic `PERSON` counting for the selected scenes are shown in Figure 10.7a, Figure 10.9a and Figure 10.11a for each DVS used in the Living-Lab.

A quantitative comparison of the automated processing results with the human annotations was conducted in order to summarize the results. However, it was not possible to calculate, for example, a classical z-score to measure how close the obtained `PERSON` count prediction p_i^{alg} is to the distribution of human annotations. As the z-score is defined as

$$\text{z-score}_i = \frac{p_i^{\text{alg}} - \mu_i^{\text{human}}}{\sigma_i^{\text{human}}} \tag{10.1}$$

with i as an index for the considered scene, while the mean μ_i^{human} and standard deviation σ_i^{human} are estimated from the human annotations. In some cases, all ten human annotators

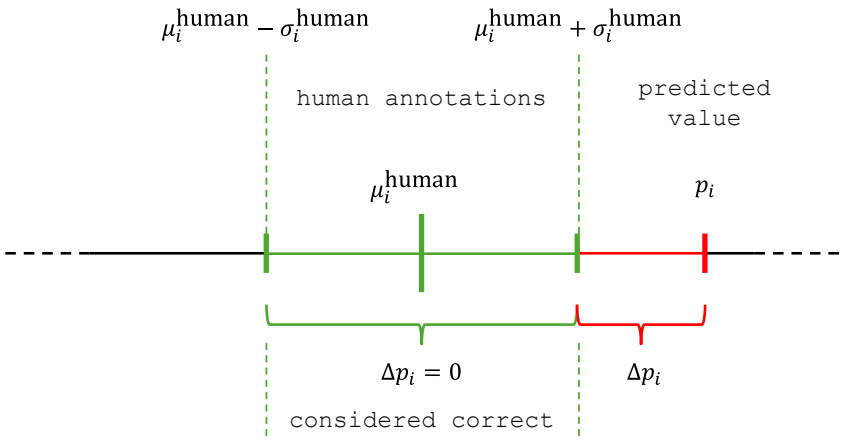


Figure 10.5: Description of evaluation metric.

Sensor	Average standard deviation in human annotations	mean($\lvert\Delta p\rvert$) of predictions		
		YOLO v8	3D-BoNet	JSNet
DVS1	0.847	1.145	**0.972**	1.188
DVS2	0.926	**0.324**	0.349	0.510
DVS3	1.033	0.881	**0.571**	1.022

Table 10.1: Comparison of `PERSON` counting results.

counted exactly the same number of people, resulting in an $\sigma_i^{\text{human}} = 0$, making the z-score undefined.

As an alternative, and to account for the varying spread in the human annotations, we consider the range of $[\mu_i^{\text{human}} - \sigma_i^{\text{human}}, \mu_i^{\text{human}} + \sigma_i^{\text{human}}]$ as correct values for the prediction, resulting in a zero error if the prediction falls within this interval. For larger deviations, the difference between the prediction and this interval limit is counted as an error. This measurement scheme is illustrated in Figure 10.5. To further summarize the results, the absolute value of this error score is averaged over all scenes per sensor.

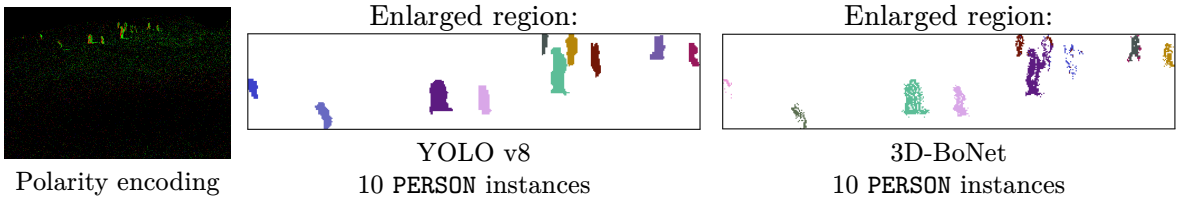
Quantitative Results

The results based on this metric are shown in Table 10.1. The processing performed by 3D-BoNet predicted the best `PERSON` counts on average across all sensors. Results were obtained that differed on average less than 0.35, 0.58, and 0.98 counted `PERSON`s outside the interval formed by the human annotations.

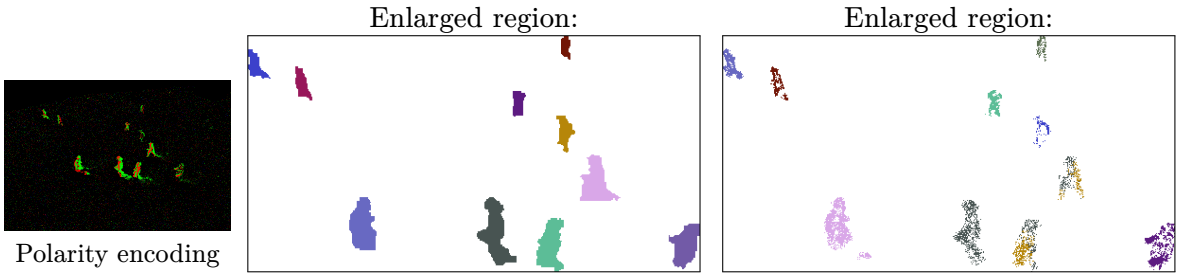
Qualitative Results

The metric results show an advantage for 3D-BoNet over JSNet. Therefore, the examples showing qualitative results in Figure 10.6 do not include results from JSNet.

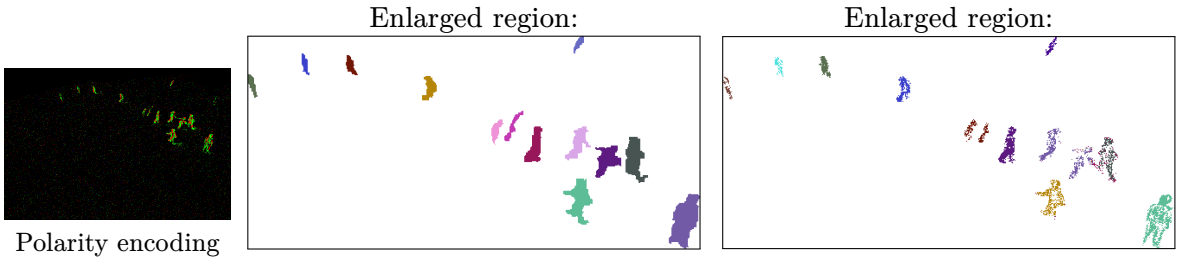
In general, all three methods, namely YOLO v8, JSNet, and 3D-BoNet, are capable of achieving visually good segmentation results in the context of the application, and thus are able to estimate the number of people with reasonable accuracy. This is especially the case for DVS2, as the comparison between human and automated results in Figure 10.7a shows. However, there are also systematic differences.



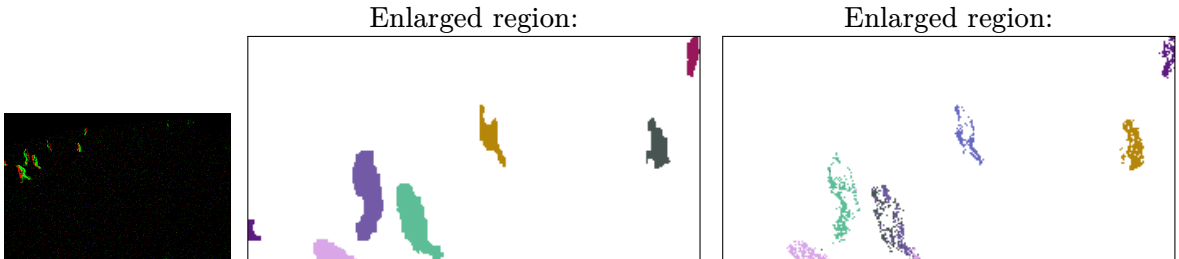
(a) Scene CMH123_0002



(b) Scene CMH123_0009



(c) Scene CMH123_0020



(d) Scene CMH123_0026

Figure 10.6: Examples of qualitative segmentation results.

Sensor	Number of input	
	2D frames	3D aRoIs
DVS1	90,000	92,502
DVS2	90,000	220,845
DVS3	90,000	124,419
Sum	270,000	437,766

Table 10.2: Number of inputs to be processed based on the sampled 90 one-minute scenes.

By design, YOLO v8 is limited to predict closed polygon masks for segmentation. This leads to inaccuracies at the event level. However, these inaccuracies are irrelevant for pure instance counting.

The predictions of JSNet and 3D-BoNet tend to correctly separate instances, but incorrectly assign single or very few outlier events across instance boundaries. In terms of the total number of events in the instances, these errors are not substantial, but these outliers lead to greater inaccuracies when considered at an instance bounding box level. Postprocessing to reassign events that are far from the center of mass of the instance would be one way to address this problem. However, these inaccuracies are also irrelevant for counting the instances.

Inference Count

Given that the selected representative scenes have a uniform length of one minute, the number of input images is identical for frame-based processing for all sensors. The integration time of 60 ms employed for image generation, coupled with a total of 90 selected scenes, results in a total number of 90,000 frames to be processed per sensor.

However, this is not the case for the 3D space-time event cloud-based processing. As a consequence of the preprocessing applied and the decomposition into aRoIs, the number of input clouds is dependent upon the number of included `PERSON`s and their spatial distribution within the individual sequences. Table 10.2 provides an overview of the number of inferences which were required for processing the evaluated scenes. For the scenes selected for evaluation, this results in an approximately 1.62-fold increase in the number of inferences required.

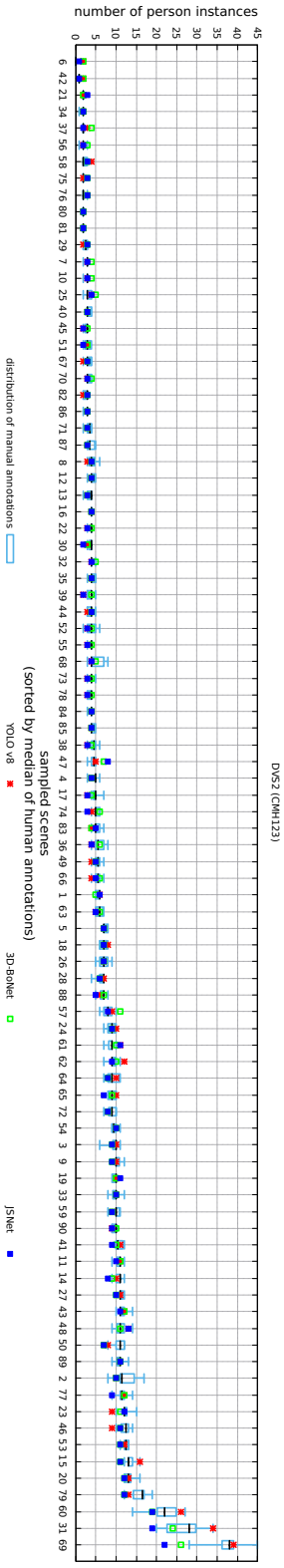
10.4.3 Typical Error Sources

Comparing the results for the individual scenes, three main categories of error sources in the people counting, and thus in the underlying segmentation, can be identified. In the following, each of these typical sources of error is illustrated, explained, and justified using one of the three sensors as an example.

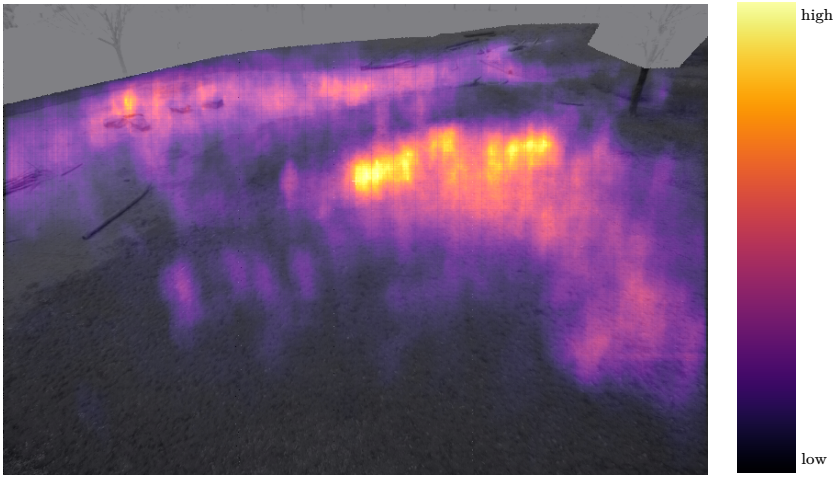
Very Crowded Scenes: Illustrated by the example of DVS2, as shown in Figure 10.7c.

For very crowded scenes, such as when the area is used by entire kindergarten groups, 3D space-time event cloud processing is limited in its results due to the preprocessing applied. In the case of many, spatially very close objects, the generated aRoIs, which serve as input for the neural networks, become very large in relation to the number of events they contain.

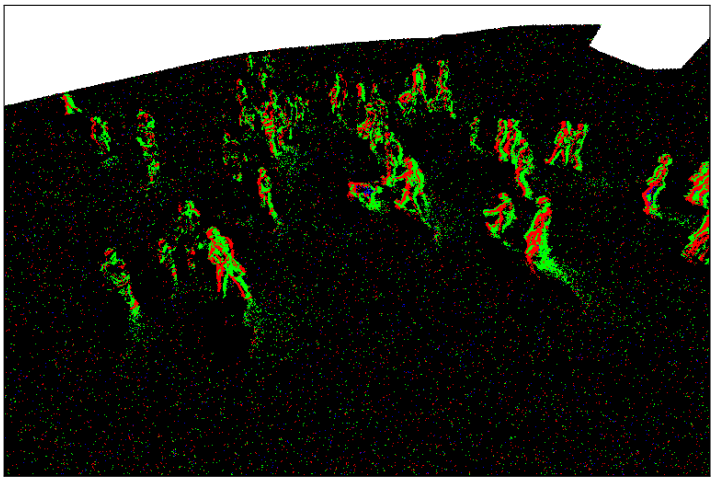
Figure 10.7d shows the generated aRoIs for the time window corresponding to Figure 10.7c. As indicated, the largest aRoI in this time window contains 37,496 individual events. For processing, each aRoI is subsampled to 2,048 events. Figure 10.8b shows a visualization of the subsampled data, clearly showing the loss of detail that prevents



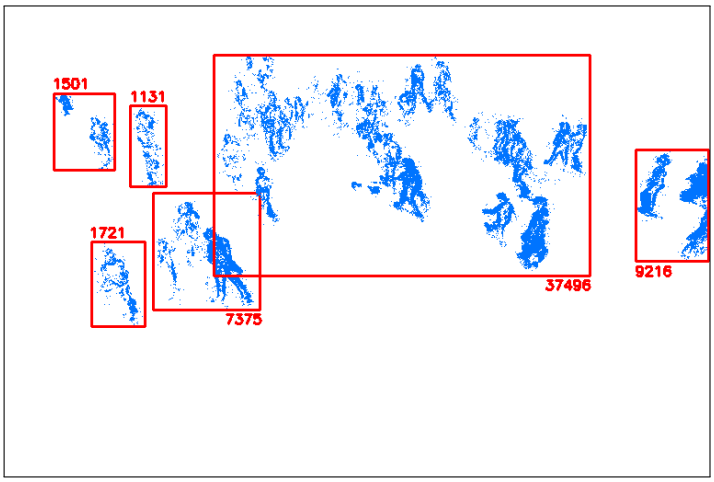
(a) Scene-based comparison ($n = 10$ human annotations)



(b) YOLO v8 segmentation histogram

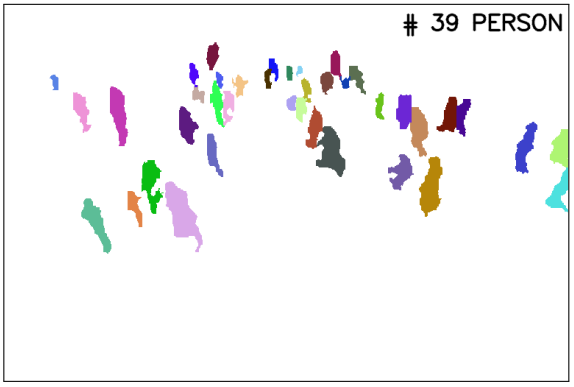


(c) Challenging scene: very crowded scene

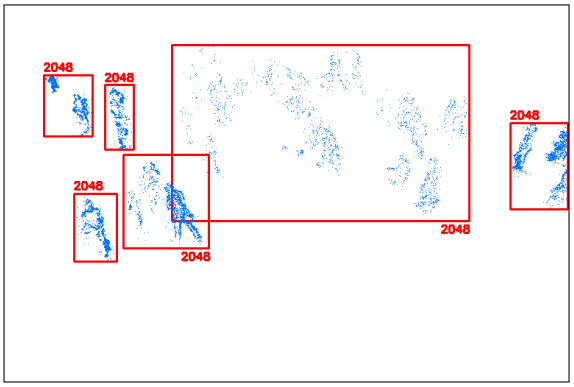


(d) Preprocessing step: aRoI generation based on event data as shown in Figure 10.7c

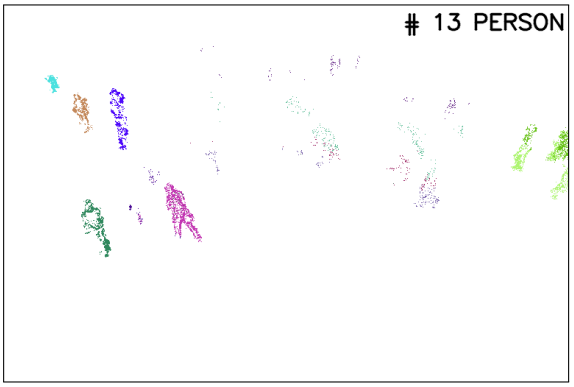
Figure 10.7: Results summary for DVS2 with typical example of a challenging scene.



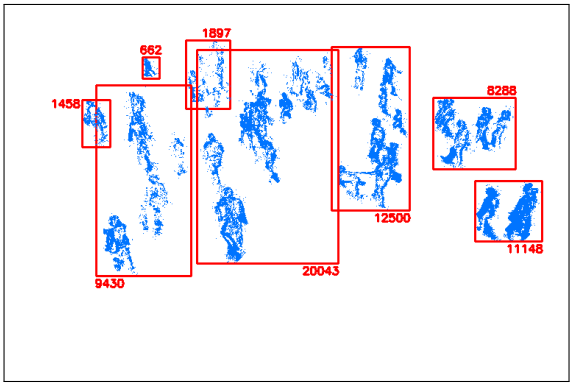
(a) YOLO v8 prediction
(based on input of Figure 10.7c)



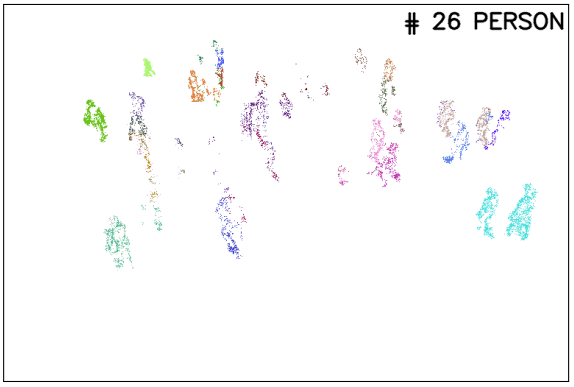
(b) Subsampled aRoI from Figure 10.7d, each with $n = 2048$ events



(c) 3D-BoNet prediction
(based on subsampled aRoI input from Figure 10.8b)



(d) *Alternative* aRoI from same crowded scene



(e) 3D-BoNet prediction
(based on subsampled aRoI input from Figure 10.8d)

Figure 10.8: Segmentation results based on crowded input scene given in Figure 10.7.

meaningful processing. Another aRoI from the same one-minute sequence is shown in Figure 10.8d and suffers from the same problem.

Therefore, the processing result of e.g. 3D-BoNet, as shown in Figure 10.8c and e, is very limited in these scenes. Frame-based processing is not limited by the number of events that need to be considered in the projection to generate the frame encoding. This gives the YOLO v8 processing an advantage in the processing very crowded scenes. The YOLO v8 segmentation result of the used example scene is shown in Figure 10.8a for reference.

Environmental Influences: Illustrated by the example of DVS3, as shown in Figure 10.9c.

The processing is performed in a multi-stage manner, i.e., environmental influences are removed prior to instance segmentation. This is necessary because these influences are not, or only to a very limited extent, taken into account in the dataset used for instance segmentation training.

Due to the nature of such multi-stage processing, errors in earlier stages are propagated through the pipeline. In some cases, where there is a strong presence of environmental influences, the UNet-based processing was not able to completely remove all environmental influences. This leads to an overestimation of the number of people in such cases for all tested approaches, both 2D and 3D, due to misclassifications in the subsequent instance segmentation.

Figure 10.9c illustrates two examples of environmental influences that were not completely removed by the applied filtering, as shown in Figure 10.9d.

The dataset on which the filtering is based, DVS-OUTLAB, was recorded in 2020, shortly after the playground was completely remodeled. At that time, the planted and seeded vegetation was not yet fully developed. Figure 10.10 shows the difference in the vegetation on the playground using orthophotos from 2021 and 2023 as an example, as no public satellite image is available for the time of the dataset acquisition in 2020. Nevertheless, the difference between 2021 and 2023 demonstrates a clear change in the vegetation on the site.

An important environmental influence that was not filtered satisfactorily was an increased amount of events triggered by grown grass moving in the wind. The influence of this type of vegetation is not satisfactorily included in DVS-OUTLAB. In retrospect, a regular update of the dataset would have been advisable, at least for the inclusion of changed vegetation.

Objects Far Away: Illustrated by the example of DVS1, as shown in Figure 10.11c.

The positions in each sensor's field of view where the most activity occurred vary greatly, as shown in Figure 10.7b, Figure 10.9b, and Figure 10.11b.

For DVS2, activity was detected over almost the entire field of view of the sensor. For DVS3, almost all of the activity occurred in the top half of the sensor's field of view. While for DVS1, the peak of detections occurred in the upper sensor row block. Due to perspective, this has a significant impact on the projected sizes of the objects.

Figure 10.11c shows examples of people in the sampled recordings that were far away from the sensor. Regarding object sizes in pixels, compare with Figure 4.2 on page 49. The processing of these small objects presents two challenges:

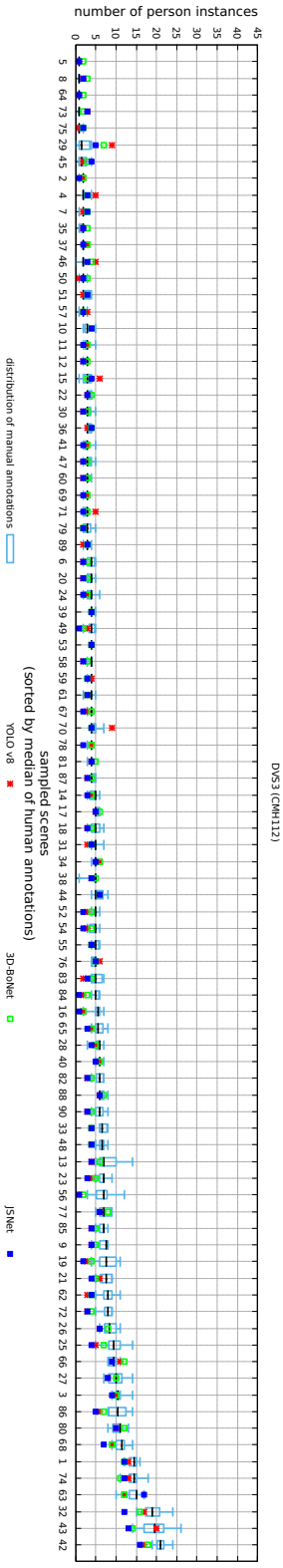
1. They tend to be removed by the applied filtering of environmental influences (cf. the missing instances in Figure 10.11d), since even for humans, their difference from sensor noise is only visible by including a larger temporal context of several seconds.

2. The remaining small instances after filtering are often misclassified as noise by the instance segmentation.

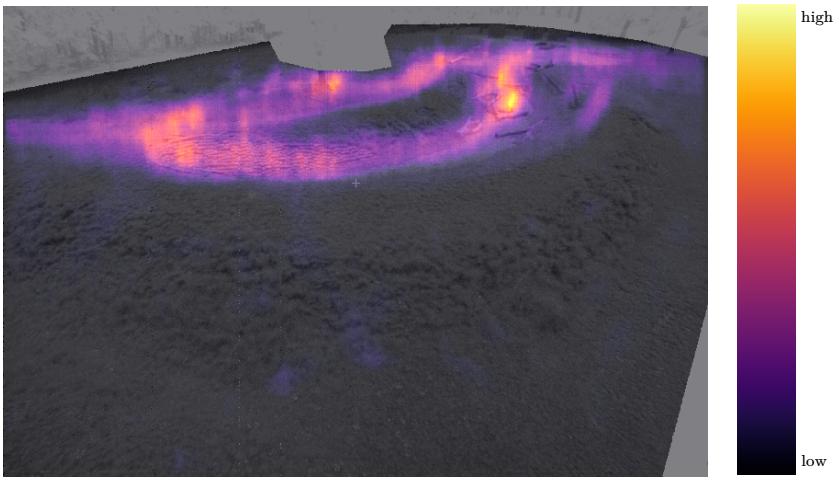
In summary, the human annotators were clearly superior to the automatic approaches in their "processing" of these small objects.

The superior results achieved for the DVS2 can be attributed to the fact that this sensor was mounted with a slightly larger inclination to the ground compared to the other sensors. This results in a significantly smaller maximum distance in the field of view, as a smaller measurement area was covered, which improves the given image scale of the objects. Figure 10.12 compares these distances and areas per sensor. The area covered by DVS2 was approximately half that of the other sensors.

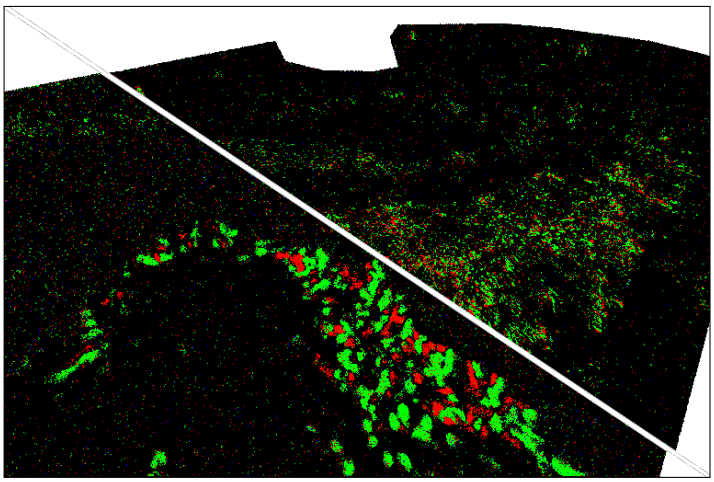
The sensor positions and orientations were originally chosen to achieve almost complete sensor coverage of the playground (see Section B.3 in the Appendix). In retrospect, due to the limited sensor resolution of only 768×512 active pixels, the area covered by each sensor should have been considered more when selecting the sensor positions. This means that the measuring field and thus the measuring ranges per sensor should have been reduced.



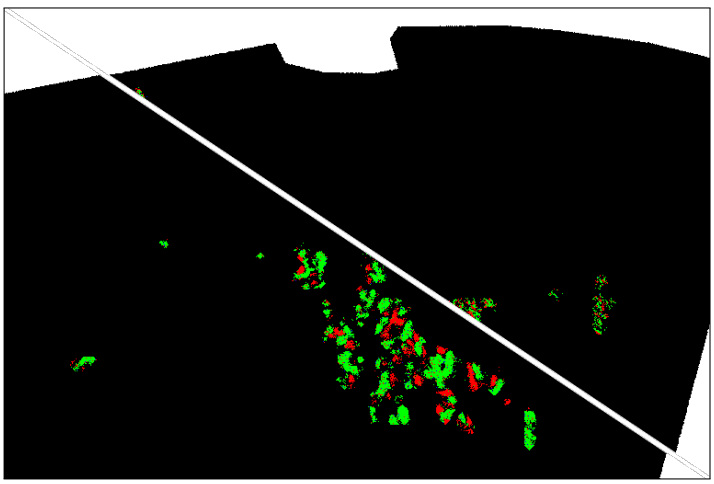
(a) Scene-based comparison ($n = 10$ human annotations)



(b) YOLO v8 segmentation histogram



(c) Challenging scene: environmental influences (two scenes are shown, diagonally separated)



(d) Preprocessing step: UNet-based pre-filtering result based on event data as shown in Figure 10.9c (two scenes are shown, diagonally separated)

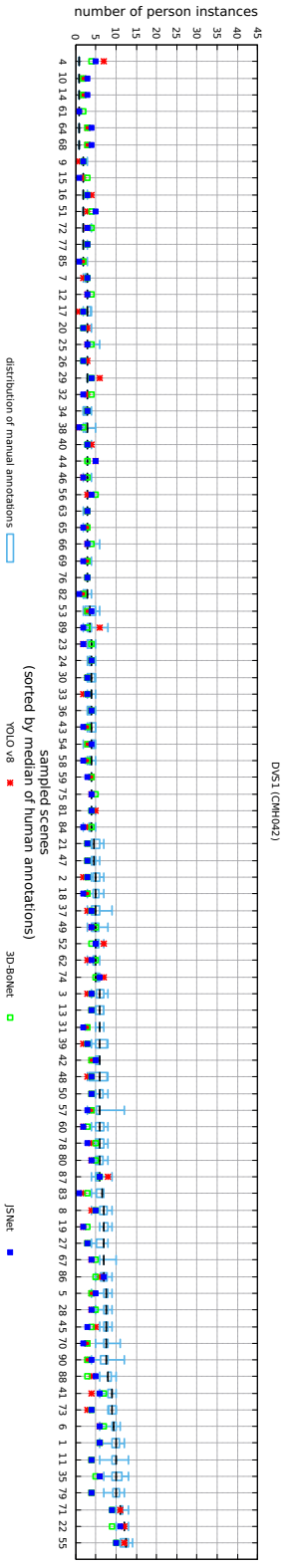
Figure 10.9: Results summary for DVS3 with typical example of a challenging scene.

(a) Year: 2021

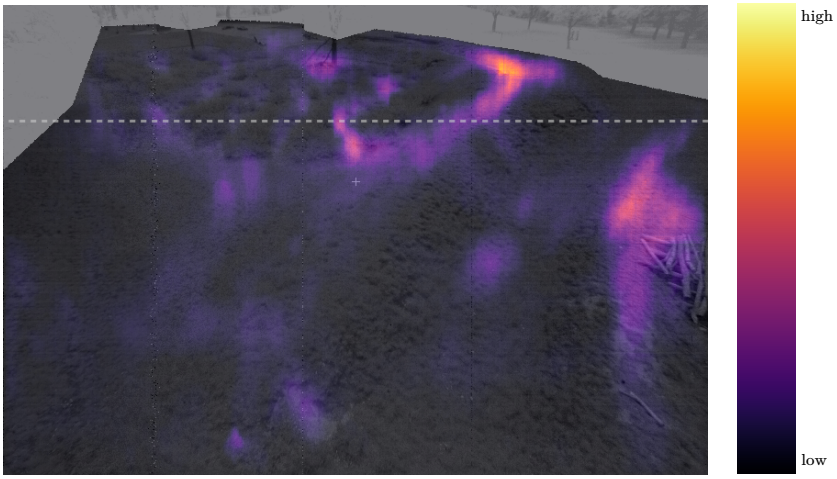
(b) Year: 2023

Figure 10.10: Comparison of historical orthophotos of the Living-Lab playground[19].

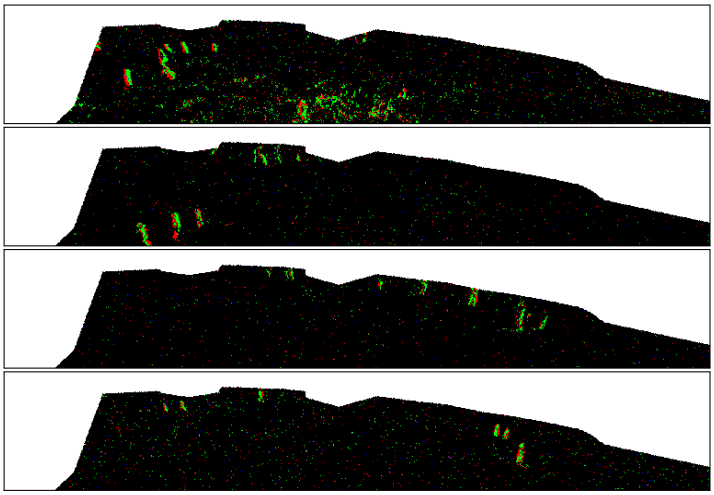
[19]Orthophotos from Figure 10.10 are provided by GEObasis NRW as part of `https://www.tim-online.nrw.de/`. Data is freely available under the `https://www.govdata.de/dl-de/zero-2-0` license.



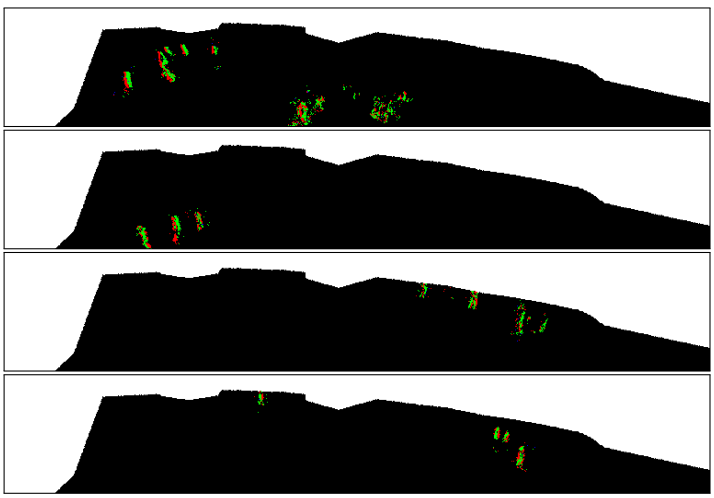
(a) Scene-based comparison ($n = 10$ human annotations)



(b) YOLO v8 segmentation histogram

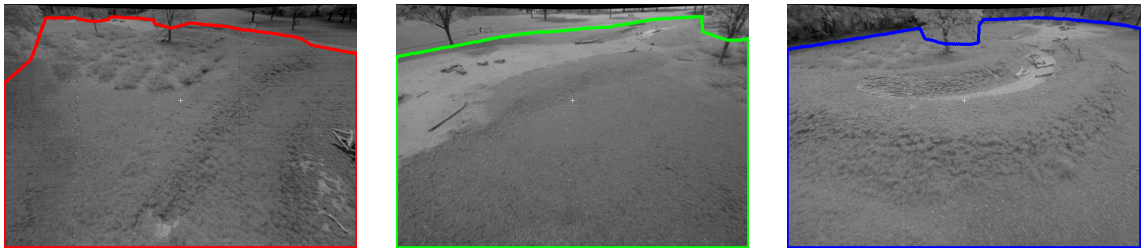


(c) Challenging scene: objects far away
(multiple scenes are shown, each cropped at upper sensor row block)



(d) Preprocessing step: UNet-based pre-filtering result based on event data as shown in Figure 10.11c
(multiple scenes are shown, each cropped at upper sensor row block)

Figure 10.11: Results summary for DVS1 with typical example of a challenging scene.



DVS1 DVS2 DVS3

(a) Grayscale scene references with highlighted mask borders

(b) DVS1

(c) DVS2

(d) DVS3

Figure 10.12: Coverage area and maximum detection distance per sensor.

Chapter 11

Spatial Distribution of Activity: Heat Map Visualization

The descriptions of the approach, methods used, and results of this chapter have previously been published in:

Bolten, T., Pohle-Fröhlich, R., Volker, D., Brück, C., Beucker, N., and Hirsch, H. (2022b). Visualization of Activity Data from a Sensor-based Long-term Monitoring Study at a Playground. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAPP)*, pages 146 – 155. INSTICC, SciTePress

In order to incorporate a user-oriented perspective into the planning and design of public spaces, urban planners need detailed knowledge of the use of an urban space. As mentioned earlier, this requires long-term monitoring of public spaces. This long-term aspect of monitoring and the use of multiple sensors generates a large amount of data. The results need to be aggregated and visualized appropriately. The key is to present it in a way that decision makers can easily understand and interpret the collected information.

Therefore, a visualization of the spatial distribution of users over the measured field was implemented. A procedure for fusing object detections into a joint map-like bird's eye view is presented.

The main contributions presented in this chapter are as follows:

- Introduction of a 3D model and simulation-based sensor view mapping to incorporate terrain modeling characteristics into the processing.
- Presentation of a processing for fusing detections from different sensors, spatial and temporal binning, and data normalization for visualization.
- Development and evaluation of different visualization variants based on a user study and presentation of a prototypical interface.

Introduction

There are several approaches to the visualization of space-time data in the literature. For example, measured parameters can be used to color individual objects of the observed infrastructure. However, the observed playground in this work is designed as a nature experience space and therefore does not contain classical playground equipment. Instead, there are different areas where children can playfully interact with natural materials. Therefore, in this practical case, representations of playground equipment cannot be used for visualization.

Scatterplots are typically used to visualize fewer recorded events at a location, such as manual observations of people on a street segment over a limited period of time. These visualizations were described and used, for example, by Whyte, who was a pioneer of systematic user analysis in urban planning [Whyte, 1979]. His study protocols [Whyte, 1980] are still leading information graphics in the field of urban studies. The use of bubble plots allows the aggregation of larger datasets by including the size and color of the shape used for the visualization. A top view of the observed area is typically used when creating visualizations.

Usage maps in the form of choropleth maps or heat maps, are often created to visualize the results of automated data collection and the resulting large amounts of data. This is a visualization technique in which the measured variable (e.g., the number of users) is displayed in different colors taken from a defined colormap. Choropleth maps display information divided into spatial political or geographic units, while heat maps are typically not limited to geographic or social boundaries.

In [Moussouri and Roussos, 2014] and [Rashed et al., 2016], heat maps were used to analyze which regions of a museum were used more by visitors. Bolleter also used heat maps to present the results of Wi-Fi tracking in public spaces to count people and map their stays and movements [Bolleter, 2017]. The activity in different areas of a student dormitory depending on the day of the week is visualized in [Rajasekaran et al., 2020] using the same technique.

In general, these heat maps can be created in both 2D and 3D form. In most cases, abstract 2D representations are preferred over 3D representations. This is certainly due to the fact that 2D representations do not require any additional interaction to interpret the data. In some projects, both options are used together, as in [Rezaei and Azarmi, 2020] to visualize tracking and distance maps.

11.1 Proposed Processing

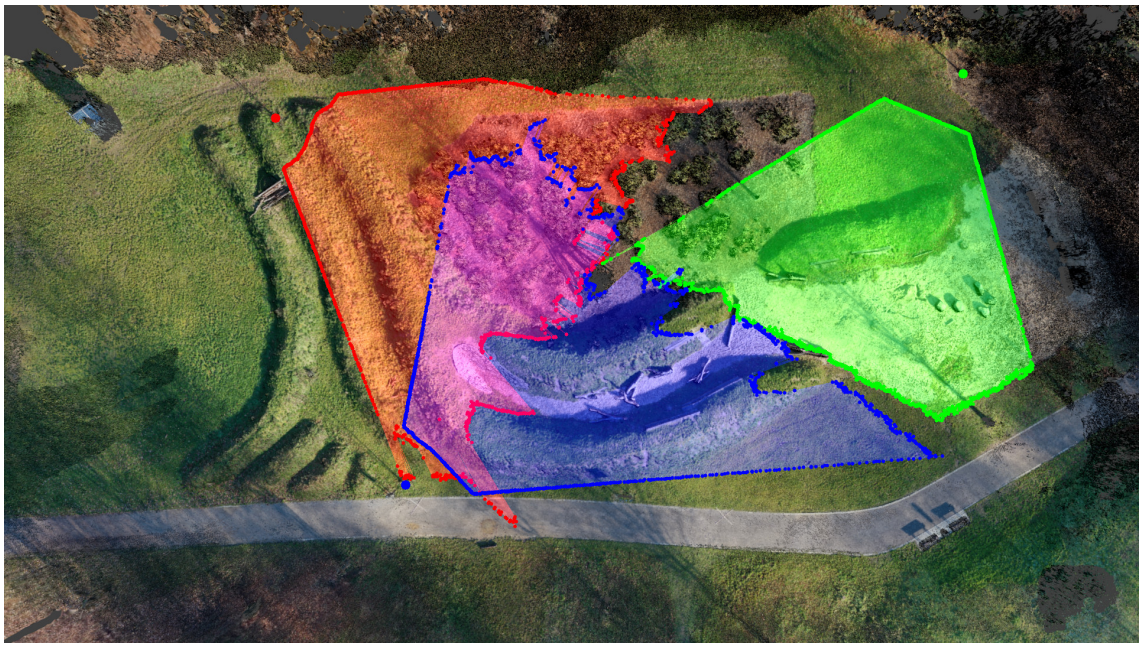
For the creation of visualizations, the sensor setup of the considered Living-Lab scenario has to be taken into account, since the processing results of the individual sensors have to be combined. The measurement setup on the Living-Lab site was designed and built in such a way that the mounted sensors could cover the entire measurement area.

Figure 10.12 shows the total area covered by each sensor. However, detections in the top pixel rows of the sensor have proven to be very error-prone due to the projected small object sizes. Figure 11.1 shows the overlap in the covered areas when the top 64 pixel rows of the actively used sensor array are ignored. As a result, there are very limited areas of overlap between the fields of view of the individual sensors, where a high and reliable level of detection quality is provided. For this reason, the integration of this multi-sensor setup into the preceding semantic processing (e.g., with respect to fusion and/or depth estimation) is very limited.

Therefore, transferring the obtained individual results into a shared representation is an alternative form of processing. For this purpose, we developed a projection of the individual detection results into a map-like bird's eye view. In the following, the developed processing to derive coordinates for a joint visualization from multiple sensors is presented.

DVS1 DVS2 DVS3

(a) Grayscale scene references with highlighted detection areas. The top 64 pixel rows are ignored in addition to the applied masking.



(b) Overlap

Figure 11.1: Adapted sensor coverage areas and overlap.

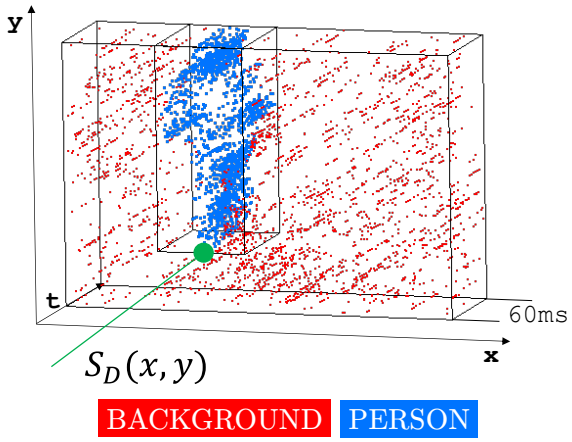


Figure 11.2: Object segmentation example with derived object detection coordinate $S_D(x, y)$ (from [**Bolten** et al., 2022b]).

(a) 3D Rendering (b) Wireframe

Figure 11.3: Calculated SFM 3D model of the playground (from [**Bolten** et al., 2022b]).

11.1.1 Data Preprocessing

One of the processing modules introduced in Part III is used to process the DVS event stream and derive object instance segmentation. This is achieved by semantic segmentation and further clustering or by direct instance segmentation. A single object class is selected and filtered for further processing.

For each selected object instance of the sensor S a detection coordinate $S_D(x, y)$ relative to the field of view of the sensor is determined. This coordinate is centered on the x-axis of the object and is located on the ground plane where the object moves. Figure 11.2 illustrates a detected person and the derived $S_D(x, y)$ coordinate.

11.1.2 Bird's Eye View Mapping

The next step is to map the derived object detection coordinates $S_D(x, y)$, which are relative to the sensor field of view, into a global representation of the playground, such as a top view or a blueprint view, as shown in Figure 3.1a on page 22.

Mapping these object detection coordinates is a challenging task due to the complex terrain modeling of the monitored playground area, including hills and valleys. A simple geometric transformation between the sensor field of view and the map representation would result in significant inaccuracies due to the lack of a flat surface plane. To obtain a more precise estimation, we generated a 3D model of the playground area and conducted the mapping using this model. In this way, the terrain modeling is included in the applied transformation.

The 3D model was created from 285 images captured during a drone overflight. The camera

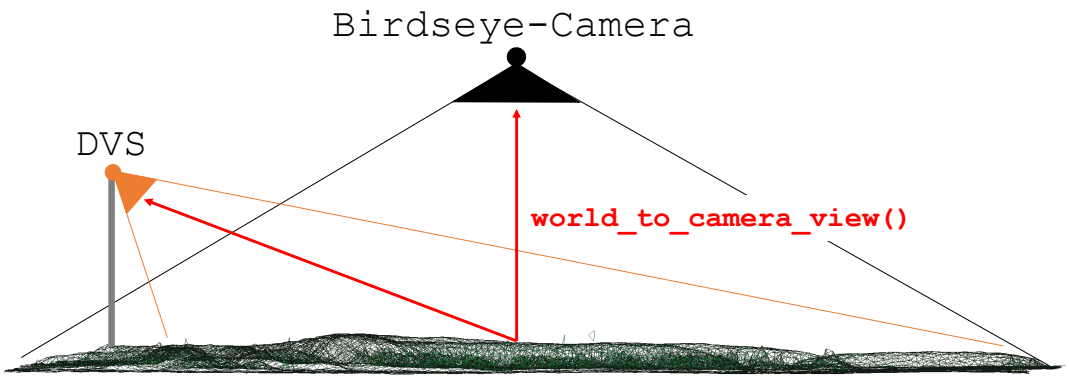


Figure 11.4: 3D model based coordinate lookup table generation (from [**Bolten** et al., 2022b]).

positions in these overlapping images were estimated using the structure from motion (SFM) technique, based on automatically detected corresponding points. The points of a 3D model were calculated based on these camera positions. Agisoft Metashape[20], a commercial software, was used for the estimation. Figure 11.3a shows a 3D rendering of the model created, while Figure 11.3b highlights its terrain characteristics.

This playground model was then used for a 3D simulation based on Blender, a free and open source 3D modeling and animation software[21]. In this simulation, camera objects were positioned and aligned with the real Dynamic Vision Sensor positions in the Living-Lab. In addition, a bird's eye camera B was positioned.

The simulation also utilized the DVS parameters, including sensor chip size, pixel resolution, and focal length of the attached lens. The purpose of this simulation was to determine the relationship between the mesh vertices of the 3D model and the corresponding pixel coordinates of each Dynamic Vision Sensor and the bird's eye camera. Based on the bird's eye camera B, a lookup table was created for each DVS, which allows a pixel mapping from $S_D(x, y) \mapsto B_D(x', y')$ within a rendered top view. This processing is shown in Figure 11.4.

11.1.3 Multi-Sensor Detection Fusion

In general, this processing results in a high-quality mapping between the (x, y)-coordinates of the sensor view and the corresponding points in the bird's eye view. However, due to terrain modeling, there are still leaps in this bird's eye mapping. This is illustrated in Figure 11.5 for one of the sensors.

For each point $S_D(x, y)$ of the sensor view, the resulting maximum distance in meters within the bird's eye projection is shown, which results from projecting the coordinates within the 8-connected neighborhood $S_D(x \pm 1, y \pm 1)$ around this point. Shifting the sensor detection coordinates by only one pixel can result in projection distances of several meters on the ridges of the terrain model. These projection distances are taken into account as an "uncertainty factor" when merging the detections from all sensors into a joint representation.

The object detections of different sensors are first synchronized based on the underlying DVS event timestamps. For synchronous detections from different sensors, it is tested whether

[20] `https://www.agisoft.com/`
[21] `https://www.blender.org/`

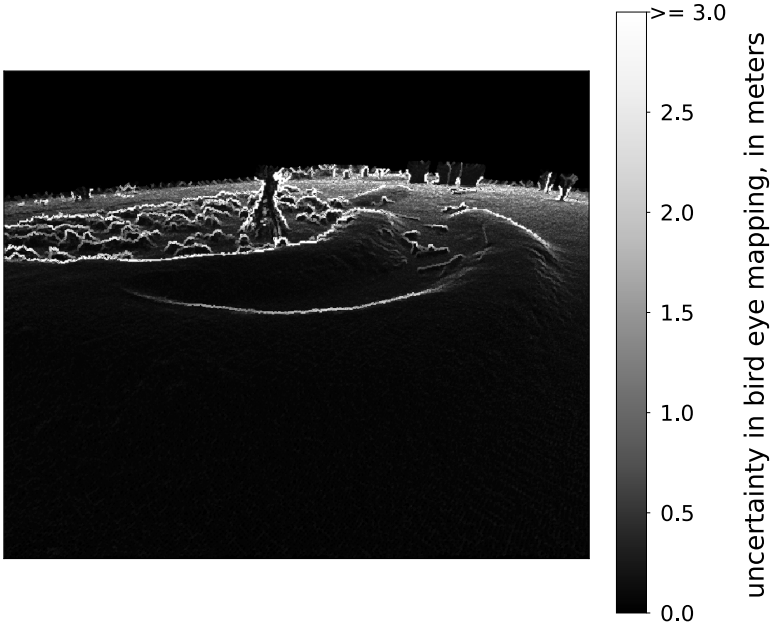


Figure 11.5: Visualization of the uncertainty in the $S_D(x, y) \mapsto B_D(x', y')$ pixel mapping (shown exemplarily for DVS3 within the Living-Lab setup, compare with Figure 3.2c on page 22; from [**Bolten** et al., 2022b]).

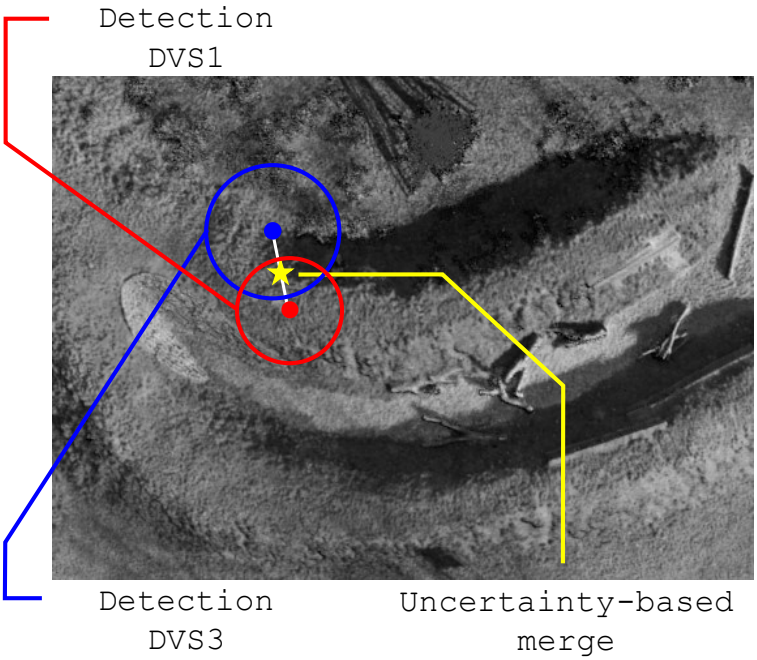


Figure 11.6: Cropped and zoomed example of uncertainty-based merging of projected bird's eye detections (from [**Bolten** et al., 2022b]).

there are overlaps with other detections within these uncertainty distances. This procedure is illustrated in Figure 11.6.

The filled circles represent synchronous detections of the sensors, while the outer circles indicate the corresponding uncertainty distances within the bird's eye mapping process. The radii of the DVS1 and DVS3 detections overlap, resulting in a merged detection for visualization purposes.

11.1.4 Spatial and Temporal Binning

The playground's bird's eye view is divided into several sub-areas using spatial binning to visualize activity levels. Activity is then determined and plotted for each sub-area. The grid used for spatial binning is approximately 0.5 to 1 square meter per bin. This selection was made in order to be able to identify small local differences in usage frequency despite the overall size of the monitored area.

The long-term observations include periods of varying lengths during which activity occurred at the site. Since both short-term and long-term aspects are of interest in the evaluation, the ability to visualize time intervals of different lengths is essential. This poses the challenge that the displayed activity levels (low to high) must be comparable to each other when visualizing periods of different lengths, e.g. using the same colormap. For this reason, we apply the following normalization technique.

In addition to the described spatial binning of the monitored area, a further temporal binning is performed. For each spatial bin, the fused detections are sorted temporally. The entire temporal interval for visualization is divided into smaller sections. The number of these smaller temporal sections containing at least one object detection is then determined. In this way, a percentage normalization between $[0, 1]$ is realized as a function of the total duration considered.

For example, an observation time for visualization of one hour and a division into temporal bins of one minute length results in a scaling of $x/60$, where x is the number of populated one-minute detection time blocks. The division into these short time segments (e.g., one minute) is possible because the semantic analysis of the DVS event stream is performed in much shorter time windows of only 60 ms.

11.2 Created Visualizations

Heat maps were selected as the visualization technique to provide decision makers with an intuitive and quick overview of the activity levels detected. All generated plots were created using the library `matplotlib` [Hunter, 2007].

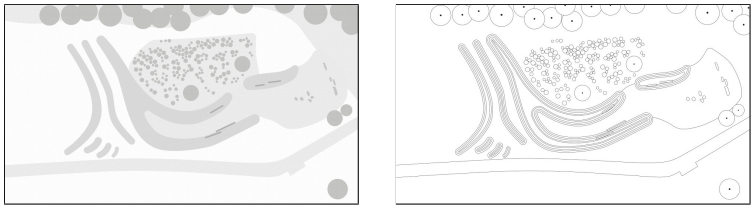
11.2.1 Heat Map Variations

Several variations were considered and then evaluated in a user study to generate these heat maps.

Level of Detail in Bird's Eye View (LoD)

To visualize the spatial location of activity detections, it is crucial to have an image or map representation of the monitored playground area. However, the background image used to represent the playground can be created at different levels of detail.

(a) RGB rendering (b) Grayscale rendering (c) RGB sketch



(d) Grayscale sketch (e) Simplified sketch

Figure 11.7: Level of detail used in the bird's eye view (from [**Bolten** et al., 2022b]).

In our work, we created photorealistic renderings based on the performed 3D Blender model simulation, as well as abstracted sketches of the monitored area. Figure 11.7 shows examples of these background images, including different levels of detail and color or grayscale variants.

3D vs. 2D Display Dimensionality

Display variants were created with a two-dimensional and a three-dimensional rendering of the plot.

In the case of the 3D display, the activity is represented by the height of a three-dimensional surface in addition to the color used. An example is shown in Figure 11.8a. To interpret these 3D views, it is necessary to change the viewport. Therefore, animations have been created that include a 360-degree rotation of the plot.

A two-dimensional image, on the other hand, allows data interpretation without this type of animation or interaction. An example of this visualization is shown in Figure 11.8b.

Spatial Binning Variants

Besides the straightforward variant of spatial (x, y)-binning into rectangular tiles, we also applied a tessellation of regular hexagons, as shown in Figure 11.9a and b. In addition, a contour-based representation has been implemented to group areas with similar usage (see Figure 11.9c).

These variants share a discontinuity in the generated representation. As an alternative, a variant based on Kernel Density Estimation (KDE) has also been implemented, which allows a gradual smoothing of the activity level. An example of this is shown in Figure 11.9d.

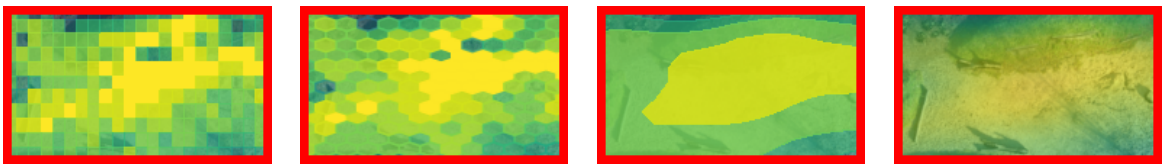
Colormap Variants

For heat maps, selecting an appropriate colormap is critical to interpreting the data [Eghteabs et al., 2017]. This selection should also take into account the needs of people who are colorblind or elderly and who may otherwise have difficulty interpreting the data [Silva et al., 2011]. Different types of colormap generation are described in the literature [Zhou and Hansen, 2015].

Colormaps can be generated procedurally, with the goal of being able to interact with as many different datasets as possible. In addition, there are perception-based rules for specific applications that have been learned over time. The display of measured temperatures on a

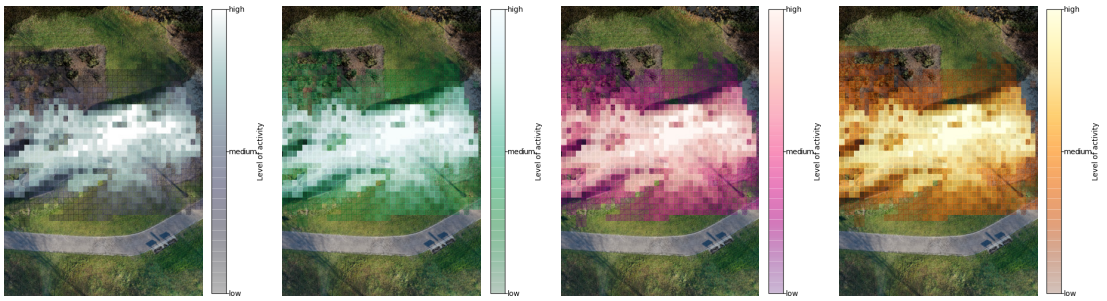
(a) 3D with animated viewport (b) 2D projection

Figure 11.8: Spatial display variants in 3D and 2D (from [**Bolten** et al., 2022b]).



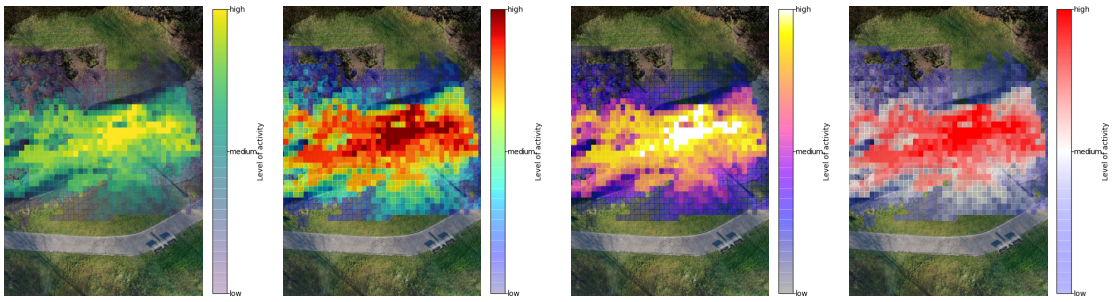
(a) Rectangles (b) Hexagons (c) Contour (d) Continuous density, resulting from KDE

Figure 11.9: Applied spatial binning and display variants. Shown is the highlighted region extracted from Figure 11.8b (from [**Bolten** et al., 2022b]).



(a) Bone (b) BuGn (c) RdPu (d) YlOrBr

Figure 11.10: Applied sequential colormaps (from [**Bolten** et al., 2022b]).



(a) Viridis (b) Jet (c) Gnuplot2 (d) Bwr

Figure 11.11: Applied diverging colormaps (from [**Bolten** et al., 2022b]).

color scale from blue to red is a prominent example of this. For our monitoring application, data-driven generation is relevant to ensure that both short-term and long-term data are interpreted in a similar way. To visualize ordinal data, sequential perceptually uniform maps should be considered that accurately reflect numerically equal distances between values for equally perceived color differences [Moreland, 2009].

For some tasks, there are user studies that have been used to develop and select appropriate colormaps. Compared to the colormaps that have long been learned and used daily by the general public (e.g., in weather forecasting), it is an open question which colormap is best suited for decision makers in terms of being easily interpretable and intuitively linked to the activity level of the area as determined by the performed monitoring.

Therefore, several simple sequential colormaps (see Figure 11.10) as well as diverging colormaps (see Figure 11.11) were considered and evaluated. For the diverging colormaps, the usual use of a center point within the double-ended gradient was omitted. This results in a continuous color fade for the detected and normalized activity level from low to high through this center point.

Perceptually uniform perceived colormaps were also taken into account in this colormap consideration. In addition, the direction of the colormap used to encode the activity level, i.e. from dark to bright or vice versa, was also considered.

A linear alpha blending with higher transparency factors for lower activity levels was used within the colormap to overlay the generated activity visualizations on the selected background plot image.

11.3 User Study

The previously described activity heat map representations and variants were evaluated in an online survey. This online survey was conducted as part of the underlying EFRE-funded research project. It was led by the participating Faculty of Design at Niederrhein University of Applied Sciences.

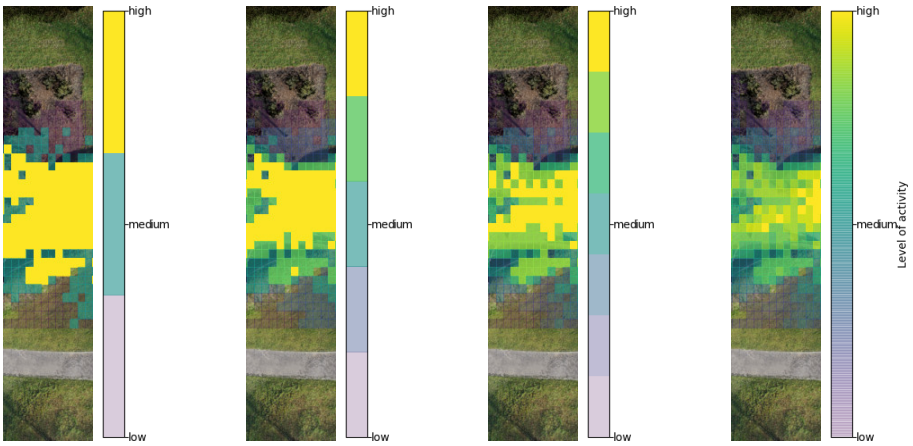
The carefully selected participants came from professional backgrounds in urban planning and interaction design, but also interested citizens were involved in order to reflect the layman's perspective. The survey was designed to test the comprehensibility of the generated activity representations by means of heat maps, which were created based on design assumptions regarding readability and experience-based knowledge. It was also intended to identify adequate default settings for future visualizations and applications.

Sixteen participants participated in the online survey. They had the opportunity to rate the different visualizations and were asked to comment on their decisions. The results reveal trends and confirm previously stated assumptions.

Level of Detail and Dimensionality

The RGB rendering of the bird's eye view image (see Figure 11.7a) was selected as the preferred variant for the level of detail of the background image. However, comments indicated that the displayed user activity could not be mapped accurately and precisely because the heat maps overlaid the image too strongly.

For the 3D view, it was found that a high level of user activity was displayed to far away from the map, making it difficult to classify.



(a) 3 color blocks (b) 5 color blocks (c) 7 color blocks (d) 256 color blocks

Figure 11.12: Used level of detail in activity colormap through further binning it into n blocks (from [**Bolten** et al., 2022b]).

Colormaps and Binning

Regarding color, it can be summarized that colormaps with diverging colors were preferentially associated with user activity. One assumption as to why these diverging colormaps were preferred over the sequential ones is that they offer a higher contrast and thus are more easily distinguishable, even for older users. This contrast discrepancy is further enhanced by the increased alpha blend factor used at lower activity levels.

However, contrary to the common assumption in the literature [Moreland, 2009], no perceptually uniform colormap was preferred. The colormap `jet` (see Figure 11.11b) was preferred by the participants. We assume that this colormap preference is indirectly based on experiential knowledge (compare with commonly used colormaps in weather forecasting, ranging from blue to red), leading to a corresponding association and increased intelligibility.

With respect to the colormaps used, a further simplification has been evaluated within the user survey. This simplification involves quantizing the continuous activity scale into coarser sections, as shown in Figure 11.12. The idea was to clearly divide the activity into blocks that can be referenced from “low” to “high”. However, the study participants clearly preferred the full 8-bit color gradient as shown in Figure 11.12d. Therefore, subdividing the color scale was not considered further, as it also leads to a loss of information and makes it more difficult to identify hotspots on the plot.

For the spatial binnings, most respondents chose to display user activity in rectangular subareas (see Figure 11.9a). This preference is consistent with established representations. Similar rectangular grids were used by the pioneer of spatial observation, William H. Whyte.

Plausibility Check

Finally, a plausibility check was performed in the online survey. Users were shown a map with three marked areas. They were asked to select the marked area where the highest user activity was measured. The correct answer was chosen by 15 out of 16 respondents. This confirms the comprehensibility of the presentation.

11.4 Interface Prototype

The studied visualizations provide a basis for the design of an integrated interface to evaluate the spatial distribution of users in the measured area. The large amount of collected and complex data can thus be made available interactively to urban planners and interested citizens.

In addition to the monitoring data collected by the installed Dynamic Vision Sensors, weather data should also be recorded. This data can contribute to a better estimation of the usage of the playground at a given time, as fewer users can be expected during cool rainy weather than during warm sunny weather.

A prototyped interface mockup is shown in Figure 11.13. This interface provides the user with the ability to select individual time frames and to focus on spatial sub-areas of the playground. In addition, details of the prevailing weather conditions are directly integrated to easily provide insight into the circumstances during the considered time period of the visualized usage on the site.

The heat map visualization was adapted based on the results of the user study. A sketch drawing of the playground was superimposed on the rendering of the user activity to improve the spatial attribution as shown in the interface mockup. This allows for a more accurate spatial mapping of the user activity.

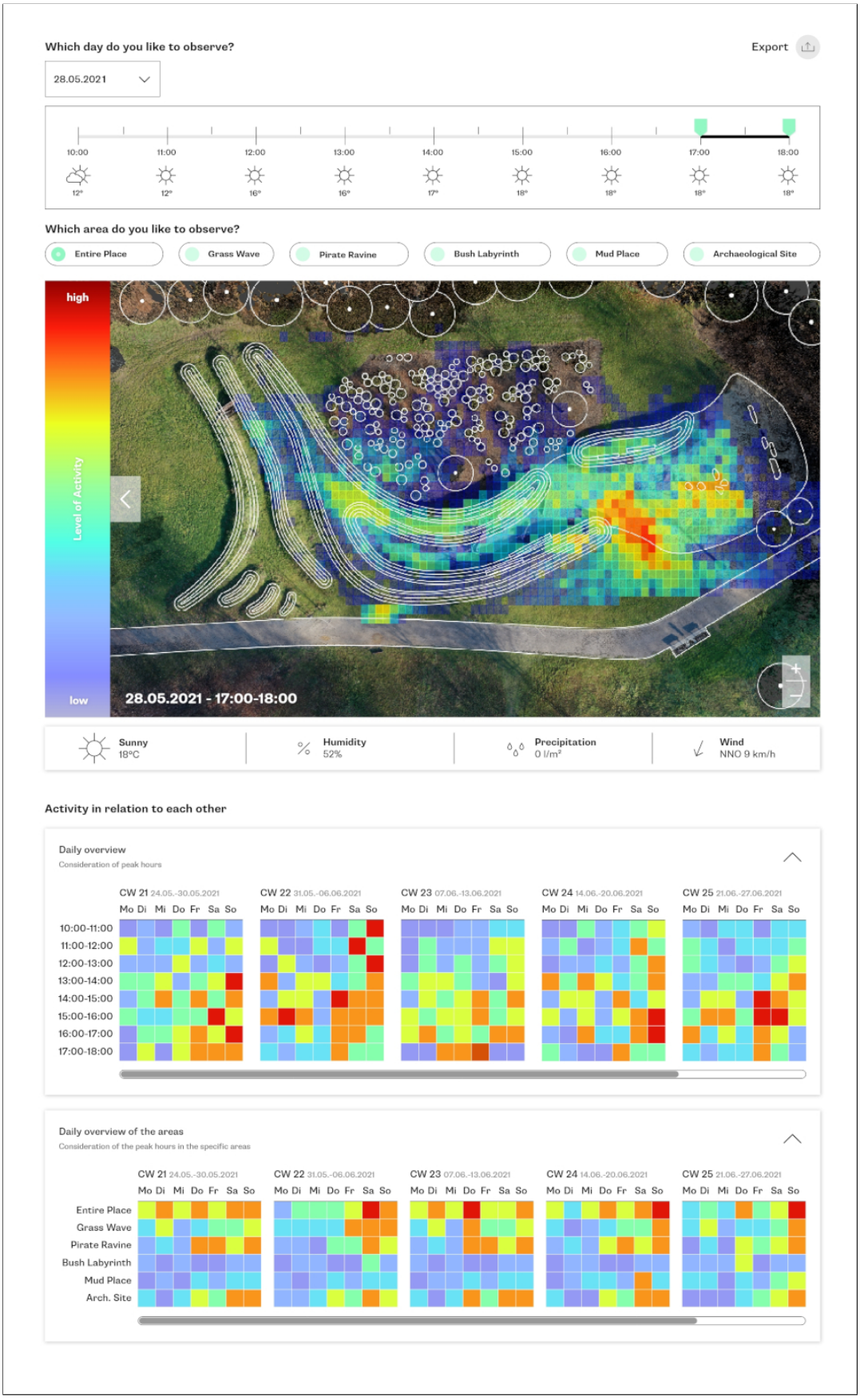


Figure 11.13: Example for prototyped user-interface (from [**Bolten** et al., 2022b]).

Summary of Part IV

Large amounts of data are generated during long-term monitoring, especially with multi-sensor measurement setups. Therefore, efficient preprocessing is required.

A processing pipeline was presented that is able to extract scenes of further interest within the performed monitoring. The objects contained in the scenes are used for decision-making. Only a subset of the included object classes, so-called objects of interest, lead to the persistence of the ongoing recording. This reduces the memory footprint of the monitoring process.

The results of the segmentation analysis performed in Part III were used to realize an implementation that takes into account the limited computing and power capacities on-site. During processing, spatial areas of interest are identified by filtering and thresholding. These areas are then processed in a downstream semantic segmentation. Based on an evaluation of unprocessed recordings, the presented pipeline was shown to be able to filter scenes with a high level of quality. On a large test dataset, an overall F1 score of approximately 0.94 was achieved for the separation of long-term recordings at the scene level. Therefore, the pipeline was directly integrated and used in the recording environment of the on-site long-term recordings.

One of the two primary parameters to be derived by the developed monitoring system is the number of users within the observation area. In order to estimate the performance of the system in this task, all recordings from the entire measurement period in 2021 were processed. The selection of a subset of scenes was necessary to keep the processing and subsequent evaluation within reasonable limits. Representative scenes were selected, including the full range of on-site and typical usage scenarios. The scenes were then annotated by ten human reviewers to provide a human-based reference of the included user volume.

A multi-stage processing approach was developed, consisting of two stages. The first stage involved the filtering of environmental influences through the application of a UNet-based approach. The second stage involved the application of 2D frame and 3D space-time event cloud-based instance segmentation, to automatically determine the included user volume. The results obtained were then compared with the aforementioned human reference. Both 2D and 3D based processing yielded satisfactory results. Typical error cases were reviewed and used to illustrate aspects that should be considered in the development of future monitoring systems, scenarios, and their designs.

The obtained qualitative segmentation results also indicated the applicability of the developed processing to extract the spatial distribution of the detected and counted objects. However, an intuitive aggregation of the derived spatial distribution of users across the measured playground is necessary to present the obtained results to urban planning and design stakeholders.

We introduced a 3D SFM model extracted from drone footage to integrate the existing terrain modeling into the processing. Based on this model, a cartographic bird's eye view of the playground was created. In addition, this model allows a precise mapping between the DVS pixel coordinates and the generated top-view image. A procedure for fusing and normalizing

temporally and spatially close objects was presented.

Different heat map visualizations were created. A user study was conducted to evaluate their comprehensibility. Finally, a prototype of an interface was presented. This interface aims at an interactive exploration of the obtained monitoring data.

Part V

Conclusion

Chapter 12

Summary and Conclusion

A proof-of-concept monitoring system based on Dynamic Vision Sensors has been developed that, by analyzing long-term observations, can create an evidence-based database on the long-term use of public space. This can serve as a basis for downstream decision-making in an urban planning process.

The development of this system had to take into account real-world aspects, as the practical application of the measurement setup in the Living-Lab setup outside of controllable laboratory conditions leads to influences that have to be dealt with.

A summary of the progress made in the development of this monitoring system is provided, along with a critical review of the limitations of the system. Additionally, suggestions for improvements and further work are presented.

12.1 Summary

Due to their technical advantages over conventional image sensors, Dynamic Vision Sensors are well-suited for outdoor applications such as the monitoring scenario considered in this work.

In particular, they provide data acquisition with minimal redundancy and low power consumption. In addition, due to their very high dynamic range, they are able to operate effectively directly under a wide range of lighting conditions without the need to adjust characteristics such as exposure time, lens aperture, or light sensitivity gain during ADC conversion, as is the case with conventional sensors. This also applies to the *intrascene* dynamic range, allowing very bright and very dark areas of the same scene to be captured simultaneously. This is particularly useful and interesting when monitoring outdoor areas.

12.1.1 Contributed Datasets

Despite these technical advantages, Dynamic Vision Sensors have not been widely used in such monitoring applications. One obstacle has been the lack of publicly available datasets required to train and evaluate processing approaches. To address this issue, we have contributed two entirely new datasets.

The DVS-OUTLAB dataset can be used for semantic segmentation in outdoor monitoring applications, as it incorporates the environmental influences that must be taken into account in real outdoor monitoring. This dataset provides over 47,000 semantically labeled DVS data patches.

We also contributed N-MuPeTS, a DVS-based dataset containing continuous recordings of up to four actors performing scenarios that arise from common tracking and segmentation

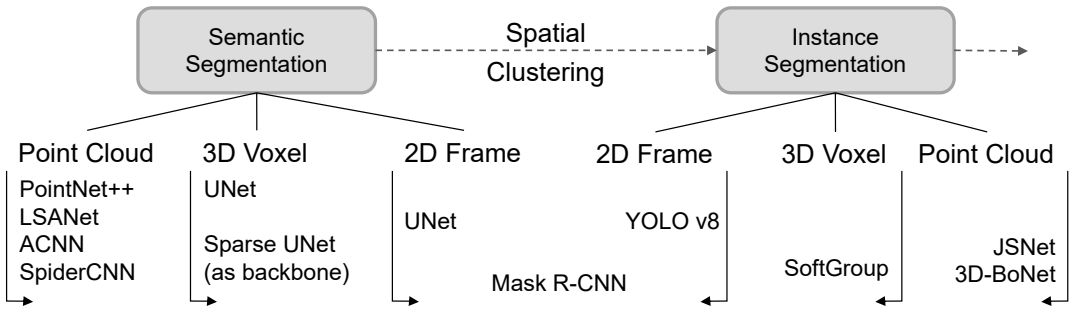


Figure 12.1: Summary of analyzed processing approaches and representations.

challenges, while providing object instance labels and activity annotations. This allowed us to leverage and evaluate deep learning-based instance segmentation approaches in the context of event-based vision.

12.1.2 Segmentation

The output of Dynamic Vision Sensors is fundamentally different from that of standard image sensors because it is asynchronous, unordered, and spatially sparse. For processing, it can be represented in different forms, including space-time event clouds, event graphs, voxel grids, or it can be projected onto an image plane to create a classical frame.

Currently, there is no de facto standard representation in the literature. For this reason, different representations, with a focus on three-dimensional space-time event clouds, and their processing with different deep learning approaches were evaluated with respect to their segmentation results in the considered monitoring context. A summary and overview of the considered processing approaches and corresponding representations is shown in Figure 12.1.

For semantic segmentation, the processing of space-time event clouds by PointNet++, the pioneering and groundbreaking approach for deep learning-based point cloud processing, was optimized in terms of network hyperparameters and input data preprocessing. This resulted in a set of recommended parameter values. The 3D point-based processing achieved weighted F1 score results of > 0.93, while outperforming a 2D frame-based Mask R-CNN reference method in both quality and runtime.

The experiments were extended to include voxel-based representations for processing. However, the selection of the voxel size to be used is a difficult and often ambiguous task. Due to the sparsity of the DVS event stream, many voxels remain empty when trying to preserve fine details in the voxelization step. The use of classical 3D convolutions on this data has not led to improvements.

Considering the task of instance segmentation, point-, voxel-, and frame-based representations and processing were also compared. The experiments, performed on two separate datasets, showed differences to be considered depending on the main objective for processing. Space-time event cloud-based processing demonstrated superior per-event accuracy for predicted instances, while frame-based processing was limited by the achieved pixel accuracies in the predicted object masks.

For example, the experiments with the DVS-iOUTLAB dataset showed a difference of 19 % in favor of point-based processing by JSNet for the $\mathrm{mAP}^{0.95}_{0.5}$ score, which takes greater account of details than the $\mathrm{mAP}^{0.5}$ score due to the inclusion of higher IoU thresholds. For the $\mathrm{mAP}^{0.5}$ score, on the other hand, there was a 9 % improvement in favor of frame-based Mask R-CNN processing.

12.1.3 Real-World Application Context

A Living-Lab-based long-term monitoring was conducted in the real-world outdoor context of a children's playground. The study resulted in a comprehensive collection of DVS monitoring data recorded between 2019 and 2021.

Since this public space is not in constant use, it was necessary to identify time segments containing events triggered by a predefined set of object classes and to store them for later, more in-depth analysis. This selection had to take into account the environmental influences included in the sensor signal, as well as the limited computing capabilities on-site. To achieve this, a processing based on 3D point-based semantic segmentation was implemented.

An evaluation on real-world data of almost 90 hours, recorded over several days under varying weather conditions, demonstrated a false positive and false negative rate of 1.5 % and 3.8 %, respectively. This proved the reliability of the on-site scene selection and provided an appropriate storage solution.

The system primarily aims to derive two main parameters in order to create an evidence-based database on the long-term use of the area to support the decision-making process in urban planning. These are the number of users and their spatial distribution within the monitoring area.

The goal of counting the number of users in the area was evaluated based on sampled representative scenes from the monitoring conducted in 2021. From the pool of approximately 630 hours of recorded DVS data, 270 minutes were sampled covering the full spectrum of occurred usage, including scenes with only one or a few people to very crowded scenes. Subsequently, 2D frame-based and 3D space-time event cloud-based instance segmentation were used to determine the number of included `PERSON` instances, which were compared to the distribution of manual counts provided by ten different human annotators. On average over all recordings evaluated, the three-dimensional processing by 3D-BoNet achieved the best results on this task. Counting based on 2D and 3D segmentation of recordings from the sensor that covered the most activity resulted in an average deviation of less than 0.33 and 0.35 counted `PERSON` instances outside the interval formed by the human reference, respectively.

Long-term monitoring systems generate a substantial quantity of data. In order to provide a meaningful and usable insight about the spatial distribution of observed space usage, an aggregation is necessary. The results of this aggregation must be easily and intuitively understood by the stakeholders who want to benefit from the monitoring.

For this purpose, a visualization was implemented based on the obtained instance segmentations. This visualization combines the detections into a joint bird's eye view of the playground and shows the different spatial distributions of usage as a heat map false color representation. The elevation modeling of the playground was integrated into the mapping between the sensor coordinates and the bird's eye view based on a 3D terrain model. The suitability of this visualization approach was evaluated in a user study and an interface prototype was created for a possible future end-user application.

12.2 Conclusion: 3D vs. 2D Event Representation

The processing of 3D space-time event clouds by deep learning-based approaches currently requires data subsampling to meet practical event count limitations. This has been achieved by introducing spatial patching in addition to temporal windowing of the data.

For this purpose, the scenes were initially divided into fixed-sized subareas, so-called Patches-of-Interest. However, this can lead to objects being cut off at the boundaries of these patches. In the course of this work, this patching procedure was modified to be adaptive in size. Based on an event-clustering step, input regions were formed in which objects are not cut into separate parts. However, this preprocessing, either in fixed-size or adaptively formed input regions, results in a higher number of inferences needed for 3D-based processing compared to a 2D-based processing counterpart. This is due to the fact that 2D-based processing is capable of processing the entire data of a temporal window without spatial patching in a single run.

In order to aggregate the spatial distribution of usage within the final projection step into a joint bird's eye view, the obtained instance segmentation quality must be considered. It was observed that the 3D-based processing of JSNet and 3D-BoNet tends to predict errors in the form of a few misclassified events across object instance boundaries. Without further postprocessing of the segmentation results, these misclassified events, even if only a few, must be weighted more heavily as errors for the bird's eye view generation than pixel inaccuracies obtained in the dense pixel masks generated by frame-based instance segmentation. This means that for the desired public space monitoring application, a higher $\mathrm{mAP}^{0.5}$ is more useful than a higher $\mathrm{mAP}^{0.95}_{0.5}$.

Especially for DVS2, which covered the most activity, only small differences in object counting based on 3D and 2D processing were observed. Therefore, considering that the implementation of standard 2D convolutions are better optimized compared to custom-implemented 3D point-based processing, from an application-oriented point of view, classical 2D frame-based processing is preferable for a commercial adaptation of the developed system.

In the course of this work, it became clear that the DVS is capable and predestined to clearly capture components triggered by environmental influences when the sensor is used in outdoor scenarios. What in the context of this dissertation is considered an "unwanted" influence to be filtered, has already led to immediate follow-up work with the goal of DVS-based insect monitoring [Pohle-Fröhlich and **Bolten**, 2023; Pohle-Fröhlich et al., 2024].

The movement of insects contains a significantly higher proportion of fast motion components compared to the objects studied on the playground. From an academic point of view, it seems promising to further investigate 3D space-time event cloud-based processing on such data, as the results obtained in this work have shown its general applicability and comparability with results from 2D frame-based processing. It can be expected that, for faster motions, the advantage of 3D methods will become more apparent.

12.3 Limitations

The use of Dynamic Vision Sensors in real-world outdoor applications is still very rare. This work can serve as a valuable reference for other researchers who intend to develop similar systems.

12.3.1 Real-World Application Context

The CeleX-IV sensor used in the developed system has a high level of background activity. This prevailing noise level must be taken into account during the system design process to ensure that the system's performance is not unduly constrained.

A strong temperature dependency of the noise level was found for the sensor used. Through a combination of spatio-temporal filtering and active temperature stabilization of the sensor, this noise level could be significantly reduced, which is very advantageous for subsequent processing. However, the electrical energy required for this active temperature stabilization

counteracts the technical advantage of a low electrical power consumption of a Dynamic Vision Sensor.

For the development of future systems, it is recommended that the sensors used and their bias settings be tested in advance at different temperatures. Promising next-generation Dynamic Vision Sensor models with improved sensor noise characteristics are already available for research and development.

The evaluation of the quality of the quantitative derivation of the user volume from the recorded real-world observation data revealed two main issues with the system used in the practical application context.

In a long-term monitoring conducted over years, especially in a context like the monitored nature-experience-oriented playground, the given infrastructure may change over time. In our case, the local vegetation changed significantly over time, which should have been taken into account in updated datasets and retrained models in order to better incorporate these changes into the processing. Therefore, we recommend a critical review and regular inspection of the measurement site to detect these changes in already considered object classes or the introduction of entirely new object classes. Furthermore, an extension of the processing to a panoptic segmentation should be considered by including the environmental influences in the datasets used for instance segmentation. In this way, the multi-stage processing of filtering environmental influences followed by instance segmentation could be replaced by a single segmentation step.

The CeleX-IV Dynamic Vision Sensors provide a usable spatial resolution of 768×512 pixels. The number of sensors used to cover the playground area was chosen as a compromise between cost, coverage, and the a priori expected distribution of users on the measurement area. The sensors were positioned to achieve an almost complete coverage of the playground area, while expecting a considerable user volume across the entire playground area. However, the actual usage behavior was heavily skewed towards the coverage area of a single sensor. This resulted in a lot of activity occurring only in the peripheral coverage areas of the other sensors, where processing is very challenging due to the very small size of the projected objects resulting from the large distances involved. In retrospect, and therefore as a recommendation, more attention should have been paid to ensuring that the detection areas per sensor cover a similarly large area, which should have been selected in a better ratio to the available pixel resolution.

12.3.2 Segmentation

The applied fixed-size partitioning of the provided sensor data into Patches-of-Interest proved to be error-prone for objects that are truncated at the boundaries of the formed patches.

However, due to the logic applied in the creation of the DVS-OUTLAB dataset, and with a processing based on such patches in mind, the provided labels were also exported in a spatially patched manner. As a result, the use of this dataset in an unpatched form is a challenge. Based on the trained segmentation networks, a complete processing of the given database would be possible. After a manual check and possible corrections of the resulting automatic segmentations, this shortcoming of the dataset can be overcome and complete, unpatched data would be available.

Increased spatial pixel resolutions of modern and upcoming Dynamic Vision Sensor models will further increase the challenges related to event subsampling required for 3D space-time event cloud-based processing.

Adaptive selection of regions for further processing can counteract the increasing total number of events due to higher spatial resolutions. However, very crowded scenes have proven

to be a challenge in this context, as the proposed decomposition fails in such scenes due to the spatial proximity of the given events. In general, a balance between the spatial resolution required for the processing objectives and the resulting number of triggered events has to be found (see also [Gehrig and Scaramuzza, 2022]).

12.4 Further Work

The evaluated long-term monitoring scenes contain only a few instances of non-`PERSON` object classes. Therefore, the performed evaluation was limited to this class.

Especially for the aggregated representation in the form of projected bird's eye heat maps, an open aspect for further work is the evaluation of methods for the consolidated visualization of activities caused by a set of multiple classes. The goal would be to allow further differentiation of the underlying activity levels per class. This could be achieved by the interactive use of different visualization layers or by the integration of class icons into the activity visualization.

The 3D space-time event clouds constructed for processing in this work, with $e_i = \{x_i, y_i, t_i\}$, are based solely on the spatio-temporal components of the event stream. Consequently, one aspect for further research may be to consider incorporating additional features, such as event polarity p_i, into this representation. Furthermore, the incorporation of fully end-to-end learned event representations is another promising area of research.

Currently, each formed input of the event stream is processed independently, as the application requirements do not strongly depend on strict continuous predictions for each of the processed 60 ms time windows. In general, processing is likely to benefit from continuity improvements when recurrent components are integrated. This is also likely to be the case for 3D-based processing with approaches such as [Min et al., 2020]. The processing of point cloud sequences by transformer architectures, e.g. [Wei et al., 2022], is another interesting approach to achieve improvements.

12.4.1 Moving Towards Object Tracking

Object tracking is an interesting extension of processing in real-world long-term monitoring setups. The high temporal resolution of Dynamic Vision Sensors results in a nearly continuous signal for moving objects in the 3D space-time output, while ideally not including static scene components.

Preliminary work has been done in [**Bolten** et al., 2019] that implements an object tracking based on event clustering. However, this approach has limitations with respect to the discrimination of different object classes and the spatial proximity of instances.

Improvements to this approach can be made based on the work performed on instance segmentation. As shown using the N-MuPeTS dataset (see Figure 5.1 on page 56), the continuity of events in the event stream almost automatically leads to object tracking if the instance segmentation is of appropriate quality.

In this case, the intersections in the resulting object trajectories pose the greatest challenge. An approach based on derived motion and velocity parameters for tracking instances across these intersections is an interesting point for further work. The N-MuPeTS dataset provides suitable scenes with corresponding motion patterns and associated annotations to directly support and enable these investigations.

Part VI

Appendices

Contents

List of Figures

List of Tables

Appendix A

Thermal Stabilization

As described in Section 3.3.1, the background noise behavior of the CeleX-IV Dynamic Vision Sensor is highly temperature dependent. This behavior is also highlighted in Figure 3.8 on page 29.

Therefore, the sensor system features an active temperature stabilization. A Peltier element, relays, and multiple temperature sensors are used to create a closed-loop controller. The central processing unit is an Arduino-based microcontroller board. The system can log temperatures, humidity, voltages, and currents over the complete duration of operation.

In the following, the design intentions and resulting hardware setup of the thermal concept are described. The control loop configuration and results from the real-world outdoor operation are presented afterwards.

A.1 Thermo-Mechanical Setup

The CeleX-IV Dynamic Vision Sensor comes with an OpalKelly XEM6310 FPGA development board[22]. This board features a Xilinx Spartan 6 FPGA and an USB 3.0 interface.

A.1.1 Heat Sources

The imaging sensor is heat-sensitive as noise increases drastically with temperature. Every component behind that operates digitally and cooling is not critical (within the operating limits).

One source of heat is of course the environment. The ambient temperature can be high during summer days. It would be possible to enclose the complete system airtight, insulate it, and cool the interiors, but this approach would entail considerable effort, increased weight, and cost. Here, we opted for an open, "breathing" system with forced air exchange.

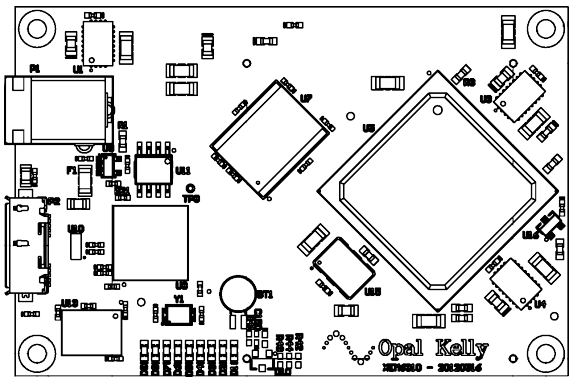
Besides external sources, multiple electrical components on the FPGA board can be identified as heat sources[23] (i.e. "self-heating"):

- Spartan-6 FPGA
- SDRAM
- USB host interface
- multiple switching regulators

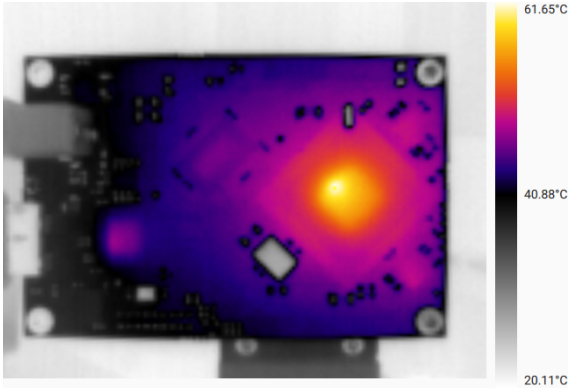
The dominating heat source is the FPGA itself. This was validated using a Teledyne FLIR Lepton 3.5 longwave infrared (LWIR) camera as shown in Figure A.1. The FPGA is located

[22] `https://opalkelly.com/products/xem6310/`

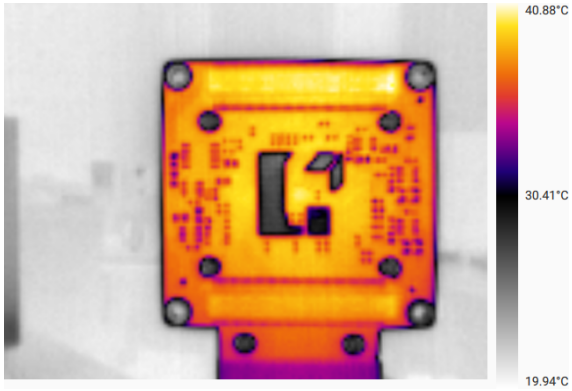
[23] `https://docs.opalkelly.com/xem6310/powering-the-xem6310/`



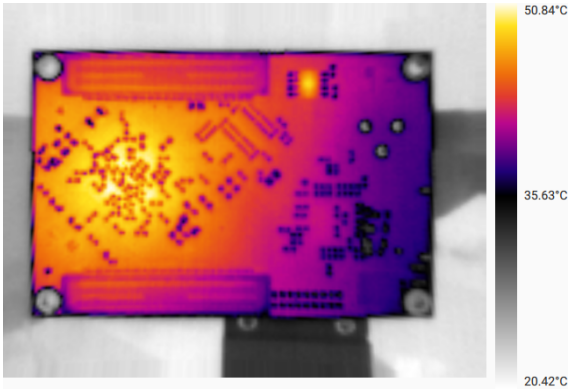
(a) Mechanical drawing of the XEM6310[24]



(b) LWIR-image of the FPGA's backside



(c) LWIR-image of the CeleX-IV sensor board's backside



(d) bottom right: LWIR-image of the FPGA's front side

Figure A.1: FPGA heat sources. All LWIR-images were taken in rapid succession at room temperature after one hour of operation.

behind the perimeter of the sensor board. This means the heat-sensitive image sensor is located in close proximity, albeit with an insulating air gap, of the system's major heat source.

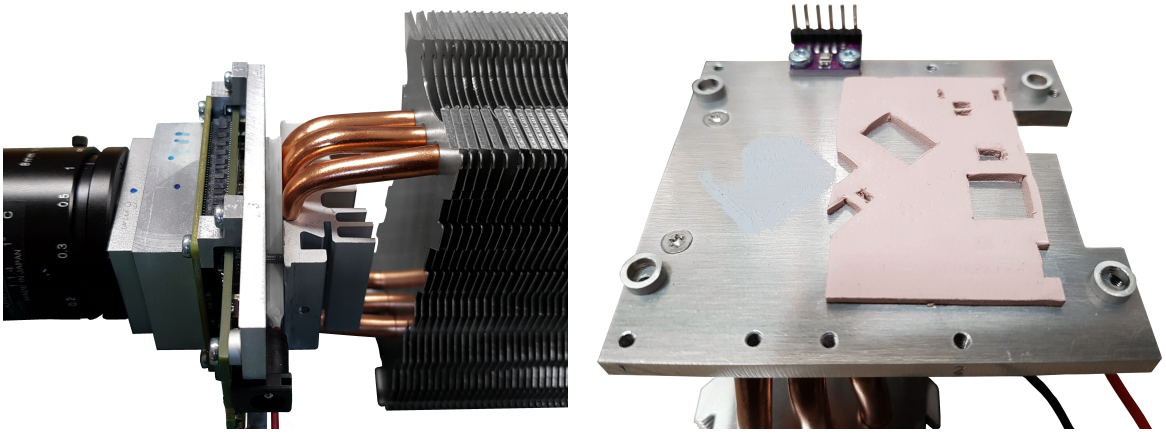
The lens is mounted via a CS-mount to C-mount adapter made out of EN AW-5083 aluminum. The walls, which the adapter sits on, are only 2 mm in thickness so that the area of contact with the sensor board is minimized while ensuring no light leakage. The sensor board is made from FR-4 or a similar material. FR-4 has a low thermal conductivity [Mohan, 2019, p. 7] and the mounting holes are electrically isolated, i.e. no contact to the copper plane is possible. Therefore, despite its close proximity, the lens itself is mostly thermally decoupled from the rest of the system and is not considered further. Later experiments (see Figure A.3) verify that the lens is only slightly cooled under ambient temperature hinting at a low thermal conductivity connection to the Peltier element. Heating from direct sunlight is no concern due to protection from the enclosure itself.

A.1.2 Heat Pump Stack

It is not easy to extract the generated heat directly from the sensor PCB because the distance to the FPGA board, dictated by the PCB connectors, is only a few millimeters. Recall that the greater heat source is the FPGA board itself. Extracting heat from the backside of the FPGA board is therefore preferred.

The chosen interface is a plate, milled from EN AW-7075 aluminum, custom fit to the

[24] adapted from `https://docs.opalkelly.com/xem6310/specifications/`



(a) Stack without a fan and thermal pads (b) Masked thermal pad

Figure A.2: Heat pump stack.

backside of the FPGA board – in the following referred to as backplate. On this metal plate, both, the FPGA board and the sensor board, are mechanically fixed.

A 40×40 mm Peltier element is mounted on the other side of the backplate. The Peltier element is driven using a H-bridge made up of two single-pole double throw relays. This enables reversal of the supply voltage which leads to reversal of the heat current and thus cooling as well as heating.

The other side of the Peltier element features an oversized heat sink with a fan. It is available for consumer market CPUs with up to 120 W TDP and offers a cheap, fitting solution to heat dissipation.

The FGPA's chip is the most protruding component and thus can be thermally connected to the backplate by heat-conducting paste alone (see Figure A.2, right image, on the left half the imprint can be seen). The air gaps between the PCBs and the backplate are filled with thermally conductive pads. Between sensor and FPGA board two layers were stacked. When necessary, cut-outs were made for protruding electrical elements like resistors and integrated circuits (see Figure A.2).

A.1.3 Forced Air Cooling

Camera lens and cooled sensor are located at the front of the case, switching converters and heat sink are located towards the rear. Two 60 mm fans create an airstream from the front to the back. The required holes are cut out from the base of the enclosure. A coarse grid prevents insects or larger particles from being sucked into the enclosure.

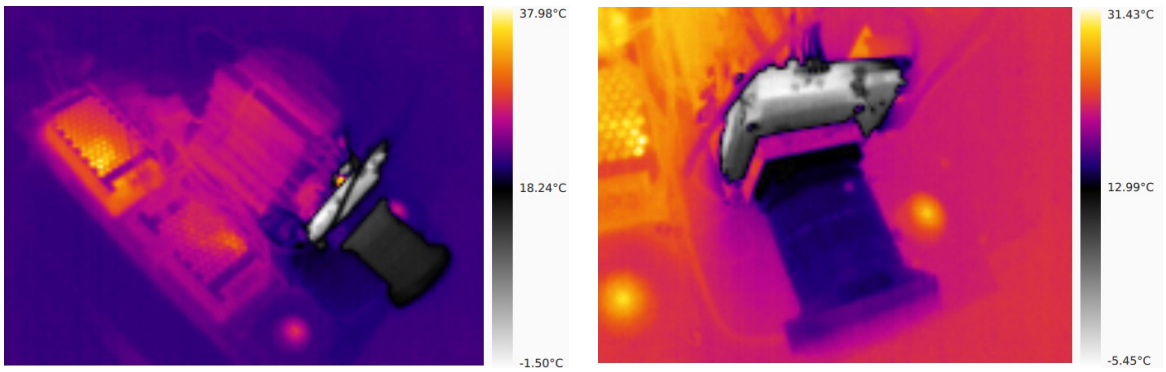
At the back of the enclosure a slit across the breadth of the case is left open. The cross-section area is matched to that of the inlets in order to not create a constriction.

This creates a temperature gradient from the front to the back and a constant exchange of air that prevents built-up of moisture.

A.2 Control Engineering

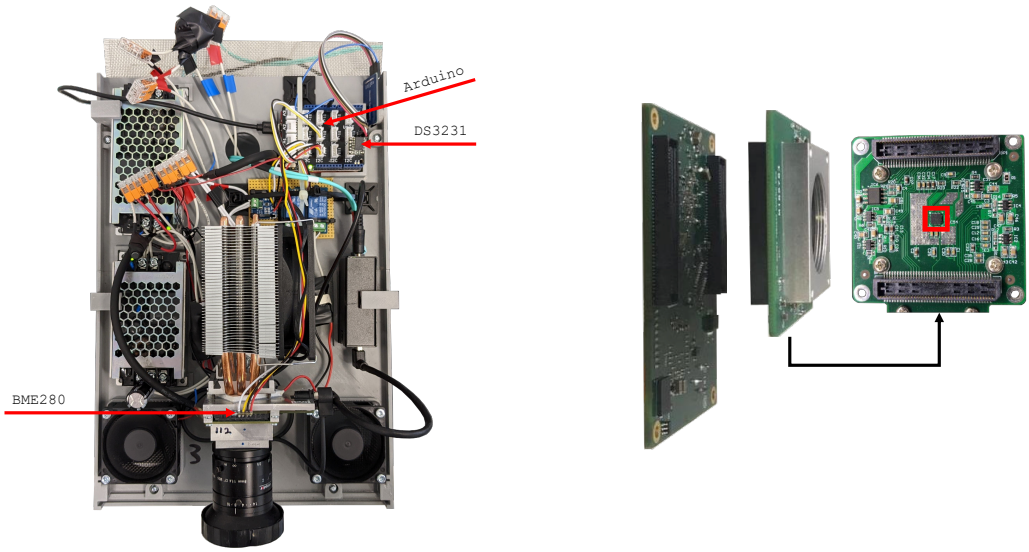
A.2.1 Instrumentation

A BME280 temperature sensor is mounted on the backplate near the "cold end" of the stack and close the FPGA chip. Using this temperature sensor a closed-loop control loop is formed. The temperature is read out via I^2C by an Arduino microcontroller.



(a) LWIR-image of the mast head assembly (b) Close-up LWIR-image of the stack

Figure A.3: LWIR-images of the system in operation.



(a) Sensor enclosure (b) CeleX-IV IMU position

Figure A.4: Temperature sensor positions.

Current flowing through the Peltier element is measured using a INA219 current sensor. The current sensor was modified to a 22 mΩ shunt resistor to increase the maximum measurable current.

For monitoring of the ambient temperature, the temperature sensor within the DS3231 real-time clock is used. This module is located at the rear and opposite of the switching regulators. The readings should therefore give an estimate of the temperature inside the enclosure, independent of influence by other components inside.

The internal temperature sensor of the Arduino microcontroller was used for verification of tendencies. As the microcontroller as a real-time system has a constant computing load, the heating up should be constant as well. All temperature sensors were calibrated across all systems to a single point, i.e. offsets relative to one of the DS3231 modules.

See Figure A.4 for an overview of the temperature sensors installed inside each enclosure.

A.2.2 PID-Controller

Two-point, three-point, and PID controllers were considered.

A two-point controller where the Peltier element is used for cooling is a simple and sufficient solution. A necessary improvement is the use of an H-bridge in order to use the Peltier element

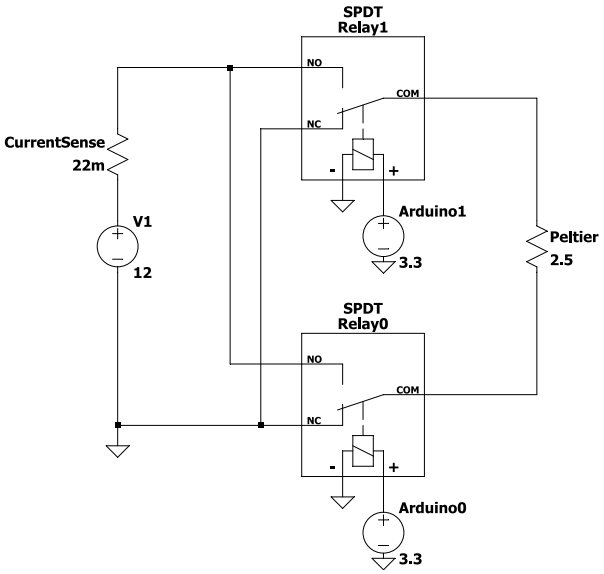


Figure A.5: Circuit diagram of a Peltier element connected to a H-bridge made up of two relays for cooling and heating. `Arduino0` and `Arduino1` are the PWM outputs of the controller board. `CurrentSense` is the shunt resistor of the INA219 current monitoring module.

for heating, too. Our circuit is shown in Figure A.5. This requires a three-point controller which is a trivial extension of the prior.

Heat conduction is an inert process but the process of cooling vs. heating is highly asymmetrical, i.e. cooling takes longer than heating. In the final setup at $T_{\text{amb}} = 22°\text{C}$ cooling from 20°C to 15°C takes three times longer than heating. During development, different setpoints were required for additional experiments and control loop performance degrades with increased setpoint temperature. Thus, we opted for a PID controller for more flexibility and better performance.

The PID control algorithm is based on the parallel form (see [Lipták, 2006, pp. 126–127]):

$$m = K_p\, e + K_i \int e\, dt + K_d \frac{de}{dt} \tag{A.1}$$

where m is the controller output, K_p, K_i, and K_d are the control parameters, t is the time, and

$$e = c - r \tag{A.2}$$

where e is the error, c is the control variable, i.e. the measured temperature, and r is the setpoint, i.e. the temperature target.

The PID controller is implemented in 8-bit integer arithmetic. The temperature reading from the BME280 on the backplate is our process variable. The setpoint was persisted in the EEPROM for outdoor operation else it can be set via USB. The final control value is a 8-bit unsigned integer that is used for pulse-width modulation (PWM) of the two relay control pins. It is derived from the calculated control value m using

$$m_{\text{8-bit}} = \begin{cases} 0 & \text{if } m < -128, \\ 255 & \text{if } m > 127, \\ \lfloor m + 128 \rfloor & \text{else.} \end{cases} \tag{A.3}$$

0 means 100 % heating power, 128 means off, and 255 means 100 % cooling power. The PWM period was set to 10 s. A dead-time of 157 ms was enforced because of the inertia of the relays.

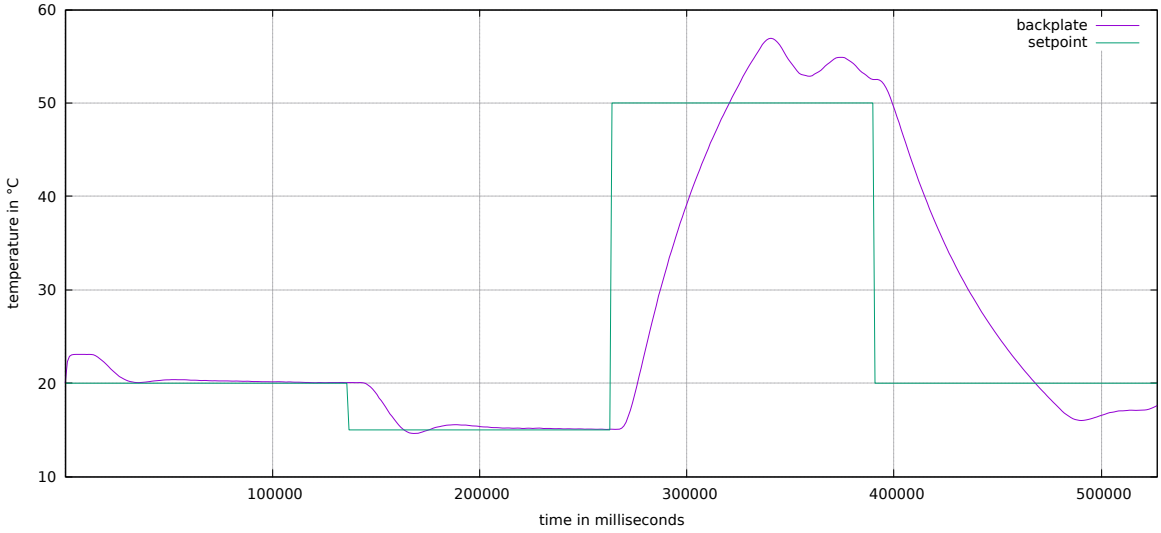


Figure A.6: Measured temperature during a sequence of different setpoints (20°C, 15°C, 50°C, and 20°C for two minutes each).

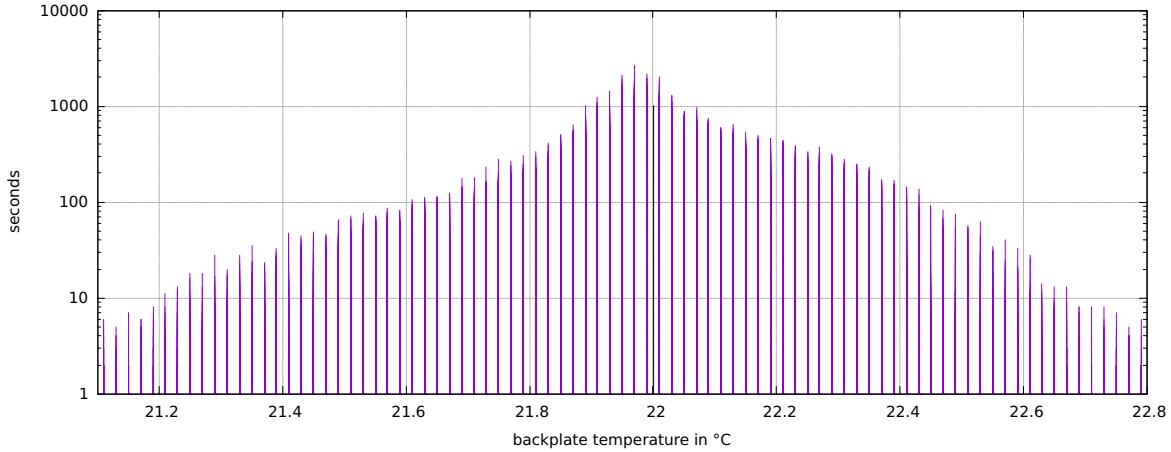


Figure A.7: Histogram of backplate temperatures within measurement period. Bins with less than 10 seconds of data are considered invalid and excluded.

Control values less or equal to 2 and greater or equal to 253 lead to the relais being switched on constantly. Tuning of the control parameters K_p, K_i, and K_d was done manually. We used rectangular control signals and a sequence of different setpoint targets (see Figure A.6) to evaluate the system's behavior. The final values are $K_p = 25$, $K_i = 0.25$, and $K_d = 0.0$. In practice, the controller therefore simplifies to an PI controller.

A.3 Retrospection

The outdoor measurement period covers 140 days of active data recording. As can be seen from Figure A.7, the temperature was well controlled within a 1°K interval. The mean recorded temperature is 21.999°C with a standard deviation of 0.256°K.

Finally, we want to discuss the systems behavior and performance using an example from our real-world measurements. Figure A.8 shows the first five minutes after powering the system on at the beginning of a day. The x-axes of all plots are synchronized. Every tic on the x-axes

Date	Precipitation height	Sunshine duration	Minimum temperature	Mean of temperature	Maximum temperature
June 17th, 2021	0 mm	13.717h	21.8°C	28.1°C	34.5°C
June 18th, 2021	0 mm	12.183h	21.1°C	27.5°C	34.2°C
June 19th, 2021	18 mm	12.933h	18.5°C	24.1°C	29.4°C

Table A.1: Excerpt of meteorological observations made in Düsseldorf on June 18th, 2021. Taken from "Historical daily station observations (temperature, pressure, precipitation, sunshine duration, etc.) for Germany, Version v23.3"[25].

Local time	Minimal temperature	Maximal temperature
09:00	26.4°C	26.9°C
09:10	26.9°C	28.0°C
09:20	28.0°C	28.9°C
09:30	28.1°C	28.8°C
09:40	28.7°C	29.1°C
09:50	28.8°C	29.7°C
10:00	28.8°C	29.7°C
10:10	28.8°C	29.4°C
10:20	29.2°C	29.6°C
10:30	29.6°C	30.1°C

Table A.2: Temperature extrema within 10-minute intervals in Düsseldorf on June 18th, 2021. Taken from "10-minute station observations of extreme temperatures for Germany, Version v23.3"[26].

represents 10 seconds, i.e. one control period in which the controller output is updated. For reference, the setpoint temperature is marked with a green line in the first plot.

June 18th, 2021, was the last in a series of hot days with tropical nights. Within the following days, severe weather conditions occurred and it cooled down. For reference, Table A.1 shows the daily reports from June 17th until and inclusive June 19th. Table A.2 lists the temperature extremes from 60 minutes before to 30 minutes after the start of operation. All observations were made in Düsseldorf which is 10 km off of the playground in Mönchengladbach.

The system was powered on at 10:00 local time. The initial temperature was measured to be 23.044°C. The temperature is 1.044°K above the setpoint. With $e = 1.044$, we get an controller output of

$$m^{(0)} = 25 \cdot 1.044 + 0.25 \cdot 1.044 \cdot 10 = 28.71. \tag{A.4}$$

This value is in a 7-bit range. In order to convert it to a duty cycle, we have to divide by $2^7 = 127$ and get 22.6 %. The Peltier element started cooling with a duty cycle of 22.6 %.

In the second period ($t = 10$ seconds), the temperature rose to 25.104°C despite our cooling effort. From past meteorological observations, it can be seen that the air temperature was between 28.8°C and 29.7°C. On powering on, the two enclosure fans were turned on and started sucking in warm air. The integral term had then accumulated the previous error multiplied by dt. The temperature was then 3.104°K too high. Updating the controller gives

$$m^{(1)} = 25 \cdot 3.104 + 0.25 \cdot (1.044 \cdot 10 + 3.104 \cdot 10) = 87.97. \tag{A.5}$$

[25] see `https://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/daily/kl/historical/`

[26] see `https://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/10_minutes/extreme_temperature/historical/`

This cycle, the duty cycle was increased to $\frac{87.97}{127} = 69.2\,\%$ resulting in a peak in power consumption of 22.4 W.

In the fourth period ($t = 30$ seconds), a small undershoot to 21.665°C was measured. The controller reduced the duty cycle to 30.8 %. The temperature overshoots slightly to 22.677°C within the next 20 seconds.

At 100 seconds after start, the system has reached a stationary state at $\approx 20 \pm 0.5\,\%$ duty cycle, drawing ≈ 5.2 – 6.7 W of electrical power. From this point on, the temperature differences are negligible for the remaining duration of operation.

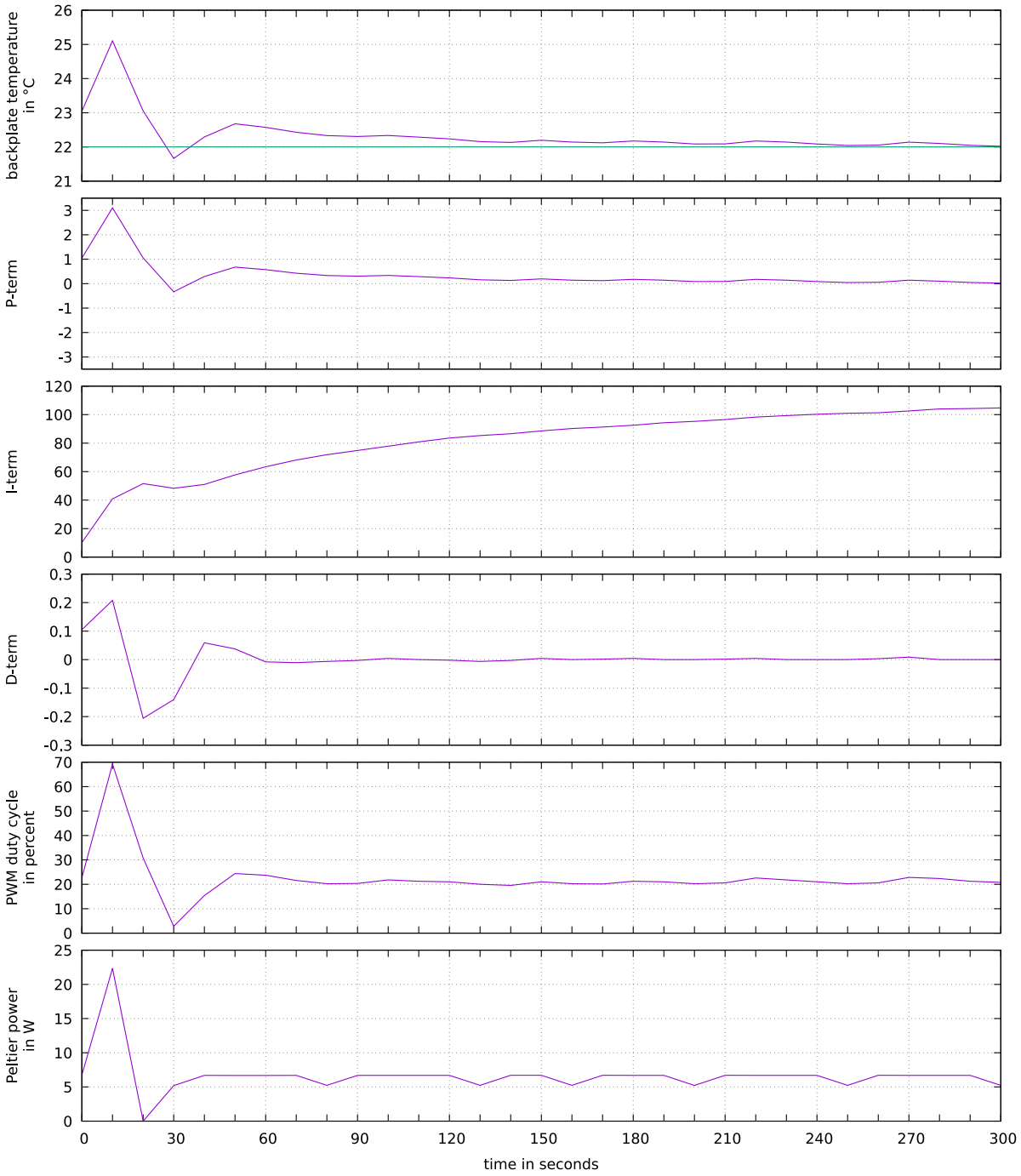


Figure A.8: Plot of logged controller metrics during the first five minutes of June 18$^{\text{th}}$, 2021, after power on. On this tropical summer day, the ambient temperature was 24.15°C. P-, I-, and D-term refer to e, $\int e\,dt$, and $\frac{de}{dt}$ respectively, i.e. before scaling by K_p, K_i, K_d.

Appendix B

Hardware Setup

At the start of the development work, different sensor modalities were considered for the realization of the measurement system to be set up. In particular, in addition to CCTV and Dynamic Vision Sensor-based systems, the following measurement techniques were considered:

Radar: Radio Detection And Ranging

A radar operates by transmitting radio waves, usually in the super high or extremely high frequency range (i.e. 3–30 GHz or 30–300 GHz), towards a target object. The waves reflect off the target and return to the radar system, where they are detected. The radar system can determine various parameters by analyzing the time it takes for the waves to return and their Doppler shift (change in frequency caused by motion). A radar system can typically provide at least the distance, velocity, and azimuth angle of detected targets.

The radar cross-section signatures are affected by the size, shape, and material characteristics of the objects being measured. By analyzing radar micro-Doppler effects, which refer to the small frequency shifts in the radar return signal caused by the micro-motion of target components, it is also possible to distinguish between different classes of targets.

Pro:

- Weather independence: Radar operates effectively in all weather conditions, including rain, fog, and darkness.
- Long range: Radar is able to detect objects over long distances, depending on factors such as the radar's power, frequency, and antenna design.

Con:

- Complexity and cost: Radar systems can be complex and expensive to design, deploy, and maintain.
- Object classification: Radar systems may currently have limitations in distinguishing between different types of objects, as this is an active and ongoing research topic.

Lidar: Light Detection And Ranging

A lidar measurement relies on emitting pulses of laser light in the near-infrared spectrum, typically between 800 and 1550 nanometers, towards a target object. The lidar sensor detects the pulses that reflect off the target. By timing the duration it takes for the light pulses to return and analyzing their properties, such as intensity, lidar systems can accurately calculate distances.

Lidar can produce detailed 3D maps of the environment and objects within it by scanning an area with a sweeping laser beam and recording the return signals.

Pro:

- High precision: Provides highly accurate 3D representations of the environment and the objects within it.
- Object classification: Object classification based on shape or size is possible.

Con:

- Cost: High resolution systems are expensive.
- Time resolution: Low frame rates are typically achieved depending on the number of scan lines configured.
- Range: Limited range compared to radar.

Infrared Thermography: Thermal Imaging

Infrared thermography is a method of detecting infrared radiation emitted by objects. This is based on the principle that all objects with a temperature above absolute zero emit infrared radiation, and the amount of radiation emitted is directly related to the object's temperature. A long-wave infrared camera can capture infrared radiation with wavelengths typically ranging from 8 to 15 micrometers and convert it into a visual representation.

Infrared thermography enables the identification of temperature variations. In the context of a monitoring application, this means, for example, that a person typically appears as a silhouette in the generated image, depending on the person's body temperature and the surrounding environment.

Pro:

- Independent of visible light: Works in low light and at night.

Con:

- Cost: High resolution systems are expensive.
- Environmental factors: Ambient temperature, humidity and air movement can affect the accuracy and reliability of measurements.
- Sensor resolution: Limited spatial resolution compared to RGB cameras.

As already described, the use of classic RGB-based CCTV camera systems is limited by regulatory and legal restrictions. Radar- and lidar-based systems were excluded from the implemented monitoring system, mainly due to the currently very high acquisition costs. For IR cameras, in addition to the low spatial resolution of affordable models, limitations in analysis must be expected, especially in midsummer, due to small temperature differences between background and objects.

In addition to the aforementioned privacy benefits, the technical characteristics of the Dynamic Vision Sensor make it a promising solution for the development of a monitoring system (see Chapter 1). For this reason, the DVS was selected as the sensor technology used in this work.

B.1 Living-Lab: System Components

A more detailed overview of the measurement setup components used in the Living-Lab is given in this section, extending the descriptions given in Section 3.1.

Basically, the system components can be semantically divided into two groups. On the one hand, there are the components for the power supply and the central data acquisition and processing. On the other hand, there are the components at the three sensor locations on the measurement area.

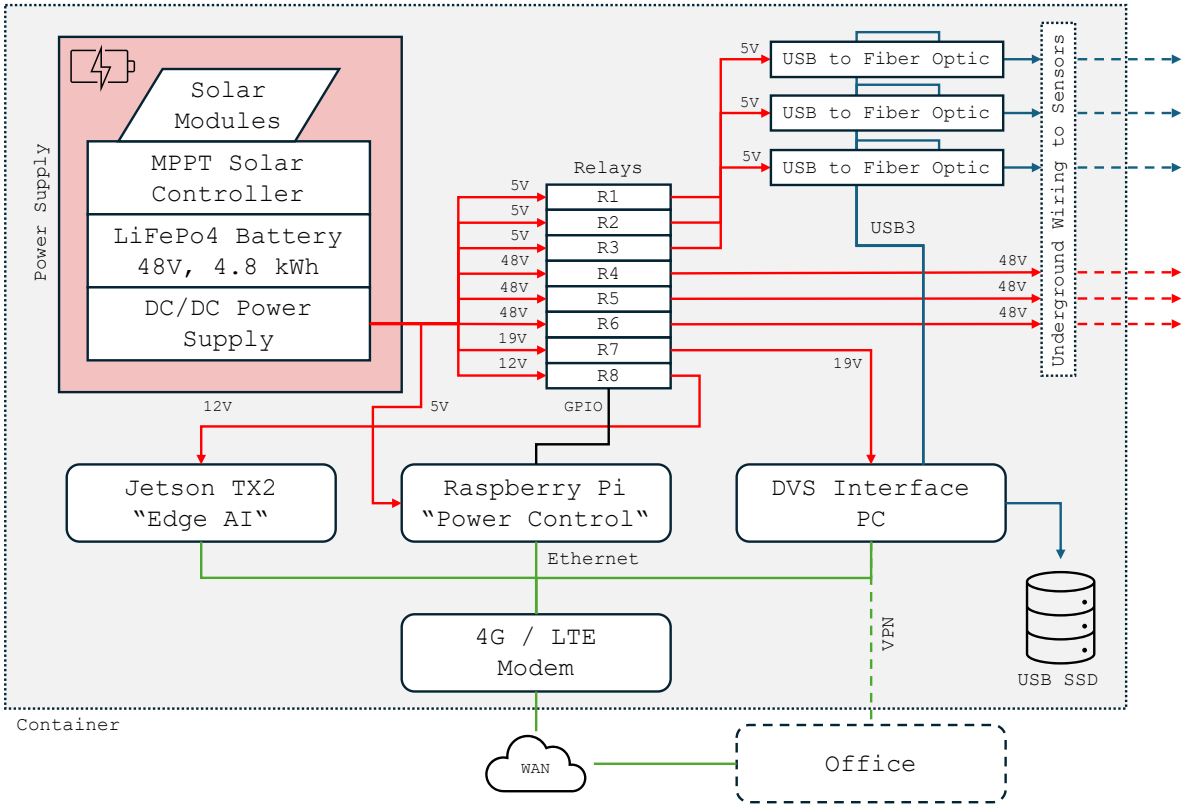


Figure B.1: Living-Lab: System components at the central point of data acquisition and processing.

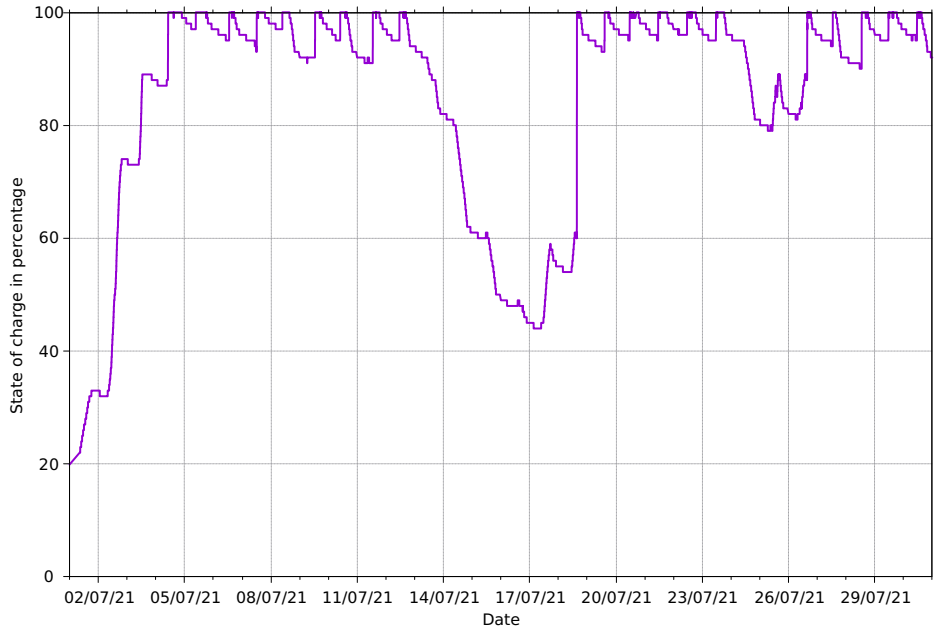


Figure B.2: State of charge of the used battery system during one month of monitoring.

B.1.1 Central Components

The power supply and data processing components were housed in an approximately $2 \times 2 \times 2$ meter quick-build container[27] located about 16.5 meters northwest of the first DVS sensor position. The container served primarily as protection from vandalism and weather, but also provided a temporary workspace for on-site development work.

Figure B.1 provides an overview of the main system components contained in this container.

Power Supply

As the Living-Lab is located in an urban forest area, a self-sufficient energy supply was realized. A system consisting of photovoltaic modules and an energy storage battery was designed and implemented.

Two solar modules, each measuring 1×1.6 meters and providing a maximum output of 275 Wp, were installed on the roof of the container. These solar modules charged a battery using a commercially available MPPT solar charge controller[28]. To reduce the risk of vandalism or theft of these modules, a fence was installed on the roof of the container to prevent climbing onto it (see Figure 3.1b on page 22).

The battery consisted of two PylonTech "US2000 Plus" modules[29]. These modules have a fully integrated battery management system. The lithium iron phosphate (LiFePO4) based battery modules each provide 2.4 kWh of energy resulting in a total of 4.8 kWh. The system operates in the extra-low voltage range. The typical discharge voltage is 48 VDC, with the range of 45 to 54 VDC. This ensures that the system carries a low risk of dangerous electric shock, making it a safe option for use in proximity of children.

The sensor masts were powered directly from the available 48 VDC battery output, while DC/DC conversion to 5, 12 or 19 VDC was used to power the other system components inside the container.

In order to save power during off-hours and to be able to reset the system components in the case of problems, the power supply to the most important system components was designed to be switchable as part of an implemented power management system. For this purpose, a relay card was developed and integrated into the system, which, controlled by a connected Raspberry Pi, allowed the individual components to be switched on and off via software.

Figure B.2 shows the state of charge of the battery system for one month during an active measurement period. It can be seen that the system was able to operate autonomously and compensate for fluctuations in solar yield as well as larger energy withdrawals on-site.

Underground Cabling

The EFRE-funded research project included a redesign of the measurement area. As part of this redesign and construction work, an underground cabling system was installed to connect the components.

This underground cabling system connects the container with the sensor locations on the playground in a star shape. Each sensor location was equipped with its own power and data connection.

[27] https://www.container-bestofsteel.de/schnellbaucontainer/schnellbaucontainer/schnellbaucontainer

[28] https://www.victronenergy.de/solar-charge-controllers/bluesolar-mppt-150-35

[29] https://en.pylontech.com.cn/download.aspx?id=199

The electrical connection of the sensor masts is made at 48 VDC, which is the output voltage of the battery. This is a good compromise between safety and the required wire gauge, which must take into account the voltage drop caused by the length of the wires.

The CeleX-IV sensor system provides only an USB 3.0 interface. The distances to be bridged are too large for a direct USB connection, e.g. the distance between the container and one of the sensor locations is about 100 meters. For this reason, a fiber optic connection was used for data communication. The USB connection was converted from copper to fiber in the container by a converter[30] and from fiber to copper in the sensor node by a corresponding counterpart. For the connected components, i.e. the USB host PC and the sensor, this conversion is transparent and does not need to be taken into account in any software.

Computing

As mentioned above, a Raspberry Pi was used as a central command and control component, switching the power supplies via relays and logging the battery state of charge. No other functions were performed by the Raspberry Pi.

Two NVIDIA Jetson TX2 modules were used on-site to accelerate the computations involved in selecting the scenes to be saved (see Chapter 9). These modules are energy-efficient edge AI computing devices that use NVIDIA graphics processors to accelerate computations. In the application scenario, which essentially consisted of PointNet++ inferences, the developer modules used were measured to consume less than 20 W each.

The main processing component, in terms of DVS acquisition and preprocessing, was performed using a Dell Optiplex 7060 Micro PC. This PC, equipped with an Intel Core i7-8700 CPU, acted as the interface PC for all three CeleX-IV sensors and is rated with a maximum power consumption of 130 W. The measurements were stored on a 2 TB USB SSD connected to this interface PC. This allowed the data storage volume to be changed quickly and easily during the measurement periods.

The computing components were connected via Gigabit Ethernet. In addition, the use of a 4G mobile router enabled a VPN-based remote connection to the components in the Living-Lab. This connection was not used to transmit the recorded monitoring data, but to supervise the system itself.

B.1.2 Sensor Nodes

Figure B.3 provides an overview of the main components at each of the sensor locations.

The incoming 48 VDC supply voltage was converted to 5 VDC or 12 VDC to power the individual components. The sensor and communication components were supplied with 5 VDC, while the 12 VDC was used for active ventilation by fans and the thermal stabilization described above using a Peltier element.

The power management implemented in the container allowed to switch on or off the entire power supply for each sensor position. However, practical tests have shown that in combination with the used fiber optic USB converters and the CeleX-IV FPGA carrier board, which provides the USB connection, the power-on sequence of these components is important. For this reason, a power supply delay for the fiber optic converter was implemented in hardware. This ensured the correct sequence, sensor before converter, when switching on the sensor mast.

[30] `https://www.lindy.de/200m-Fibre-Optic-USB-3-0-Extender.htm?websale8=ld0101.ld011101&pi=42707`

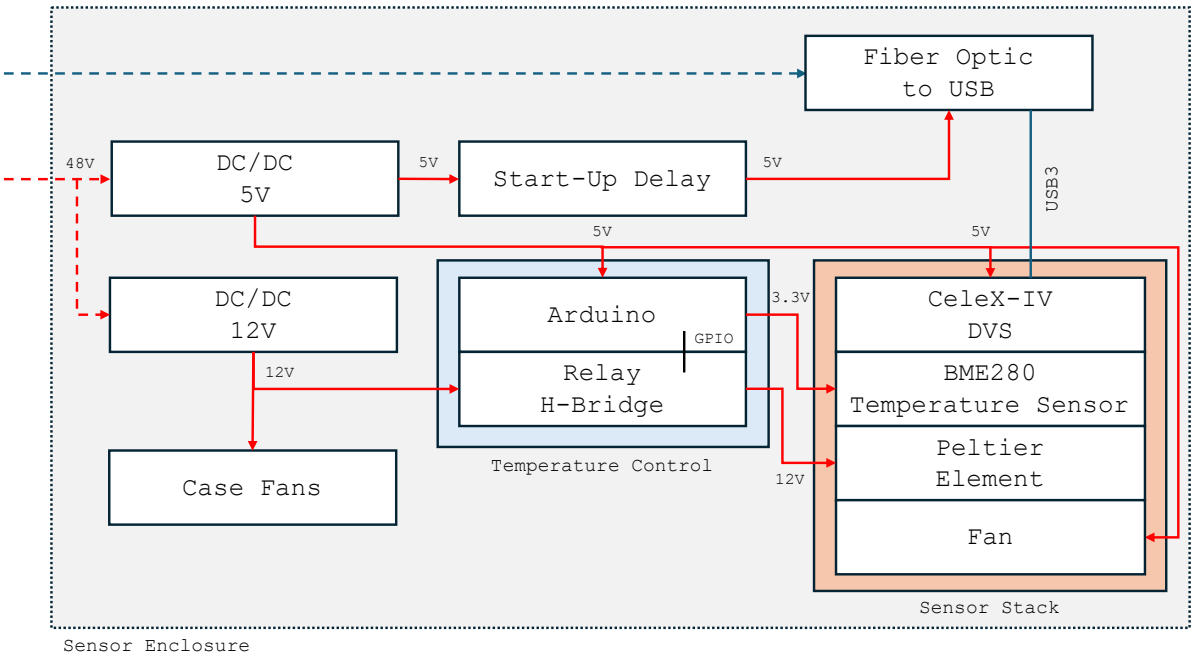


Figure B.3: Living-Lab: System components inside the sensor enclosure.

B.2 Sensor Enclosure

A custom enclosure was designed and 3D printed to protect the sensor components from the elements. The design and fabrication of this enclosure was a collaborative effort with colleagues in the Faculty of Design as part of the EFRE-funded research project.

The sensors were mounted on the top of street lamp posts, which were erected on the site for our purposes, at a height of approximately 6 meters (see Figure B.4a). In addition to simply housing and protecting the sensor components, there were other requirements for this enclosure.

The sensor components had to be mounted securely and robustly. On the one hand to protect people in the measurement field from falling parts and on the other hand to support the processing. The field of view of the sensors should remain constant during a measurement period due to the calibration performed for mapping (see Section 11.1.2).

Reflections in the sensor's field of view had to be avoided. Therefore, a light barrier was installed directly behind the transparent protective glass of the enclosure.

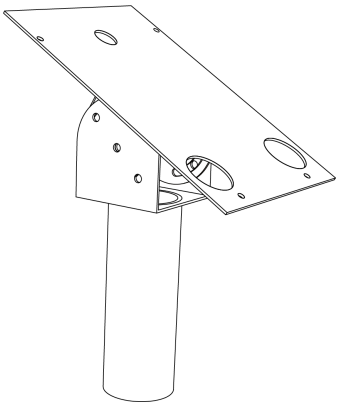
For easy installation and fine adjustment in the field, the pitch axis of the sensor setup had to be easily adjustable. For this reason, a pitch-adjustable mounting head (see Figure B.4b) was developed to hold the actual sensor enclosure.

A 3D-printed mounting bed was mounted on top of this head, to which the individual components discussed in the previous section were attached (see Figure B.4c). The 3D printing was mainly carried out with ASA material in order to achieve high UV and temperature stability of the manufactured components.

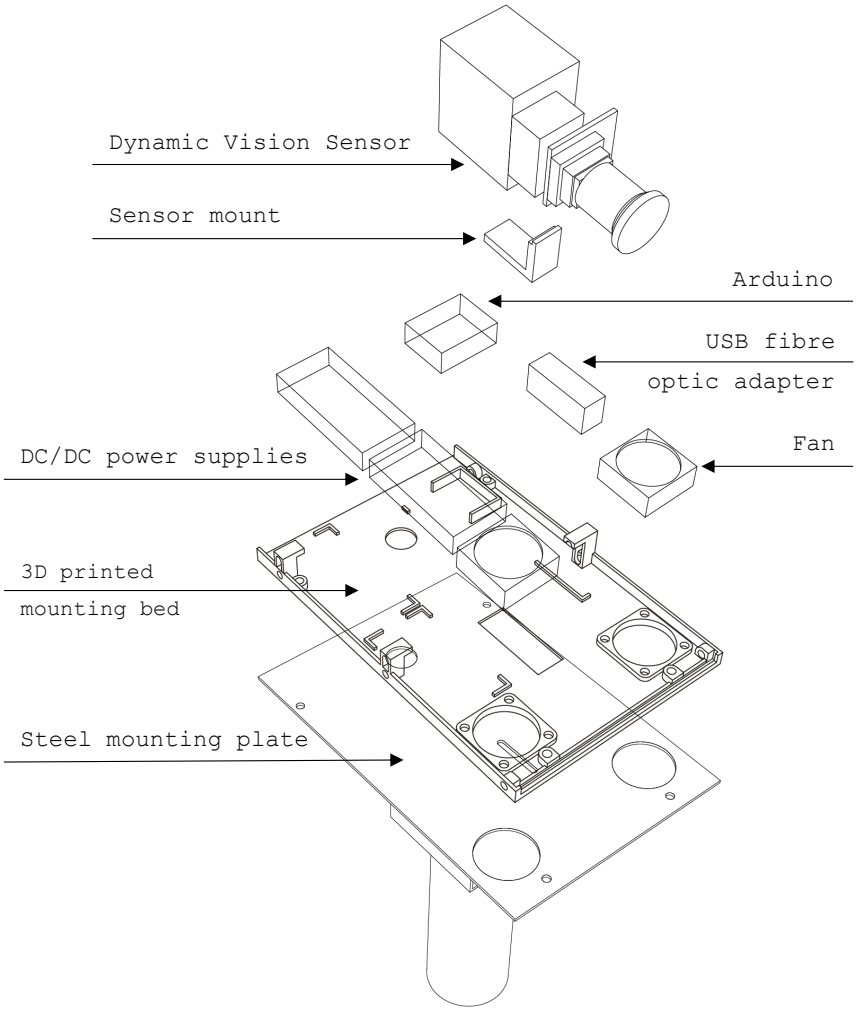
The designed enclosure is shown in Figure B.5. Note the overhang of the housing cover over the protective glass of the sensor. This increases protection against environmental influences, including raindrops falling on the protective glass and affecting the sensor signal.

A forced airflow is generated by the fans in the front of the enclosure and an open slot in the bottom of the rear enclosure cover. This serves to cool the components and also prevents

(a) Streetlamp pole mount

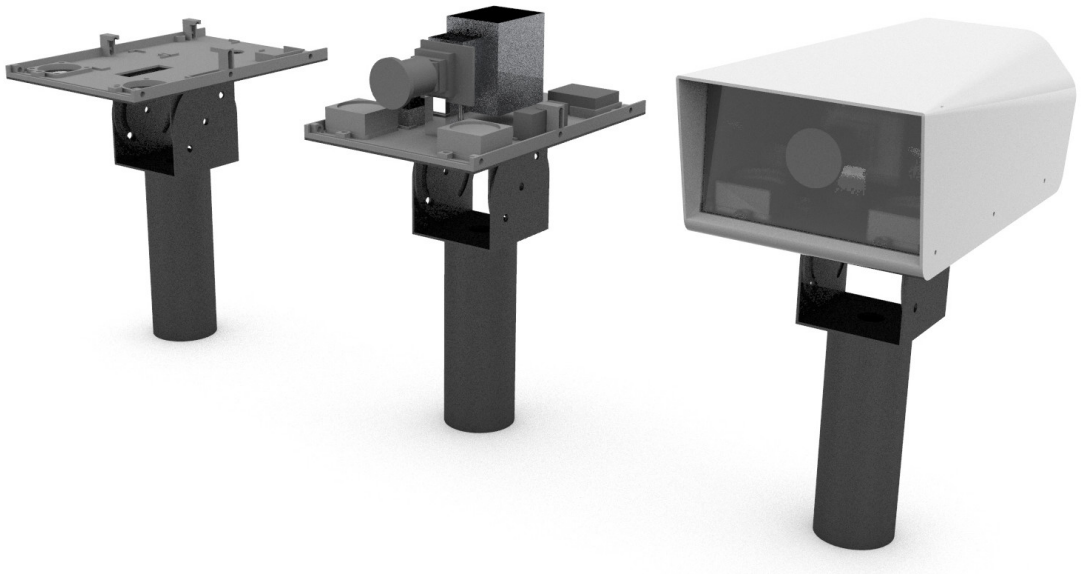


(b) Pitch adjustable mounting plate

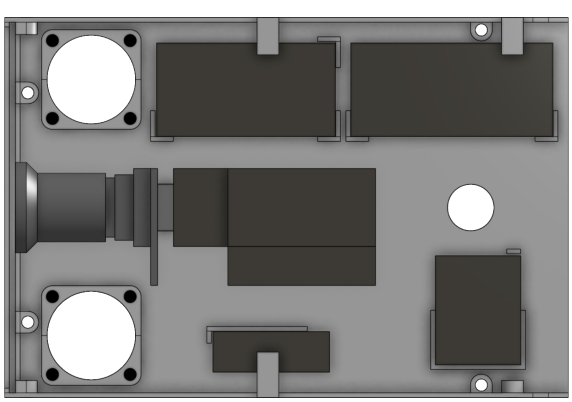


(c) Exploded view of components

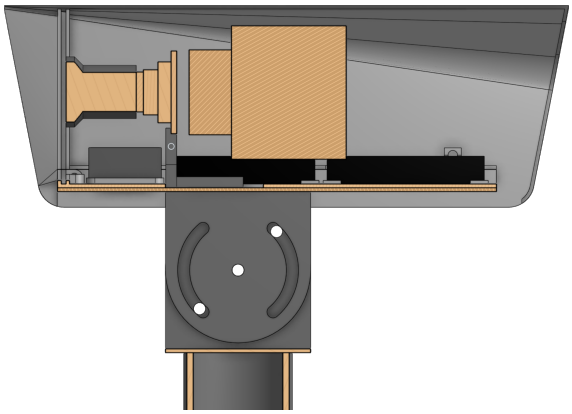
Figure B.4: Sensor mount and system components.



(a) Rendering at different assembly levels



(b) Top view



(c) Side (cutaway) view

Figure B.5: CAD model of the sensor enclosure.

the protective glass from fogging up and impairing visibility. All areas left open for ventilation were equipped with insect meshes to prevent insects from entering.

The case was manufactured and painted in a light gray color to minimize heat absorption from the sun in the summer. Rounded shapes were used to reduce the wind load on the enclosure, thereby helping to reduce the induced ego-motion component in windy conditions.

B.3 Sensor Positioning

The sensor locations had to be determined for planning the underground cabling before the measurement area was remodeled, i.e. before a 3D SFM model of the real and final area could be created (see Section 11.1.2).

The layout of the sensor positions required a compromise between

(a) the number of Dynamic Vision Sensors to be installed,
(b) the exact position on the measurement area, and
(c) the mounting height of the sensors

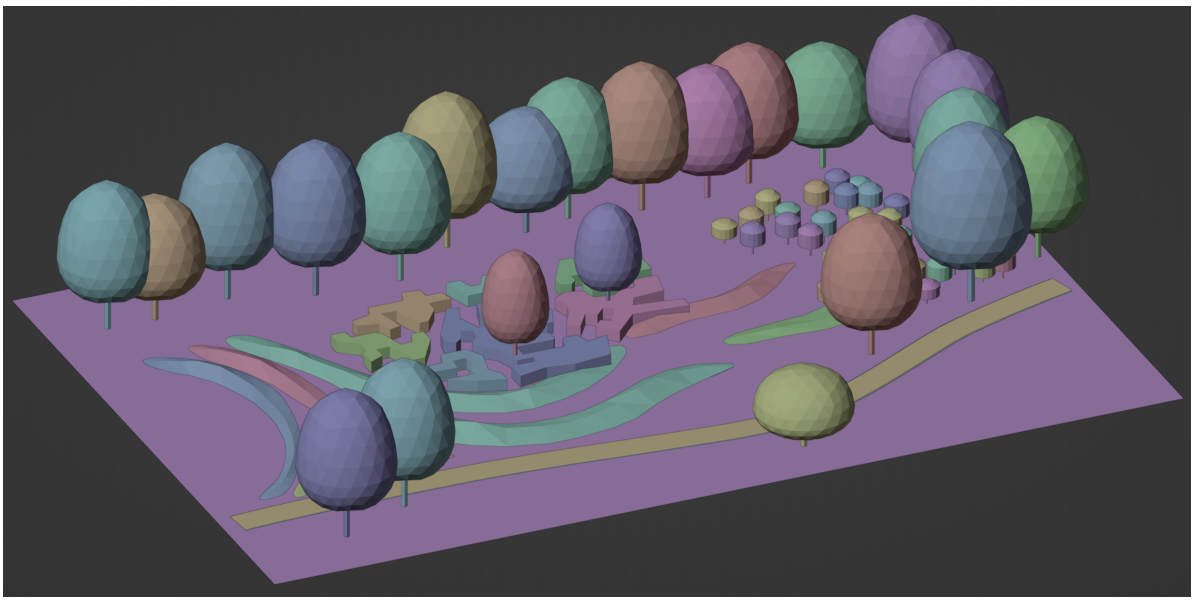
in order to achieve the best possible sensor coverage. It was also necessary to take into account the expected occlusions in the field of view of the sensors due to the planned terrain modeling of the playground.

For this purpose, a highly abstracted 3D model of the planned area was created, since the planning of the measurement area was only carried out in the form of a 2D map (see Figure 3.1a on page 22). The resulting model is shown in Figure B.6a.

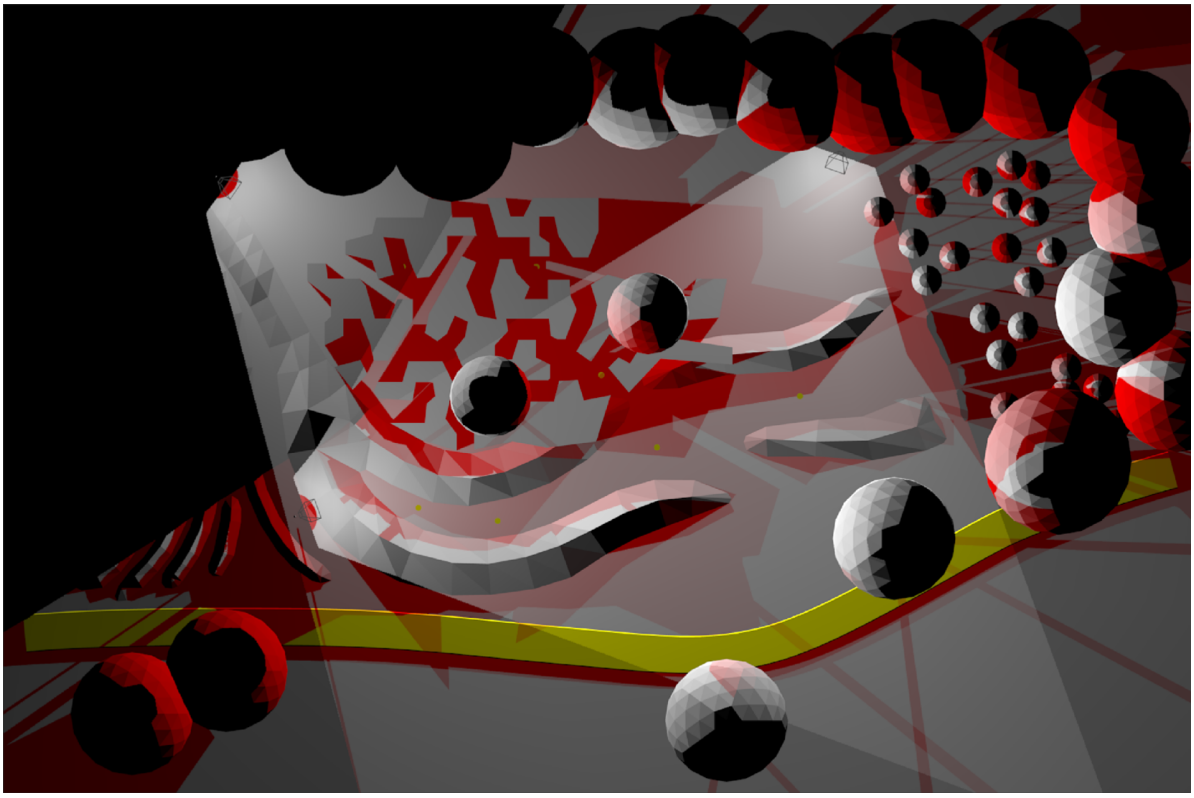
This model was used in a 3D simulation created using the software Blender to generate occlusion maps of the area for planning. The simulation took into account the basic sensor characteristics of the CeleX-IV Dynamic Vision Sensor and the lens used. An example of such an occlusion map is shown in Figure B.6b. Here, a higher degree of occlusion is represented by a higher saturation of the included reds.

Within this simulation, different sensor locations and mounting heights were simulated (see Figure B.6c–e).

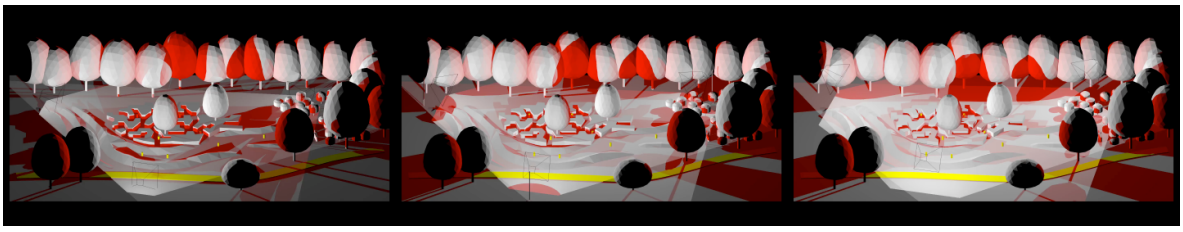
As a result of this process, the final configuration shown in Figure 3.1a was selected.



(a) Simplified 3D playground model



(b) Simulated occlusion map



(c) 5 m mounting height (d) 10 m mounting height (e) 15 m mounting height

Figure B.6: Blender-based 3D sensor position modeling and optimization prior to playground redesign.

Appendix C

Software Implementation

For the development of event-driven applications, no standard software library has yet been established. Typically, user programs are based mainly on the SDKs provided by the sensor manufacturers.

For the CeleX-IV Dynamic Vision Sensor, the manufacturer provides a C++ based SDK[31]. Based on this SDK, a comprehensive DVS processing application has been developed. In the following sections, we will briefly introduce the basic components of the SDK and the structure of implemented extensions.

The C++ application was developed using several software libraries, including Boost[32] for threading, file system abstractions, I/O operations, and argument parsing. Additionally, the Point Cloud Library[33] was used for event processing and to render the event stream in 3D. OpenCV[34] was used to generate frame-based event representations and 2D visualizations.

The third-party components were selected with respect to their license terms. Both free and commercial use of the developed components is possible.

The developed and customized application was used for basic event processing, while standalone Python applications were utilized for deep learning-based processing such as training, inference, and segmentation evaluations. Data exchange formats and protocols were used for communication between the applications.

C.1 CeleX-IV SDK: Structure and Basic Processing

During development, the provided CeleX-IV SDK underwent extensive changes, optimizations, and bug fixes. These changes were made to increase the program's efficiency and stability. As a result, the descriptions that follow may differ slightly from the publicly available source code of the manufacturer's SDK. However, the basic structure, functionality and process remain unaffected.

Figure C.1a shows the main classes of the CeleX-IV SDK, with the `CeleX4` class as the central component. This class communicates with the FPGA sensor carrier board through the `FrontPanel`. It also instantiates and manages various threads.

It is important to note that in SDK terminology, the term `"frame"` (e.g., `setEventFrameTime(...)`, `onFrameDataUpdated(...)` or `FrameData`) does not refer to classic 2D image

[31] `https://github.com/CelePixel/CeleX4-OpalKelly`
[32] `https://github.com/boostorg/boost`
[33] `https://github.com/PointCloudLibrary/pcl`
[34] `https://github.com/opencv/opencv`

frames. Instead, it refers to the time "frames" of the event stream that are transmitted and processed.

The `FrontPanel` from OpalKelly, which manufactures the FPGA board, provides an application programming interface (API) to the FPGA board. The `CeleX4` class utilizes the `FrontPanel` to configure the sensor's basic operating properties. The class implements several methods to abstract these configurations in application programs. After initialization, the sensor operates independently and writes its output continuously to the SDRAM of the FPGA board. The sensor output can also be accessed through the `FrontPanel` API.

Figure C.1b displays the process interactions in the CeleX-IV SDK, which are executed until an event data structure is passed to a user defined application logic.

The `CeleX4` class instance must repeatedly call `blockPipeOut(...)` provided by the `FrontPanel` API to read the FPGA's SDRAM. If a binary save is requested within the SDK logic, the acquired SDRAM content is passed to an instance of a `DataRecorderThread`, which will dump the binary content to a file.

The acquired data is passed to an instance of a `DataProcessThread`, which processes it by calling methods of a `FPGADataProcessor` member. This step converts the binary data format provided by the sensor, or more precisely by the FPGA board, into C++ data structures, namely `EventData` and `FrameData`.

The resulting data is then passed as `CeleX4ProcessedData` to a `CeleX4DataManager`, which informs a `SensorDataObserver` instance about new, available event data via a previously registered callback. At this point, user-defined application logic takes over for further processing.

C.2 Processing Pipeline and Stage Concept

To achieve flexible and reusable event data processing, we designed and implemented a software architecture that allows event "frames" to be processed in stages.

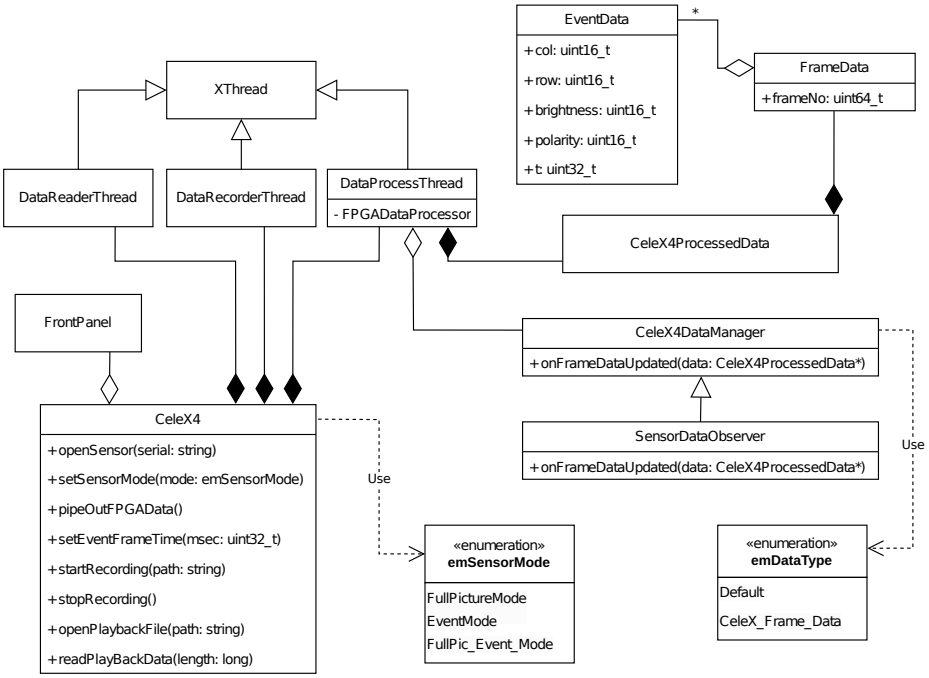
Events are processed in sequential stages, such as filtering or enriching them with derived attributes, and then passed to a subsequent stage for further processing. Other Dynamic Vision SDKs now typically offer this type of processing architecture[35], while the CeleX-IV SDK does not. This architecture type provides flexibility, reusability, and a clear separation of processing responsibilities. This results in improved maintainability and expandability of the codebase.

The architecture, shown in Figure C.2a, was developed to enable a CeleX-based processing using this concept. It essentially consists of two interfaces, the `IStage` and the `IData`, and the classes derived from them.

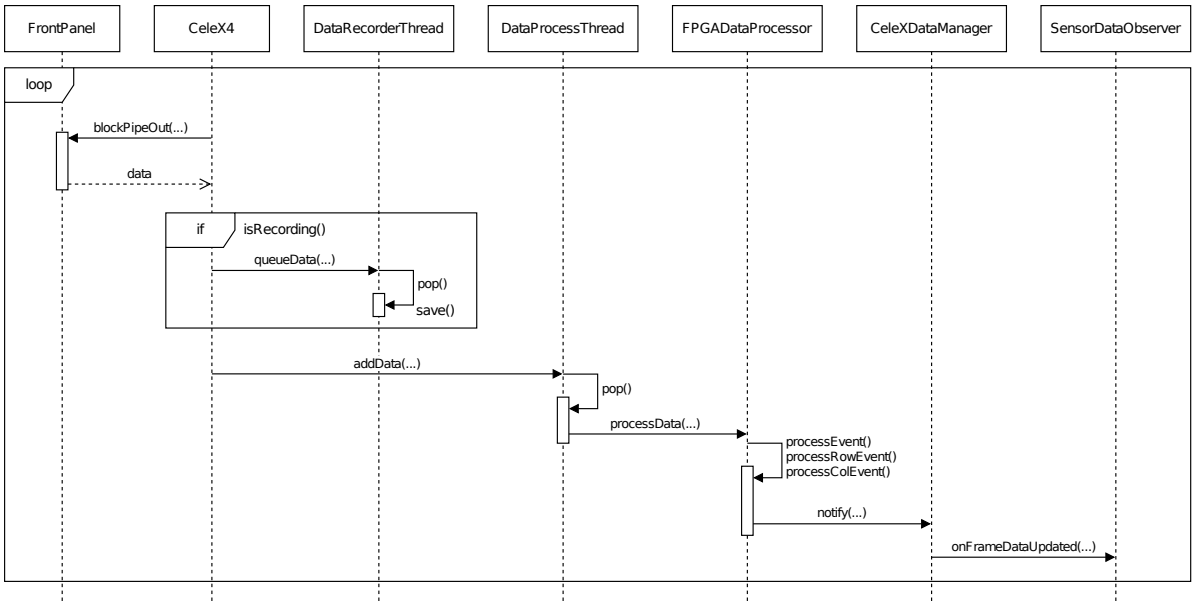
The classes derived from `IData` represent a type of exchangeable and expandable data container available for processing. The `IStage` class is a subclass of `XThread`, which is provided by the CeleX-IV SDK. This design allows for parallel processing of individual processing steps since each stage is executed by its own thread.

The `Pipeline` class, which is also derived from `IStage`, fulfills a special function. It manages the structure and ensures a FIFO order for processing event data. Each processing stage takes data from the front of an event deque, if available, and must implement its own

[35] such as "jAER" [Delbruck, 2008],
or `https://dv-sdk.inivation.com/master/sdk/dv-modules/index.html`
or `https://docs.prophesee.ai/stable/guides/sdk_pipelines/index.html`



(a) UML class diagram of simplified main SDK components



(b) UML sequence diagram of the main SDK interactions for obtaining event data

Figure C.1: Main structure and processing components of CeleX-IV SDK.

method `performStageLogic(IData*)`. The processed data is then appended to the end of the deque of the subsequent stage.

A mechanism was implemented to configure the structure of pipeline stages at runtime based on a JSON description, which further increases flexibility. This feature allows for the dynamic configuration of the processing flow without requiring the C++ application to be recompiled.

Figure C.2b illustrates the described pipeline and stage logic based on the processing sequences involved, using the following JSON declaration of a processing pipeline:

```
{
    "SDK": { ... },
    "ApplicationLogic": { ... },
    "Pipeline": [
        {
            "name": "StageTimeFilter",
            "config": {
                "msThres": 10.0
            }
        },
        {
            "name": "StageCSVExport",
            "config": {
                "outPath": "/tmp/csv/"
            }
        }
    ]
}
```

Listing C.1: Sample JSON description for dynamically initializing a processing pipeline.

The `StageClear` is automatically added to every configured pipeline, as shown in Figure C.2b. Its purpose is to release the event data that has been allocated to the memory heap at the end of processing.

C.3 Custom Data Export and Reader Concept

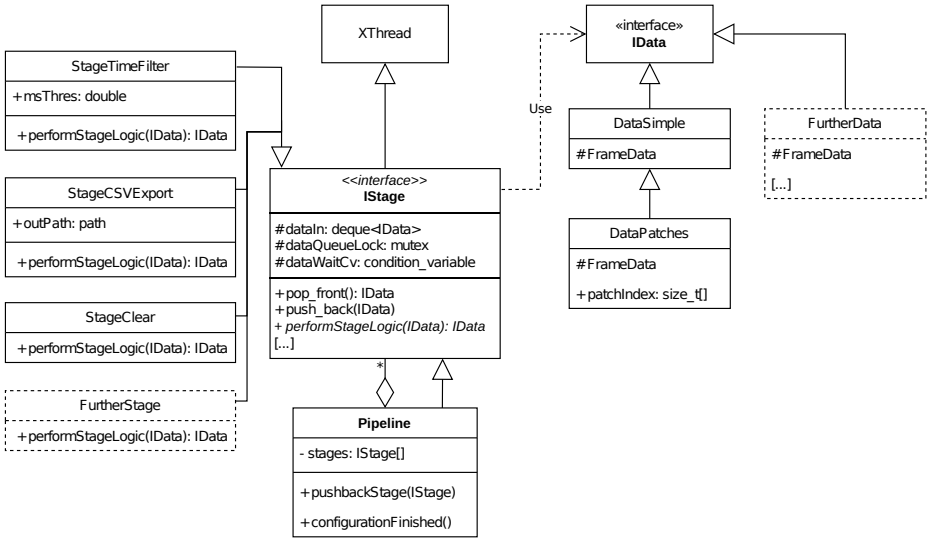
As shown in Figure C.1 and described above, the CeleX-IV SDK allows direct writing of binary dumps of event data. However, the data stored here consists of unprocessed, raw binary data that was directly read from the SDRAM of the FPGA interface board. This means that this data corresponds to the data that was exchanged over the USB connection. At this stage, no processing has been done by the SDK, including the creation of data structures containing the actual C++ event data structures. Thus, the SDK board tools do not support the storage of event data that has already been (pre)processed in the processing pipeline.

Therefore, a custom data writer and binary data format were implemented to store pre-processed data, e.g. time-filtered event data, or processing results like object labels obtained by segmentation.

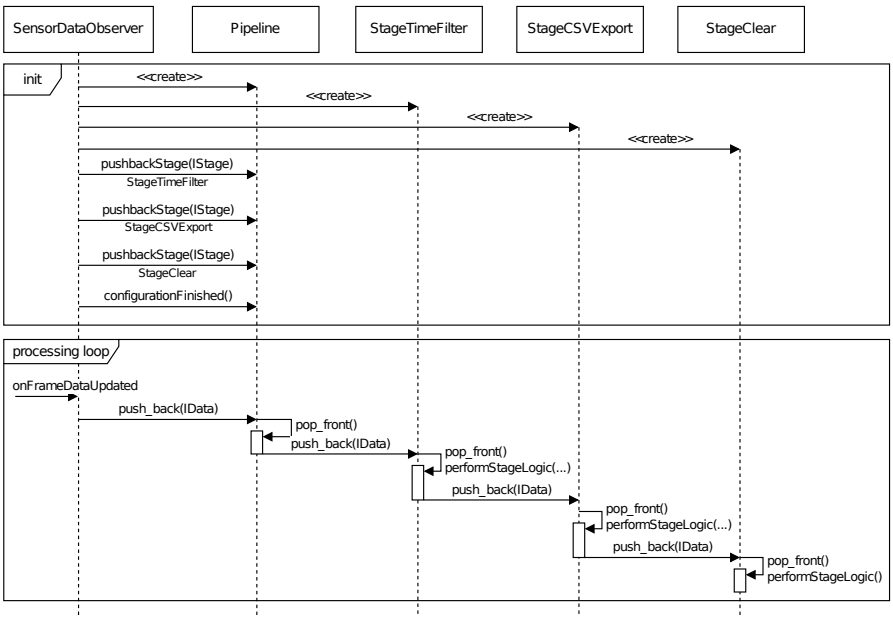
This custom writer is implemented as its own `IStage` and thus can be used directly in the processing pipeline. A number of binary formats have been defined, containing events and different attributes obtained during processing. An automatic selection of a suitable data decoder is achieved by proving meta information for each stored file.

The stored data is directly written as a gzip[36] compressed binary stream to reduce the resulting file sizes and to allow for a longer recording period on-site at the Living-Lab, previously a manual exchange of the used external SSD was necessary. Additionally, a file rollover

[36] `https://www.gnu.org/software/gzip/`



(a) UML class diagram of simplified main pipeline components



(b) UML sequence diagram of the processing in configured pipeline

Figure C.2: Structure of the developed and implemented generalized processing pipeline.

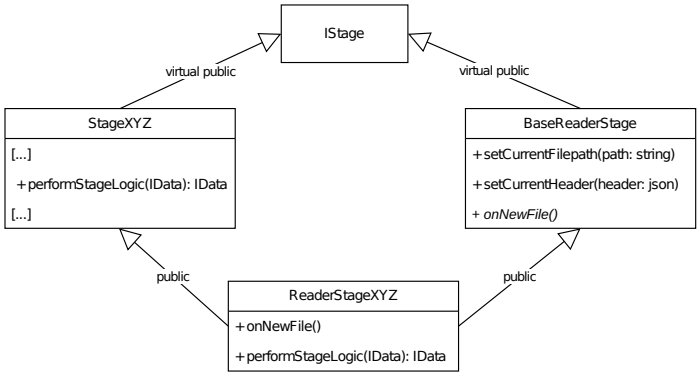


Figure C.3: Diamond inheritance design pattern for custom data reader.

mechanism based on a maximum file size was implemented to ensure that the files remain manageable during the performed long-term monitoring process.

Reader Concept

In order to reuse the processing stages implemented in the main CeleX-IV acquisition application, a reader application based on the same source tree was designed and implemented.

This design enables the abstraction of different `ReaderStages` in a `Pipeline` instance by inheriting a `BaseReaderStage` from the described `IStage` interface. By inheriting specific `ReaderStage` using C++'s multiple inheritance mechanism from the `BaseReaderStage` and the corresponding "normal" stage implementation, it is possible to reuse all custom implemented `IStage` processing stages in this reader context.

To prevent any ambiguity in this multiple inheritance from `IStage`, the *diamond inheritance design pattern*, as shown in Figure C.3, was utilized.

This allows for one-to-one adoption of the `performStageLogic(...)` implementation from the main stage, or to customize the behavior for custom written data by overwriting it in the `ReaderStage`.

C.4 Living-Lab: PointNet++ Inference

Chapter 9 describes a procedure for selecting recordings in the measurement field that should be saved for further analysis. This processing involves a PointNet++ inference to obtain a semantic segmentation of the captured event data. As described in this chapter, this inference is performed on an edge-AI computing device, the NVIDIA Jetson TX2, which allows for low-power processing. Appendix B provides additional details that two of these TX2 modules were used on-site.

The two modules had to process the event data from the three CeleX-IV Dynamic Vision Sensors on the playground. In the worst case, 16 inferences per sensor had to be performed simultaneously, resulting in a total of 48 inferences for all sensors.

A processing stage was implemented that encapsulates the event data into an HTTP request, which is then sent to these modules for further processing. This HTTP communication incorporates an HTTP load balancer proxy, which allows for fair processing of inference requests while ensuring high utilization of each TX2 module.

Appendix D

Datasets

The following sections provide supplementary information on the DVS-OUTLAB (see Chapter 4) and N-MuPeTS (see Chapter 5) datasets that extends the information provided in the main text.

D.1 DVS-OUTLAB: Event Count Statistics

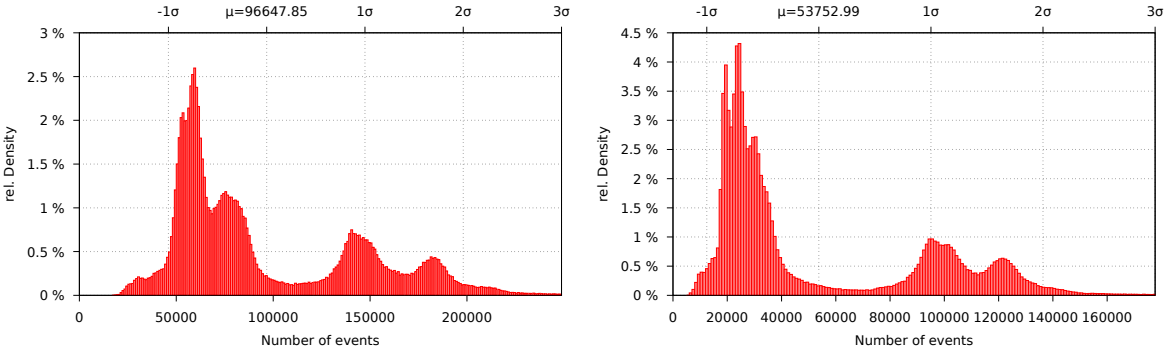
This section is based on the supplementary material provided with [**Bolten** et al., 2021].

In Section 6.1.2, event count requirements for the processing of space-time event clouds using PointNet++ are described.

Figure D.1a illustrates the total number of events per 60 ms time window in the entire sensor field of view of 768×512 pixels, based on the staged scenes contained in the DVS-OUTLAB dataset. Figure D.1b shows the number of events remaining after applying a 10 ms spatio-temporal time filter as preprocessing.

Due to the high number of events, it was necessary to introduce a spatial patching in addition to temporal windowing in order to process this data. Therefore, patches of 192×128 pixels in size were formed. Based on the DVS-OUTLAB dataset, the event counts that occurred in these patches and the effects of the applied spatio-temporal time filtering are shown in Figure D.2.

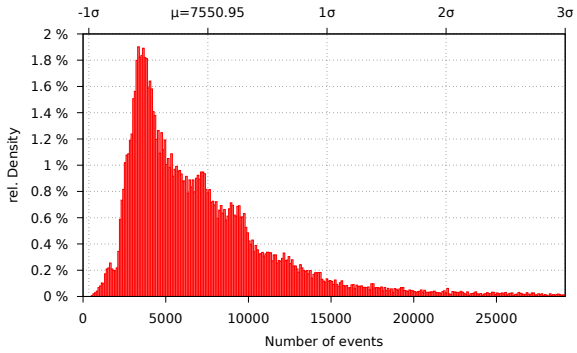
This analysis, primarily the determined average count, formed the basis for the selection of the event subsampling target count used in the point-based processing approaches.



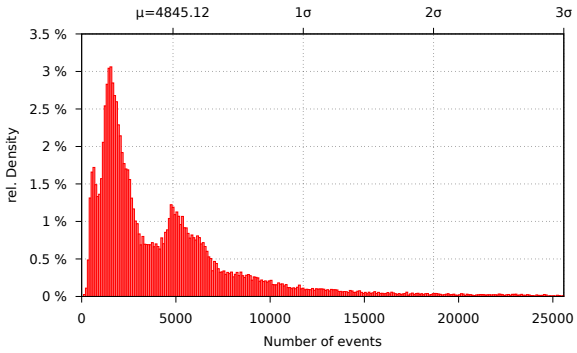
(a) Only staged scenes, unfiltered

(b) Only staged scenes, 10 ms time-filtered

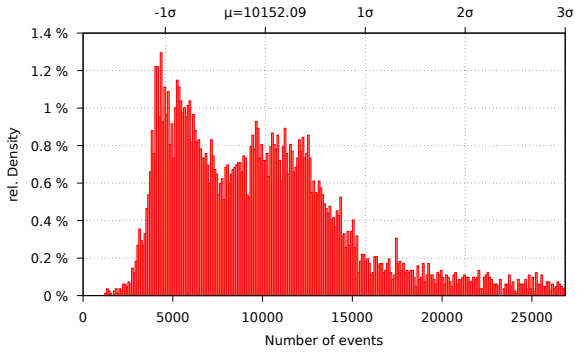
Figure D.1: Event counts in staged scenes of DVS-OUTLAB database (complete 768 px × 512 px sensor view; time window of 60 ms).



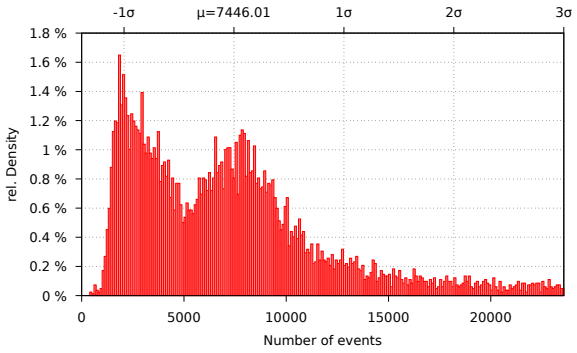
(a) Complete DVS-OUTLAB, unfiltered



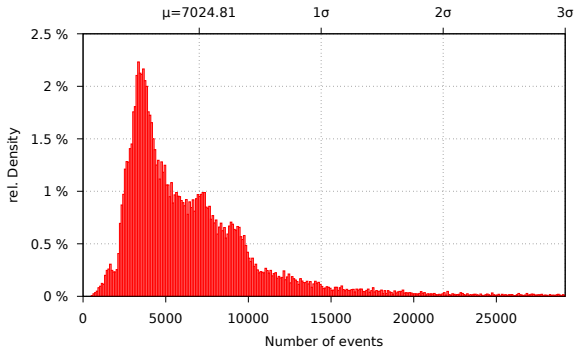
(b) Complete DVS-OUTLAB, 10 ms time-filtered



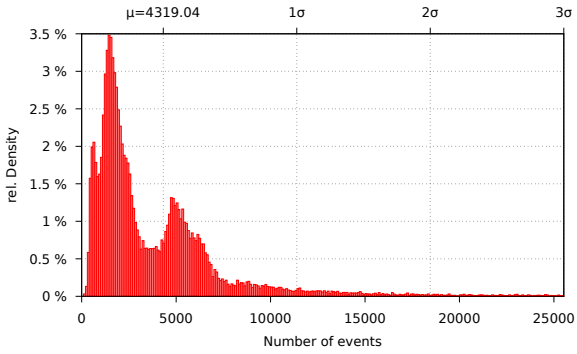
(c) Only staged scenes, unfiltered



(d) Only staged scenes, 10 ms time-filtered



(e) Only environmental influences, unfiltered



(f) Only environmental influences, 10 ms time-filtered

Figure D.2: Event counts in DVS-OUTLAB database (per spatial patch of 192 px × 128 px; time window of 60 ms).

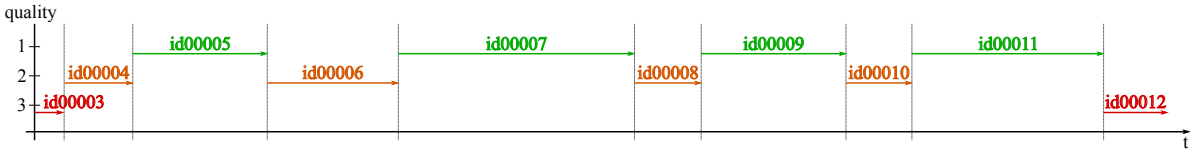


Figure D.3: Possible sequence merge from `Seq. id00004` to `Seq. id00011` when using quality ≤ 2 (quality 1 in green, quality 2 in orange, quality 3 in red).

D.2 N-MuPeTS: Masks, Statistics and Sequences Annotations

The following sections are based on the supplementary material provided with [**Bolten** et al., 2023a].

D.2.1 Label Mask Generation

The main part of the dataset preparation is based on the automated generation of label masks from APS color images using color features provided by the actors' clothing (see Section 5.3). The quality of this processing step cannot be determined quantitatively due to the lack of references such as manual ground truth annotations. However, subjective and qualitative evaluations are possible. Figure D.4 shows randomly sampled results of the label generation. These examples demonstrate the high quality of the labels generated.

The left column shows a cropped APS input image, while the middle column shows the automatically generated label mask. The mapping quality achieved between the APS and DVS fields of view also plays an important role in the generation of the dataset. Examples are shown in the right column of the figure. The corresponding time window of the DVS event stream is shown as a projection along time. DVS events are shown in gray unless they are under the assigned label mask. In these cases, the events are colored according to the label mask.

D.2.2 Statistics

In Section 5.4, different quality classes are defined for the recorded dataset sequences. While the main text only summarizes the quality level 1, which reflects scenes without any issues, Tables D.1–D.6 show the duration and occurrence statistics including the imperfect quality levels 2 and 3. The information here is cumulative with respect to higher quality classes. This means, for example, that the information for quality class 2 also includes class 1.

D.2.3 Dataset Sequences

As stated, the dataset is divided into individual sequences by distinguishing different quality classes. A new sequence is started each time the quality class is changed. Multiple annotations can be assigned to a single sequence. For example, a single sequence can contain WALKING and OCCLUSION (compare to given annotation file in Listing D.1).

Due to the high-quality requirement, the scenes of quality class 1 are naturally rather short. Especially for further processing, a longer sequence length may be of interest. By including quality class 2, temporally longer sequences can be formed. Temporally consecutive sections of quality 1 and 2 can be combined as long as they are not interrupted by quality 3. Figure D.3 shows this exemplarily, in which sequences `Seq. id00004` to `Seq. id00011` can be combined.

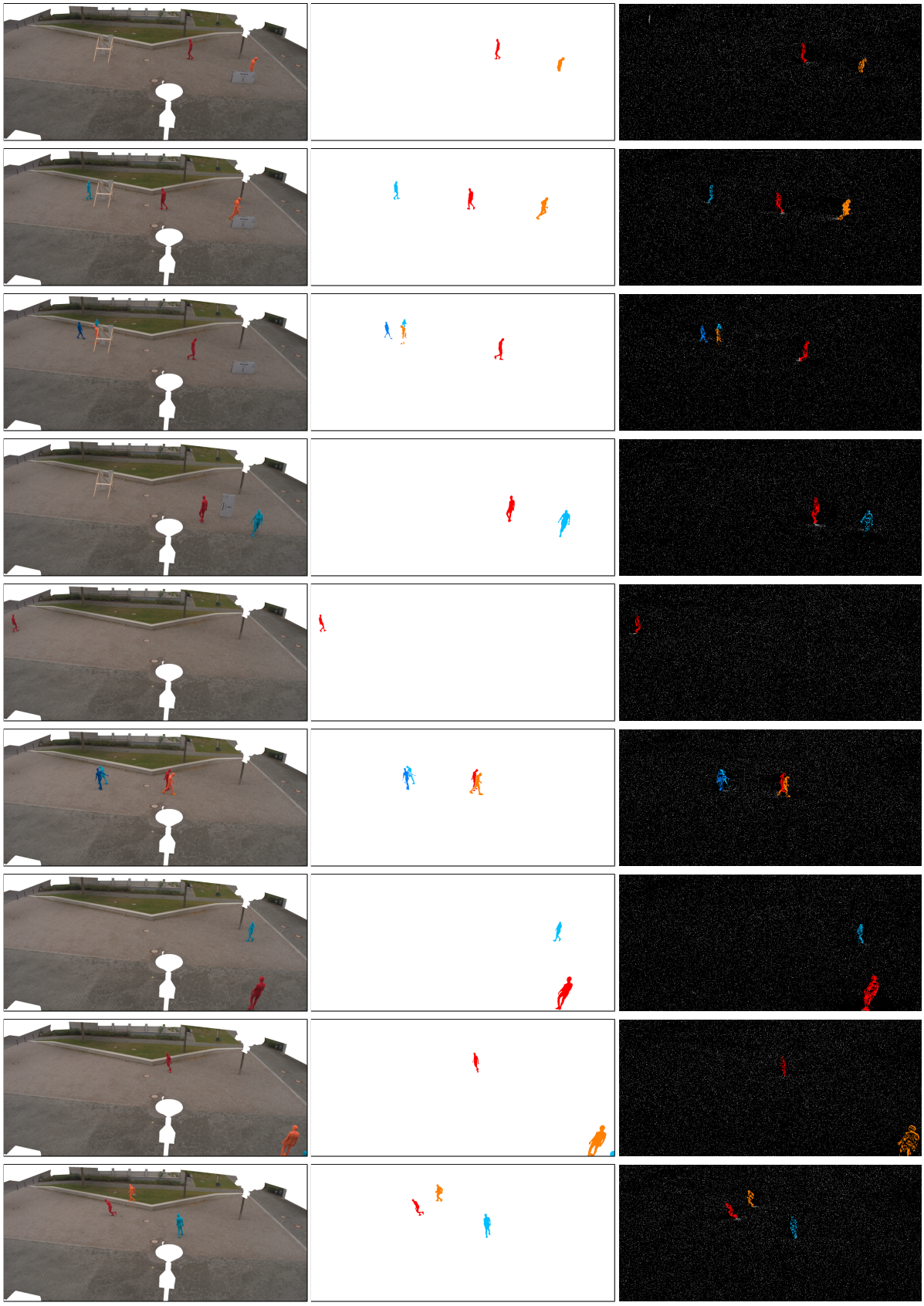


Figure D.4: N-MuPeTS: APS color segmentation examples (best viewed digitally zoomed).

Quality Class 2

RED	ORANGE	CYAN	BLUE	Cumulative duration: per color combination	Cumulative duration: sum per person count
					378
•				385	1052
	•			182	
		•		259	
			•	227	
•	•			222	782
•		•		46	
•			•	156	
	•	•		260	
	•		•	85	
		•	•	14	
•	•	•		589	831
•	•		•	154	
•		•	•	57	
	•	•	•	30	
•	•	•	•	406	406

Table D.1: Cumulative durations per color combination in quality class 2.

Annotation	Number of sequences	Mean duration	Cumulative duration
RED	230	8.8	2016
ORANGE	212	9.1	1928
CYAN	173	9.6	1661
BLUE	137	8.2	1127
BACKGROUND	71	5.3	378
STANDING	100	3.2	325
WALKING	658	4.1	2694
RUNNING	142	3.8	542
RANDOM	19	8.7	166
HELIX	14	5.2	72
FAR	79	3.9	305

Table D.2: Duration statistics per annotation in quality class 2.

Annotation	Number of occurences
OCCLUSION	239
EXERCISING	9
KNEELING	13
STOOPED	24
WAVING	10
CROSSING	228
MEET	50
SIDE BY SIDE	166

Table D.3: Occurrence statistics per annotation in quality class 2.

Quality Class 3

RED	ORANGE	CYAN	BLUE	Cumulative duration per color combination	Cumulative duration sum per person count
					1140
•				531	1479
	•			339	
		•		338	
			•	271	
•	•			321	1004
•		•		73	
•			•	186	
	•	•		279	
	•		•	122	
		•	•	23	
•	•	•		765	1048
•	•		•	177	
•		•	•	73	
	•	•	•	34	
•	•	•	•	473	473

Table D.4: Cumulative durations per color combination in quality class 3.

Annotation	Number of sequences	Mean duration	Cumulative duration
RED	270	9.6	2599
ORANGE	262	9.6	2510
CYAN	200	10.3	2058
BLUE	163	8.3	1359
BACKGROUND	120	9.5	1140
STANDING	117	3.4	400
WALKING	786	4.4	3465
RUNNING	171	4.1	705
RANDOM	31	7.2	223
HELIX	21	4.6	98
FAR	87	4.3	377

Table D.5: Duration statistics per annotation in quality class 3.

Annotation	Number of occurences
OCCLUSION	300
EXERCISING	12
KNEELING	14
STOOPED	37
WAVING	11
CROSSING	282
MEET	55
SIDE BY SIDE	181

Table D.6: Occurrence statistics per annotation in quality class 3.

D.2.4 Dataset Annotations

Annotations (see Section 5.4.4) have been created for each recorded sequence and are provided in separate JSON files. The JSON description starts with a block that contains details about the entire sequence.

In addition to the sequence ID as a counter, the quality class assignment, and the total length of this sequence, the `aggregated_annotation` field provides a union of all annotations within this sequence for convenience. An example is given in the following listing:

```
    {
      "id": 338,
      "quality": 1,
      "length": 81,
      "aggregated_annotation": [
          "occlusion",
          "walking"
      ],
```

This is followed by a detailed annotation description per person. For each person, a list of start and end blocks describes the annotation performed in its temporal context. These start and end information refer to a 25 ms sliding time window (analogous to the "frame number" of an APS camera) of the corresponding and plain DVS recording:

```
  "colors": {
    "CYAN": [
        {
            "start": 13440,
            "end": 13520,
            "annotation": [
                "walking"
            ]
        }
    ],
    "BLUE": [],
    "ORANGE": [],
    "RED": [
        {
            "start": 13440,
            "end": 13514,
            "annotation": [
                "walking"
            ]
        },
        {
            "start": 13515,
            "end": 13520,
            "annotation": [
                "occlusion",
                "walking"
            ]
        }
    ]
  }
}
```

Listing D.1: JSON annotation of a single sequence in N-MuPeTS.

Appendix E

Segmentation

The following sections provide supplementary information, such as specific network configurations or advanced evaluation results from the segmentation evaluations conducted in Part III.

E.1 PointNet++ Semantic Segmentation

The following sections are based on the supplementary material provided with [**Bolten** et al., 2022a].

E.1.1 Network Configurations

In Section 6.3.3, key aspects with an estimated high impact on the results expected from a PointNet++ architecture were identified and ranked as follows:

1. the configured layer depth,
2. the number of events in the first Set Abstraction layer,
3. the variation of spatio-temporal scaling and the selection of corresponding radii,
4. the modification of considered temporal neighborhoods.

Based on these key aspects an optimization of the 3D processing hyperparameters was performed. The exact network configurations tested in this greedy optimization are summarized in Table E.1. Table E.2 summarizes the configurations of the final evaluated 3D-based processing approaches.

The intent of providing these details is to eliminate any ambiguity in comprehension and to support potential reproducibility.

E.1.2 Extended Result: Detailed per-class F1 Scores

In order to present the results in a more readable and clear manner, the object classes in the evaluations performed in the main body of this dissertation have been grouped. The three categories formed are *background*, *objects*, and *environmental influences*. Metric scores were given for these aggregated groups, and due to the very uneven distribution of events per class, a weighted F1 score was used.

For completeness, the unweighted F1 scores for all ten classes are provided in Table E.3 and Table E.4 for the experiments conducted in Chapter 6.

(a) Layer experiments

PointNet++(1024, 3L):	SA(1024, 0.1, [32, 32, 64]) → SA(256, 0.3, [64, 64, 128]) → SA(16, 0.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
PointNet++(1024, 4L):	SA(1024, 0.1, [32, 32, 64]) → SA(256, 0.2, [64, 64, 128]) → SA(64, 0.4, [128, 128, 256]) → SA(16, 0.8, [256, 256, 512]) → FP([256, 256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
PointNet++(1024, 5L):	SA(1024, 0.1, [32, 32, 64]) → SA(512, 0.2 [64, 64, 128]) → SA(256, 0.4, [128,128,256]) → SA(64, 0.6, [256, 256, 512]) → SA(16, 0.8, [512, 512, 1024]) → FP([256, 256]) → FP([256, 256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
PointNet++(1024, 6L):	SA(1024, 0.1, [32, 32, 64]) → SA(512, 0.2, [64, 64, 128]) → SA(256, 0.3, [128, 128, 256]) → SA(128, 0.4, [256, 256, 512]) → SA(64, 0.6, [512, 512, 1024]) → SA(16, 0.8, [1024, 1024, 2048]) → FP([256, 256]) → FP([256, 256]) → FP([256, 256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])

(b) Point count experiments

PointNet++(512, 3L):	SA(512, 0.1, [32, 32, 64]) → SA(256, 0.3, [64, 64, 128]) → SA(16, 0.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
PointNet++(1024, 3L):	SA(1024, 0.1, [32, 32, 64]) → SA(256, 0.3, [64, 64, 128]) → SA(16, 0.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
PointNet++(2048, 3L):	SA(2048, 0.1, [32, 32, 64]) → SA(256, 0.3, [64, 64, 128]) → SA(16, 0.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
PointNet++(3072, 3L):	SA(3072, 0.1, [32, 32, 64]) → SA(256, 0.3, [64, 64, 128]) → SA(16, 0.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])

(c) Input scaling experiments

S^T_{cube} PointNet++(2048, 3L):	SA(2048, 0.1, [32, 32, 64]) → SA(256, 0.3, [64, 64, 128]) → SA(16, 0.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
S^T_{tScaled} PointNet++(2048, 3L):	SA(2048, 9.6, [32, 32, 64]) → SA(256, 28.8, [64, 64, 128]) → SA(16, 76.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
S^T_{native} PointNet++(2048, 3L):	SA(2048, 9.6, [32, 32, 64]) → SA(256, 28.8, [64, 64, 128]) → SA(16, 76.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])

(d) Spatio-temporal weighting experiments

Time weight $\gamma = 1$ PointNet++(2048, 3L):	SA(2048, 9.6, [32, 32, 64]) → SA(256, 28.8, [64, 64, 128]) → SA(16, 76.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
Time weight $\gamma = 3.2$ PointNet++(2048, 3L):	Weighted spatio-temporal distance: $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + 3.2 \cdot (t_i - t_j)^2}$
Time weight $\gamma = 20$ PointNet++(2048, 3L):	Weighted spatio-temporal distance: $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + 20 \cdot (t_i - t_j)^2}$

Table E.1: PointNet++ network configurations (compare to syntax used in [Qi et al., 2017b]).

PointNet++(2048, 3L):	SA(2048, 9.6, [32, 32, 64]) → SA(256, 28.8, [64, 64, 128]) → SA(16, 76.8, [128,128,256]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
A-CNN(2048, 3L):	MR(2048, [[0.0, 4.8], [4.8, 9.6]], [[32, 32, 64], [64, 64, 128]]) → MR(256, [[7.2, 14.4], [21.6, 28.8]], [[64, 64, 128], [128, 128, 256]]) → MR(16, [[19.2, 38.4], [57.6, 76.8]], [[128, 128, 256], [256, 256, 512]]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
LSANet(2048, 3L):	LSA(2048, 9.6, [32, 32, 64], [32, 32]) → LSA(256, 28.8, [64, 64, 128], [32, 32]) → LSA(16, 76.8, [128, 128, 256], [32, 32]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 10])
SpiderCNN(*, 3L):	BallQuery(9.6) → SpiderConv(32) → SpiderConv(64) → SpiderConv(128) → Top-2 → FC([256, 256, 128, 10])

Table E.2: 3D network variant configurations (syntax adapted and inspired from [Qi et al., 2017b]).

(a) Layer experiments

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
3 layers	0.95	**0.76**	**0.59**	**0.80**	0.70	**0.75**	0.41	0.88	0.83	0.28	**0.70**
4 layers	0.95	0.74	0.53	**0.80**	0.73	0.73	0.49	**0.89**	0.83	0.22	0.69
5 layers	0.95	0.75	0.56	**0.80**	0.76	**0.75**	**0.50**	**0.89**	**0.84**	0.22	**0.70**
6 layers	**0.96**	0.75	0.52	**0.80**	**0.77**	0.72	0.36	**0.89**	0.81	**0.31**	0.69

(b) Point count experiments

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
n = 512	0.95	0.74	0.49	0.71	0.70	0.70	0.20	0.86	0.82	0.23	0.64
n = 1024	0.95	**0.76**	0.59	0.80	0.70	**0.75**	0.41	0.88	0.83	0.28	0.70
n = 2048	**0.96**	**0.76**	0.60	**0.86**	0.79	**0.75**	**0.60**	**0.90**	**0.84**	**0.37**	0.74
n = 3072	**0.96**	**0.76**	**0.64**	**0.86**	**0.80**	**0.75**	**0.60**	0.89	**0.84**	0.35	**0.75**

(c) Input scaling experiments

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
S^T_{cube}	0.96	0.76	0.60	**0.86**	0.79	0.75	0.60	0.90	0.84	0.37	0.74
S^T_{tScaled}	**0.97**	**0.83**	**0.78**	0.80	0.86	**0.82**	**0.65**	0.92	0.88	0.52	**0.80**
S^T_{native}	**0.97**	0.82	0.73	0.85	**0.87**	0.80	0.63	**0.93**	**0.89**	**0.53**	**0.80**

(d) Spatio-temporal weighting experiments

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
Time weight $\gamma = 1$	**0.97**	**0.82**	**0.73**	0.85	**0.87**	0.80	**0.63**	**0.93**	0.89	**0.53**	**0.80**
Time weight $\gamma = 3.2$	**0.97**	**0.82**	0.69	**0.86**	0.86	**0.81**	**0.63**	**0.93**	**0.90**	0.49	0.79
Time weight $\gamma = 20$	0.96	0.80	0.70	**0.86**	0.81	0.79	0.60	0.91	0.86	0.31	0.76

Table E.3: Detailed per-class F1 scores in PointNet++ parameter optimization experiments.

(a) 3D Network variants

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
PointNet++	**0.97**	**0.83**	0.78	0.80	0.86	**0.82**	**0.65**	0.92	0.88	0.52	**0.80**
A-CNN	**0.97**	**0.83**	0.77	**0.81**	**0.87**	**0.82**	0.58	**0.93**	**0.89**	**0.58**	**0.80**
LSANet	**0.97**	**0.83**	**0.82**	**0.81**	**0.87**	**0.82**	0.57	0.92	**0.89**	0.54	**0.80**
SpiderCNN	0.95	0.74	0.51	0.53	0.80	0.73	0.41	0.83	0.76	0.32	0.66

(b) Mask R-CNN with ResNet50 backbone

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
Binary	**0.95**	0.82	0.73	0.60	0.85	0.85	0.38	0.90	0.87	0.58	0.75
Polarity	**0.95**	**0.84**	0.78	**0.61**	0.88	**0.87**	0.34	**0.95**	**0.89**	0.60	0.77
Frequency	**0.95**	0.82	0.70	0.59	0.88	0.85	0.45	0.93	0.88	0.62	0.77
MTC	**0.95**	**0.84**	**0.82**	0.59	**0.89**	0.86	**0.57**	0.93	0.88	**0.65**	**0.80**

(c) Mask R-CNN with ResNet101 backbone

Configuration	BACKGROUND	PERSON	DOG	RAIN	TREE	BICYCLE	SPORTSBALL	INSECT	BIRD	TREE_SHADOW	Mean
Binary	**0.95**	0.82	0.59	0.60	0.85	0.85	0.27	0.89	0.86	0.59	0.73
Polarity	**0.95**	**0.84**	**0.76**	0.60	0.88	**0.86**	0.42	**0.95**	**0.89**	0.62	0.78
Frequency	**0.95**	0.83	0.70	0.60	**0.89**	0.85	0.52	0.92	0.88	0.63	0.78
MTC	**0.95**	**0.84**	0.70	**0.61**	0.88	**0.86**	**0.59**	0.93	0.86	**0.64**	**0.79**

Table E.4: Detailed per-class F1 scores for network variants and 2D Mask R-CNN baseline.

E.2 Instance Segmentation

The following sections are based on the supplementary material provided with [**Bolten** et al., 2024].

E.2.1 Event Count Statistics: aRoI

In Section 8.2.1, a size-adaptive Region-of-Interest extraction algorithm is proposed. This algorithm yields spatial patches of event data depending on the included objects. Instead of extracting fixed-sized patches, this approach prevents objects from being sliced into separate parts, while grouping spartially close objects into the same region.

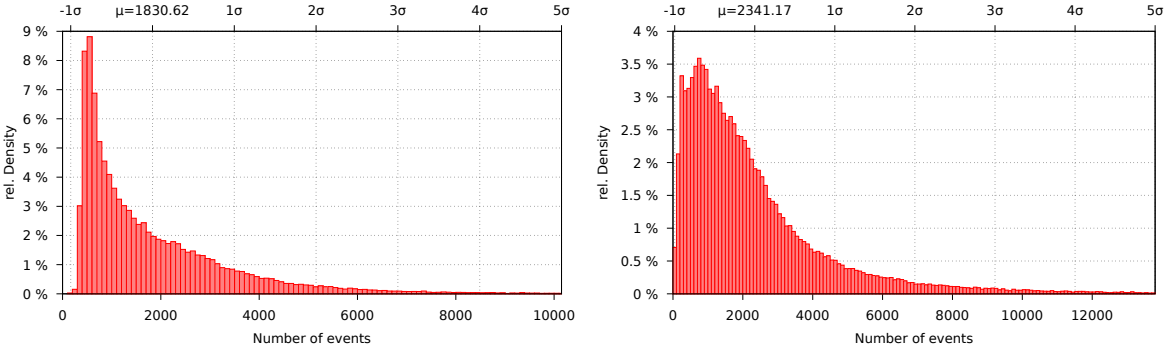
In line with the other event statistics given in this appendix, the number of events occurring in these aRoIs for the considered datasets is given in Figure E.1. Additional statistical parameters for these aRoIs are listed below:

DVS-iOUTLAB dataset aRoIs:		**N-MuPeTS dataset aRoIs:**	
Mean:	1830.6182	Mean:	2341.1705
Std Dev:	1663.9163	Std Dev:	2291.8931
Minimum:	126.0000	Minimum:	1.0000
Maximum:	24700.0000	Maximum:	28219.0000
Quartile:	664.0000	Quartile:	864.0000
Median:	1265.0000	Median:	1700.0000
Quartile:	2452.0000	Quartile:	2967.0000

E.2.2 Extended Result

In Section 8.3.3, an intentionally challenging data subset was created to evaluate the selected instance segmentation methods on the N-MuPeTS dataset. This subset contains only sequences in which at least one of the actors is occluded or in which the actors are spatially very close to each other. Table E.5 shows the resulting metrics for the *complete* test set of the N-MuPeTS dataset.

The object classes contained and evaluated in the DVS-iOUTLAB dataset were also presented in aggregated form in the evaluations presented in the main body. In order to provide complete, unaggregated results, Table E.6 shows the semantic segmentation results and Table E.7 shows the instance segmentation results in detail for all included object classes.



(a) DVS-iOUTLAB dataset (b) N-MuPeTS dataset

Figure E.1: Event counts for time Δ=10 ms filtered aRoIs (60 ms event time windows).

Network	Configuration	Semantic quality weighted F1 score NOISE	Semantic quality weighted F1 score PERSON	Instance quality PERSON mIoU	Instance quality PERSON $AP^{0.95}_{0.5}$	Instance quality PERSON $AP^{0.5}$	Instance quality PERSON $AP^{0.75}$
(a) Baseline method: PointNet++ with spatial clustering							
PointNet++ Clustering	in 2048 events	**0.93**	**0.96**	0.84	0.52	0.66	0.55
	in 1024 events	**0.93**	**0.96**	0.85	0.52	0.65	0.54
(b) Space-time event cloud-based methods							
JSNet	4 layers in 2048 events	**0.93**	**0.96**	0.86	0.70	0.87	0.75
	4 layers in 1024 events	0.92	0.95	0.84	0.64	0.82	0.70
3D-BoNet	4 layers in 2048 events	**0.93**	**0.96**	0.86	**0.72**	0.86	**0.76**
	4 layers in 1024 events	0.91	0.95	0.83	0.62	0.78	0.66
(c) Voxel-based method							
SoftGroup	voxel grid	0.88	0.95	**0.88**	0.71	0.83	0.75
(d) Frame-based methods							
Mask R-CNN	polarity	0.83	0.91	0.76	0.50	0.87	0.56
	MTC	0.83	0.91	0.76	0.52	0.88	0.58
YOLO v8	polarity	0.84	0.93	0.75	0.64	**0.91**	0.75
	MTC	0.84	0.93	0.76	0.64	**0.91**	0.64

Table E.5: Segmentation results on *complete* N-MuPeTS test set (60 ms event time window).

Network	Configuration	Semantic quality; weighted F1 score				
		PERSON	DOG	BICYCLE	SPORTSBALL	NOISE
(a) Baseline method: PointNet++ with spatial clustering						
PointNet++ Clustering	in 2048 events	0.95	0.87	0.86	0.94	0.93
	in 1024 events	0.95	0.85	0.85	0.94	0.93
(b) Space-time event cloud-based methods						
JSNet	4 layers in 2048 events	0.96	0.90	0.88	0.97	**0.95**
	4 layers in 1024 events	0.94	0.80	0.80	0.96	0.93
3D-BoNet	4 layers in 2048 events	0.96	0.86	0.90	0.93	0.93
	4 layers in 1024 events	0.95	0.85	0.89	0.93	0.92
(c) Voxel-based method						
SoftGroup	voxel grid	**0.98**	**0.98**	**0.97**	**0.98**	0.94
(d) Frame-based methods						
Mask R-CNN	polarity	0.94	0.94	0.94	0.94	0.87
	MTC	0.94	0.94	0.93	0.95	0.87
YOLO v8	polarity	0.94	0.90	0.93	0.86	0.86
	MTC	0.94	0.86	0.93	0.85	0.86

Table E.6: Semantic segmentation results per class on DVS-iOUTLAB dataset (60 ms event time window).

Network	Configuration	Instance quality: mIoU PERSON	mIoU DOG	mIoU BICYCLE	mIoU SPORTSBALL	$AP^{0.95}_{0.5}$ PERSON	$AP^{0.95}_{0.5}$ DOG	$AP^{0.95}_{0.5}$ BICYCLE	$AP^{0.95}_{0.5}$ SPORTSBALL
(a) Baseline method: PointNet++ with spatial clustering									
PointNet++ Clustering	in 2048 events	0.82	0.70	0.73	0.85	0.47	0.52	0.53	0.77
	in 1024 events	0.83	0.74	0.74	0.88	0.47	0.52	0.53	0.78
(b) Space-time event cloud-based methods									
JSNet	4 layers in 2048 events	**0.91**	0.91	0.72	0.93	**0.84**	0.71	0.79	0.91
	4 layers in 1024 events	0.89	**0.92**	0.57	**0.94**	0.77	0.55	0.69	0.81
3D-BoNet	4 layers in 2048 events	0.85	0.77	0.78	0.87	0.75	0.63	0.67	0.78
	4 layers in 1024 events	0.84	0.80	0.76	0.88	0.72	0.63	0.66	0.77
(c) Voxel-based method									
SoftGroup	voxel grid	0.86	0.88	**0.90**	0.93	**0.84**	**0.87**	**0.87**	**0.94**
(d) Frame-based methods									
Mask R-CNN	polarity	0.78	0.79	0.79	0.84	0.51	0.60	0.63	0.74
	MTC	0.77	0.75	0.78	0.83	0.50	0.61	0.59	0.73
YOLO v8	polarity	0.80	0.73	0.82	0.66	0.64	0.65	0.66	0.45
	MTC	0.81	0.83	0.80	0.67	0.65	0.57	0.69	0.42

Table E.7: Instance segmentation results per class on DVS-iOUTLAB dataset (60 ms event time window).

E.3 Consideration of a Further Application Context: Autonomous Driving Recordings

The descriptions of the approach, methods used, and results of this section have previously been published in:

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2023c). Semantic Segmentation on Neuromorphic Vision Sensor Event-Streams Using PointNet++ and UNet Based Processing Approaches. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 178. INSTICC, SciTePress

In order to go beyond the sole application scenario of a stationary long-term monitoring of public spaces, a small selection of the segmentation methods presented in Part III were also tested in another application context. For this purpose, the processing of Dynamic Vision sensor recordings from road traffic was chosen. Methods for semantic segmentation were trained and evaluated based on the "DDD17" dataset [Binas et al., 2017].

E.3.1 DDD17 Dataset Subset

The authors of the Ev-SegNet segmentation approach [Alonso and Murillo, 2019] published with their work a subset of the DDD17 dataset, which is extended by semantic labels. These labels were used in our experiments.

The DDD17 dataset contains sequences of recordings obtained from a moving car in traffic (see Figure E.2). These recordings were made with a DAVIS346B Dynamic Vision Sensor. Therefore, they provide a spatial resolution of 346×260 pixels. However, since the bottom 60 pixel rows included the dashboard of the used car, they were cropped to 346×200 pixels.

The DDD17 dataset subset processed by Alonso and Murillo contains 15,950 sequences for training and 3,890 for testing, each corresponding to a 50 ms section of the event stream. For these sequences, the authors used a conventional 2D segmentation CNN to automatically generate pixel-wise semantic labels based on the grayscale images provided by the DAVIS sensor. In this way, they created ground truth labels for six different classes:

1. `CONSTRUCTION/SKY`,
2. `OBJECTS` (like street signs or light poles),
3. `NATURE` (like trees),
4. `HUMANS`,
5. `VEHICLES` and
6. `STREET`.

The DVS event data was published by Alonso and Murillo in a form that provides only a 2D frame representation of the event stream. Furthermore, the generated labels are also only available in the form of 2D frames. Thus, they are not directly usable for the generation of other event representations.

Therefore, utilizing the *native* DDD17 event stream recordings, we first propagated the labels of the Ev-SegNet data subset back to the original event stream from DDD17. For each 50 ms of the event stream the corresponding 2D label was transferred to all underlying events. This results in annotations per-event in the form of $(x, y, t, p, \text{label})$.

PointNet++(4096, 5L)	SA(2048, 17.3, [32, 32, 64]) →
PointNet++(8192, 5L)	SA(4096, 17.3, [32, 32, 64]) →
followed by	SA(1024, 34.6, [64, 64, 128]) → SA(256, 69.2, [128,128,256]) → SA(64, 103.8, [256,256,512]) → SA(16, 138.4, [512,512,1024]) → FP([256, 256]) → FP([256, 128]) → FP([256, 256]) → FP([256, 128]) → FP([128, 128, 128, 128, 6])

Table E.8: PointNet++ configuration summary (compare to syntax used in [Qi et al., 2017b]).

E.3.2 Experiments and Evaluation

The methods discussed in Chapter 7 were selected for further evaluation in this application context. This involves the semantic segmentation of the event stream using UNet-architectures and PointNet++. Extending the work of Section 7.2, the following describes the experiments performed on the DDD17 dataset.

Network Training

The network layer configurations and training hyperparameters utilized in the DDD17 experiments were as follows:

PointNet++ Processing Network:

For the DDD17 data, due to the larger spatial input dimension (346×200 pixels vs. 192×128 pixels per processed region in DVS-OUTLAB), we adjusted the network configuration to account for the resulting higher event count. Additionally, we trained and tested two PointNet++ configurations with previous subsampling to 8192 and 4096 events, respectively.

The specific PointNet++ configurations used are summarized in Table E.8.

UNet Processing Network:

Apart from the number of classes and the spatial resolution, the parameter configuration remained unchanged compared to the DVS-OUTLAB experiments.

Evaluation Metrics

In addition to the logic described in Section 6.3.2 and Section 7.2, the following simple post-processing was performed before evaluation:

If a non-void class prediction is made, but there is no event present, the prediction is interpreted as void and ignored. If a void class prediction is made, but an event is present, this prediction is reinterpreted and considered as the dominant class for evaluation. This class is `CONSTRUCTION/SKY` for the DDD17 data.

Results and Comparison

The summary of results for the subset of labeled DDD17 dataset sequences is given in Table E.9. On this dataset, PointNet++ processing achieves weaker results in contrast to the UNet variations and the dataset authors' Ev-SegNet reference.

Network	CONSTRUCTION	OBJECTS	NATURE	HUMANS	VEHICLES	STREET	Macro-Avg	Weighted-Avg
(a) Ev-SegNet baseline [Alonso and Murillo, 2019], metric recalculated to match proposed evaluation								
Ev-SegNet	**0.916**	0.229	0.712	**0.670**	0.850	**0.727**	**0.696**	**0.876**
(b) PointNet++ results								
PNet(8192, 5L)	0.842	0.088	0.516	0.398	0.743	0.619	0.534	0.771
PNet(4092, 5L)	0.840	0.103	0.521	0.464	0.748	0.600	0.546	0.766
(c) Unfiltered UNet results								
UNet 2D	0.886	0.266	0.686	0.577	0.835	0.703	0.659	0.829
UNet 2D 64ch	0.895	0.285	0.723	0.533	0.849	0.723	0.668	0.843
UNet 3D Voxel	0.898	**0.301**	**0.729**	0.572	0.847	0.719	0.678	0.843
(d) Spatio-temporal filtered (time 10ms) UNet results								
UNet 2D	0.882	0.265	0.673	0.557	0.846	0.660	0.647	0.826
UNet 2D 64ch	0.896	0.289	0.713	0.590	**0.862**	0.681	0.672	0.846
UNet 3D Voxel	0.895	0.296	0.713	0.568	0.858	0.680	0.668	0.843

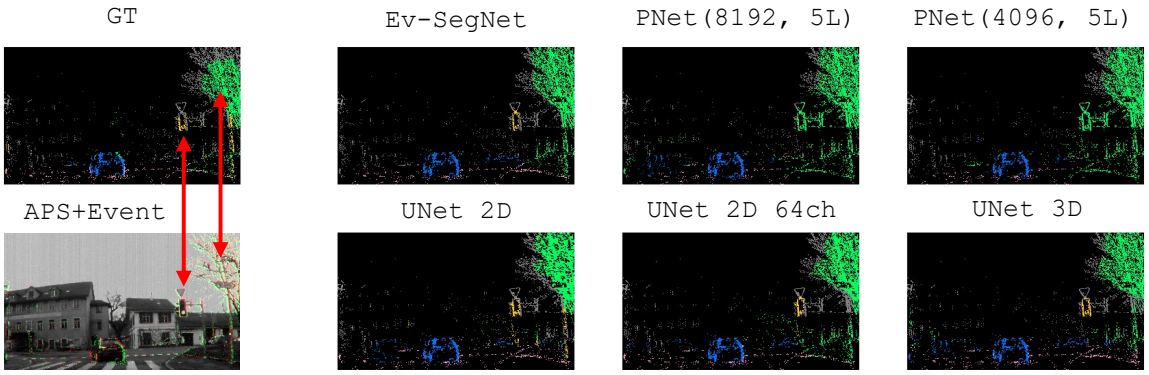
Table E.9: Results on subset of DDD17 dataset.

There is a noticeable difference in the results of the OBJECTS class for the PointNet++ processing. This class of the dataset contains for example lampposts, street signs or traffic lights.

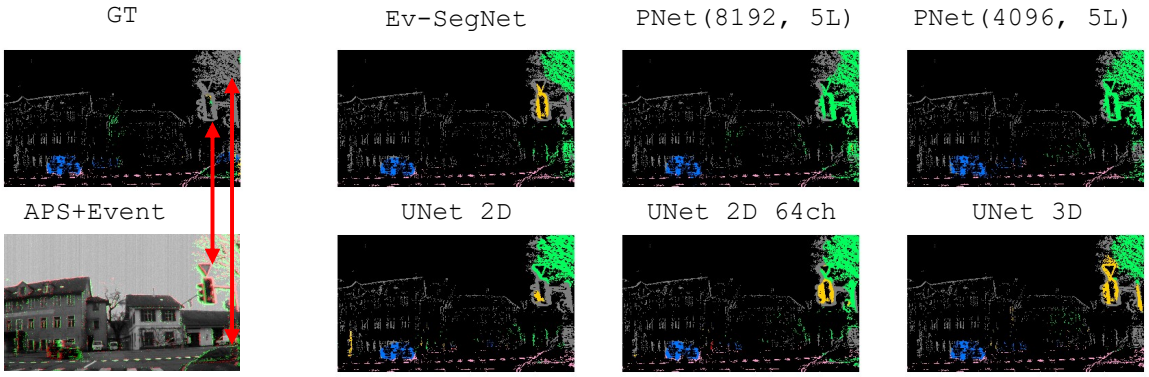
Although the PointNet++ configurations used here were adjusted in the number of points to be considered in the input cloud and the first Set Abstraction layer, as well as in the number of layers themselves, this suggests that such fine details were not fully captured. Due to the high number of triggered events in the autonomous driving context of this dataset (compared to the static sensor in DVS-OUTLAB monitoring) and the larger spatial input (346×200 pixels vs. 192×128 pixels), the encoder/decoder approach of UNet-based processing is preferable. The PointNet++ processing relies on considering sufficiently representative events selected by Farthest Point Sampling and building a corresponding neighborhood, which is difficult in this situation.

However, when considering the overall results on the DDD17 subset, the given quality of the ground truth label must be taken into account. These labels were generated by Alonso and Murillo through an automatic processing. Out of a total of nearly 12 hours of material from the DDD17 dataset, about 15 minutes were labeled in this way, and the ground truth (GT) labels obtained are not completely accurate and consistent over time.

Figure E.2 shows an example of included artifacts in the GT labels using two examples that are separated by only a short period of time. The annotations of the included traffic sign and the tree (marked by red arrows) are different.



(a) Ground truth vs. prediction at t_i



(b) Ground truth vs. prediction at t_i + 1.25 seconds

CONSTRUCTION or SKY | OBJECTS | NATURE | HUMAN | VEHICLE | STREET

Figure E.2: Visualization of GT labeling quality of DDD17 subset from [Alonso and Murillo, 2019] and predictions of trained networks. Note the inconsistent GT labeling of the marked traffic sign and trees between the timestamps shown (best viewed digitally zoomed; from [**Bolten** et al., 2023c]).

List of Acronyms

ADC	Analog-to-Digital Converter.
AER	Address Event Representation.
AI	Artificial Intelligence.
API	Application Programming Interface.
APS	Active Pixel Sensor.
ASA	Acrylnitril-Styrol-Acrylat 3D printing material.
BA	Background Activity.
CAD	Computer-Aided Design.
CCTV	Closed Circuit Television.
CIS	CMOS Image Sensor.
CMOS	Complementary Metal-Oxide-Semiconductor.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
DVS	Dynamic Vision Sensor.
EEPROM	Electrically Erasable Programmable Read-Only Memory.
EFRE	Europäischer Fonds für regionale Entwicklung (European Regional Development Fund).
FIFO	First In, First Out.
FN	False Negative.
FoV	Field of View.
FP	
	FP False Positive.
	FP Feature Propagation layer.
FPGA	Field Programmable Gate Array.
FPS	
	FPS Farthest Point Sampling.
	fps Frames per second.
GPU	Graphics Processing Unit.
GT	Ground Truth.
HDR	High Dynamic Range.
HSV	Hue Saturation Value.
IMU	Inertial Measurement Unit.
IoU	Intersection over Union.
IR	Infrared.

JSON JavaScript Object Notation.

KDE Kernel Density Estimation.

Lidar Light detection and ranging.
LoD Level of Detail.
LSTM Long Short-Term Memory.
LWIR Longwave Infrared.

mAP Mean Average Precision.
Meps Mega events per second.
MLP Multilayer Perceptron.
MOT Multi-Object Tracking.
MPPT Maximum Power Point Tracking.
ms Millisecond.
MTC Merged-Three-Channel event representation.

NIR Near Infrared.
NTP Network Time Protocol.

PCB Printed Circuit Board.
PID Proportional–Integral–Derivative controller.
PoI Patch-of-Interest.
PRE Percentage of Remaining Events.
PWM Pulse-Width Modulation.

Radar Radio detection and ranging.
ReLU Rectified Linear Unit.
RGB Red Green Blue.
RoI
- **aRoI** Adaptive Region-of-Interest.
- **RoI** Region-of-Interest.

RTC Real Time Clock.

SA Set Abstraction layer.
SDK Software Development Kit.
SDRAM Synchronous Dynamic Random Access Memory.
SFM Structure From Motion.
SGD Stochastic Gradient Descent.
SNN Spiking Neural Network.
SNR Signal-to-Noise Ratio.
SPDT Single Pole, Double Throw relay.
SSD
- **SSD** Single-Shot Detector.
- **SSD** Solid-State-Drive.

TDP Thermal Design Power.
TN True Negative.
TP True Positive.

USB Universal Serial Bus.
UV Ultraviolet light.

VDC Volts of Direct Current.
VLSI Very Large Scale Integrated Circuits.

List of Publications

The following papers were published in the course of this work:

Datasets

- **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE

- **Bolten**, T., Neumann, C., Pohle-Fröhlich, R., and Tönnies, K. (2023a). N-MuPeTS: Event Camera Dataset for Multi-Person Tracking and Instance Segmentation. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 290 – 300. INSTICC, SciTePress

Semantic and Instance Segmentation

- **Bolten**, T., Lentzen, F., Pohle-Fröhlich, R., and Tönnies, K. (2022a). Evaluation of Deep Learning based 3D-Point-Cloud Processing Techniques for Semantic Segmentation of Neuromorphic Vision Sensor Event-streams. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 179. INSTICC, SciTePress

- **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2023c). Semantic Segmentation on Neuromorphic Vision Sensor Event-Streams Using PointNet++ and UNet Based Processing Approaches. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 178. INSTICC, SciTePress

- **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2024). Instance Segmentation of Event Camera Streams in Outdoor Monitoring Scenarios. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 452 – 463. INSTICC, SciTePress

Application Scenario: Long-Term Monitoring

- **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2023b). Semantic Scene Filtering for Event Cameras in Long-Term Outdoor Monitoring Scenarios. In Bebis, G. et al., editors, *18th International Symposium on Visual Computing (ISVC), Advances in Visual Computing*, volume 14362 of *Lecture Notes in Computer Science*, pages 79 – 92, Cham. Springer Nature Switzerland

- **Bolten**, T., Pohle-Fröhlich, R., Volker, D., Brück, C., Beucker, N., and Hirsch, H. (2022b). Visualization of Activity Data from a Sensor-based Long-term Monitoring Study at a Playground. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAPP)*, pages 146 – 155. INSTICC, SciTePress

Further Publications

The following publications are related to Dynamic Vision Sensors and were co-authored during the course of this dissertation. However, they are outside the main scope of the work:

- **Bolten**, T., Pohle-Fröhlich, R., and Tönnies, K. (2019). Application of Hierarchical Clustering for Object Tracking with a Dynamic Vision Sensor. In Rodrigues, J. et al., editors, *International Conference on Computational Science (ICCS), Computational Science*, volume 11540 of *Lecture Notes in Computer Science*, pages 164 – 176, Cham. Springer

- Pohle-Fröhlich, R. and **Bolten**, T. (2023). Concept Study for Dynamic Vision Sensor Based Insect Monitoring. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 411 – 418. INSTICC, SciTePress

- Pohle-Fröhlich, R., Gebler, C., and **Bolten**, T. (2024). Stereo-Event-Camera-Technique for Insect Monitoring. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 375 – 384. INSTICC, SciTePress

Bibliography

Abdulla, W. (2017). Mask R-CNN for Object Detection and Instance Segmentation on Keras and Tensorflow. `https://github.com/matterport/Mask_RCNN`.

Acin, L., Jacob, P., Simon-Chane, C., and Histace, A. (2023). VK-SITS: Variable Kernel Speed Invariant Time Surface for Event-Based Recognition. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5 (VISAPP)*, pages 754 – 761. INSTICC, SciTePress.

Afshar, S., Hamilton, T. J., Tapson, J., van Schaik, A., and Cohen, G. (2019). Investigation of Event-Based Surfaces for High-Speed Detection, Unsupervised Feature Extraction, and Object Recognition. *Frontiers in Neuroscience*, 12(1047).

Alonso, I. and Murillo, A. C. (2019). EV-SegNet: Semantic Segmentation for Event-Based Cameras. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1624 – 1633. IEEE.

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., and Modha, D. (2017). A Low Power, Fully Event-Based Gesture Recognition System. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388 – 7397. IEEE. `https://research.ibm.com/interactive/dvsgesture/`.

Baldwin, R. W., Almatrafi, M., Asari, V., and Hirakawa, K. (2020). Event Probability Mask (EPM) and Event Denoising Convolutional Neural Network (EDnCNN) for Neuromorphic Cameras. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1698 – 1707. IEEE.

Bardow, P., Davison, A. J., and Leutenegger, S. (2016). Simultaneous Optical Flow and Intensity Estimation from an Event Camera. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 884 – 892. IEEE.

Barranco, F., Fermuller, C., and Ros, E. (2018). Real-Time Clustering and Multi-Target Tracking Using Event-Based Sensors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5764 – 5769. IEEE.

Barranco, F., Teo, C. L., Fermuller, C., and Aloimonos, Y. (2015). Contour Detection and Characterization for Asynchronous Event Sensors. In *2015 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 486 – 494. IEEE.

Berner, R. and Delbruck, T. (2011). Event-Based Pixel Sensitive to Changes of Color and Brightness. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(7):1581 – 1590.

Berthelon, X., Chenegros, G., Finateu, T., Ieng, S.-H., and Benosman, R. (2018). Effects of Cooling on the SNR and Contrast Detection of a Low-Light Event-Based Camera. *IEEE Transactions on Biomedical Circuits and Systems*, 12(6):1467 – 1474.

Bi, Y. and Andreopoulos, Y. (2017). PIX2NVS: Parameterized Conversion of Pixel-domain Video Frames to Neuromorphic Vision Streams. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1990 – 1994. IEEE.

Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., and Andreopoulos, Y. (2019). Graph-Based Object Classification for Neuromorphic Vision Sensing. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 491 – 501. IEEE.

Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., and Andreopoulos, Y. (2020). Graph-Based Spatio-Temporal Feature Learning for Neuromorphic Vision Sensing. *IEEE Transactions on Image Processing*, 29:9084 – 9098.

Binas, J., Neil, D., Liu, S.-C., and Delbruck, T. (2017). DDD17: End-To-End DAVIS Driving Dataset. arXiv preprint, arXiv:1711.01458. Accepted paper at International Conference on Machine Learning (ICML), Workshop on Machine Learning for Autonomous Vehicles (MLAV).

Bisulco, A., Cladera Ojeda, F., Isler, V., and Lee, D. D. (2020). Near-Chip Dynamic Vision Filtering for Low-Bandwidth Pedestrian Detection. In *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 234 – 239. IEEE.

Biswas, S. D., Kosta, A., Liyanagedera, C., Apolinario, M., and Roy, K. (2024). HALSIE: Hybrid Approach to Learning Segmentation by Simultaneously Exploiting Image and Event Modalities. In *2024 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 5952 – 5962. IEEE.

Bolleter, J. (2017). Counter intelligence: evaluating Wi-Fi tracking data for augmenting conventional public space–public life surveys. *Australian Planner*, 54(2):134 – 144.

Bolten, T., Lentzen, F., Pohle-Fröhlich, R., and Tönnies, K. (2022a). Evaluation of Deep Learning based 3D-Point-Cloud Processing Techniques for Semantic Segmentation of Neuromorphic Vision Sensor Event-streams. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 179. INSTICC, SciTePress.

Bolten, T., Neumann, C., Pohle-Fröhlich, R., and Tönnies, K. (2023a). N-MuPeTS: Event Camera Dataset for Multi-Person Tracking and Instance Segmentation. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 290 – 300. INSTICC, SciTePress.

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2019). Application of Hierarchical Clustering for Object Tracking with a Dynamic Vision Sensor. In Rodrigues, J. et al., editors, *International Conference on Computational Science (ICCS), Computational Science*, volume 11540 of *Lecture Notes in Computer Science*, pages 164 – 176, Cham. Springer.

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2021). DVS-OUTLAB: A Neuromorphic Event-Based Long Time Monitoring Dataset for Real-World Outdoor Scenarios. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1348 – 1357. IEEE.

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2023b). Semantic Scene Filtering for Event Cameras in Long-Term Outdoor Monitoring Scenarios. In Bebis, G. et al., editors, *18th International Symposium on Visual Computing (ISVC), Advances in Visual Computing*, volume 14362 of *Lecture Notes in Computer Science*, pages 79 – 92, Cham. Springer Nature Switzerland.

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2023c). Semantic Segmentation on Neuromorphic Vision Sensor Event-Streams Using PointNet++ and UNet Based Processing Approaches. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 168 – 178. INSTICC, SciTePress.

Bolten, T., Pohle-Fröhlich, R., and Tönnies, K. (2024). Instance Segmentation of Event Camera Streams in Outdoor Monitoring Scenarios. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 452 – 463. INSTICC, SciTePress.

Bolten, T., Pohle-Fröhlich, R., Volker, D., Brück, C., Beucker, N., and Hirsch, H. (2022b). Visualization of Activity Data from a Sensor-based Long-term Monitoring Study at a Playground. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAPP)*, pages 146 – 155. INSTICC, SciTePress.

Boretti, C., Bich, P., Pareschi, F., Prono, L., Rovatti, R., and Setti, G. (2023). PEDRo: an Event-based Dataset for Person Detection in Robotics. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4065 – 4070. IEEE.

Brandli, C., Berner, R., Yang, M., Liu, S.-C., and Delbruck, T. (2014). A 240×180 130 dB $3\,\mu$s Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333 – 2341.

CelePixel (2018). *CeleX-IV Datasheet*. CelePixel Technology Co., Ltd.

Chaney, K., Cladera, F., Wang, Z., Bisulco, A., Hsieh, M. A., Korpela, C., Kumar, V., Taylor, C. J., and Daniilidis, K. (2023). M3ED: Multi-Robot, Multi-Sensor, Multi-Environment Event Dataset. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4016 – 4023. IEEE.

Chen, G., Cao, H., Conradt, J., Tang, H., Rohrbein, F., and Knoll, A. (2020a). Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception. *IEEE Signal Processing Magazine*, 37(4):34 – 49.

Chen, G., Cao, H., Ye, C., Zhang, Z., Liu, X., Mo, X., Qu, Z., Conradt, J., Röhrbein, F., and Knoll, A. (2019a). Multi-Cue Event Information Fusion for Pedestrian Detection With Neuromorphic Vision Sensors. *Frontiers in Neurorobotics*, 13:10.

Chen, J., Meng, J., Wang, X., and Yuan, J. (2020b). Dynamic Graph CNN for Event-Camera Based Gesture Recognition. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1 – 5. IEEE.

Chen, L., Li, X., Fan, D., Cheng, M., Wang, K., and Lu, S. (2019b). LSANet: Feature Learning on Point Sets by Local Spatial Aware Layer. arXiv preprint, arXiv:1905.05442.

Chen, S., Fang, J., Zhang, Q., Liu, W., and Wang, X. (2021). Hierarchical Aggregation for 3D Instance Segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15467 – 15476. IEEE.

Chen, S. and Guo, M. (2019). Live Demonstration: CeleX-V: A 1M Pixel Multi-Mode Event-Based Sensor. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1682 – 1683. IEEE.

Chen, S., Tang, W., Zhang, X., and Culurciello, E. (2012). A 64 × 64 Pixels UWB Wireless Temporal-Difference Digital Image Sensor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(12):2232 – 2240.

Cheng, W., Luo, H., Yang, W., Yu, L., Chen, S., and Li, W. (2019). DET: A High-Resolution DVS Dataset for Lane Extraction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1666 – 1675. IEEE.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213 – 3223. IEEE.

Czech, D. and Orchard, G. (2016). Evaluating Noise Filtering for Event-based Asynchronous Change Detection Image Sensors. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 19 – 24. IEEE.

Damien, J., Hubert, K., and Frederic, C. (2019). Convolutional Neural Network for Detection and Classification with Event-based Data. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5 (VISAPP)*, pages 200 – 208. INSTICC, SciTePress.

Darmont, A. (2013). *High Dynamic Range Imaging: Sensors and Architectures*. SPIE Press, Bellingham, WA.

De Tournemire, P., Nitti, D., Perot, E., Migliore, D., and Sironi, A. (2020). A Large Scale Event-based Detection Dataset for Automotive. arXiv preprint, arXiv:2001.08499.

Delbruck, T. (2008). Frame-free dynamic digital vision. In *Proceedings of Intl. Symposium on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pages 21 – 26.

Delmerico, J., Cieslewski, T., Rebecq, H., Faessler, M., and Scaramuzza, D. (2019). Are We Ready for Autonomous Drone Racing? the UZH-FPV Drone Racing Dataset. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6713 – 6719. IEEE.

Detlefsen, N. S., Borovec, J., Schock, J., Jha, A. H., Koker, T., Liello, L. D., Stancl, D., Quan, C., Grechkin, M., and Falcon, W. (2022). TorchMetrics - Measuring Reproducibility in PyTorch. *Journal of Open Source Software*, 7(70):4101. `https://github.com/Lightning-AI/torchmetrics`.

Dorn, C., Dasari, S., Yang, Y., Kenyon, G., Welch, P., and Mascareñas, D. (2017). Efficient Full-Field Operational Modal Analysis Using Neuromorphic Event-Based Imaging. In Harvie, J. M. and Baqersad, J., editors, *Shock & Vibration, Aircraft/Aerospace, Energy Harvesting, Acoustics & Optics, Volume 9*, pages 97 – 103. Springer International Publishing, Cham.

Du, B., Li, W., Wang, Z., Xu, M., Gao, T., Li, J., and Wen, H. (2021). Event Encryption for Neuromorphic Vision Sensors: Framework, Algorithm, and Evaluation. *Sensors*, 21(13):4320.

Eckardt, F. (2014). *Stadtforschung: Gegenstand und Methoden*. Springer Fachmedien Wiesbaden.

Eghteabs, C., Ritter, B., Parikh, A., and Lischke, L. (2017). Effects of Color and Threshold on User Perception of Heat Maps. In *Proceedings of the 2017 ACM Conference Companion Publication on Designing Interactive Systems*, pages 275 – 279. ACM.

Engelmann, F., Bokeloh, M., Fathi, A., Leibe, B., and Niessner, M. (2020). 3D-MPA: Multi-Proposal Aggregation for 3D Semantic Instance Segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9028 – 9037. IEEE.

Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *2004 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 178 – 187. IEEE.

Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., Mostafalu, P., Brady, F., Chotard, L., LeGoff, F., Takahashi, H., Wakabayashi, H., Oike, Y., and Posch, C. (2020). A 1280×720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with $4.86\,\mu$m Pixels, 1.066 GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline. In *2020 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 112 – 114. IEEE.

Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., and Scaramuzza, D. (2022). Event-based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154 – 180.

Gallego, G., Rebecq, H., and Scaramuzza, D. (2018). A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3867 – 3876. IEEE.

Gallego, G. and Scaramuzza, D. (2017). Accurate Angular Velocity Estimation With an Event Camera. *IEEE Robotics and Automation Letters*, 2(2):632 – 639.

García, G. P., Camilleri, P., Liu, Q., and Furber, S. (2016). pyDVS: An extensible, real-time Dynamic Vision Sensor emulator using off-the-shelf hardware. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1 – 7. IEEE.

Gehl, J. and Svarre, B. (2013). *How to Study Public Life*. Island Press, Washington, Covelo, London.

Gehrig, D., Gehrig, M., Hidalgo-Carrio, J., and Scaramuzza, D. (2020). Video to Events: Recycling Video Datasets for Event Cameras. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3583 – 3592. IEEE.

Gehrig, D., Loquercio, A., Derpanis, K., and Scaramuzza, D. (2019). End-to-End Learning of Representations for Asynchronous Event-Based Data. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5632 – 5642. IEEE.

Gehrig, D. and Scaramuzza, D. (2022). Are High-Resolution Event Cameras Really Needed? arXiv preprint, arXiv:2203.14672.

Gehrig, M., Aarents, W., Gehrig, D., and Scaramuzza, D. (2021). DSEC: A Stereo Event Camera Dataset for Driving Scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947 – 4954.

Ghasemzadeh, M. and Shouraki, S. B. (2023). Semantic Segmentation Using Events and Combination of Events and Frames. In Ghatee, M. and Hashemi, S. M., editors, *Artificial Intelligence and Smart Vehicles. International Conference on Artificial Intelligence and Smart Vehicles (ICAISV)*, volume 1883 of *Communications in Computer and Information Science*, pages 167 – 181, Cham. Springer Nature Switzerland.

Graça, R. and Delbruck, T. (2021). Unraveling the paradox of intensity-dependent DVS pixel noise. arXiv preprint, arXiv:2109.08640. accepted paper at International Image Sensors Workshop (IISW), Online event.

Graça, R., McReynolds, B., and Delbruck, T. (2023a). Optimal biasing and physical limits of DVS event noise. arXiv preprint, arXiv:2304.04019. accepted paper at International Image Sensors Workshop (IISW), Scotland, United Kingdom.

Graça, R., McReynolds, B., and Delbruck, T. (2023b). Shining light on the DVS pixel: A tutorial and discussion about biasing and optimization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4045 – 4053. IEEE.

Graham, B., Engelcke, M., and van der Maaten, L. (2018). 3D Semantic Segmentation With Submanifold Sparse Convolutional Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9224 – 9232. IEEE.

Graham, B. and van der Maaten, L. (2017). Submanifold Sparse Convolutional Networks. arXiv preprint, arXiv:1706.01307.

Guo, M., Chen, S., Gao, Z., Yang, W., Bartkovjak, P., Qin, Q., Hu, X., Zhou, D., Uchiyama, M., Fukuoka, S., Xu, C., Ebihara, H., Wang, A., Jiang, P., Jiang, B., Mu, B., Chen, H., Yang, J., Dai, T. J., Suess, A., and Kudo, Y. (2023). A 3-Wafer-Stacked Hybrid 15 MPixel CIS + 1 MPixel EVS with 4.6 GEvent/s Readout, In-Pixel TDC and On-Chip ISP and ESP Function. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 90 – 92. IEEE.

Guo, M., Ding, R., and Chen, S. (2016). Live Demonstration: A Dynamic Vision Sensor with direct logarithmic output and full-frame picture-on-demand. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 456 – 456. IEEE.

Guo, M., Huang, J., and Chen, S. (2017). Live Demonstration: A 768×640 pixels 200 Meps Dynamic Vision Sensor. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 618 – 618. IEEE.

Guo, S. and Delbruck, T. (2023). Low Cost and Latency Event Camera Background Activity Denoising. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):785 – 795.

Guo, S., Kang, Z., Wang, L., Li, S., and Xu, W. (2020a). HashHeat: An O(C) Complexity Hashing-based Filter for Dynamic Vision Sensor. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 452 – 457. IEEE.

Guo, S., Wang, L., Chen, X., Zhang, L., Kang, Z., and Xu, W. (2020b). SeqXFilter: A Memory-efficient Denoising Filter for Dynamic Vision Sensors. arXiv preprint, arXiv:2006.01687.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2021). Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338 – 4364.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *2017 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2980 – 2988. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770 – 778. IEEE.

Hossain, M., Leminen, S., and Westerlund, M. (2019). A systematic review of living lab literature. *Journal of Cleaner Production*, 213:976 – 988.

Hu, Y., Binas, J., Neil, D., Liu, S.-C., and Delbruck, T. (2020). DDD20 End-to-End Event Camera Driving Dataset: Fusing Frames and Events with Deep Learning for Improved Steering Prediction. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1 – 6. IEEE.

Hu, Y., Liu, H., Pfeiffer, M., and Delbruck, T. (2016). DVS Benchmark Datasets for Object Tracking, Action Recognition, and Object Recognition. *Frontiers in Neuroscience*, 10:405.

Hu, Y., Liu, S.-C., and Delbruck, T. (2021). v2e: From Video Frames to Realistic DVS Events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1312 – 1321. IEEE.

Huang, J., Guo, M., and Chen, S. (2017). A Dynamic Vision Sensor with Direct Logarithmic Output and Full-frame Picture-On-Demand. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1142 – 1146. IEEE.

Huang, X., Kachole, S., Ayyad, A., Naeini, F. B., Makris, D., and Zweiri, Y. (2024). A neuromorphic dataset for tabletop object segmentation in indoor cluttered environment. *Scientific Data*, 11(1):127.

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90 – 95.

Islam, M. Z., Islam, M., and Rana, M. S. (2015). Problem Analysis of Multiple Object Tracking System: A Critical Review. *International Journal of Advanced Research in Computer and Communication Engineering*, 4:374 – 377.

Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.-W., and Jia, J. (2020). PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4866 – 4875. IEEE.

Jiang, Z., Xia, P., Huang, K., Stechele, W., Chen, G., Bing, Z., and Knoll, A. (2019). Mixed Frame-/Event-Driven Fast Pedestrian Detection. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8332 – 8338. IEEE.

Jin, C.-B., Do, T. D., Liu, M., and Kim, H. (2018). Real-Time Action Detection in Video Surveillance using a Sub-Action Descriptor with Multi-Convolutional Neural Networks. *Journal of Institute of Control, Robotics and Systems*, 24:298 – 308.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). YOLO by Ultralytics. `https://github.com/ultralytics/ultralytics`, v8.0.0, AGPL-3.0.

Kachole, S., Alkendi, Y., Baghaei Naeini, F., Makris, D., and Zweiri, Y. (2023). Asynchronous Events-based Panoptic Segmentation using Graph Mixer Neural Network. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4083 – 4092. IEEE.

Kachole, S., Huang, X., Naeini, F. B., Muthusamy, R., Makris, D., and Zweiri, Y. (2024). Bimodal SegNet: Fused instance segmentation using events and RGB frames. *Pattern Recognition*, 149:110215.

Khodamoradi, A. and Kastner, R. (2018). O(N)-Space Spatiotemporal Filter for Reducing Noise in Neuromorphic Vision Sensors. *IEEE Transactions on Emerging Topics in Computing*, 9:15 – 23.

Kim, J., Bae, J., Park, G., Zhang, D., and Kim, Y. M. (2021). N-ImageNet: Towards Robust, Fine-Grained Object Recognition with Event Cameras. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2126 – 2136. IEEE.

Kirillov, A., He, K., Girshick, R., Rother, C., and Dollar, P. (2019). Panoptic Segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9396 – 9405. IEEE.

Kirkland, P., Manna, D., Vicente, A., and Di Caterina, G. (2022). Unsupervised Spiking Instance Segmentation on Event Data Using STDP Features. *IEEE Transactions on Computers*, 71(11):2728 – 2739.

Kogler, J. (2016). *Design and Evaluation of Stereo Matching Techniques for Silicon Retina Cameras*. PhD thesis, TU Wien. `https://dx.doi.org/10.34726/HSS.2016.36281`.

Komarichev, A., Zhong, Z., and Hua, J. (2019). A-CNN: Annularly Convolutional Neural Networks on Point Clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7421 – 7430. IEEE.

Kong, D., Fang, Z., Hou, K., Li, H., Jiang, J., Coleman, S., and Kerr, D. (2022). Event-VPR: End-to-End Weakly Supervised Deep Network Architecture for Visual Place Recognition Using Event-Based Vision Sensor. *IEEE Transactions on Instrumentation and Measurement*, 71:1 – 18.

Lagorce, X., Meyer, C., Ieng, S.-H., Filliat, D., and Benosman, R. (2015). Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1710 – 1720.

Lagorce, X., Orchard, G., Galluppi, F., Shi, B. E., and Benosman, R. B. (2017). HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346 – 1359.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, 86(11):2278 – 2324.

Li, C., Brandli, C., Berner, R., Liu, H., Yang, M., Liu, S.-C., and Delbruck, T. (2015). Design of an RGBW Color VGA Rolling and Global Shutter Dynamic and Active-Pixel Vision Sensor. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 718 – 721. IEEE.

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128×128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566 – 576.

Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2017). Enhanced Deep Residual Networks for Single Image Super-Resolution. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132 – 1140. IEEE.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *European Conference on Computer Vision (ECCV)*, volume 8693 of *Lecture Notes in Computer Science*, pages 740 – 755, Cham. Springer International Publishing.

Linares-Barranco, A., Gomez-Rodriguez, F., Villanueva, V., Longinotti, L., and Delbruck, T. (2015). A USB3.0 FPGA Event-Based Filtering and Tracking Framework for Dynamic Vision Sensors. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2417 – 2420. IEEE.

Lipták, B. G. (2006). *Instrument Engineers' Handbook, Volume Two: Process Control and Optimization*. CRC press.

Litzenberger, M., Posch, C., Bauer, D., Belbachir, A. N., Schon, P., Kohn, B., and Garn, H. (2006). Embedded Vision System for Real-Time Object Tracking using an Asynchronous Transient Vision Sensor. In *2006 IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop (DSPWS)*, pages 173 – 178. IEEE.

Liu, M. and Delbruck, T. (2018). Adaptive Time-Slice Block-Matching Optical Flow Algorithm for Dynamic Vision Sensors. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1 – 12.

Liu, M. and Qian, P. (2021). Automatic Segmentation and Enhancement of Latent Fingerprints Using Deep Nested UNets. *IEEE Transactions on Information Forensics and Security*, 16:1709 – 1719.

Liu, S. and Dragotti, P. L. (2023). Sensing Diversity and Sparsity Models for Event Generation and Video Reconstruction from Events. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12444 – 12458.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *European Conference on Computer Vision (ECCV)*, volume 9905 of *Lecture Notes in Computer Science*, pages 21 – 37, Cham. Springer International Publishing.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., and Kim, T.-K. (2021). Multiple Object Tracking: A Literature Review. *Artificial Intelligence*, 293:103448.

Luo, Z., Branchaud-Charron, F., Lemaire, C., Konrad, J., Li, S., Mishra, A., Achkar, A., Eichel, J., and Jodoin, P.-M. (2018). MIO-TCD: A New Benchmark Dataset for Vehicle Classification and Localization. *IEEE Transactions on Image Processing*, 27(10):5129 – 5141.

Mahowald, M. (1992). *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. PhD thesis, California Institute of Technology. `https://resolver.caltech.edu/CaltechTHESIS:09122011-094355148`.

Maqueda, A. I., Loquercio, A., Gallego, G., Garcia, N., and Scaramuzza, D. (2018). Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5419 – 5427. IEEE.

Marcireau, A., Ieng, S.-H., Simon-Chane, C., and Benosman, R. B. (2018). Event-Based Color Segmentation With a High Dynamic Range Sensor. *Frontiers in Neuroscience*, 12:135.

McGlinchy, J., Johnson, B., Muller, B., Joseph, M., and Diaz, J. (2019). Application of UNet Fully Convolutional Neural Network to Impervious Surface Segmentation in Urban Environment from High Resolution Satellite Imagery. In *2019 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3915 – 3918. IEEE.

McMahon-Crabtree, P. N. and Monet, D. G. (2021). Commercial-off-the-shelf event-based cameras for space surveillance applications. *Applied Optics*, 60(25):G144 – G153.

McReynolds, B. J., Graça, R. P., and Delbruck, T. (2022). Experimental methods to predict dynamic vision sensor event camera performance. *Optical Engineering*, 61(7):074103.

Mead, C. (1989). *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., USA.

Miao, S., Chen, G., Ning, X., Zi, Y., Ren, K., Bing, Z., and Knoll, A. (2019). Neuromorphic Vision Datasets for Pedestrian Detection, Action Recognition, and Fall Detection. *Frontiers in Neurorobotics*, 13:38.

Min, Y., Zhang, Y., Chai, X., and Chen, X. (2020). An Efficient PointLSTM for Point Clouds Based Gesture Recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5760 – 5769. IEEE.

Mishra, A., Ghosh, R., Principe, J. C., Thakor, N. V., and Kukreja, S. L. (2017). A Saccade Based Framework for Real-Time Motion Segmentation Using Event Based Vision Sensors. *Frontiers in Neuroscience*, 11:83.

Mitrokhin, A., Fermüller, C., Parameshwara, C., and Aloimonos, Y. (2018). Event-Based Moving Object Detection and Tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6895 – 6902. IEEE.

Mitrokhin, A., Hua, Z., Fermüller, C., and Aloimonos, Y. (2020). Learning Visual Motion Segmentation Using Event Surfaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14402 – 14411. IEEE.

Moeys, D. P., Corradi, F., Li, C., Bamford, S. A., Longinotti, L., Voigt, F. F., Berry, S., Taverni, G., Helmchen, F., and Delbruck, T. (2018). A Sensitive Dynamic and Active Pixel Vision Sensor for Color or Neural Imaging Applications. *IEEE Transactions on Biomedical Circuits and Systems*, 12(1):123 – 136.

Mohan, S. (2019). Thermal Comparison of FR-4 and Insulated Metal Substrate PCB for GaN Inverter. `https://www.ti.com/lit/an/tida030/tida030.pdf`.

Mondal, A., Shashant, R., Giraldo, J. H., Bouwmans, T., and Chowdhury, A. S. (2021). Moving Object Detection for Event-based Vision using Graph Spectral Clustering. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 876 – 884. IEEE.

Moreland, K. (2009). Diverging Color Maps for Scientific Visualization. In Bebis, G. et al., editors, *International Symposium on Visual Computing (ISVC), Advances in Visual Computing*, volume 5876 of *Lecture Notes in Computer Science*, pages 92 – 103, Berlin, Heidelberg. Springer Berlin Heidelberg.

Moussouri, T. and Roussos, G. (2014). Mobile Sensing, BYOD and Big Data Analytics: New technologies for audience research in museums. *Participations: journal of audience and reception studies*, 11(1):270 – 285.

Mueggler, E., Huber, B., and Scaramuzza, D. (2014). Event-based, 6-DOF Pose Tracking for High-Speed Maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2761 – 2768. IEEE. `https://www.youtube.com/watch?v=LauQ6LWTkxM&t`.

Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., and Scaramuzza, D. (2017). The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM. *The International Journal of Robotics Research*, 36(2):142 – 149.

Müllner, D. (2013). fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 53(9):1 – 18.

Najibi, M., Lai, G., Kundu, A., Lu, Z., Rathod, V., Funkhouser, T., Pantofaru, C., Ross, D., Davis, L. S., and Fathi, A. (2020). DOPS: Learning to Detect 3D Objects and Predict Their 3D Shapes. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11910 – 11919. IEEE.

Nakamura, J., editor (2005). *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, Boca Raton, FL.

Nozaki, Y. and Delbruck, T. (2017). Temperature and Parasitic Photocurrent Effects in Dynamic Vision Sensors. *IEEE Transactions on Electron Devices*, 64(8):3239 – 3245.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.-C., Lee, J. T., Mukherjee, S., Aggarwal, J. K., Lee, H., Davis, L., Swears, E., Wang, X., Ji, Q., Reddy, K., Shah, M., Vondrick, C., Pirsiavash, H., Ramanan, D., Yuen, J., Torralba, A., Song, B., Fong, A., Roy-Chowdhury, A., and Desai, M. (2011). A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video. In *2011 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3153 – 3160. IEEE.

Ojeda, F. C., Bisulco, A., Kepple, D., Isler, V., and Lee, D. D. (2020). On-Device Event Filtering with Binary Neural Networks for Pedestrian Detection Using Neuromorphic Vision Sensors. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3084 – 3088. IEEE.

Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9:437.

Padala, V., Basu, A., and Orchard, G. (2018). A Noise Filtering Algorithm for Event-Based Asynchronous Change Detection Image Sensors on TrueNorth and Its Implementation on TrueNorth. *Frontiers in Neuroscience*, 12:118.

Paredes-Valles, F. and de Croon, G. C. H. E. (2021). Back to Event Basics: Self-Supervised Learning of Image Reconstruction for Event Cameras via Photometric Constancy. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3445 – 3454. IEEE.

Piątkowska, E., Belbachir, A. N., Schraml, S., and Gelautz, M. (2012). Spatiotemporal Multiple Persons Tracking using Dynamic Vision Sensor. In *2012 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 35 – 40. IEEE.

Pohle-Fröhlich, R., Bohm, A., Ueberholz, P., Korb, M., and Goebbels, S. (2019). Roof Segmentation based on Deep Neural Networks. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 326 – 333. INSTICC, SciTePress.

Pohle-Fröhlich, R. and **Bolten**, T. (2023). Concept Study for Dynamic Vision Sensor Based Insect Monitoring. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4 (VISAPP)*, pages 411 – 418. INSTICC, SciTePress.

Pohle-Fröhlich, R., Gebler, C., and **Bolten**, T. (2024). Stereo-Event-Camera-Technique for Insect Monitoring. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3 (VISAPP)*, pages 375 – 384. INSTICC, SciTePress.

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). 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.

Pranav, M., Zhenggang, L., and Shah Shishir, K. (2020). A Day on Campus - An Anomaly Detection Dataset for Events in a Single Camera. In Ishikawa, H., Liu, C.-L., Pajdla, T., and Shi, J., editors, *Asian Conference on Computer Vision (ACCV)*, volume 12627 of *Lecture Notes in Computer Science*, pages 619 – 635, Cham. Springer International Publishing.

Pérez-Carrasco, J. A., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., and Linares-Barranco, B. (2013). Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing - Application to Feedforward ConvNets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2706 – 2719.

Qi, C. R., Su, H., Kaichun, M., and Guibas, L. J. (2017a). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77 – 85. IEEE.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5105 – 5114. Curran Associates Inc. `https://github.com/charlesq34/pointnet2`.

Rajasekaran, R., Rasool, F., Srivastava, S., Masih, J., and Rajest, S. S. (2020). Heat Maps for Human Group Activity in Academic Blocks. In Haldorai, A., Ramu, A., and Khan, S. A. R., editors, *Business Intelligence for Enterprise Internet of Things*, Communication and Computing, pages 241 – 251. Springer International Publishing, Cham.

Rashed, M. G., Suzuki, R., Yonezawa, T., Lam, A., Kobayashi, Y., and Kuno, Y. (2016). Tracking Visitors in a Real Museum for Behavioral Analysis. In *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 80 – 85. IEEE.

Rebecq, H., Gehrig, D., and Scaramuzza, D. (2018). ESIM: an Open Event Camera Simulator. In Billard, A., Dragan, A., Peters, J., and Morimoto, J., editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 969 – 982. PMLR.

Rebecq, H., Ranftl, R., Koltun, V., and Scaramuzza, D. (2019). Events-To-Video: Bringing Modern Computer Vision to Event Cameras. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3852 – 3861. IEEE.

Rebecq, H., Ranftl, R., Koltun, V., and Scaramuzza, D. (2021). High Speed and High Dynamic Range Video with an Event Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):1964 – 1980.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779 – 788. IEEE.

Reinbacher, C., Graber, G., and Pock, T. (2016). Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation. In Wilson, R. C. et al., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 9.1–9.12. BMVA Press.

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137 – 1149.

Rezaei, M. and Azarmi, M. (2020). DeepSOCIAL: Social Distancing Monitoring and Infection Risk Assessment in COVID-19 Pandemic. *Applied Sciences*, 10(21):7514.

Richardson, I. E. (2010). *The H.264 Advanced Video Compression Standard*. Wiley.

Rodríguez-Gomez, J. P., Eguíluz, A. G., Martínez-de Dios, J. R., and Ollero, A. (2020). Asynchronous Event-Based Clustering and Tracking for Intrusion Monitoring in UAS. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8518 – 8524. IEEE.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234 – 241, Cham. Springer International Publishing.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211 – 252.

Sabater, A., Montesano, L., and Murillo, A. C. (2022). Event Transformer. A sparse-aware solution for efficient event data processing. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2677 – 2686. IEEE.

Scheerlinck, C., Barnes, N., and Mahony, R. (2019). Continuous-Time Intensity Estimation Using Event Cameras. In Jawahar, C., Li, H., Mori, G., and Schindler, K., editors, *Asian Conference on Computer Vision (ACCV)*, volume 11365 of *Lecture Notes in Computer Science*, pages 308 – 324, Cham. Springer International Publishing.

Schmid, C. (2015). Für eine transdisziplinäre Erforschung der urbanen Wirklichkeit. *Archithese*, Heft 02/2015:18 – 25.

Schraml, S. and Belbachir, A. N. (2010). A Spatio-temporal Clustering Method Using Real-time Motion Analysis on Event-based 3D Vision. In *2010 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 57 – 63. IEEE.

Sekikawa, Y., Hara, K., and Saito, H. (2019). EventNet: Asynchronous Recursive Event Processing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3882 – 3891. IEEE.

Selle, K. (2010). *Stadträume im Wandel. Einführung in die Diskussion um eine zentrale Aufgabe der Stadtentwicklung*. Edition Stadt-Entwicklung. Rohn, Detmold. Neuaufl. von: Was ist los mit den öffentlichen Räumen?

Serrano-Gotarredona, T. and Linares-Barranco, B. (2015). Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details. *Frontiers in Neuroscience*, 9:481.

Sharma, R., Saqib, M., Lin, C. T., and Blumenstein, M. (2022). A Survey on Object Instance Segmentation. *SN Computer Science*, 3(6):499.

Silva, S., Santos, B. S., and Madeira, J. (2011). Using color in visualization: A survey. *Computers & Graphics*, 35(2):320 – 333.

Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., and Benosman, R. (2018). HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1731 – 1740. IEEE.

Snyder, C. and Do, M. (2019). STREETS: A Novel Camera Network Dataset for Traffic Flow. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 10242 – 10253. Curran Associates, Inc.

Son, B., Suh, Y., Kim, S., Jung, H., Kim, J.-S., Shin, C., Park, K., Lee, K., Park, J., Woo, J., Roh, Y., Lee, H., Wang, Y., Ovsiannikov, I., and Ryu, H. (2017). A 640 × 480 Dynamic Vision Sensor with a 9 μm Pixel and 300 Meps Address-Event Representation. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66 – 67. IEEE.

Sony (2018). *IMX477 Datasheet*. Sony Semiconductor Solutions Corporation.

Steffen, L., Reichard, D., Weinland, J., Kaiser, J., Roennau, A., and Dillmann, R. (2019). Neuromorphic Stereo Vision: A Survey of Bio-Inspired Sensors and Algorithms. *Frontiers in Neurorobotics*, 13:28.

Stoffregen, T., Gallego, G., Drummond, T., Kleeman, L., and Scaramuzza, D. (2019). Event-Based Motion Segmentation by Motion Compensation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7243 – 7252. IEEE.

Stoffregen, T. and Kleeman, L. (2018). Simultaneous Optical Flow and Segmentation (SOFAS) using Dynamic Vision Sensor. arXiv preprint, arXiv:1805.12326. Accepted paper at 2017 Australasian Conference on Robotics and Automation.

Suh, Y., Choi, S., Ito, M., Kim, J., Lee, Y., Seo, J., Jung, H., Yeo, D.-H., Namgung, S., Bong, J., Yoo, S., Shin, S.-H., Kwon, D., Kang, P., Kim, S., Na, H., Hwang, K., Shin, C., Kim, J.-S., Park, P. K. J., Kim, J., Ryu, H., and Park, Y. (2020). A 1280 × 960 Dynamic Vision Sensor with a 4.95-μm Pixel Pitch and Motion Artifact minimization. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1 – 5. IEEE.

Sun, Z., Messikommer, N., Gehrig, D., and Scaramuzza, D. (2022). ESS: Learning Event-based Semantic Segmentation from Still Images. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T., editors, *European Conference on Computer Vision (ECCV)*, volume 13694 of *Lecture Notes in Computer Science*, pages 341 – 357, Cham. Springer Nature Switzerland.

Taverni, G., Paul Moeys, D., Li, C., Cavaco, C., Motsnyi, V., San Segundo Bello, D., and Delbruck, T. (2018). Front and Back Illuminated Dynamic and Active Pixel Vision Sensors Comparison. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(5):677 – 681.

United Nations (2018). World Urbanization Prospects: The 2018 revision. Department of Economic and Social Affairs: Population Division.

United Nations (2019). World Population Prospects 2019: Highlights, Key findings and advance tables. Department of Economic and Social Affairs: Population Division.

Van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of machine learning research*, 9(86):2579–2605.

Vasco, V., Glover, A., Mueggler, E., Scaramuzza, D., Natale, L., and Bartolozzi, C. (2017). Independent Motion Detection with Event-driven Cameras. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 530 – 536. IEEE.

Vasudevan, A., Negri, P., Linares-Barranco, B., and Serrano-Gotarredona, T. (2020). Introduction and Analysis of an Event-Based Sign Language Dataset. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, pages 675 – 682. IEEE.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B. (2019). MOTS: Multi-Object Tracking and Segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7934 – 7943. IEEE.

Vu, T., Kim, K., Luu, T. M., Nguyen, T., and Yoo, C. D. (2022). SoftGroup for 3D Instance Segmentation on Point Clouds. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2708 – 2717. IEEE. `https://github.com/thangvubk/SoftGroup`.

Wan, J., Xia, M., Huang, Z., Tian, L., Zheng, X., Chang, V., Zhu, Y., and Wang, H. (2021). Event-Based Pedestrian Detection Using Dynamic Vision Sensors. *Electronics*, 10(8):888.

Wang, L., Chae, Y., Yoon, S.-H., Kim, T.-K., and Yoon, K.-J. (2021a). EvDistill: Asynchronous Events To End-Task Learning via Bidirectional Reconstruction-Guided Cross-Modal Knowledge Distillation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 608 – 619. IEEE.

Wang, L., Liu, Z., Zhang, Y., Yang, S., Shi, D., and Zhang, Y. (2022). FusionSeg: Motion Segmentation by Jointly Exploiting Frames and Events. In Khanna, S., Cao, J., Bai, Q., and Xu, G., editors, *Pacific Rim International Conference on Artificial Intelligence (PRICAI): Trends in Artificial Intelligence*, volume 13631 of *Lecture Notes in Computer Science*, pages 267 – 280, Cham. Springer Nature Switzerland.

Wang, Q., Zhang, Y., Yuan, J., and Lu, Y. (2019). Space-Time Event Clouds for Gesture Recognition: From RGB Cameras to Event Cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1826 – 1835. IEEE.

Wang, X., Zhang, R., Kong, T., Li, L., and Shen, C. (2020). SOLOv2: Dynamic and Fast Instance Segmentation. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 17721 – 17732. Curran Associates, Inc.

Wang, Y., Zhang, X., Shen, Y., Du, B., Zhao, G., Cui Lizhen, L. C., and Wen, H. (2021b). Event-Stream Representation for Human Gaits Identification Using Deep Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3436–3449.

Watson, A. B. (2014). A formula for human retinal ganglion cell receptive field density as a function of visual field location. *Journal of Vision*, 14(7):15.

Wei, Y., Liu, H., Xie, T., Ke, Q., and Guo, Y. (2022). Spatial-Temporal Transformer for 3D Point Cloud Sequences. In *2022 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 657 – 666. IEEE.

Whyte, W. H. (1979). New York and Tokyo: A Study in Crowding. *Real Estate Issues*, 4(2):117.

Whyte, W. H. (1980). *The Social Life of Small Urban Spaces*. Conservation Foundation, Washington, D.C.

Wu, W., Qi, Z., and Fuxin, L. (2019a). PointConv: Deep Convolutional Networks on 3D Point Clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613 – 9622. IEEE.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019b). Detectron2. `https://github.com/facebookresearch/detectron2`.

Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., and Luo, P. (2020). PolarMask: Single Shot Instance Segmentation With Polar Representation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12190 – 12199. IEEE.

Xie, E., Wang, W., Ding, M., Zhang, R., and Luo, P. (2022). PolarMask++: Enhanced Polar Representation for Single-Shot Instance Segmentation and Beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5385 – 5400.

Xu, J., Zou, J., and Gao, Z. (2018a). Comment on "Temperature and Parasitic Photocurrent Effects in Dynamic Vision Sensors". *IEEE Transactions on Electron Devices*, 65(7):3081 – 3082.

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. (2018b). SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *European Conference on Computer Vision (ECCV)*, volume 11212 of *Lecture Notes in Computer Science*, pages 90 – 105, Cham. Springer International Publishing.

Xu, Y., Zhou, X., Chen, S., and Li, F. (2019). Deep Learning for Multiple Object Tracking: A Survey. *IET Computer Vision*, 13(4):355 – 368.

Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., and Trigoni, N. (2019). Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 6737 – 6746. Curran Associates, Inc. `https://github.com/Yang7879/3D-BoNet`.

Zhang, J., Liu, H., Yang, K., Hu, X., Liu, R., and Stiefelhagen, R. (2023). CMX: Cross-Modal Fusion for RGB-X Semantic Segmentation With Transformers. *IEEE Transactions on Intelligent Transportation Systems*, 24(12):14679 – 14694.

Zhang, P., Zhu, S., and Lam, E. Y. (2024). Event encryption: rethinking privacy exposure for neuromorphic imaging. *Neuromorphic Computing and Engineering*, 4(1):014002.

Zhao, L. and Tao, W. (2020). JSNet: Joint Instance and Semantic Segmentation of 3D Point Clouds. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12951 – 12958. `https://github.com/dlinzhao/JSNet`.

Zhou, L. and Hansen, C. D. (2015). A Survey of Colormaps in Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(8):2051 – 2069.

Zhou, Y., Gallego, G., Lu, X., Liu, S., and Shen, S. (2021). Event-Based Motion Segmentation With Spatio-Temporal Graph Cuts. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):4868–4880.

Zhu, A. Z. (2019). *Event-Based Algorithms For Geometric Computer Vision*. PhD thesis, Publicly Accessible Penn Dissertations. `https://repository.upenn.edu/edissertations/3566`.

Hilfsmittel

Bei der Erstellung dieser Arbeit wurden Rechtschreib-, Grammatik- und Zeichensetzungsfehler auf Basis des Korrekturwerkzeugs “DeepL Write”[37] geprüft und verbessert.

Ehrenerklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 19. Dezember 2024

Tobias Bolten

[37] `https://www.deepl.com/write`