FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

An asynchronous Data Interface for Event-based Stereo Matching

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computergraphik & Digitale Bildverarbeitung

eingereicht von

Harald Reingruber
Matrikelnummer 0726257

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: A.o.Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Dr. Ahmed Nabil Belbachir

Wien, 07.09.2011

(Unterschrift Verfasser) (Unterschrift Betreuer/in)

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, am 31. August 2011

Harald Reingruber

Contents

Erklärung iii

Abstract vi

Kurzfassung vii

Acknowledgments ix

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Objective of this Work . . . 6
- 1.3 Contribution . . . 6
- 1.4 Scope . . . 7
- 1.5 Outline . . . 7

2 Bio-inspired Computer Vision 9
- 2.1 Human Vision . . . 9
- 2.2 Event-based Vision Sensor . . . 13
- 2.3 Address-Event Representation . . . 17
- 2.4 Space-time Representation . . . 19

3 Stereo Vision 22
- 3.1 Stereo Vision . . . 22
  - 3.1.1 Epipolar Geometry . . . 22
  - 3.1.2 Stereo Processing Pipeline . . . 23
- 3.2 Event-based Stereo Vision . . . 27

4 Asynchronous local Address-Event Buffer 30
- 4.1 Basic Idea . . . 30
- 4.2 Concept . . . 31
- 4.3 Implementation . . . 32
- 4.4 Output . . . 35

5 Evaluation and Results 38

5.1 Stereo Rig Calibration 38
5.2 Acquiring Ground Truth Data 41
5.2.1 Structured Light 41
5.2.2 Calibrated Object 44
5.2.3 Rectangular Object 45
5.3 Evaluation 47
5.4 Results 49

6 Conclusion and Outlook **51**
6.1 Conclusion 51
6.2 Possible follow up Research 52
6.3 Personal Experience 52

Bibliography **54**

Abstract

Computer vision systems operate by capturing sequences of frames which are processed frame by frame and in most cases pixel by pixel. The human brain does not operate frame-wise. Each ganglion cell sends spikes to the visual cortex when its activity level reaches a certain threshold. During the last decade researchers have developed bio-inspired sensors which mimic human visual sensing. In event-based vision, the neuronal spikes are represented by events that are generated when the relative change of light intensity exceeds a certain threshold. The output of an event-based vision sensor is a stream of events, generated by autonomous pixels, which fire them as soon as they occur and do not wait for an artificial, periodic frame time. Additionally, redundant information like static image areas is suppressed, hence only data from dynamic areas is generated.

Current Stereo Vision concepts based on Address-Event Representation (AER) are abandoning the advantages of this asynchronous data representation by buffering incoming events into artificially introduced pseudo-frames. One aim of this thesis is to design an asynchronous data interface for event-based stereo matching which preserves these advantages. The second goal is to make this data interface applicable to motion at different velocities in the sensor's field of view.

A ground-truth comparison between the state of the art approach and the one presented by this work, has been performed in order to analyze feasibility and improvements by the presented approach.

After analyzing different methods for acquiring comparable ground-truth data from dynamic scenes, it turned out that capturing ground-truth data from scenes containing moving test objects with complex geometry remains a topic for follow-up research, as its extent exceeds the scope of this work. An evaluation approach using simple test objects, finally led to a feasible evaluation. Due to the test object simplifications, the evaluation was not able to reveal improvements in stereo matching accuracy regarding varying object movement velocities, as it was not able to sufficiently stress the drawbacks of the state of the art solution.

Nevertheless, the principal aim, an asynchronous data interface, was achieved and as visible in the evaluation result, without negative impact on stereo matching accuracy.

Kurzfassung

Computer Vision Systeme zeichnen Bildsequenzen auf, welche Bild für Bild und Pixel für Pixel verarbeitet werden. Das menschliche Gehirn arbeitet andererseits nicht bildweise. Jede Ganglionzelle sendet autonom Spikes zum visuellen Cortex, wenn ihre Aktivität einen Schwellwert erreicht. Im letzten Jahrzehnt entwickelten Wissenschaftler bio-inspirierte Sensoren, welche das menschliche Sehempfinden imitieren. Bei Event-based Vision werden die neuronalen Spikes als Events dargestellt, welche generiert werden wenn die relative Änderung der Lichtintensität einen Schwellwert überschreitet. Die Ausgabe eines Event-based Vision Sensors ist ein Stream von Events, erzeugt von autonomen Pixels, die gefeuert werden sobald sie auftreten und daher nicht abhängig sind von einer künstlichen, periodischen Frame-dauer. Weiters wird redundante Information, wie statische Bildbereiche, unterdrückt, bzw. werden nur Daten von dynamischen Bereichen erzeugt.

Aktuelle, auf Address-Event Representation (AER) basierende, Stereo Vision Konzepte verwerfen die Vorteile dieser asynchronen Datendarstellung, weil ankommende Events in künstlich eingeführte Pseudo-Frames gepuffert werden. Ein Ziel dieser Arbeit ist ein asynchrones Daten Interface für Event-based Stereo Matching zu designen, welche die meisten dieser Vorteile erhält. Das zweite Ziel ist, dieses Daten Interface für Bewegungen von unterschiedlichen Geschwindigkeiten einsetzbar zu machen.

Es wurde ein Ground-truth Vergleich, zwischen State of the Art Verfahren und jenem das in dieser Arbeit vorgestellt wird, durchgeführt um die Machbarkeit und Verbesserung dieses Verfahrens zu analysieren.

Nach der Analyse einiger Methoden um vergleichbare Ground-truth Daten von dynamischen Szenen aufzuzeichnen, hat sich herausgestellt dass das Aufzeichnen von Ground-truth Daten von Szenen die bewegte Testobjekte mit komplexer Geometrie beinhalten ein Follow-up Forschungsthema bleiben wird, da es den Umfang dieser Arbeit sprengt. Eine Auswertungsmethode mit einfachen Testobjekte hat schließlich zu einer machbaren Auswertung geführt. Durch die Vereinfachung des Testobjektes konnte die Auswertung aber keine Verbesserungen der Stereo Matching Genauigkeit bezüglich unterschiedlicher Bewegungsgeschwindigkeiten enthüllen, da es die Nachteile der State of the art Lösung nicht stark genug hervorheben konnte.

Nichtsdestotrotz wurde das Hauptziel, Implementierung des asynchronen

Daten Interfaces, erreicht und weist, wie aus dem Auswertungsergebnis sichtbar ist, keine negative Auswirkung auf die Stereo Matching Genauigkeit auf.

Acknowledgments

First of all, I would like to thank Nabil Belbachir and the Austrian Institute of Technology (AIT) for giving me the opportunity to work on this topic and providing me with all the necessary tools and infrastructure. Special thanks to Nabil, for having confidence in me, sharing your experience and expertise with me and for the flexibility you showed when I came up with necessary changes in directions.

Thanks a lot to Stephan Schraml for getting me started on the hardware and introducing me to the details of the currently implemented stereo matching algorithm. Thanks to Bernhard Kohn, Peter Schön, Gerhard Gritsch, Martin Litzenberger, Rainer Wohlgenannt, Nikolaus Donath, Daniel Bauer, Gervin Pedico, Daniel Matolin and Roman Gmeiner for all the enriching discussions.

Thanks to Peter Hamilton, Andreas Gschwandtner and Thomas Pieldner for the mutual support while exploring the solutions to the problems we were working on. Thanks to all my friends for the patience while me being busy with work and not joining leisure activities. Thanks to Karin for cheering my up during tough times.

Thanks to Robert Sablatnig for supervising my work and for challenging me with the right questions with the aim to lead me towards excellent solutions.

Most thanks deserve my parents for (not only financially) supporting my studies and my decisions all the time. I am also thankful to the Austrian education system and the Ministry for Education, Science and Research to enabling me to study full-time while being financially independent.

Chapter 1

Introduction

The title of *Tobi Delbrück's* article [11] in *The Neuromorphic Engineer* emphasizes the main difference of event-based vision approaches in comparison to common computer vision:

Freeing vision from frames

Frame-based (traditional) computer vision evolved historically from the invention of the photo camera by adding the temporal dimension to the retrieval of video sequences. While this is suitable for reproducing video sequences on a viewing device, it requires unnecessary additional processing for computer vision applications, which is a disadvantage especially for real-time applications. Furthermore, human or mammalian vision does not operate in a frame-based manner and does not process the data pixel-by-pixel, but rather the optical nervous system fires nerve impulses when the optical receptors receive an optical stimulus [33].

The representation of image data as periodic frames is inevitable in common computer vision. Event-based vision in contrast, abandons the frame concept and considers only image regions where relative light intensity has changed [11]. Since pixels only generate events when their relative intensity change reaches a certain threshold, image regions which do not contain new information do not generate data. An exception to this principle is events generated because of sensor noise.

Event-based vision sensors produce data only where the relative light intensity changes beyond a certain threshold. Basically, in the case of a stationary camera, contours of moving objects are visible as long as there is a difference in intensity between the object and the background. In general, every change in light reflection, or direct light radiation can be responsible for a change in intensity.

Considering the before mentioned characteristics, event-based vision techniques offer new possibilities for developing highly-responsive applications. A more detailed explanation of the event-based vision principles is provided later in this work.

1.1 Motivation

Common digital image or video acquisition techniques generate densely sampled image data, whereby the index of each sample denotes implicit spatial and temporal neighborhood relations. Moreover, the spatial and temporal relations are equidistant, hence common image processing algorithms might take the implicit spatial and temporal coherency of image data for granted.

Event-based computer vision on the other hand generates sparse image data, whereby the index of each event has no relation to spatial neighboring events and the temporal relation is monotonically increasing but not equidistant. The spatial and temporal relation between each event is explicitly defined by the event's x, y coordinates and its timestamp.

Because of this, there is still an unresolved question for event-based vision research, as to whether the best solution is to adapt common image processing methods, in order to operate on this event-based data representation, or if we need to find new groundbreaking methods which operate on event-based data directly, without any adaption or conversion. A more detailed explanation of event-based data representation is provided in Chapter 2.

One way to establish a link between common image processing algorithms and this event-based data representation is by periodically rendering the event-based data into frames. As a result, the image data is processable as usual. But this is achieved by neglecting the asynchronous nature, which the event-based data representation originally has.

Initiated by our cooperation partner, the *Austrian Institute of Technology (AIT)*, this work investigates how the data conversion process can be modified in order to preserve the benefits of an asynchronous interface. In order to evaluate the approach proposed by this work, the focus is set on stereo vision as an application.

Another problem addressed in this investigation is that in order to convert to a frame-based representation, a certain frame rate demands a certain object movement velocity. Objects moving faster than the required velocity through the sensor's field of view will result in cluttered object edges comparable to motion blur as in common computer vision. The edges are cluttered, because the object moves through the field of view of multiple pixels, frame by frame. On the other hand, if objects move slower through the visible area than the required velocity, edges will appear jagged or vanish completely. This is due to the fact that the edge passes the field of view of only one pixel during multiple frames and the events will be distributed over the first frames. The frame in which each pixel will fire depends on when each pixel of the edge reaches the threshold and fires the event. Due to minimal differences in sensitivity of the individual pixels, they might not fire at the same time, hence their timestamp might not be identical.

The velocity of an object passing through the field of view of each pixel in image space is calculated using the thin lens equation 1.1 [18], derived from

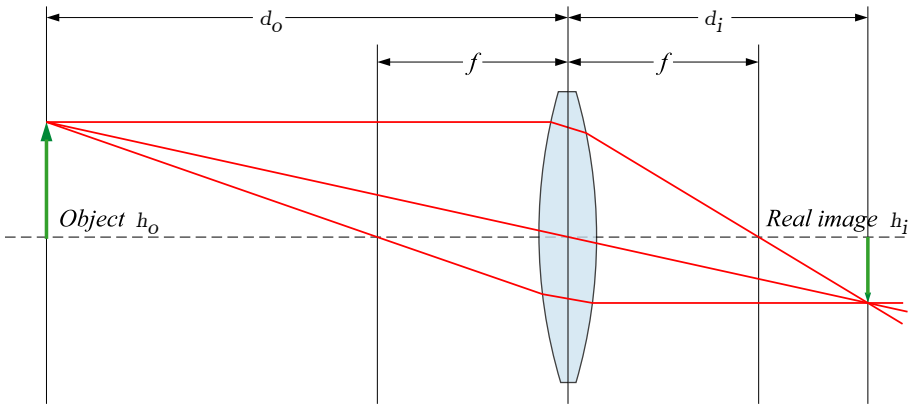


Figure 1.1: The thin lens equation defines a simplified geometrical relation between an object and its image. From http://en.wikipedia.org/wiki/File:Lens3.svg.

the relationship illustrated in Figure 1.1, whereby h_o denotes the object height, h_i denotes the image height, d_o denotes the object distance and f denotes the focal length.

$$\frac{h_i}{h_o} = \frac{f}{d_o - f} \tag{1.1}$$

Derived from Equation 1.1, h_o and h_i can be replaced by the object velocity v_o and image velocity v_i as depicted in Equation 1.2.

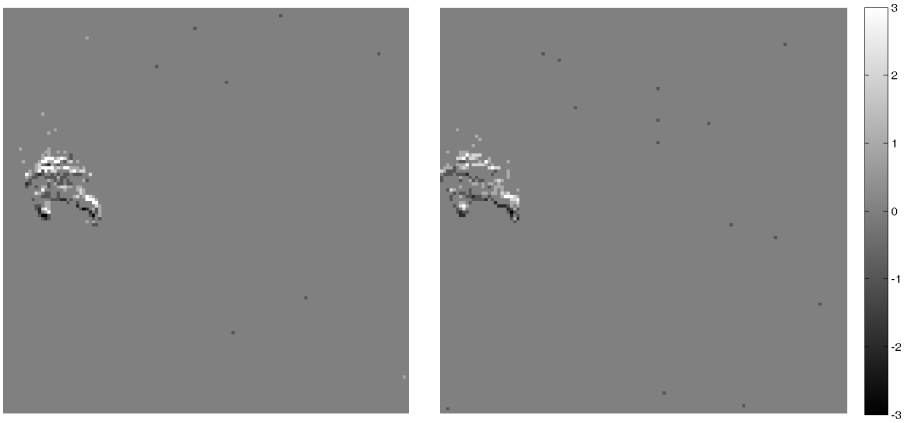
$$\frac{v_i}{v_o} = \frac{f}{d_o - f} \tag{1.2}$$

The image velocity v_i of unit meter per second $[m/s]$ can be converted to the unit pixel per second $[px/s]$ by dividing it by pixel pitch p of the sensor. Then, it is converted to the unit pixel per frame $[px/F]$ by multiplying it by the frame duration denoted as t. The image velocity in pixel per frame is denoted as V_i, depicted in Equation 1.3.

$$V_i = \frac{v_i}{p} \cdot t \tag{1.3}$$

As mentioned above, a certain periodic frame rate is only feasible for a specific object velocity. In image space, this velocity is 1 pixel per frame. Let V_i from Equation 1.3 be 1, as depicted in Equation 1.4, the suitable object speed v_o for a frame duration t ($= 1/\text{fps}$) can be calculated by transforming and applying Equations 1.5 – 1.8.

$$1 = \frac{v_i}{p} \cdot t \tag{1.4}$$



(a) (b)

Figure 1.2: Pedestrian, acceptable edges, left (a) and right (b) sensor, frame duration 20 ms. The images are rendered from accumulated Address-Events, whereby the intensity value denotes the sum of positive and negative events at this location.

$$\frac{p}{t} = v_i \tag{1.5}$$

$$\frac{\frac{p}{t}}{v_o} = \frac{f}{d_o - f} \tag{1.6}$$

$$\frac{p}{t} = \frac{f}{d_o - f} \cdot v_o \tag{1.7}$$

$$v_o = \frac{\frac{p}{t}}{\frac{f}{d_o - f}} \tag{1.8}$$

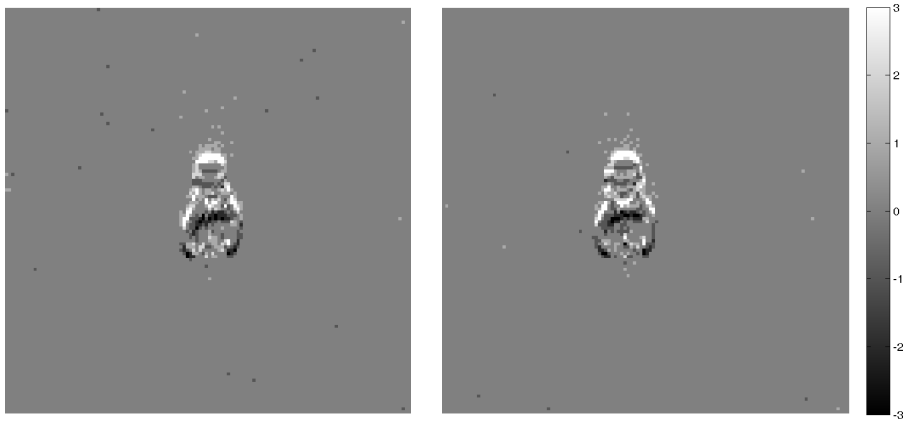
If the expected object velocity is given, the suitable frame duration t is calculated as

$$t = \frac{p}{\frac{f}{d_o - f} \cdot v_o} \tag{1.9}$$

Figure 1.2 illustrates a walking pedestrian, rendered from accumulated Address-Event data with a frame duration of 20 ms. The edges of the moving person are visible and appear sharp, so it can be assumed that the frame duration fits the movement velocity.

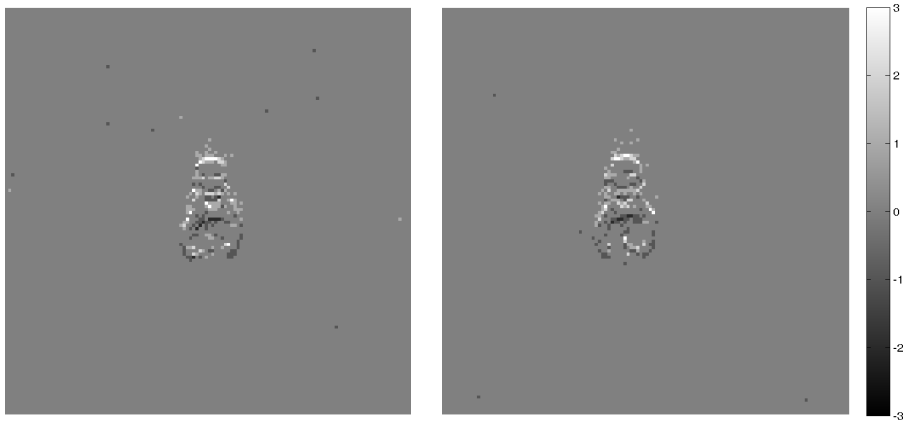
The cyclist in Figure 1.3 on the other hand, rendered again using a frame duration of 20 ms, moves too fast for the chosen frame duration. For this object movement velocity a frame duration of 7 ms is more suitable. Figure 1.4 illustrates the difference.

If the pedestrian, on the other hand, is rendered using only 7 ms frame duration, the amount of Address-Events visible in each frame is too low to



(a) (b)

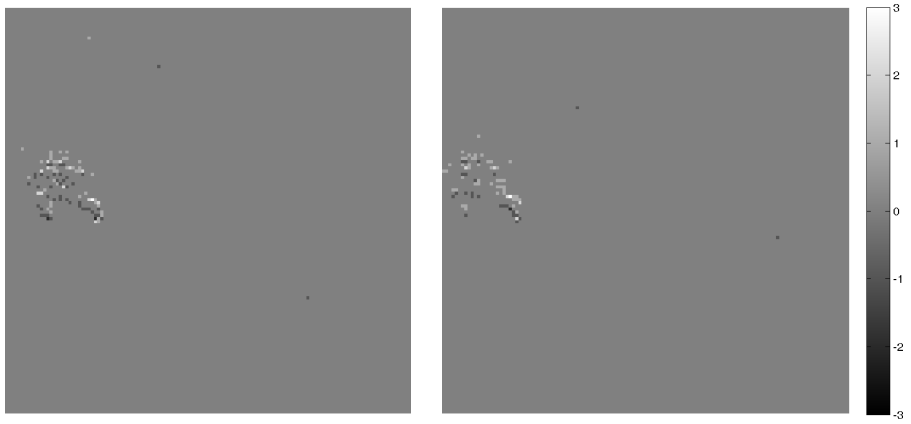
Figure 1.3: Cyclist, cluttered edges, left (a) and right (b) sensor, frame duration 20 ms. The intensity value denotes the sum of positive and negative events at this location.



(a) (b)

Figure 1.4: Cyclist, acceptable edges, left (a) and right (b) sensor, frame duration 7 ms. The intensity value denotes the sum of positive and negative events at this location.

result in acceptable images, as Figure 1.5 shows. These examples illustrate the problem when Address-Event data with movement of varying velocity is rendered with the same frame duration.



(a) (b)

Figure 1.5: Pedestrian, jagged edges, left (a) and right (b) sensor, frame duration 7 ms. The intensity value denotes the sum of positive and negative events at this location.

1.2 Objective of this Work

The targeted result of this thesis is a data interface which preserves the image quality (in regard to the input for stereo matching) over a broad range of object movement velocities and as a result increases stereo matching accuracy. A ground-truth comparison of the proposed concept to state of the art AER stereo matching approaches (periodic frame duration) will be presented in order to show the feasibility and improvement achieved by this work.

As one aim of this investigation is to increase stereo matching accuracy, test disparity map data is recorded and validated against generated ground-truth data.

1.3 Contribution

A novel asynchronous data interface approach for event-based stereo vision data is the main contribution of this work. The interface uses image space local Address-Event buffers, which allow the efficient application of a local event density threshold. This way, the interface provides asynchronous access to the image data and independence from the velocity of the object in motion at the same time. A detailed description of the asynchronous interface realization is provided in Chapter 4.

Another contribution is the evaluation of ground-truth validation approaches for event-based stereo image data of dynamic scenes. The structured light method for ground-truth data acquisition requires static scenes,

and is hence not applicable for event-based vision since static scenes do not produce data (in case of static camera position, like in this work). The first approach implemented is a ground-truth comparison using a 3D digitized object that was moved along the track of a toy train. The second approach was further refined by using a rectangular object with regular edges in order to overcome spatial alignment problems between the captured data and the 3D model used for comparison. See Chapter 5 for more details on the evaluation.

Additionally, a space-time visualization tool is presented in Chapter 2. This application visualizes the Address-Event data in real-time as points in the space-time cube (2D space plus 1D time), so that the data can be analyzed without any prior conversion (except the 3D to 2D projection).

The approach of using blinking circles as a calibration pattern for camera calibration is also a contribution presented by this work. This approach solves the problem of static calibration patterns not being visible for these type of image sensors. The implemented calibration pattern is described in more detail in Chapter 5.

1.4 Scope

Improving the image quality of the input data to the later stereo matching process is the principal scope of this thesis. Improving other elements in the stereo calculation process chain, like improving the stereo matching algorithm itself is not within the scope of this work. Processing performance and real-time capability is outside the scope of this thesis as well, in order keep the complexity on a suitable level for a master thesis.

1.5 Outline

After this brief overview of the topic and the purpose of this thesis, the following chapter provides a brief introduction to the human vision system including the structure of the retina and an overview to the principle of the visual nervous system. The second part of Chapter 2 provides a summary of event-based vision and depicts the Address-Event data structure specification relevant for this work. Chapter 2 concludes with a description of the space-time representation and a presentation of a visualization tool implemented for the analysis of event-based image data.

Afterwards, an introduction to stereo vision, including epipolar geometry and the stereo processing pipeline, is presented in Chapter 3, followed by the state of the art of current research in event-based stereo vision.

Chapter 4 presents the concept of the proposed asynchronous data interface, the local Address-Event buffer, and depicts the implementation in pseudo code. Additionally, exemplary output results of the implemented

interface are illustrated in Chapter 4.

Chapter 5 describes the calibration of the stereo system used in the experiment including the resulting intrinsic and extrinsic camera parameters. The different approaches to the ground-truth data acquisition process are also described in this chapter, as well as the actual evaluation process of this interface. In the evaluation section, the performance between the proposed system and a state of the art approach are compared relative to ground-truth data followed by the presentation and an interpretation of these results.

A conclusion arguing about the achievement of the defined objectives and providing reasons why one of the goals has not been reached is provided in Chapter 6. Finally, the outlook on possible further research completes the final chapter of this thesis.

Chapter 2

Bio-inspired Computer Vision

There are examples in research where mimicking natural processes resp. natural functionality is attempted. Examples for biological inspired technology are the lotus paint inspired by the lotus plant [2], gecko tape [14] and automotive reflectors inspired by cat eyes [22]. The following sections provide a summary of how human vision works and what recent research in the area of computer vision has learned and adopted from nature. The technique considered by this work, event-based bision, is an example of a technique which applies some of these concepts.

2.1 Human Vision

The eye is the principal organ for the sense of sight and from a technical point of view has remarkable characteristics. The eye has photoreceptors that are able to sens even single photons. The range of recognizable illumination is from 10^{-6} cd/m² (e. g. cloudy sky at night) to between 10^{4} cd/m² and 10^{5} cd/m² (e. g. snow in sunlight) [29], which means it has a dynamic range of 200 dB compared to the dynamic range of average digital camera sensors of less than 70 dB.

The human eye is an optical system comparable to a camera. Likewise, it contains an *iris*, a circular muscle to control the amount of light entering the eye and a *lens*, to bundle light rays allowing a sharp image. Additionally, in front of the lens there is the *pupil*, which appears black from the outside because of the retina's light absorbing ability [26]. Both pupil and iris are covered and protected by the *cornea*. The *sclera* is known as "the white of the eye" [26]. Figure 2.1 illustrates the basic structure of the eye.

The part which is most interesting for the design of bio-inspired vision sensors is where the projected image is perceived and where the visual stimuli are transmitted to the visual cortex of the brain. The visual perception process takes place in the *retina*, which is located on the inner backside of the eye. The *optic nerve* is responsible for the transmission of the image

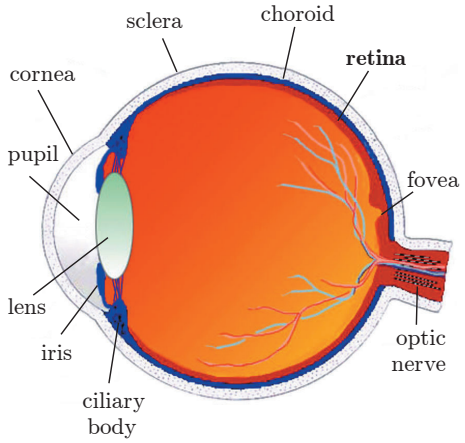


Figure 2.1: Anatomy of the eye. From [25].

data after it has been perceived.

The retina

The *retina* is structured in distinct layers and is hardly 0.5 millimeter thick. There are three layers of nerve cells and 2 layers of synapses. Further details on the synapses layers can be found in neuroscience literature such as [33]. The *photoreceptors*, which are the sensing unit, are located on the back side of the retina, close to the *pigment epithelium* (Figure 2.2) and make up the first nerve layer. Therefore, light has to travel through all other layers, before activating them.

There exist two kinds of photoreceptors in the human retina:

Rods – responsible for low-light vision

Cones – responsible for daylight bright-colored vision [25].

The *fovea* contains most of the cones. The dense alignment of cones in the fovea allows sharp daylight vision and is mainly responsible for color vision, whereas the wide distribution of rods in the retina allows recognition of slight low-light changes in a wide field of view. This is in particular useful at night or in dark surroundings. There are three different kinds of cones, one responding to red light, one responding to green light and one to blue light.

The photoreceptors are connected to the *biploar cells*, which form the middle nerve layer. The photoreceptors release neurotransmitters to the bipolar cells under dark conditions and stop transmitting if light strikes the receptors [25]. The bipolar cells can be grouped into two categories. ON bipolar cells inhibit if they receive neurotransmitters from the photoreceptor (dark) and activate if the transmission stops (light). OFF bipolar cells work the other way round, they activate if neurotransmitters are received (dark)

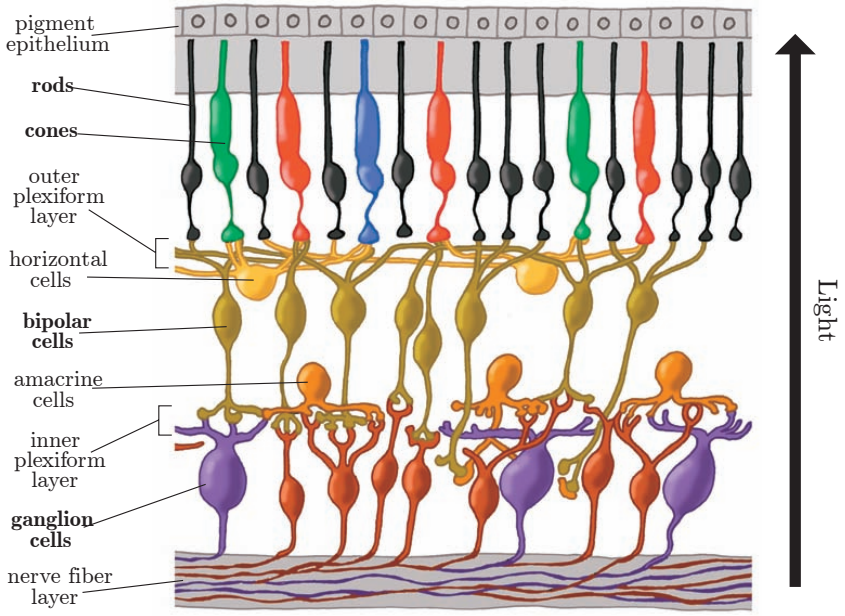


Figure 2.2: Arrangement of neural cells like rods, cones and ganglion cells in the retina. From [25].

and get deactivated if transmission from the photoreceptor stops [23].

The outer layer is composed of *ganglion cells*. The human retina holds between only 1 million and 1.5 million ganglion cells, compared to between 100 and 130 million photoreceptors (numbers vary in literature between authors). In the fovea, the ganglion cells receive input from only a single or a few photoreceptors, whereas in more peripheral areas thousands of photoreceptors transmit to one ganglion cell. This ensures a high spatial resolution in the fovea.

The ganglion cells can be distinguished between ON-center and OFF-center cells [23] and X-type and Y-type cells (also the terms P-type and M-type are found in literature) [33]. Resting ganglion cells fire at a base rate, excitation or inhibition increase or decrease the rate at which they fire. They have a circular receptive field, whereby ON-center cells excite if their center is exposed to light and inhibit if the remaining area, which is called the surround, is stimulated. OFF-center cells respond inversely. Figure 2.3 illustrates the ganglion cell responses to different light conditions.

The behavior described above is valid for X-type cells which fire as long as the light stimulation continues, Y-type cells, on the other hand, respond to the onset and offset of light [33]. Figure 2.4 illustrates the ganglion cell fire rate for X-type and Y-type ganglion cells.

In a nutshell, different layers of nerve cells in the retina apply prepro-

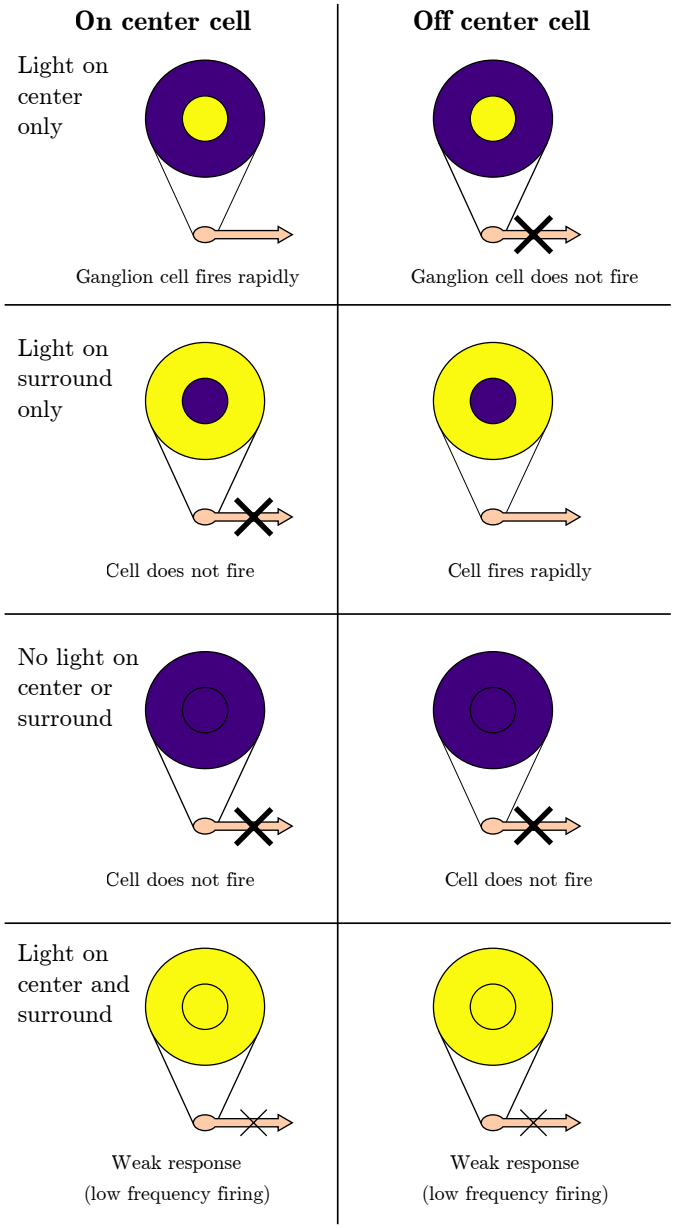


Figure 2.3: Response of ganglion cells according to different light conditions in their receptive field. From http://commons.wikimedia.org/wiki/File:Receptive_field.svg. **Note:** In the discussion of the Wikipedia article "Receptive field" it is questioned whether the off center cell responses for the cases *light on center and surround* and *dark on center and surround* are correct in this diagram. Whether or not this is true, in the authors personal opinion there is no harm done for this basic overview, simply used to demonstrate the motivation of bio-inspired vision sensors.

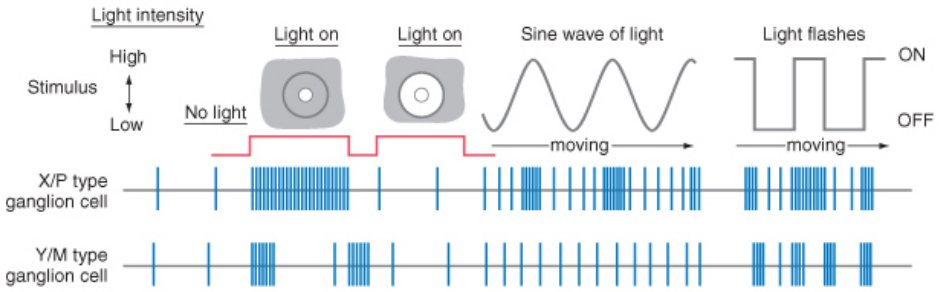


Figure 2.4: X-type cells fire rapidly during light stimulation of the receptive field center. Y-type cells, on the other hand, respond best between changes in light conditions. From [33].

cessing tasks, like detecting spatial changes in contrast (e. g. edges) and temporal changes in light exposure, before transmitting the nerve signals to the visual cortex in the brain. This preprocessing step also reduces the amount of signals necessary to transmit the image to the brain, which is later responsible for the remaining image procession and interpretation. The methods of processing the visual information, first in the human retina by the neuron network, and later in the visual cortex are highly parallel, which, combined with the aforementioned preprocessing, makes human vision very efficient.

Note: This section has to be considered as a simplified summary used simply to outline the basic principles of human vision, as this is sufficient for the purpose of this work. Should a more in depth summary be needed, the latest literature in visual neuroscience, like [33] or [23], is recommended.

2.2 Event-based Vision Sensor

For computer and machine vision, as well as for human vision, the objective of retrieving information from the sensed visual stimuli does not implicitly require capturing a visual, viewable image, for example in the form of a two dimensional matrix of absolute intensity or color values. This is mainly required in order to present the image data on a device, like a screen, viewable for humans.

In the late 1980s, a new interdisciplinary field evolved called *neuromorphic engineering*. Coined by Carver Mead [37], neuromorphic engineering has drawn inspiration from Biology, Physics, Mathematics, Computer Science and Engineering in order to design artificial neural systems, such as the vision system. As depicted in the previous section, the human vision system processes visual stimuli in a massively parallel and data-driven way

[29]. Data-driven processing signifies that transmission is initiated if new information is sensed, in contrast to periodic polling by the sensing unit.

If a computer or machine vision application only requires detecting a change or movement in the field of view of the camera, static image areas may be neglected. Hence, capturing full intensity or color coded frames is not necessary. Research in neuromorphic engineering strives to mimic the principles of biological vision in order to overcome limitations resulting from frame-based vision systems.

The first event-based bision sensor, the *silicon retina*, was developed by Misha Mahowald and Carver Mead in [35] and [37]. The term *silicon retina* is an analogy to the chip intending to mimic the human retina. On the chip, the photoreceptors, which are in this case photodiodes, are composed of silicon. According to [29], Carver Mead's and Misha Mahowald's silicon retina was only a demonstration of concept which was unusable for real applications, as it only sensed high contrast stimuli, like blinking LEDs.

According to [32], asynchronous vision sensors prior to the temporal contrast sensor were merely useful for demonstrations of concepts, but no useful application has been implemented. Therefore, they will not be discussed in further detail.

Temporal contrast sensor

A group of researchers from the Institute of Neuroscience of the University of Zürich are one of the main contributors in the area of event-based vision sensors [28]. [27] and [30] present their first developments before presenting in [29] the temporal contrast sensor. Applications like the pencil balancer [8] demonstrate the new abilities of the sensor, in this case the low latency of the feedback loop.

The main difference of the temporal contrast image sensor in contrast to traditional image sensors, is the integrated logic in each pixel. Figure 2.5 shows the layout of the temporal contrast sensor chip including a magnification of the pixel's integrated circuits. The programmable bias generators control parameters like ON-threshold and OFF-threshold.

Basically, the pixel's integrated circuit is designed in three parts [29], as shown in Figure 2.6. The photoreceptor part uses a photodiode which produces the photocurrent I proportional to incoming light intensity. The photocurrent I is converted logarithmically into voltage V_p. Further, the photoreceptor part is coupled with a differencing amplifier to determine whether light intensity is increasing or decreasing (V_{diff}). If the voltage V_{diff} reaches the ON- or OFF-Event threshold, the ON or OFF comparator switches and an ON or OFF event is generated. Afterwards, the differentiator is reset.

Figure 2.7 illustrates the principle of the pixel's operation. The upper graph displays the logarithmic photocurrent I and the lower graph displays the corresponding output of the differentiator and marks the points when

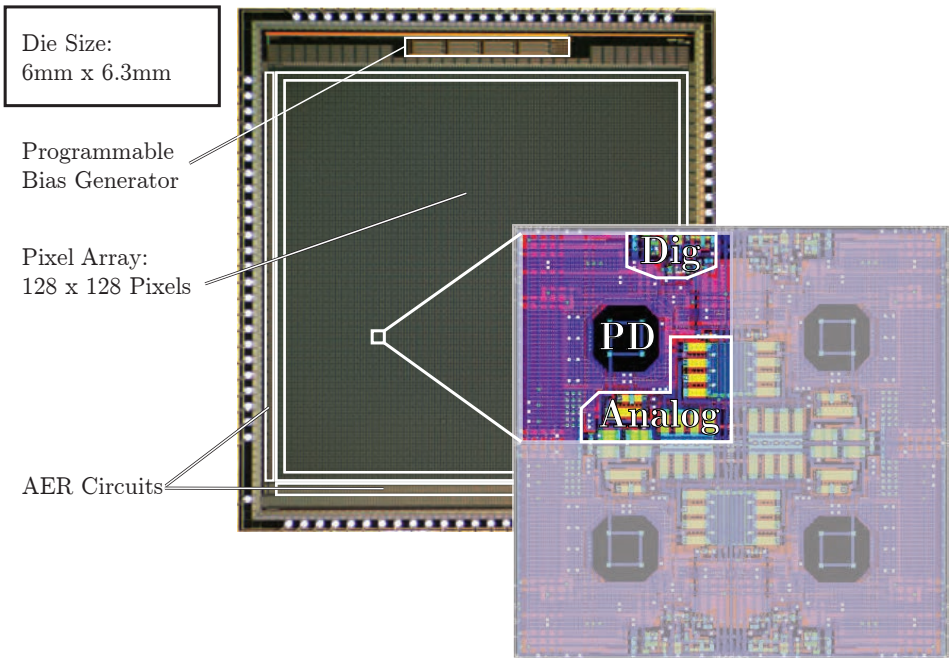


Figure 2.5: Die photograph of the 128 px × 128 px temporal contrast sensor. The magnification displays a block of 4 pixels, where the circuits are arranged quad-mirror-symmetric with the photodiode (PD), analog and digital parts. From [29].

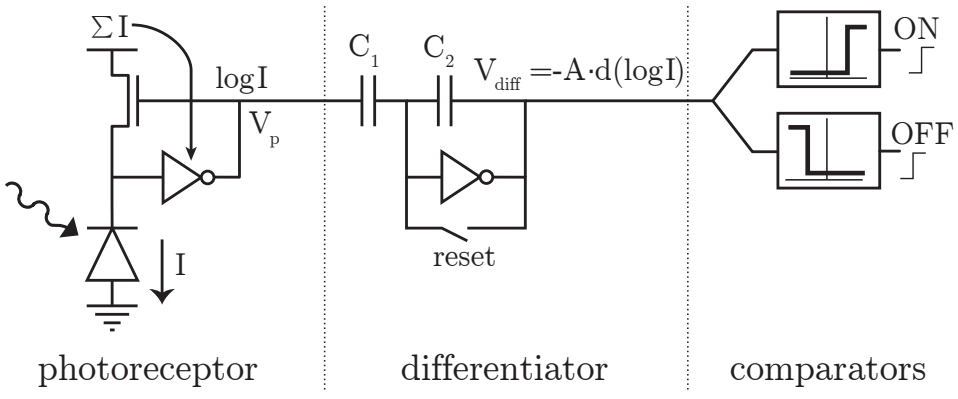


Figure 2.6: Principal design of the temporal contrast sensor. From [29].

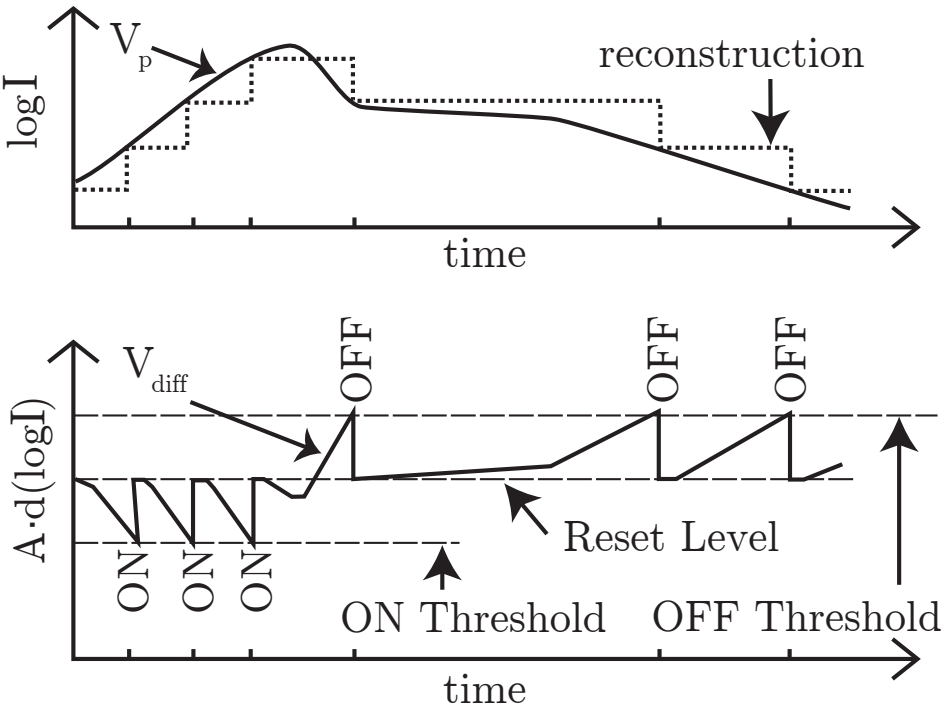


Figure 2.7: Address-Event generation sequence. From [29].

ON or OFF events are generated.

The integration of circuit logic for each pixel enables each pixel to autonomously detect changes in light intensity and let them fire events as soon as changes are detected, regardless of artificial frame times. Another advantage is that static image areas do not generate data, except for cumulated sensor noise which leads to the generation of noise events.

The logarithmic response to light intensity enables the sensor to detect intensity changes in a dynamic range of more than 120 dB. Figure 2.8 displays a comparison of the temporal contrast sensor's Address-Event image to a conventional camera with different illumination durations. The left half of the scene is illuminated by 780 lux and the right half by 5.8 lux. The temporal contrast sensor delivers readable data for both halves whereas the conventional camera only produces acceptable results for either half, depending on the exposure time.

The event response latency depends on the bias configuration and on the illumination [29]. With a "fast" bias configuration, the latency varies between 400 μs and 15 μs whereas a "slow" bias configuration results in a latency between 4 ms and 1 ms.

For the non-simplified version of the pixel's circuit diagram and a more

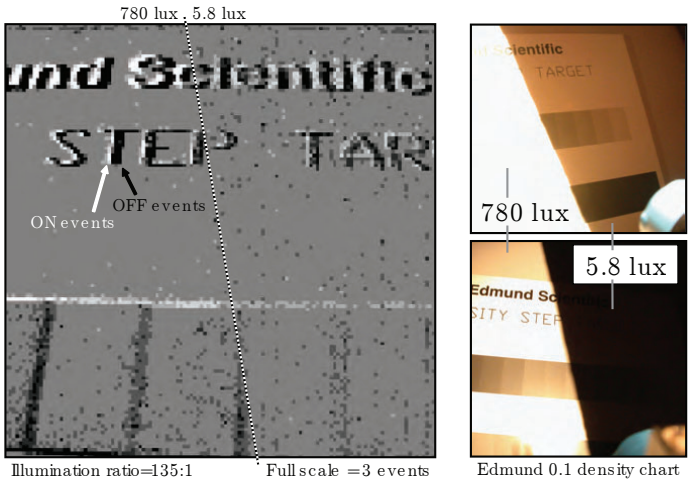


Figure 2.8: The dynamic range of the temporal contrast sensor (left) versus the dynamic range of a conventional digital camera (right). The left part of the scene is illuminated by 780 lux whereas the right part of the scene is illuminated by 5.8 lux. With the temporal contrast sensor both parts are visible using identical settings, whereas the conventional camera over- or underexposes one part. From [29].

detailed explanation of how it works, a look into [29] is recommended.

A research team at the Austrian Institute of Technology recently developed an event-based vision sensor of Quarter-VGA resolution (304 px × 240 px) called Asynchronous Time-based Images Sensor (ATIS). Besides increased resolution, this sensor combines the temporal contrast approach and the transmission of absolute gray-scale values, in each pixel's logic. For further detail please take a look into [39], [41] and [40].

In the literature, various terms have evolved, each emphasizing unique characteristics of event-based vision sensors. [11] and [10] presented the Temporal contrast Sensor, [38] [12] use the term event-based vision sensor, [10], [38], [8] and [45] call it a Dynamic Vision Sensor and [31] uses the term Asynchronous Vision Sensor. The community has not agreed on one specific term yet, in this work the term *event-based vision* is used.

2.3 Address-Event Representation

The sensing pixels from event-based image sensors generate events, when the light-intensity changes. The pixels can be interpreted as neurons, as the principal idea of these were sensors derived from neuromorphic circuits. These neurons require a point-to-point interconnection between the input layer and further processing layer(s). For a high number of neurons, this

is complex to implement in hardware and also inefficient as the expected number of concurrent communicating neurons is sparse [4].

In the neuromorphic engineering community, a protocol for the communication between neuromorphic chips called *Address-Event Representation* (AER) has been developed. In AER, each neuron is identified by a unique address and the spikes of these neurons are events which include the spiking neuron's address. Depending on the implementation, the event data can be extended by additional information (e. g. timestamp, coordinates, signal payload).

The AER, rather, specifies the communication sequence, and not exactly how it is implemented in hardware. [4] is a tutorial on how to design AER-based interchip communication channels.

Although mainly optical AER applications like the silicon retina are pointed out in this work, there are also other applications like the silicon cochlear [7].

As this work proposes an algorithm for AER processing, it focuses on the event data structure of AER, rather than going into detail about the AER communication protocol from the hardware's point of view. AER does not define the data content of the events, as it is not dedicated to a specific application, like vision in this case.

Researchers of the Neuroinformatics group of the Austrian Institute of Technology have developed a stereo vision camera, with 2 integrated event-based vision sensors based on the technology developed at the Institute of Neuroscience of the University of Zürich. The event data structure of this stereo event-based vision sensor device is illustrated in Figure 2.9 [24,46]. In comparison to the data structure used by the temporal contrast sensor, the data structure of the event-based stereo vision sensor is extended by fields required for stereo vision. The visual information in AER is transmitted as a stream of Address-Events, where each event is transmitted as soon as it occurs and as soon as the transmission media is ready. On this hardware platform, an Address-Event is an 8 byte (two 32 bit words) data structure consisting of the following values:

x – the pixel's x coordinate

y – the pixel's y coordinate

timestamp – the time (in μs) when the change in intensity was detected

p – polarity, whether it is an ON or OFF event

c – channel, whether it is from the left or right sensor

The depth field in Figure 2.9 is in gray font since it is only set if the stereo computation of the stereo camera is activated, which has been disabled for this analysis.

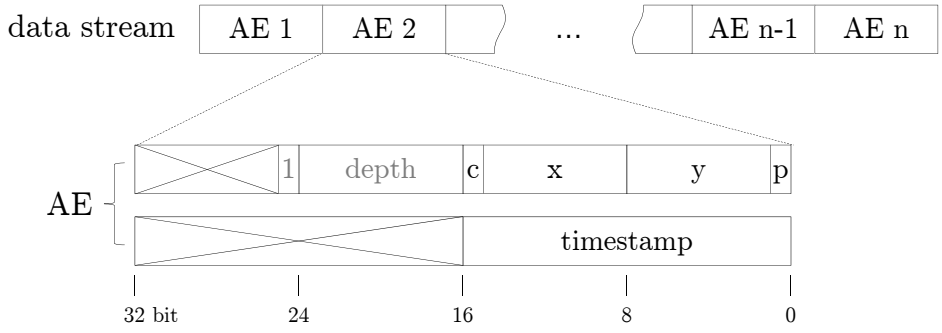


Figure 2.9: Address-Event data structure.

2.4 Space-time Representation

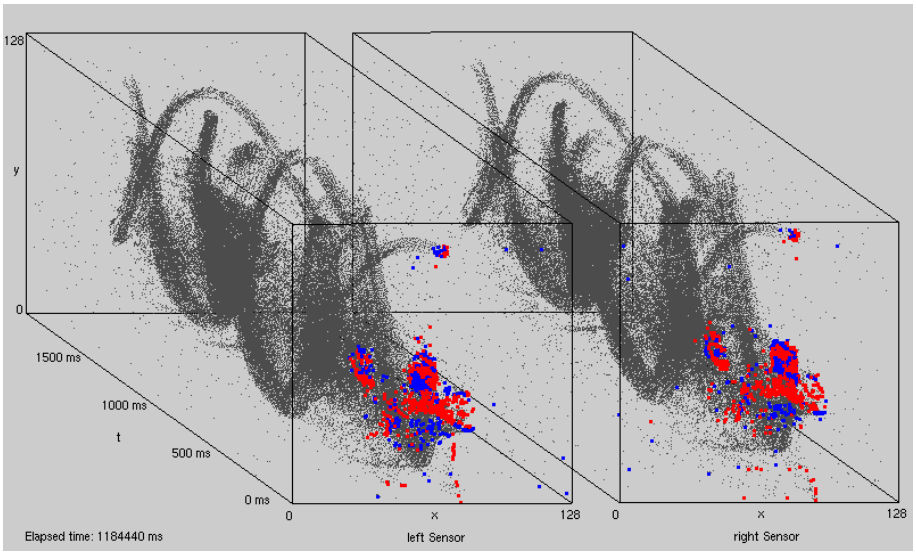
Data transmitted by event-based vision sensors is uniquely defined by the dimensions x, y and t. Each event can be considered as a point in this 3-dimensional space. Visualizing the data as a video stream requires the introduction of artificial frames and choosing parameters like frame duration which is not implicitly given by the data. Considering this, the most natural visualization approach for event-based vision data, is displaying the events as points in a space-time cube. This way, no additional assumption has to be made.

Real-time Space-time Visualization Tool

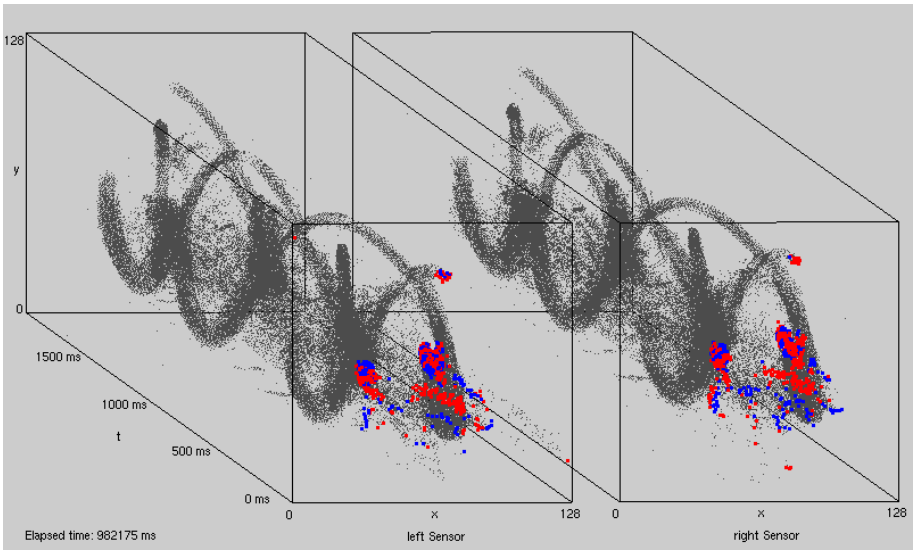
In order to gain insight from the data, a space-time visualization for Mathworks Matlab® has been developed. The purpose of this visualization tool is to make the structure of the space-time point cloud visible and support better understanding of the data for development of event-based algorithms.

Figure 2.10 displays a screenshot of the space-time visualization, whereby (a) displays the raw AER data from the stereo vision system and (b) displays the filtered AER data using the background activity filter described in [10]. The background activity filter reduces the amount of noise events which, as an example, can be useful for accumulating events over a long period of time into one frame, like capturing the trajectory of a moving object [10].

Basically, the visualization tool receives Address-Event data from a network interface as binary data transmitted via the User Datagram Protocol (UDP) and stores it in a ring buffer in the local memory. A rendering loop reads the buffer each cycle and renders the events of $\Delta t \leq 20\,\text{ms}$ as big points, whereby ON-events are rendered in red and OFF-events are rendered in blue. The remaining events are rendered as small black dots, but only if their timestamp is within $\Delta t \leq 2000\,\text{ms}$. These parameters ($\Delta t$ of



(a)



(b)

Figure 2.10: Space-time visualization tool, which displays the left and right channel's AER data of a juggling person, using the stereo vision system described in the following chapter. The most recent 20 ms are illustrated as big colored points, ON-events in red and OFF-events in blue color. (a) displays the raw AER data from the stereo vision system, (b) displays the filtered AER data using the background activity filter described in [10].

most recent events and visible time) are configurable via the graphical user interface.

The Space-time Visualization Tool directly renders the data received from the UDP input stream, which allows real-time visualization of the data sent by the stereo system. As the Matlab[1] libraries are too bulky for the fluent visualization of the AER data from the stereo system, the rendering loop has been extracted into a Java component, using Java Bindings for OpenGL (JOGL) as a graphics Application Programming Interface (API). The combination of Java plus JOGL has been chosen because of native support to access Java components from within Matlab applications. Likewise, the data retrieving loop has been extracted into a Java component, in order to minimize the risk of input buffer overflows, resp. the risk of data loss.

Summary

This chapter has presented the ideas and principles of biological inspired computer vision and how it has been inspired by human resp. mammalian vision. The first section summarized the principles of the visual system and especially focused on the functionality of the retina and the visual nervous system. The second section presented the history and the state of the art in event-based vision. Further, the temporal contrast sensor was presented, and additionally its hardware characteristics as well as the event generation logic were depicted. The latter sections described the idea of the Address-Event protocol and the Address-Event data structure used in this implementation, as well as how to visualize Address-Event data in the space-time cube.

The essential point of this chapter is that like the retina applies preprocessing on the sensed stimuli before transmitting signals to the visual cortex, the pixels of event-based vision sensors integrate logic for preprocessing as well and transmit data only for the dynamic part of the observed scene.

[1] Matlab version 7.0 (R14)

Chapter 3

Stereo Vision

This chapter introduces the principle and current state of the art of stereo vision and furthermore gives an overview of current investigations into event-based stereo vision.

3.1 Stereo Vision

Humans accomplish depth perception when the brain computes the space disparity of image points between the left eye image and the right eye image. This disparity is inversely proportional to the distance of the perceived object, hence it converges to zero at infinite distance.

Machine or computer vision systems use the same principle for depth computation, whereas the eyes are replaced by two (preferably identical) cameras. However, instead of the brain, a microprocessor or computer is responsible for disparity computation.

The reason why the disparity can be used to compute distances lies in the geometrical relations of epipolar geometry, which is briefly described in the following section. Afterwards, the typical steps of stereo processing are explained.

3.1.1 Epipolar Geometry

The epipolar geometry describes the relation between two projections (images) of the same scene. Figure 3.1 depicts a stereoscopic image acquisition system based on two pinhole cameras. O_L and O_R denote the camera centers of the left and right camera. The line joining those centers, which is called baseline, intersects the two image planes in their epipoles e_L and e_R. Basically, epipolar geometry between two views is the geometry of the intersection of image planes with the family of planes having the baseline as their axis [17]. As can be seen in Figure 3.1, an arbitrary point X in 3 dimensional space is, from the perspective of the left camera, the projected

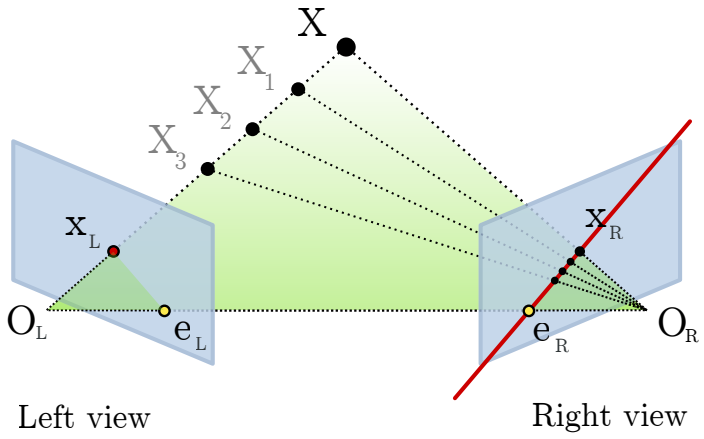


Figure 3.1: Epipolar geometry. From http://en.wikipedia.org/wiki/File:Epipolar_geometry.svg.

point x_L on the left image plane, which is the intersection of the line defined by X and O_L and the image plane. As in a stereo image acquisition system, only x_L is known (and not X), and the set of points defined by the ray from O_L through x_L denote possible positions of X (e. g. X_1, X_2, X_3, etc.). All these points are coplanar with the centers O_L and O_R and the plane formed by these points is called the *epipolar plane*. The intersection of the epipolar plane with the image plane is defined as the *epipolar line*. All epipolar lines intersect at the epipole [17]. This geometric relation is useful for computing stereo correspondence, and moreover reduces the matching problem of an arbitrary point in one image to an one dimensional search along the corresponding epipolar line in the other image. This forms the *epipolar constraint*:

> Each image point x_i of a space point X lies in the image plane *only* on the corresponding epipolar line [9].

As illustrated in Figure 3.2, given the point X and its projections on the left and right image planes x_L and x_R in image plane coordinates (u, v), the disparity d is defined as [48]

$$d = u_L - u_R \tag{3.1}$$

3.1.2 Stereo Processing Pipeline

Stereo processing systems include the following steps:

1. Image undistortion and epipolar rectification

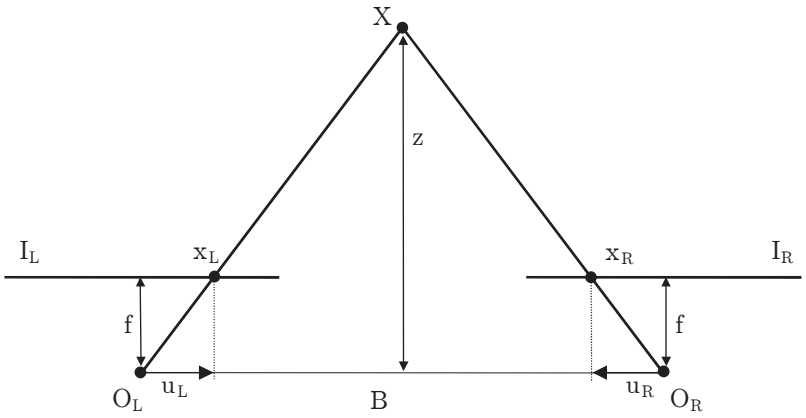


Figure 3.2: Stereo geometry (top view) for the axis parallel case. From [48].

2. Stereo matching
3. Stereo reconstruction

If no absolute distance measures are required (e. g. if only relative depth differences are required for image segmentation), the last step can be skipped as the resulting disparity map is sufficient for relative comparison.

Step 1 – Image Undistortion and Epipolar Rectification

The thin lens equation (see Chapter 1) is only a simplified geometrical model of the projection in image acquisition systems. Real camera lenses introduce distortions into the image, which have to be considered if measurements in image space are used to deduct distances in world space. The resulting distortion is a combination of radial distortion and tangential distortion. Camera calibration algorithms (see Chapter 5) compute the radial and tangential distortion parameters in order to undistort the image.

Epipolar rectification is the transformation and rotation of the left and right image plane to a common coplanar plane so that their corresponding epipolar lines and the image scan lines become collinear [9]. Figure 3.2 illustrates the stereo geometry for the axis parallel case (after rectification). The transformation operations of image undistiortion and epipolar rectification can be combined and precomputed once using a lookup table, storing the mapping of source to destination coordinates, for fast application when undistorting each frame.

Step 2 – Stereo Matching

As explained before, if the projected point on the left image plane x_L and the projected point on the right image plain x_R is known, the point in 3 dimensional space X can be computed. Finding this correspondence is the goal of the stereo matching step. This is the crucial part in the pipeline, because the final accuracy depends on the matching accuracy.

The existing approaches for stereo matching can be divided into two types, depending on the type of output they generate [9]:

Dense or area-based stereo matching – Stereo disparity is calculated for each pixel with the result of a dense stereo map.

Sparse or feature-based stereo matching – Stereo disparity is calculated only for corresponding features of both views. Depending on the application, either the disparity values only for these feature points are sufficient or their disparity values are propagated to their surrounding pixels. For example, image segmentation is applied and similar disparity values are assigned to pixels of the same image region.

The stereo matching approaches can also be grouped into *local* and *global* methods. Local methods compare regions of the left and right images, whereas global methods minimize global cost functions in order to compute the disparity map. This work focuses on local, dense (resp. area-based) stereo matching, whereas feature-based or sparse stereo matching is not within the scope of this work.

Note: As the image sensor already filters static image regions (see Chapter 2) before stereo matching is applied, and even though a dense stereo matching approach is applied in this work, the result is a sparse disparity map as only areas were events have been received are taken into account for stereo matching.

The stereo matching algorithm used in this investigation is based on the *Sum of Absolute Differences* (SAD) [48] as a cost measurement. SAD was chosen because an existing implementation of SAD was available for reuse. Moreover, the choice of the stereo matching approach is secondary since the focus of this work is on the influence of the data interface. SAD is a block matching algorithm (also known as window-based) whereby the absolute differences between image blocks along the epipolar line are computed and the positions with minimal absolute difference are chosen as stereo correspondence candidates. A similar approach is the *Sum of Squared Differences* (SSD) [36] where, as the name implies, the minimal sum of squared differences is used for computing the best match.

An overview of stereo matching algorithms and their comparison is provided by [42].

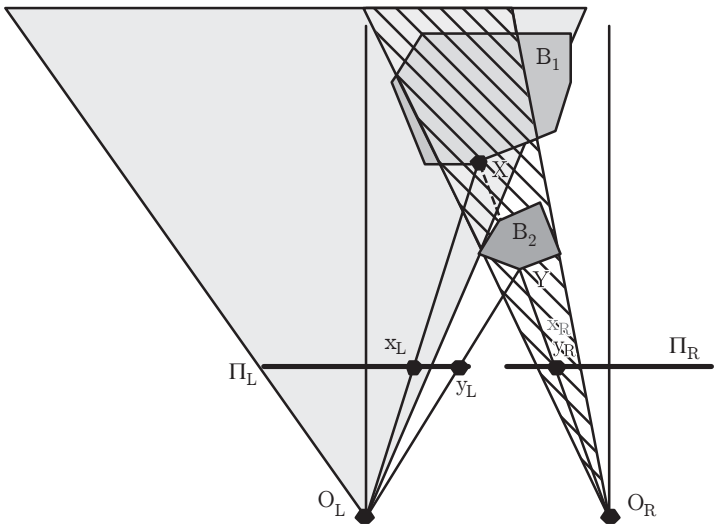


Figure 3.3: Mutual occlusions of objects. From [9].

Stereo Matching Problems: Stereo matching faces conceptional problems which are not solvable (except by post-processing the result) with the information of only two images from a stereo acquisition system.

Occlusion – More precisely, binocular half-occlusion [9] is what the problem of 3D objects being only visible from one of the two viewing positions is called. The problem is visualized in Figure 3.3, where an object B_1 is partially occluded by the object B_2. The point x_L, which is a projection of the 3D point X on the surface of the object B_1, cannot be matched with any other point on the right image, as it is not visible.

Occluded pixels can be detected using the *Left-Right Consistency* check (LRC) [9]. In this method, the matching process is performed using the left image as a reference image and additionally a second time using the right image as reference. If the disparity difference between $|d_L(u,v)|$ and $|d_R(u-d_l(u,v),v)|$ is higher than a specific threshold, then this point might be occluded.

Untextured Regions – Homogeneous regions without descriptive information result in matching errors as well.

Horizontal aligned Textures – Problems also occur if the pixel data is only descriptive in the vertical direction, but not along the horizontal axis. E. g. an edge parallel to the epipolar lines.

Repetitive Patterns – If patterns along epipolar lines are not unique, they might result in false matches.

(Specular) Reflections – Reflections are view dependent, hence they can

not be matched correctly.

There are more phenomenons responsible for stereo matching inaccuracy which are not described in detail here. For example *Image Sensor Noise*, *Sampling Artifacts*, *Different Illumination*, etc.

State of the Art: The window size of *window-based matching* approaches, like the one described above, controls the smoothness of the resulting disparity map. Large window sizes reduce the problem of untextured regions and repetitive patterns whereas small window sizes reduce the errors along depth discontinuities. The works of [13] and [20] present approaches using *adaptive windows* which combine the benefits of small and large windows by computing window sizes for each pixel individually in a flexible way.

Current state of the art approaches introduce a weight function, which increases and decreases the influence of matching window pixels by the likelihood that they are of the same disparity as the matching window's center pixel. These *Adaptive Support Weights* are computed in different manners. [49] states the assumption that pixels of similar color and small Euclidean distance are most likely to have the same disparity. Based on this assumption, an exponential weight function of the sum of similarity and distance is presented.

[21] states the assumption that points of the same disparity share a certain level of connectivity. Connectivity, in this case, is denoted as the sum of color differences along a path, whereby a low value represents high connectivity. For example, a path along homogeneous color results in high connectivity whereas a path which crosses an edge results in low connectivity. The mentioned work proposes a weight function based on the geodesic distance, which computes the length of the shortest path in the color volume.

Step 3 – Stereo Reconstruction

The relation between disparity and depth is derived by the transformation of similar triangle equations of parallel stereo acquisition systems (see Figure 3.2). The depth map is computed by applying Equation 3.2 to the disparity map:

$$z = \frac{fB}{d} \tag{3.2}$$

whereby z denotes the distance along the z–axis, f denotes the focal length, B denotes the baseline and d denotes the disparity.

3.2 Event-based Stereo Vision

[15] proposes a stereo disparity computation approach of event-based data using time domain encoded signals and presents an experimental implemen-

Figure 3.4: Stereo event-based vision sensor. From *Austrian Institute of Technology (AIT)*

tation using computer generated AER data. In [47], a complete stereo vision system with stereo disparity computation from event-based data implemented in hardware, is presented. This work uses the *Normalized SAD* (NSAD) as the stereo matching mechanism. Figure 3.4 illustrates the stereo vision system presented in the paper mentioned and which is also used in this work. This stereo vision system has been chosen because it was a requirement from the Austrian Institute of Technology, our cooperation partner. As [1] describes, the data of this event-based stereo vision sensor is an asynchronous stream of Address-Events (AE), which has to be converted to periodic frames of frame length DT, whereby DT determines the temporal resolution of this stereo vision system. Therefore, each frame is initialized (with zero for signed values or half of the maximum representable value of the data type used) and each event within the time scope of the frame is accumulated by adding $+1$ for ON events or -1 for OFF events at the corresponding pixel location. Hence, the gray value is proportional to the number of events with identical coordinates per frame.

State of the Art: [24] presents experimental area-based and feature-based stereo vision approaches for an automotive pre-crash warning system for side impacts using AE data, where SAD is used for area-based and a segment center matching approach is used for feature-based stereo matching.

A real-time tracking system using event-based stereo vision is presented in [45] and demonstrates experimental results for people tracking. The same authors present a clustering algorithm for event-based stereo data in [44] and a real-time pedestrian and cyclist classification algorithm (also based on event-based stereo data) in [3].

All event-based stereo vision approaches that are found in current state of the art literature use periodic frame conversion (i. e. a fixed frame duration) in order to apply traditional stereo vision algorithms. However, [24] states the future intention of investigating in a stereo vision approach that directly processes the AE data, without frame generation strategies.

Summary

This chapter has presented the principles of epipolar geometry and how these principles are used for stereo disparity computation. A detailed description of the stereo processing pipeline is presented as well as the limitations of stereo matching. The most crucial and therefore interesting part of the stereo processing pipeline is the stereo matching section. The different stereo matching approaches can be grouped into dense or sparse stereo matching, as well as local or global stereo matching approaches.

The last section of this chapter described how these stereo vision concepts are applied for event-based image data and concluded that the state of the art event-based stereo vision approaches use synchronous resp. periodic frame conversion in order to apply stereo processing algorithms.

Chapter 4

Asynchronous local Address-Event Buffer

One aim of this work is to provide AER-based input data for stereo matching, which is robust regarding variable object movement velocity. Additionally, this aim includes providing the data in an asynchronous manner, which means that it is not coupled to a frame creation period. This chapter describes the methodology of this work and provides rationales as to why the chosen approach is a solution to the problem defined in Chapter 1.

4.1 Basic Idea

The advantage of an asynchronous interface is that it retains the asynchronous manner of the event stream for the later application which processes the image data. This way, the application which uses the data decides how often it querys the recent data, for example depending on how fast it processes each frame. This adds more flexibility in contrast to periodic frame generation, which is object movement velocity dependent.

A preliminary idea for fulfilling the asynchronous interface requirement could be implementing a sliding window approach on the event stream, with overlapping time scope for each frame. But even if this approach provides an asynchronous interface, it would not solve the problem of object movement with varying velocities.

In order to handle cases of objects with different velocity in the image sensors field of view, a local method is preferable over a global method in order to gain acceptable results for each object.

As described in Chapter 1, if the event-based image data is rendered periodically into frames, depending on the frame duration, motion at only a certain velocity delivers acceptable results. If the object moves faster the edges will become cluttered and lacking in detail, if it moves too slowly the edges will become jagged or completely vanish between frames.

Fast movement results in many events which clutter the resulting frame. This problem is addressed in this work by locally limiting the event density using a threshold. If a region exceeds the density threshold, the oldest events in the region will be removed. If the density of events is kept low, the edges continue to appear sharp.

To provide an asynchronous interface, all events are kept in buffers (one for each pixel) and removed when the local density threshold is exceeded. The current frame may be queried at any time and provides a frame rendered from all events still residing in the buffers.

The drawback of this concept is that image noise accumulates over time and when the moving object has already passed the field of view of one pixel, the corresponding buffer is never emptied. This problem is addressed by a second rule, which is a timestamp threshold. The timestamp threshold defines the maximum age of events in the buffer. If an event exceeds the threshold it will be removed.

4.2 Concept

The Address-Event data is processed as a stream, whereby each event is immediately stored in one of the buffers, where the image data which is currently valid is stored. Applications using this data interface can retrieve the currently valid image data at any time. This way, the application decides how frequently it retrieves the image data, for example, based on how long it needs to process the data.

The addition and removal of Address-Events to the buffer needs to be efficient, therefore a First-In-First-Out (FIFO) buffer is constructed for each pixel. This way, the events are stored chronologically for each possible image space coordinate, and allows the removal of the oldest events from a certain location by direct access.

In order to efficiently evaluate whether the *local density threshold* is exceeded, a map providing the current local density for each pixel is kept up to date at all times. This is achieved by increasing the count of each pixel that is within the region of Address-Events which have been added to a buffer, and vice versa for each Address-Event which has been removed from a buffer. Figure 4.1 illustrates the local density map.

A similar principle is applied to the evaluation of the *timestamp threshold* to ensure it is performed efficiently. Another map, which provides the oldest timestamp of the currently valid AEs at each location, is also kept up to date at any point in time. This is achieved by setting the timestamp map value to the timestamp of events which are added to an empty buffer, and if an event is removed from a buffer the timestamp is replaced by the value of the next valid event from this buffer.

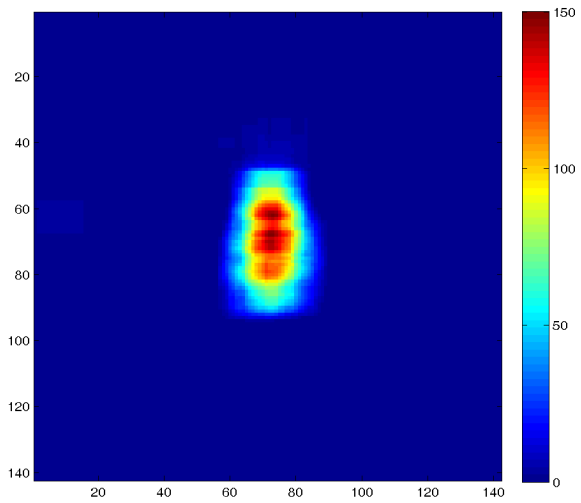


Figure 4.1: Local event density Map. The color scale denotes the absolute sum of events in the local neighborhood.

The list below enumerates the main responsibilities of the algorithm which processes the event stream.

1. Count how many AEs are stored in buffers in the local neighborhood per pixel (the *local density threshold*).
2. Store the oldest AE timestamp per pixel (for the *timestamp threshold*).
3. Store the AEs in FIFO buffers per pixel.
4. Remove oldest AEs of the regions where the *local density threshold* is exceeded.
5. Remove AEs older than the maximum delta timestamp.

4.3 Implementation

The implemented algorithm has the following configurable parameters:

local density threshold – The number of AEs per region which are maximally valid at any point in time can be configured with the *local density threshold*. The local density threshold depends on the region size parameter and on the spatial frequency of the image data that produces the Address-Events. If the expected spatial frequency of the moving objects is high, a high density threshold is required. For low spatial frequencies, for example a moving object measured as constant intensity values, a low density threshold is required.

timestamp threshold – The maximally valid delta between the current time and the oldest timestamp of an AE in the buffers can be configured with the *timestamp threshold*. If the *timestamp threshold* parameter is set too high, too much noise will be accumulated over time. If the *timestamp threshold* is set too low, slow movements are not stored as continuous image data. Hence, the *timestamp threshold* is determined by the lowest velocity which needs to be reproducible. In other words, it needs to be higher than the duration of the slowest object passing through one pixel in sensor image space.

region size – The window size which is treated as a region by the algorithm can be configured with the *region size* parameter. The *region size* adjusts the granularity of local adaption to different movements in the sensors field of view. If the *region size* is too big, for example much bigger than the expected moving objects, the number of events will vary depending on how many moving objects are within this region. On the other hand, if the *region size* is too small, the local density variance is most likely high, since it depends on the local contours of the moving objects. Hence, if the region size is too low, a chosen *local density threshold* will lack in generalizability to the locally varying amount of events.

timestamp threshold evaluation period – The time period between the periodical evaluation of the *timestamp threshold*. As this parameter defines the temporal evaluation of the timestamp threshold, which itself is already a temporal parameter, it makes sense to choose the *timestamp threshold evaluation period* in relation to the *timestamp threshold*, for example 10% of *timestamp threshold*.

Program 4.1 shows the design of the implemented algorithm in pseudo code. First the data structures used are initialized, separately for both the left and right data channel. The `localDensityMap` is initialized with zeros, the `timestampMap` is initialized with infinity and the `buffers` are initialized with empty lists. After initializing the data structures, the Address-Event data is processed. For each event the data structures corresponding to the events channel are used (line 4). Then, a matrix of ones of *region size* with center at x and y of the current Address-Event is added to increase the count in the `localDensityMap` (line 11). Afterwards, the event is added to the `buffer` at position x and y of the Address-Event. If the `buffer` at this position was empty before the processing the current Address-Event, the `timestampMap` at the Address-Events position is set to the `timestamp` value of the event.

Later on, the `localDensityMap` is evaluated in order to retrieve the positions where the *local density threshold* is exceeded (line 18). In fact, as the threshold is evaluated after each Address-Event added to the buffer, only

Program 4.1: Asynchronous buffer for Address-Event data (in pseudo code)

```
1  initializeDataStructures();
2  foreach addresEvent in stream
3    // split left and right channel
4    if (addressEvent.channel == LEFT)
5      useLeftDataStructures();
6    else
7      useRightDataStructures();
8    end
9
10   // store event in buffer and update maps
11   localDensityMap.increaseEventCountForRegionWithCenterAt(addressEvent.
       position, regionSize);
12   if (buffers[addressEvent.position].isEmpty())
13     timestampMap[addressEvent.position] = addressEvent.timestamp;
14   end
15   buffers[addressEvent.position].add(addressEvent);
16
17   // evaluate density threshold
18   areaExceedingThreshold = localDensityMap.getPositionsExceedingDensity(
       localDensityThreshold);
19   if (!areaExceedingThreshold.isEmpty())
20     removeOldestEventOfRegion(areaExceedingThreshold);
21   end
22
23   // evaluate timestamp threshold
24   if (timeElapsed(evaluateTimestampThresholdPeriod))
25     removeEventsExceedingTimestampThreshold(timestampThreshold);
26   end
27 end
```

the region surrounding the events position needs to be evaluated, as there is only a density change in this region. If there are events exceeding the threshold, the function `removeOldestEventOfRegion` is called (see Program 4.2). Within this function, since pixels can simultaneously exceed the threshold, first the center of the exceeding area is determined. Then, the oldest events from the region with this center is removed and the maps are updated.

Whether the *timestamp threshold evaluation period* has elapsed, since the last time the *timestamp threshold* has been evaluated, is checked at the end of the loop (line 24). If this is the case, the function `removeEventsExceeding-TimestampThreshold` is called. This function evaluates the `timestampMap` to get the positions where Address-Events in the buffer exceed the *timestamp threshold*. Then, for each buffer at those positions, all Address-Events exceeding the *timestamp threshold* are removed and the maps updated accordingly.

Program 4.2: Pseudo code function implementation

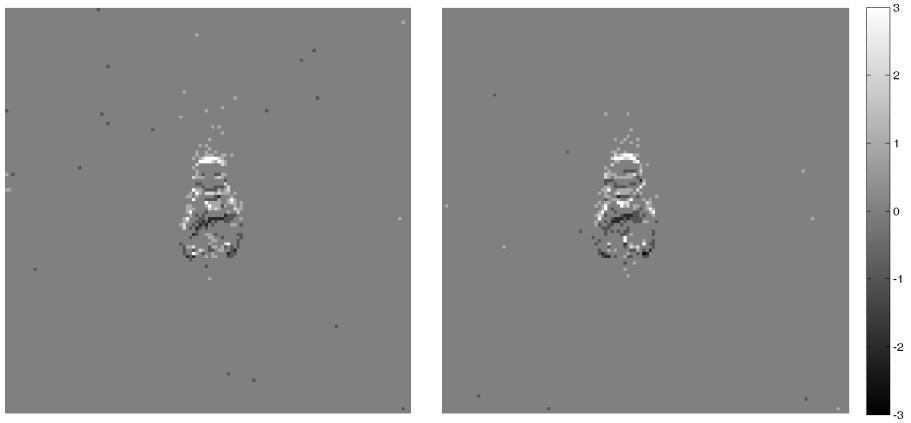
```
function removeOldestEventOfRegion(areaExceedingThreshold)
  center = getCenter(areaExceedingThreshold);
  oldestEvent = timestampMap.
    getOldestEventPositionFromRegionWithCenterAt(center, regionSize);
  removeEventAtPosition(oldestEvent);
end
```

```
function removeEventsExceedingTimestampThreshold(timestampThreshold)
  exceedingTimestamps = timestampMap.
    getPositionExceedingTimestampThreshold(timestampThreshold);
  foreach position in exceedingTimestamps
    while (buffers[position].getAt(1).timestamp < currentTime-
    timestampThreshold)
      removeEventAtPosition(position);
    end
  end
end
```

```
function removeEventAtPosition(position)
  buffers[position].removeFirst();
  if (buffers[position].isEmpty())
    timestampMap[position] = INFINITY;
  else
    timestampMap[position] = buffers[position].getAt(1).timestamp;
  end
  localDensityMap.decreaseEventCountForRegionWithCenterAt(position,
    regionSize);
end
```

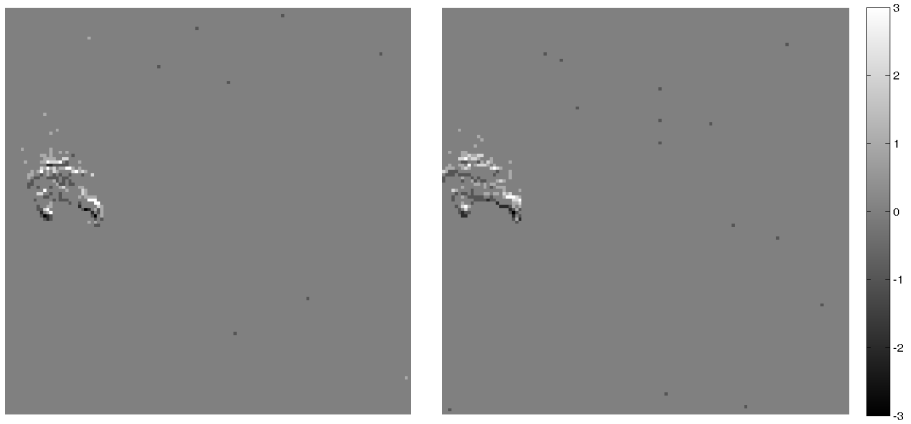
4.4 Output

Figure 4.2 and Figure 4.3 show the result of the example Address-Event data used for the figures presented in Chapter 1 rendered with the presented asynchronous data interface, both using the same set of parameters. The cyclist is more clearly visible compared to the 20 ms frame duration conversion displayed in Figure 1.3 and the output of the pedestrian is still comparable to the (already acceptable) output visible in Figure 1.2. Compared to the figures presented in Chapter 1, the cyclist rendered with the asynchronous data interface is close to the example rendered with 7 ms frame duration and the pedestrian rendered with the asynchronous data interface is close to the example rendered with 20 ms frame duration. This shows that identical parameter settings can be used without prior estimation of the expected object movement velocity.



(a) (b)

Figure 4.2: Cyclist, left (a) and right (b) sensor, asynchronous data interface. The intensity value denotes the sum of positive and negative events at this location.



(a) (b)

Figure 4.3: Pedestrian, left (a) and right (b) sensor, asynchronous data interface. The intensity value denotes the sum of positive and negative events at this location.

Summary

This chapter described the general idea, the concept and the implementation of the asynchronous local Address-Event buffer. How events are stored into buffers has been described as well as how the *local density threshold* and the *timestamp threshold* are evaluated. Section 4.3 described the parameters for fine tuning the presented approach and how they influence the eventual outcome. Furthermore, the design of the implemented algorithm is

stated in pseudo code. The last section demonstrated the impact of applying the asynchronous local Address-Event buffer on the example data used in Chapter 1.

The essence of this chapter is that limiting the number of events based on the local event density provides flexibility regarding object movement velocity and avoids the necessity of a priori chosen frame durations. This way the resulting image data keeps appearing sharp because clutter from fast object movement and jagged contours from slow object movement are reduced.

Chapter 5

Evaluation and Results

This chapter describes how the impact on the stereo matching accuracy of the proposed algorithm has been evaluated and presents the results of this evaluation. The evaluation is the crucial part of this work and reveals whether the intended improvement of the presented approach holds or if the presented ideas need further enhancement.

The asynchronous data interface has been evaluated by performing a ground truth comparison. For this evaluation, the output of stereo matching has been used for comparison, as the proposed approach claims to improve stereo matching accuracy for scenes with varying object movement velocity. It is evaluated in comparison with the current (synchronous) Address-Event conversion approach by computing the relative average error to the ground truth data.

5.1 Stereo Rig Calibration

If the same optical system is used for ground truth data acquisition as for capturing the test data, then no calibration is needed since the comparison can be performed directly with the disparity values and since both data depend on the same distortion parameters. If the ground truth data does not originate from the same optical system and the disparity values are back projected from absolute measures of length, as in this case, the stereo system needs to be calibrated. The distortion and camera parameters resulting from the calibration are used to undistort the captured test data, in order to be able to compare the data to the ground-truth model.

For this work, the *Camera calibration toolbox for matlab* [5] has been used to calibrate the optical stereo system. To establish a relation between image coordinates and world coordinates, extrinsic and intrinsic camera parameters need to be known. The *extrinsic parameters* determine where the camera is located in space and how it is oriented with regard to the world coordinate system. The *intrinsic parameters* determine the relation between image

coordinates (in pixels) and the camera coordinate system.

Varying sets of intrinsic camera parameter models can be found in current literature. This work adheres to the intrinsic camera parameters of [5] which are a modification of the ones described in [19].

Extrinsic Camera Parameters:

Rotation Parameters (θ, ϕ, ψ) – The 3 rotation angles *yaw, pitch and tilt* determine how the camera coordinate system is rotated with regard to the world coordinate system.

Translation Parameters (t_x, t_y, t_z) – The 3 translation parameters form a vector which determines the location of the camera's coordinate system origin with regard to the world coordinate system.

Intrinsic Camera Parameters:

Focal Length (f_x, f_y) – The *focal length* denotes the distance between the image plane and the center of projection (whether the image plane is in focus). The focal length determines the projection factor. In [5], the *focal length* is of unit pixels. If the image sensors pixels are squared then the horizontal and the vertical focal lengths are equal.

Principal Point (C_x, C_y) – The *principal point* coordinates determine the optical center of the lens, which is also the center of the radial lens distortion.

Skew (α) – The *skew* coefficient determines the angle between x and y axis of the image coordinates.

Radial Lens Distortion (k_1, k_2) – The 2^{nd} and 4^{th} order coefficients of the radial distortion polynomial are denoted by k_1 and k_2. Depending on the polarity of k_1, if positive the radial distortion results in barrel distortion (parallel lines are distorted radially out of the center, like the contour of a barrel) and pincushion distortion if negative (image borders are distorted radially to the center, like the contour of a cushion).

Tangential Lens Distortion (k_3, k_4) – The radial distortion is due to the shape of the lens, whereas, the tangential distortion on the other hand, is because of manufacturing inaccuracy resulting in the lens not being exactly parallel to the sensor plane [6].

The scale factors s_u and s_v, which are used in intrinsic camera parameter models in the current literature, are in this case already incorporated linearly in f_x and f_y. These scale factors are the conversion factors between the coordinates in pixel units and in metric units. The factors are in pixels per unit (resp. the reciprocal of the pixel pitch) [17].

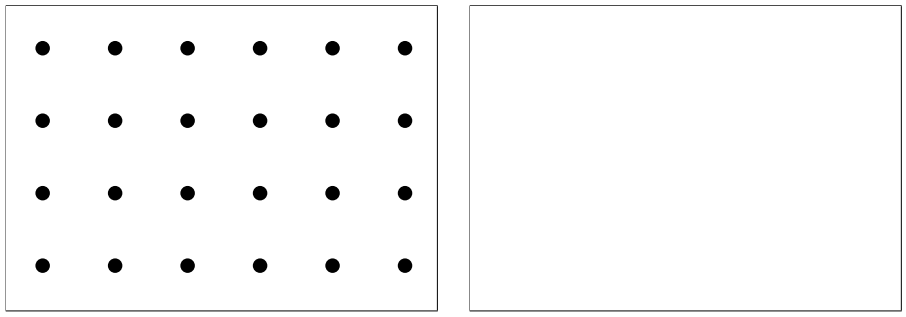


Figure 5.1: Calibration pattern for event-based vision sensors. (a) and (b) are the two frames of an infinite animation which has been used for calibrating the stereo rig.

Calibrating Stereo Event-based Vision Sensors: For camera calibration, a mapping between world coordinate distances and image space distances is required, in order to estimate the unknown extrinsic and intrinsic parameters. The mapping is established by capturing image data from a scene containing a calibration pattern with known world coordinates (e. g. a checkerboard pattern with known edge length) and with features detectable in the later image (e. g. corners or intersections).

The *Camera calibration toolbox for matlab* [5] uses the Harris corner detector [16], which detects local curvature maxima by approximating the eigenvalues of the second-moment matrix, which are used for detecting the calibration pattern points.

For the calibration of event-based vision sensors, a checkerboard pattern could not be used since only changes in intensity can be captured. After some experimentation, a LCD monitor displaying an animation of vanishing and reappearing black circles turned out to be a usable calibration pattern. Figure 5.1 illustrates the frames of the calibration pattern used. The black circles are arranged within the frame so that their resulting horizontal and vertical offset on the screen is exactly 5 cm.

[5] uses an iterative gradient descent optimization algorithm to estimate the intrinsic and extrinsic camera parameters from the detected points of the calibration pattern. Each optimization step decreases the reprojection error over all camera parameters.

The calibration pattern was captured from 24 different camera positions using the stereo rig. This means 24 calibration images for each camera were acquired, each one containing 24 calibration points, resulting in 576 calibration points in total per image sensor.

Table 5.1 lists the parameters estimated by the *Camera calibration toolbox for matlab* [5] that were used for undistorting the test image data. Figure 5.2

Table 5.1: Result of the stereo rig calibration

Left Camera – Intrinsic Parameters

Parameter	*Unit*	*Calibration result*	*Estimated uncertainty* ($\pm 3\sigma$)
(f_x, f_y)	[px]	(102.58, 102.56)	± (1.085, 1.086)
(C_x, C_y)	[px]	(59.46, 62.97)	± (0.652, 0.537)
(k_1, k_2)	[1]	(-0.2063, 0.1327)	± (0.00864, 0.01191)
(k_3, k_4)	[1]	(-0.0002, 0.0011)	± (0.00099, 0.00114)

Right Camera – Intrinsic Parameters

Parameter	*Unit*	*Calibration result*	*Estimated uncertainty* ($\pm 3\sigma$)
(f_x, f_y)	[px]	(103.07, 102.94)	± (1.082, 1.077)
(C_x, C_y)	[px]	(59.35, 68.58)	± (0.659, 0.590)
(k_1, k_2)	[1]	(-0.2136, 0.1436)	± (0.00941, 0.01468)
(k_3, k_4)	[1]	(0.0017, 0.0004)	± (0.00103, 0.00102)

Extrinsic Parameters

Parameter	*Unit*	*Calibration result*	*Estimated uncertainty* ($\pm 3\sigma$)
(θ, ϕ, ψ)	[°]	(0.77, -0.49, 0.80)	± (0.356, 0.328, 0.040)
(t_x, t_y, t_z)	[px]	(-129.44, -1.17, 0.65)	± (0.628, 0.573, 1.688)

illustrates the resulting radial distortion model from the calibration process. The tangential distortion model is depicted by Figure 5.3. Figure 5.4 shows the calibration set-up and illustrates the locations of the calibration patterns in world space.

5.2 Acquiring Ground Truth Data

In order to evaluate stereo matching performance, highly accurate disparity maps (or depth maps) are required as ground truth data. The following sections describe approaches considered for the evaluation of the proposed algorithm.

5.2.1 Structured Light

The most commonly used technique to acquire stereo matching ground truth data is *structured light* [43]. Structured light uses projected light patterns

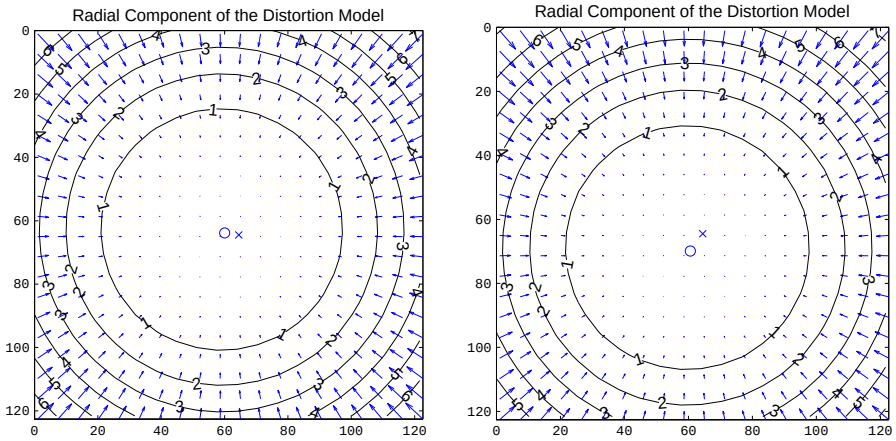


Figure 5.2: Radial distortion model of left (a) and right (b) image sensor. Generated with [5]

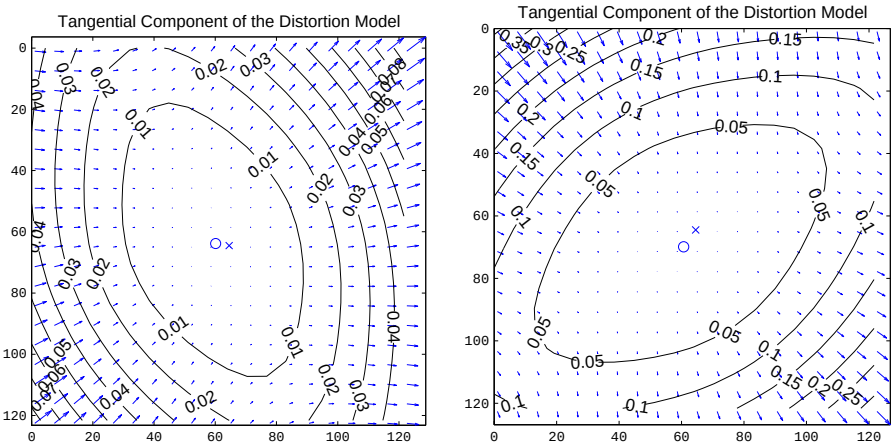


Figure 5.3: Tangential distortion model of left (a) and right (b) image sensor. Generated with [5]

to solve the correspondence problem of stereo vision. For example, [43] describes projecting a series of black and white stripe patterns onto the scene, such that each projection denotes one bit of the resulting binary code. The latter correspondence search is reduced to finding pixels with equal code in the left and right image. Figure 5.5 illustrates the principle of the described structured light pattern.

The drawback of the structured light method is that it is only applicable to static scenes, hence this technique is not a feasible solution for acquiring

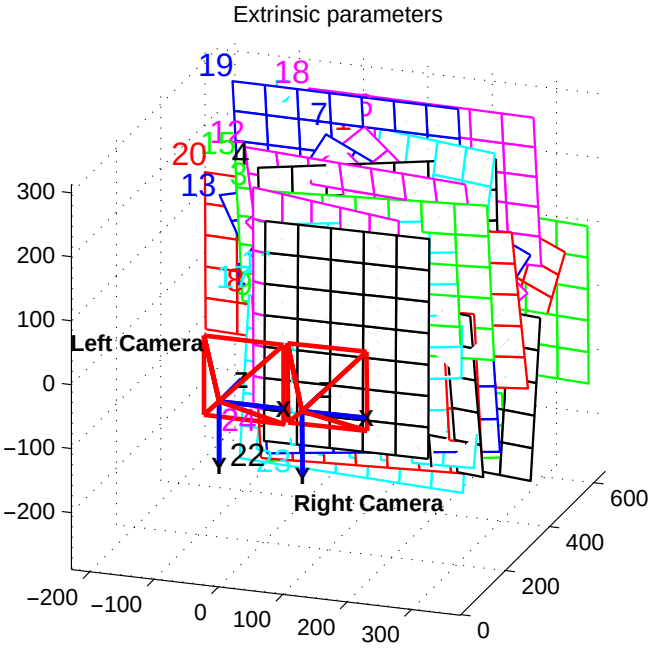


Figure 5.4: Estimated calibration set-up (extrinsic parameters). Generated with [5]

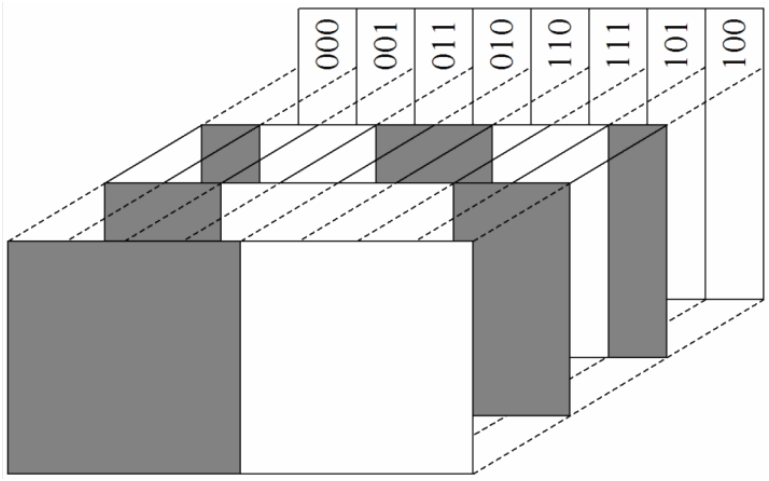


Figure 5.5: Structured light, binary projection pattern. From http://en.wikipedia.org/wiki/File:13-stripes-s.png.

ground truth data for event-based stereo vision.

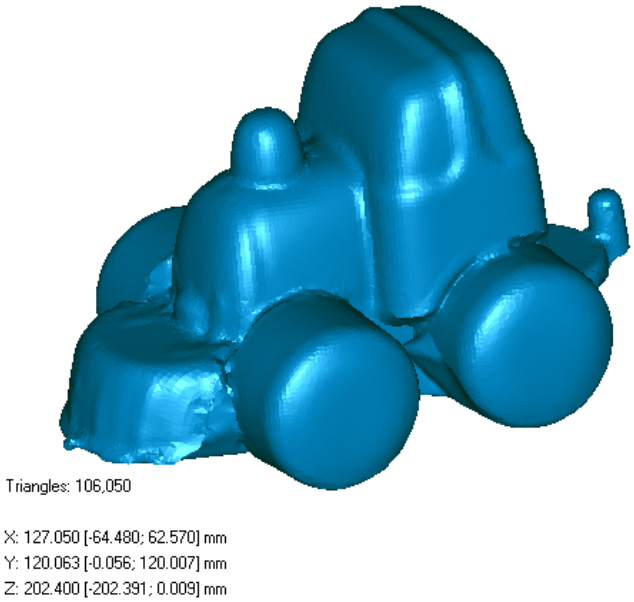


Figure 5.6: 3D digitized toy car model.

5.2.2 Calibrated Object

The Structured Light technique is not applicable due to required the dynamics of the scene. Since something needs to be moving in order to be visible for event-based vision systems, moving objects with known geometry is one way to acquire ground-truth data.

For this work, the engine part of a toy train was used for a controllable movement of objects. Toy trains allow reproducible test runs, as they follow a fixed track and their velocity is configurable. As claimed, the proposed algorithm is more robust in terms of varying object movement velocity, such that test data with varying velocity is required.

First, a plastic toy car was used for the ground-truth evaluation, because its geometry is smooth and not too complex. It was scanned with a laser-beam triangulation 3D digitizer. The captured point cloud was later converted into a 3D triangle mesh. Figure 5.6 illustrates the 3D model of the toy car.

From this 3D triangle mesh, a depth map (resp. disparity map) needs to be generated. [34] describes how the ray tracing program *POV-Ray* can be used to render 3D scenes to a depth map. The depth map generated with *POV-Ray* can be converted into a disparity map by transforming Equa-

tion 5.1 into Equation 5.2.

$$Z = \frac{f \cdot B}{d} \tag{5.1}$$

$$d = \frac{f \cdot B}{Z} \tag{5.2}$$

The problem with this test object was that due to imperfect alignment of the ground-truth model with the captured test data, a feasible comparison was not possible. The event-based vision sensor only generates image data if the light intensity changes and therefore only object or texture boundaries are visible in the image data. Object boundaries, more precisely depth discontinuities, are the regions with the highest depth resp. disparity variance. The consequence would be a failed comparison, if the difference between the expected object position of the ground-truth model and the captured scene were to lead to a displacement of the object locations in the disparity maps of even one pixel. Initial tests led to the result that this high accuracy requirement could not be reached, and therefore a simpler test object was required.

5.2.3 Rectangular Object

An object with axis parallel surfaces (camera coordinate system axis) should overcome the above described problem, since small registration inaccuracy between ground-truth and test data does not lead to high disparity map differences. A pyramid assembled of black and white Lego® bricks was chosen.

The reason for using black and white bricks can be attributed to the fact that their high contrast results in strong edges in the event-data stream. The previously described toy train was also used to move the object for this evaluation attempt. Intensity changes that are parallel to the movement direction do not generate event data, and therefore the pyramid was mounted on the toy train so that one of the diagonals of the pyramid points in the movement direction.

POV-Ray was used again to generate the ground-truth disparity maps, but this time the 3D model was created manually, using a 3D modeling software (see Figure 5.7 and Figure 5.8). In order to reduce errors caused by poorly predicting the pyramid position in 3D space, the following simplifications of the test data acquisition scene set up was applied. The event-based stereo vision sensor was mounted above the train track, so that the pyramid would move parallel to the stereo system trough the field of view and the train moves in the direction of the stereo system's y axis. Additionally, the event-based stereo vision system was positioned so that the left image sensor was exactly in the middle of the track, resp. the pyramid peak was in the middle of the left sensors field of view. The reason for this decision was,

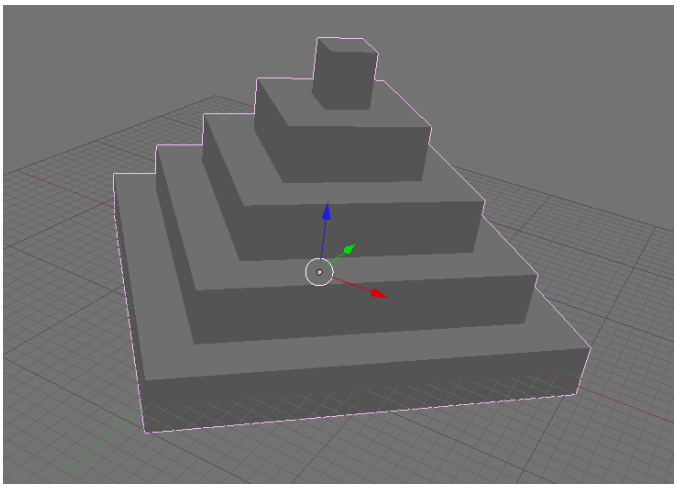


Figure 5.7: Pyramid 3D model.

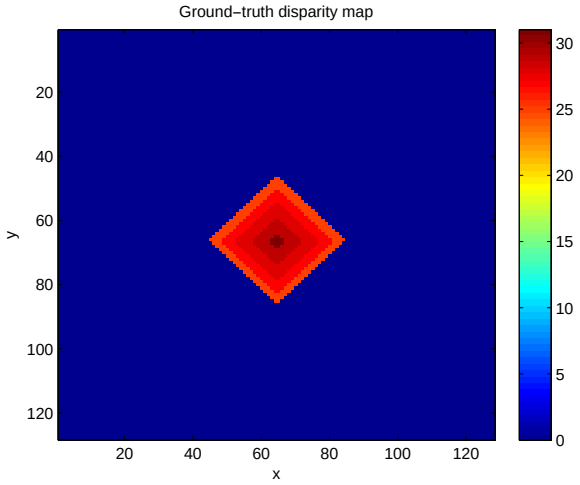


Figure 5.8: Ground-truth disparity map of the pyramid 3D model.

in this work the disparity map is based on the left sensor image, describing the stereo disparity to the right sensor image. In other words, the resulting disparity map denotes the same perspective as the left sensor image. Hence, with the pyramid peak in the middle of the left sensor's field of view (resp. the optical center) the ground-truth model is easier to predict as the pyramid is captured from the left sensor's top view perspective. Figure 5.9 displays how the stereo rig was set up.

In order to acquire a statistically representative amount of test data, the data was acquired in 2 overnight recording sessions: one moving the pyramid

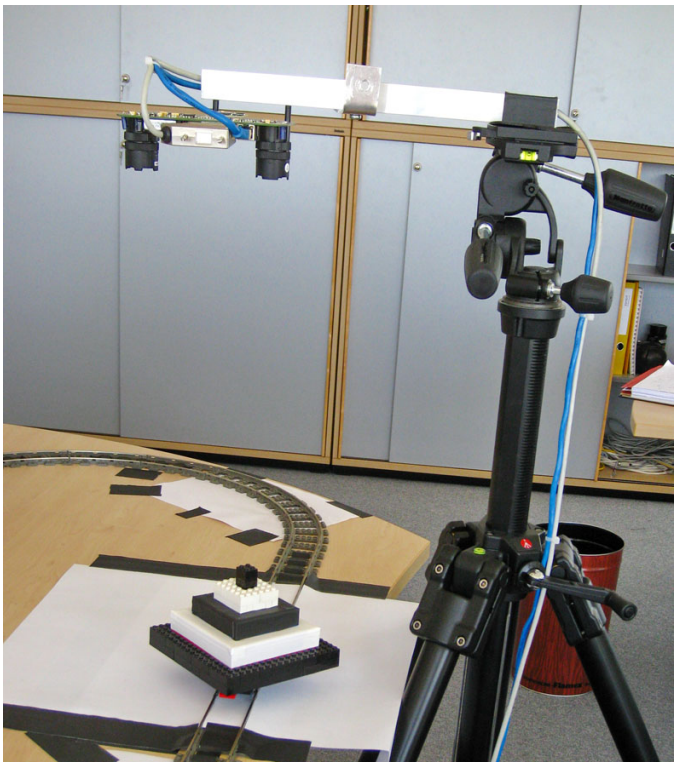


Figure 5.9: Stereo rig with event-based stereo vision sensor used for capturing test data.

slowly and one moving the pyramid quickly, each running for 7 hours. The acquisition occured overnight in order to minimize influence of changing light conditions.

5.3 Evaluation

The evaluation part proves that the goals were achieved and that the proposed improvements hold.

The test data and the ground-truth data was acquired using the methods previously described. This data is the basis for the evaluation. As described in the introduction, the aim of this work is to improve stereo matching accuracy for scenes with varying object movement velocity.

In order to evaluate whether the goals were achieved, the comparison between the test data and the ground-truth data was performed first using the currently used data interface (periodic frame conversion) and afterwards using the proposed asynchronous data interface. The error rate of each approach was used for comparison. The evaluation was executed for test data with a high object movement velocity and low object movement velocity.

First, the frames which were to be used for comparison were extracted. Therefore, only the frame for which the object was exactly in the center of the stereo systems field of view when the moving object passed by, was selected for each run. This is required in order to generate test data that is precisely aligned to the centered pyramid of the ground truth data. The frame is selected by evaluating the number of events over time from the event stream. This function denotes an equally distributed amount of noise events, a rising edge as the pyramid enters the field of view, a peak as the pyramid passes through the optical center, and a falling edge as the pyramid leaves the field of view. This event histogram, due to noise events, is not a smooth function. Initial tests showed that the function peaks did not conform with the index of the center frame of the train resp. pyramid pass. Therefore, it was decided to use the middle frame between the rising edge and the falling edge.

The middle frame was identified by applying a threshold to the function, which is higher than the event rate between the train resp. pyramid passing by (noise event rate) and lower than the event rate as the train was passing through the field of view. The first derivative of this binary function denotes 1 for each rising edge and -1 for each falling edge. The left and right frames, located at the middle between each rising and falling edge, could be extracted for comparison. The disparity maps of the test data frames were computed by stereo matching the left and right frames.

The resulting disparity maps still display a variation in the pyramid center location in the direction of movement. The disparity maps were cropped to the pyramid's bounding box, in order to center the pyramid. Therefore, a binary mask of the ground-truth image was used as a convolution kernel. The peak of the convolution result denotes the pyramid center in the computed disparity map. The disparity maps were cropped with these convolution peaks as the window center. This way, the small offsets of the pyramid location were compensated. Additionally, the binary mask of the ground-truth disparity map was applied on the cropped disparity maps in order to eliminate image data resulting from noise events.

The ground-truth disparity computed from the depth map rendered by POV-ray had decimal number values but the result from stereo matching were integer values (pixels). Therefore, the ground-truth disparity map needed to be rounded to integer values. In order to reduce evaluation error due to rounding errors, 1 pixel tolerance was introduced by applying the *floor* and the *ceiling* function to the decimal disparity map resulting in two ground-truth disparity maps.

The performance of each disparity map was measured by dividing the number of pixels used for comparison by the number of pixels which correspond to one of the two integer ground-truth disparity maps.

Table 5.2: Evaluation result

Data interface	*Velocity*	*Frames*	*Avg. pixel*	*Performance*	σ
Periodic	slow	925	379.51	70.22 %	5.28 %
Periodic	fast	5721	676.32	72.40 %	4.45 %
Asynchronous	slow	925	336.66	68.86 %	5.24 %
Asynchronous	fast	5721	419.09	72.25 %	5.26 %

5.4 Results

Table 5.2 shows the result of the evaluation process. The column *Data interface* contains the type of data interface which was used for this comparison. The type *Periodic* denotes the data interface currently used which renders the Address-Event data in a synchronous manner. *Asynchronous* denotes the type of data interface that is proposed in this work. The column *Velocity* denotes from which test run (*slow* or *fast*) the data originated. The *Frames* column contains the number of disparity maps that are computed from the Address-Event data and used for comparison. Each disparity map denotes a pass of the toy train moving the pyramid through the stereo systems field of view. Since each acquisition session was of the same duration, more disparity maps were considered for evaluation of the fast runs. The *Avg. pixel* column denotes the average number of pixels per disparity map which are defined in the disparity map, in other words, the number of pixels for which the stereo matching process of the input frames delivered a disparity value. The effect of the asynchronous interface is visible, since the average number of pixels for the fast run using the asynchronous data interface decreased to a level close to the average number of pixels of the slow runs. The column *Performance* denotes the average percentage of correct disparity values per disparity map. The σ column denotes the standard deviation of the percentage of correct disparity values.

The presented result does not reveal statistical significant improvement in stereo matching accuracy for the presented asynchronous data interface. The following chapter states an interpretation as to why no proof for an improvement was measurable by the performed evaluation.

Summary

This chapter described why and how stereo acquisition systems are calibrated and how this can be performed on stereo event-based vision sensors using an LCD monitor displaying blinking circles. The second section compared different ground-truth data acquisition approaches. Since the structured light method is only feasible for static scenes, the acquisition of ground-truth data

by moving a 3D digitized test object along the track of a toy train was evaluated. The alignment of the test data with the digitized ground-truth model turned out to be unstable, which led to using a more simple test object. The use of a pyramid assembled out of Lego bricks allowed a feasible comparison because small registration inaccuracies do not lead to high disparity map differences. The third section described how the error rates of the periodic frame conversion approach and the presented asynchronous interface compared to the ground-truth data were computed. The last section presented the stereo processing error rates for each data interface, compared to the ground-truth data.

All in all, this chapter demonstrated that acquiring ground-truth data for event-based stereo processing is challenging. The bottom line is that the presented result denotes an asynchronous data interface of comparable stereo matching accuracy to the state of the art approach, but it did not reveal statistical significant improvement in stereo matching accuracy.

Chapter 6

Conclusion and Outlook

The following sections provide an interpretation of the results presented in Chapter 5 and suggest possible further research based on the achievements of this work.

6.1 Conclusion

As described in the first chapter, this work targeted two goals. First, an asynchronous data interface for event-based vision data was to be designed, as the current state of the art only describes the processing in a synchronous manner. This goal was achieved with at least equally good stereo matching performance. The second goal of this work was the improvement of stereo matching accuracy for scenes with varying object movement using the implemented asynchronous data interface.

The proof for the second goal, the improved stereo matching accuracy, was not evident in the presented evaluation result. However, the result implies that stereo matching accuracy did not decrease when the presented asynchronous data interface was used. Hence, the first goal, the asynchronous data interface, was achieved with comparable stereo matching accuracy to the state of the art Address-Event conversion approaches.

Why has stereo matching accuracy not improved?

As Table 5.1 shows, the performance even increased for the test runs with high velocity, independent of the data interface used. This contradicts the expected effect of high velocity object movement on the quality of the input image data for stereo matching. On the other hand, the visual examples of Address-Event data rendered with too high a frame duration for the captured object movement velocity, demonstrate a loss in image detail due to the clutter resulting from the overaccumulation of events. The outcome when using too low a frame duration for conversion is even worse regarding the resulting image data quality.

These arguments lead to the assumption that the presented evaluation approach is not able to stress the consequences of the effects in order to unveil the benefits of the presented approach.

Clutter introduced because of fast motion (or high frame duration) results in edges smearing in the direction of motion. Due to this, edges become noisy areas. Since this kind of clutter is consistent in both (left and right) images and the chosen objects surfaces are parallel to the axis, results in correct disparity value computed from the cluttered stereo event data.

Moreover, due to the before mentioned smearing effect for fast motion, the number of pyramid surface pixels used for ground-truth comparison increases compared to the number of pyramid edge pixels. This also explains the slightly increasing stereo matching performance for higher object movement velocity, as depth discontinuities, like at the pyramid edges, are error prone for stereo matching.

As a conclusion it can be assumed that the test object geometry simplification necessary in order to retrieve comparable test data reduced the effect of varying object movement velocity on the performance of stereo matching. The improvement of the asynchronous data interface would be measurable in scenes where the clutter introduced by fast motion covers up detail behind it, which would furthermore result in different disparity values. Therefore, more complex scenes, which, on the other hand, are harder to evaluate (as described in the previous chapter), are required.

6.2 Possible follow up Research

Both a fundamental investigation into the acquisition of highly accurate ground truth disparity maps for dynamic scenes of arbitrary complexity and comparison approaches of this ground-truth data with Address-Event data would be an interesting topic for further research. The implementation of a more complex evaluation approach exceeded the scope of this work, and therefore working on the scale of this larger scope would be possible in follow up work.

Furthermore, computation performance of the presented approach was not within the scope of this investigation. The presented approach was only prototyped using Matlab® and therefore is not optimized for real-time processing. Computation optimization of the current approach and the implementation of it on a platform suitable for real-time applications would furthermore be interesting for further analysis.

6.3 Personal Experience

During the execution of this project, I gained some interesting insight into a novel computer vision resp. image sensing technique. Working with cut-

ting edge technology, like event-based vision sensors, was really fascinating and I am curious about the future development of this technique and its applications. On the other hand, it was also challenging to work on a topic where the research community is not large enough yet, as literature covering a broader diversity would have been helpful.

Additionally, acquiring profound experience in implementing stereo vision concepts and learning the (current) limits to these concepts, was exciting and will certainly be a valuable experience.

Rather unexpected for me was the lesson learned that the evaluation part can sometimes be the most challenging part. I think the awareness of this is beneficial for the execution of all kind of projects.

In a nutshell, I am happy with the outcome of my work, even though not all goals were accomplished.

Bibliography

[1] Ambrosch, K., M. Humenberger, S. Olufs, and S. Schraml: *Embedded stereo vision.* In Belbachir, A.N. (ed.): *Smart Cameras*, ch. 8, pp. 137–157. Springer, Berlin, DE, 2009.

[2] Barthlott, W., Z. Cerman, and A.K. Stosch: *Der Lotus-Effekt Selbstreinigende Oberflächen und ihre Übertragung in die Technik.* Biologie in unserer Zeit, 34(5):290–296, September 2004.

[3] Belbachir, A.N. and N. Brändle: *Real-time classification of pedestrians and cyclists for intelligent counting of non-motorized traffic.* In *Proceedings of the Intl. Conference on Computer Vision and Pattern Recognition*, pp. 45–50, San Francisco, US, June 2010.

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

[5] Bouguet, J.Y.: *Camera calibration toolbox for matlab*, May 2004. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[6] Bradski, G. and A. Kaehler: *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly, Sebastopol, US, 2008.

[7] Chan, V., S.C. Liu, and A. van Schaik: *AER ear: A matched silicon cochlea pair with address event representation interface.* IEEE Transactions on Circuits and Systems I: Regular Papers, 54(1):48–59, January 2007.

[8] Conradt, J., M. Cook, R. Berner, P. Lichtsteiner, R. Douglas, and T. Delbruck: *A pencil balancing robot using a pair of aer dynamic vision sensors.* In *Proceedings of the Intl. Symposium on Circuits and Systems*, pp. 781–785, Taipei, TW, May 2009.

[9] Cyganek, B. and J.P. Siebert: *An Introduction to 3D Computer Vision Techniques and Algorithms.* Wiley, West Sussex, UK, 2009.

[10] Delbrück, T.: *Frame-free dynamic digital vision.* In *Proceedings of the Intl. Symposium on Secure-Life Electronics*, pp. 21–26, Tokyo, JP, March 2008.

[11] Delbrück, T. and P. Lichtsteiner: *Freeing vision from frames.* The Neuromorphic Engineer, 3(1):3–4, May 2006.

[12] Delbrück, T., B. Linares-Barranco, E. Culurciello, and C. Posch: *Activity-driven, event-based vision sensors.* In *Proceedings of the Intl. Symposium on Circuits and Systems*, pp. 2426–2429, Paris, FR, June 2010.

[13] Fusiello, A., V. Roberto, and E. Trucco: *Efficient stereo with multiple windowing.* In *Proceedings of the Intl. Conference on Computer Vision and Pattern Recognition*, pp. 858–863, San Juan, PR, June 1997.

[14] Geim, A.K., S.V. Dubonos, I.V. Grigorieva, K.S. Novoselov, A.A. Zhukov, and S.Y. Shapoval: *Microfabricated adhesive mimicking gecko foot-hair.* Nature Materials, 2(7):461–463, July 2003.

[15] Häfliger, P. and F. Bergh: *An integrated circuit computing shift in stereo pictures using time domain spike signals.* In *Proceedings of the IEEE NorChip Conference*, Copenhagen, DK, November 2002.

[16] Harris, C. and M. Stephens: *A combined corner and edge detector.* In *Proceedings of the fourth Alvey Vision Conference*, pp. 147–151, Manchester, UK, September 1998.

[17] Hartley, R. and A. Zisserman: *Multiple View Geometry in Computer Vision.* Cambridge University Press, Cambridge, UK, second ed., 2004.

[18] Hecht, E.: *Optik.* Oldenbourg Wissenschaftsverlag, München, DE, 4th ed., 2005.

[19] Heikkilä, J. and O. Silvén: *A four-step camera calibration procedure with implicit image correction.* In *Proceedings of the Intl. Conference on Computer Vision and Pattern Recognition*, pp. 1106–1112, San Juan, PR, June 1997.

[20] Hirschmüller, H., P.R. Innocent, and J. Garibaldi: *Real-time correlation-based stereo vision with reduced border errors.* International Journal of Computer Vision, 47(1):229–246, April–June 2002.

[21] Hosni, A., M. Bleyer, M. Gelautz, and C. Rhemann: *Local stereo matching using geodesic support weights.* In *Proceedings of the Intl. Conference on Image Processing*, pp. 2069–2072, Cairo, EG, November 2009.

[22] Jedicke, P.: *Great Inventions of the 20th Century.* Chelsea House Pub., New York, US, 2007.

[23] Kandel, E.R., J.H. Schwartz, and T.M. Jessell: *Principles of Neural Science*. McGraw-Hill, New York, US, 4th ed., 2000.

[24] Kogler, J., C. Sulzbachner, and W. Kubinger: *Bio-inspired stereo vision system with silicon retina imagers*. In *Proceedings of the Intl. Conference on Computer Vision Systems*, pp. 174–183, Liège, BE, October 2009.

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

[26] Kolb, H., E. Fernandez, and R. Nelson: *Webvision: The Organization of the Retina and Visual System*. National Library of Medicine, Salt Lake City, US, 2007. http://webvision.med.utah.edu.

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

[28] Kramer, J.: *An on/off transient imager with event-driven, asynchronous read-out*. In *Proceedings of the Intl. Symposium on Circuits and Systems*, vol. 2, pp. II–165–II–168, Phoenix-Scottsdale, US, May 2002.

[29] Lichtsteiner, P.: *An AER Temporal Contrast Vision Sensor*. PhD thesis, Eidgenössischen Technischen Hochschule (ETH), Zürich, CH, 2006.

[30] Lichtsteiner, P., T. Delbrück, and J. Kramer: *Improved ON/OFF temporally differentiating address-event imager*. In *Proceedings of the Intl. Conference on Electronics, Circuits and Systems*, pp. 211–214, Tel Aviv, IL, December 2004.

[31] Lichtsteiner, P., C. Posch, and T. Delbrück: *A 128 × 128* 120 dB *30mW asynchronous vision sensor that responds to relative intensity change*. In *Proceedings of the Intl. Solid-State Circuits Conference*, pp. 2060–2069, San Francisco, US, February 2006.

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

[33] Lynch, J.C., J.J. Corbett, and J.B. Hutchins: *The visual system*. In Haines, D.E. (ed.): *Fundamental Neuroscience for Basic and Clinical Applications*, ch. 20, pp. 311–333. Elsevier, Philadelphia, US, 3rd ed., 2005.

[34] Mackay, D.: *Generating synthetic stereo pairs and a depth map with PoVRay*. Techn. rep., Defence Research and Development Canada, Suffield, CA, December 2006.

[35] Mahowald, M.: *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function.* PhD thesis, California Institute of Technology, Pasadena, US, 1992.

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

[37] Mead, C.: *Analog VLSI and neural systems.* Addison Wesley, Boston, US, 1989.

[38] Posch, C.: *Detectors, pixels, and signal processing.* In Belbachir, A.N. (ed.): *Smart Cameras*, ch. 4, pp. 53–80. Springer, Berlin, DE, 2009.

[39] Posch, C., D. Matolin, and R. Wohlgenannt: *An asynchronous time-based image sensor.* In *Proceedings of the Intl. Symposium on Circuits and Systems*, pp. 2130–2133, Seattle, US, May 2008.

[40] Posch, C., D. Matolin, and R. Wohlgenannt: *High-DR frame-free PWM imaging with asynchronous AER intensity encoding and focal-plane temporal redundancy suppression.* In *Proceedings of the Intl. Symposium on Circuits and Systems*, pp. 2430–2433, Paris, FR, June 2010.

[41] Posch, C., D. Matolin, and R. Wohlgenannt: *A QVGA 143 dB dynamic range asynchronous address-event pwm dynamic image sensor with lossless pixel-level video compression.* In *Proceedings of the Intl. Solid-State Circuits Conference*, pp. 400–401, San Francisco, US, February 2010.

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

[43] Scharstein, D. and R. Szeliski: *High-accuracy stereo depth maps using structured light.* In *Proceedings of the Intl. Conference on Computer Vision and Pattern Recognition*, pp. 195–202, Madison, US, June 2003.

[44] Schraml, S. and A.N. Belbachir: *A spatio-temporal clustering method using real-time motion analysis on event-based 3D vision.* In *Proceedings of the Intl. Conference on Computer Vision and Pattern Recognition*, pp. 57–63, San Francisco, US, June 2010.

[45] Schraml, S., A.N. Belbachir, N. Milosevic, and P. Schön: *Dynamic stereo vision for real-time tracking.* In *Proceedings of the Intl. Symposium on Circuits and Systems*, pp. 1409–1412, Paris, FR, June 2010.

[46] Schraml, S., N. Milosevic, and P. Schöen: *Smarteye StereoVision Sensor: Intelligente Kamera für Echtzeit Stereo Vision.* Tech. Rep. ARC-IT-0203, AIT Austrian Institute of Technology, Vienna, AT, March 2007.

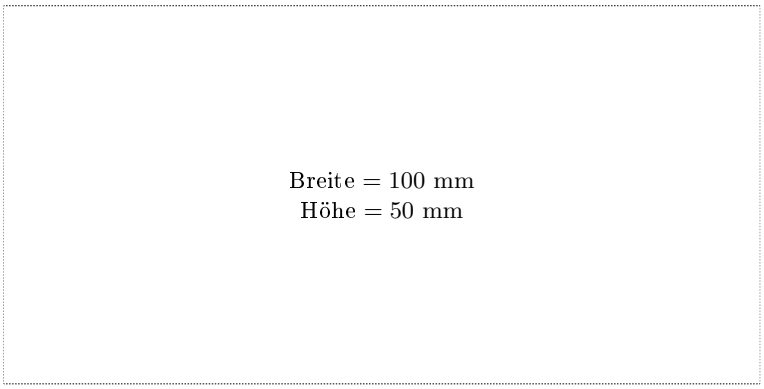
[47] Schraml, S., P. Schön, and N. Milosevic: *Smartcam for real-time stereo vision - address-event based embedded system.* In *Proceedings of the Intl. Conference on Computer Vision Theory and Applications*, pp. 466–471, Barcelona, ES, March 2007.

[48] Schreer, O.: *Stereoanalyse und Bildsynthese.* Springer, Heidelberg, DE, 2005.

[49] Yoon, K.J. and I.S. Kweon: *Locally adaptive support-weight approach for visual correspondence search.* In *Proceedings of the Intl. Conference on Computer Vision and Pattern Recognition*, pp. 924–931, San Diego, US, June 2005.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —