Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

Event-Based Depth Reconstruction Using Stereo Dynamic Vision Sensors

Lukas Everding

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Werner Hemmert

Prüfende der Dissertation:
1. Prof. Dr.-Ing. Jörg Conradt
2. Prof. Dr.-Ing. Darius Burschka

Die Dissertation wurde am 25.06.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 31.10.2018 angenommen.

Abstract

Depth information is used in a variety of technical applications nowadays. Well-known examples are autonomous driving and driver assistance or robots working independently in factories and warehouses. One natural way of reconstructing depth of objects in the environment is stereo vision. The main challenge in stereo vision is the matching problem, i.e. identifying the points on two images that are projections of the same point in the world. Although solving the matching problem has been subject to much research effort, achieving low latency and a high temporal resolution remains a challenging problem. It requires high-frame rate cameras and very powerful computers which are potentially costly, bulky and power demanding. Overall, this is not desirable or even viable for autonomous systems. This thesis tackles the problem of low latency by utilizing neuromorphic sensors for capturing visual data. Event-based vision sensors operate in an asynchronous fashion, with very low-latency updates. The only information they convey is where and exactly when illumination changes occur. This reduces data rates and opens up the possibility of solving the stereo matching problem and computing depths in a light-weight, real-time capable way. The structure of event-based data is, however, very different from conventional frame data, and new algorithms are required to solve computer vision tasks.
In this thesis different algorithms for event-based stereo matching are developed, evaluated and compared. It investigates local, neighborhood correlation based algorithms, global event-cloud registration methods and feature-based matching. In order to execute the latter, a fast and robust feature extractor for event streams, which detects and tracks lines, is developed and its accuracy and persistence are evaluated. Based on the tracker, a line matching method is introduced, that allows low latency stereo matching even in cluttered scenes and in the presence of noise.
In the final part, a potential application for event-based stereo vision outside of robotics is explored: a mobility aid for visually impaired people. Depth-labeled events are translated to virtual three dimensional sounds that deliver the information about the surrounding environment to the user. The performance of the device is evaluated and compared to other available electronic travel aids.

Zusammenfassung

Viele moderne technische Anwendungen nutzen Entfernungs- und Tiefeninformation. Zu den bekannteren Beispielen zählen autonome Fahrzeuge, Fahrassistenz und Roboter in Lagerhäusern und Fabriken. Eine natürliche Methode Entfernungen zu schätzen ist räumliches Sehen, auch stereoskopisches Sehen genannt, die Beobachtung der Umgebung mit zwei gegen einander verschobene Sensoren. Das zentrale Problem, das es bei der Anwendung von räumlichen Sehen zu lösen gilt, ist das Korrespondenzproblem, d.h. zu bestimmen, welche Punkte in der Bildebene der jeweiligen Sensoren zum selben Punkt in der Welt gehören. Das Korrespondenzproblem ist seit langer Zeit Gegenstand der Forschung im Bereich des maschinellen Sehens. Hohe zeitliche Auflösung und geringe Latenzen zu erreichen, ist jedoch nach wie vor eine herausfordernde Aufgabe, zu deren Lösung hohe Bildraten und leistungsfähige Computer herangezogen werden müssen. Für autonome Systeme ist dies oftmals nicht möglich und andere Ansätze werden gebraucht. Diese Arbeit untersucht Möglichkeiten ein schnelles, stereoskopisches System mit Hilfe von ereignisbasierten Dynamic-Vision-Sensoren zu entwickeln. Dynamic-Vision-Sensoren unterscheiden sich grundlegend von konventionellen Kameras. Ihre Pixel arbeiten asynchron und unabhängig von einander. Das ermöglicht extrem geringe Latenzen in der Größenordnung von wenigen Mikrosekunden. Sie übertragen keine Information über die Lichtintensität, sondern lediglich über die Änderung der Intensität in Form von sogenannten Ereignissen. Dadurch wird die Datenrate stark reduziert, was die Möglichkeit eröffnet das Korrespondenzproblem mit vergleichsweise geringem Rechenaufwand in Echtzeit zu lösen. Die Datenstruktur der Dynamic-Vision-Sensoren unterscheidet sich jedoch fundamental von der konventioneller Kameras. Neue Algorithmen zur Lösung von Problem des maschinellen Sehens sind daher erforderlich.

In dieser Arbeit werden verschiedene Stereoalgorithmen für ereignisbasiertes maschinelles Sehen entwickelt, evaluiert und verglichen. Lokale, korrelationsbasierte Algorithmen, globale Ereigniswolkenregistrierungsmethoden und merkmalsbasiertes Abgleichen werden behandelt. Um letzteres zu ermöglichen wird eine neuartige, schnelle und robuste Methode entwickelt Linien in Ereignisströmen zu erkennen und zu verfolgen; ihre Genauigkeit und Verfolgungspersistenz werden ausgewertet. Dann wird die Methode verwendet um einen linienbasierten Algorithmus zu entwickeln, der das Korrespondenzproblem auch in komplexen, verrauschten Szenen mit geringer Latenz lösen kann.

Der letzte Teil der Arbeit konzentriert sich auf eine Anwendung außerhalb von autonomen System und Robotik. Ein elektronische Orientierungshilfe für Blinde wird entworfen. Dafür werden Ereignisse mit Entfernungsinformation versehen und dem Anwender als virtueller drei-dimensionaler Ton übertragen. Diese Töne übertragen die Information über die Umgebung. Der Gerät wird evaluiert und mit anderen elektorischen Orientierungshilfen verglichen.

Contents

Acronyms

AER	Address Event Representation.
APS	Active Pixel Sensor.
CPD	Coherent Point Drift.
CPU	Central Processing Unit.
DVS	Dynamic Vision Sensor.
EM	Expectation Maximization.
ETA	Electronic travel aid.
FAST	Features from accelerated segment test.
GMM	Gaussian Mixture Model.
HRTF	Head-related Transfer Function.
ICP	Iterative Closest Point Algorithm.
NCC	Normalized Cross Correlation.
PCA	Principal Component Analysis.
SAD	Sum of Absolute Differences.
SIFT	Scale-invariant feature transform.
SSD	Sum of Square Differences.
SURF	Speeded-up robust features.

1 Introduction

Visual perception is one of the primary ways we use to orient ourselves in unknown environments. The deduction of high level information from raw visual stimuli does not seem to pose a challenging problem for humans. We can easily identify objects and avoid obstacles when walking. Large parts of the brain are dedicated to processing visual information to enable us to perform such tasks effortlessly. In stark contrast, extracting information from cameras is a difficult problem for robots and computers that gave rise to a whole field of research: computer vision.

In a world that gets more and more automated it is important to create autonomous systems that are robust and safe so they can operate independently or in cooperation with humans. A central requirement is that these systems can sense and adopt to changes in their environment in a rapid and reliable way. One essential task in enabling autonomous systems to do so is to accurately estimate the distance to objects in the surroundings, so that a robot, for example, knows where to grasp, does not trip over obstacles and stops moving when a person gets in its way. This task - finding fast ways to reconstruct depth - is the objective of this thesis.

Nowadays, there are different types of depth detectors. They can be principally classified into active and passive. Active systems emit a signal and measure the environmental reaction to estimate depth. The principle used here is measuring the time of flight. The time it takes the emission, e.g. a laser pulse or an ultrasound wave, to be reflected back from an object is used to infer the distance to the object. Other techniques project patterns or structured light (usually in the infrared spectrum) and infer depth from the perceived deformation to those patterns. While being in widespread use and delivering depth maps of good quality, active sensing approaches share a number of drawbacks: they need to constantly emit a signal which requires a certain amount of power. The amount of power necessary increases with the desired distance to be detectable. Furthermore, if multiple systems of the same type are used in the same area, their signals potentially interfere, causing confusion in the distance measurement and reporting wrong values; the number of systems that can operate in the same area is, therefore, limited. Finally, these systems tend to be large, especially if they require a battery to be operational over a longer time.

The second class of depth sensors are passive. Passive depth sensors observe the environment without sending signals. They deploy cameras and use optic flow, parallax, structure from motion or stereo vision to estimate depth. In comparison with active sensor, passive sensors (i.e. cameras) are usually smaller, do not require as much power and do not project a signal to the environment, so multiple cameras can be used in the same area without interfering with each other.

The approach chosen in this work is a passive one, based on stereo vision. The common stereo setup consists of two parallel, coplanar cameras. They observe the same scene from a slightly shifted position. This shift causes points in the view of one camera to be horizontally displaced with respect to objects in the other camera. This displacement is referred to as *disparity*. The amount of disparity is dependent on the distance of the object to the cameras. So, by knowing the disparity of a point, its distance can be inferred ana-

lytically. The main task in stereo matching is therefore, finding correspondences between points in the two views, this is known as the *stereo matching problem*. Solving the stereo matching problem is still a challenging problem and the computational costs associated with solving it are often large. In traditional computer vision, several techniques have been devised to solve the stereo matching problem. They range from local correlation based approaches over feature extraction and matching and global approaches like graph cuts (these methods will be introduces in more detail in Chapter 2). All these techniques operate on data from conventional CCD or CMOS camera. These cameras capture frames with a fixed update rate, i.e. every pixel will be read out synchronously and the visual information is transmitted as full image frame; this process happens independently from what is observed. But in many cases large parts of the observed scene change only very little between two frames resulting in the cameras to capture a lot of redundant data. All this data has to be processed by the subsequent computer vision system, essentially wasting a large amount of resources on processing redundant data. Considering the contrary case of parts of the environment that change very rapidly, all information about what happened in between frames is lost or causes motion blur; the pixels are only sampled at specific times. In conclusion, conventional cameras can be oversampling and undersampling at the same time. This is where event-based visions systems come into play. These systems capture visual information based on a entirely different principle. Every pixel works independently and only emits information (a so called *event*) when it perceives a change in illumination. There is no fixed sampling rate, so the information can be transmitted with a very low latency of the order of tens of microseconds. Data will only be generated when there is a change, otherwise pixels remain silent, which results in a sparse data rate and eliminates redundancies. The structure of data of conventional cameras and event-based sensors is fundamentally different: instead of a series of two-dimensional frames with fixed time intervals, data is conveyed as a quasi-continuous stream of vision events. This makes the development of new algorithms necessary, so the advantages of event-based sensors become accessible.

Objective and Outline

This thesis explores different algorithms for stereo matching in an event-based framework. The focus lies on developing possibilities to equip autonomous systems with real-time capable passive depth sensing, so not only accuracy of the results, but also computational costs will be important. This thesis is structured to guide the reader from the basics of computer vision to event-based sensors and stereo matching algorithms for them: Chapter 2 discusses a basic camera model, calibration and undistortion. It then explains the fundamental geometry of a stereo setup along with the problems that are to be solved and argues that depth estimation is equivalent to solving the stereo matching problem. Chapter 3 shows solution from conventional computer vision and discusses shortcomings of the cameras. It then presents the event-based vision sensors, gives an overview over the current state of the art in event-based stereo matching and summarizes related work. Chapter 4 and Chapter 5 form the main contribution of this thesis; they introduce the algorithms developed towards solving stereo matching in event-based cameras. First, algorithms that work on the level of single events or unstructured groups of events are analyzed. Then, a method for feature extraction from event streams is developed. Finally, a method for stereo matching in feature space is presented and analyzed. Chapter 6 shows how the research of this thesis can have an impact on society. Finally, Chapter 7 con-

cludes this work by summarizing the results and relating them to current state of research.

Figure 1.1: A stereo vision setup consisting of two event-based vision sensors in a rigid frame

2 Preliminaries

This chapter gives a brief introduction to the essential terms and concepts on which the thesis is based. It covers the mathematical description of cameras, their calibration and image rectification, then continues with the fundamental geometry of a stereo setup and established stereo matching algorithms from conventional computer vision, which serve as a partial basis for some of the ones developed for event-based sensors. Finally, event-based sensors are portrayed and compared to conventional cameras and their properties.

2.1 A mathematical camera model

A camera is an object that projects the three dimensional world onto a two-dimensional plane, the image plane. Mathematical descriptions of these projections are knows as camera models. There exists a variety of different camera models of which a basic one often used in computer vision is the *pinhole model*. We will briefly introduce this model here loosely following the more detailed explanation to be found in [1]. The pinhole model is a linear model of projection; all rays, originating from points in the world, are projected through one point, the *center of projection* $\mathbf{c}$. For our consideration, three frames of reference are of interest: the world coordinates $\mathbf{x}_w = (x_w, y_w, z_w)^T$ that describe the world independently of the camera and its parameters, the camera coordinates $\mathbf{x}_c = (x_c, y_c, z_c)^T$ in which the camera is at rest and the center of projection lies at the origin, and the image coordinates $\mathbf{p} = (u, v)^T$ which describe positions within the image plane. Let us begin with the transformation between the world coordinates and the camera coordinates. The camera has an arbitrary position in the world, therefore a general Euclidean transformation is needed. It consists of a rotation matrix R and a shift vector $\mathbf{t}$.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + \mathbf{t} \tag{2.1}$$

We are now in the camera-centric frame of reference. Two geometric structures deserve special attention in this coordinate system. First, consider the plane parallel to the x-y-plane at $z = f$, with constant $f \in \mathbb{R}^+$ we will call this plane the *image plane*. Second, the axis that passes through the center of projection and stands perpendicular on the image plane; it is known as *optical axis*. To map a three dimensional point $x_c = (x_c, y_c, z_c)^T$ to the image plane the intersection of the line connecting x_c with c is computed: x_c is mapped to $(fx_c/z, fy_c/z, f)^T$. By ignoring the z coordinate that is always f by construction, we arrive at the camera projection from 3D to 2D:

$$(x, y, z)^T \mapsto (fx/z, fy/z) \tag{2.2}$$

Note, that dividing by z is a non-linear operation. In order to write the transformation in a linear fashion a new set of coordinates is required.

2.1.1 Homogeneous coordinates

A point in d-dimensional Euclidean space is represented by a vector of size d. We now introduced *homogeneous coordinates*, in which an extra dimension is added. By definition the same point is represented by the homogeneous coordinate that contains 1 as $(\mathrm{d}+1)^{\mathrm{th}}$ value, i.e. the three dimensional point (x, y, z) in Euclidean coordinates and $(x, y, z, 1)$ in homogeneous coordinates are identical. We additionally define that all coordinate tuples that differ only in a scalar factor represent the same point, i.e. $(x, y, z, 1)$ and $(2x, 2y, 2z, 2)$ (or $(kx, ky, kz, k), k \neq 0$ in general) are coordinates for the same point. To transform a point (kx, ky, kz, k) back to a Euclidean coordinate system, we need to divide by k and prune the last dimension[1].

Let us now reconsider the mapping 2.2. It can be written as simple linear transformation when using homogeneous coordinates:

$$\begin{pmatrix} fx_c \\ fy_c \\ z_c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} \tag{2.3}$$

This transformation so far assumes that the intersection of the image plane with the optical axis (the *principal point*) coincides with the origin of the image plane coordinates $p = (u, v)^T$, which in practice may not be the case. Therefore, a shift of the mapping by the coordinate of the principal points is applied, such that the general mapping expressed in homogeneous coordinates becomes:

$$\begin{pmatrix} fx_w + z_w p_x \\ fy_w + z_w p_y \\ z_w \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{2.4}$$

The left 3×3-block of the transformation matrix is called the *camera calibration matrix*. This matrix contains the *intrinsic* parameters f, p_x, p_y which will be estimated later to calibrate the camera.

$$K = \begin{pmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

The complete mapping from world points to image point is now concisely described as linear transformation in homogeneous coordinates:

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \left(\begin{array}{c|c} K & \mathbf{0} \end{array} \right) \left(\begin{array}{c|c} R & t \\ \hline \mathbf{0} & 1 \end{array} \right) \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}, \tag{2.6}$$

where horizontal and vertical lines separate matrix blocks. The image coordinates are given as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x_i / z_i \\ y_i / z_i \end{pmatrix} \tag{2.7}$$

[1] This is not possible if $k = 0$; homogeneous coordinates where $k = 0$ are used to represent and distinguish so called points at infinity. They are not used further in this thesis, so we will omit them here.

2.1.2 Undistortion and Calibration

The pinhole model discussed above is a simple model for theoretical analysis, it does not describe the imaging properties of real cameras properly. The model assumes an point-shape (extensionless) pinhole through which all rays travel. Real camera lenses, however, have finite aperture, which makes the ray courses imperfect with respect to the pinhole model. The most prevalent deviation from the model is radial distortion. This type of distortion is symmetric around a center of distortion, image areas are magnified in dependence of their distance from the center of distortion. This can be compensated for by a correction process that transforms the measured image to the image that would have been measured using a perfect pinhole camera. After undistortion, the data from the physical camera can be treated as if it was generated by a pinhole camera.

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \tag{2.8}$$

The radial distortion depends only on the distance r from the center of distortion $(x_c, y_c)^T$.

$$\begin{aligned} \hat{x} &= x_c + L(r)(x - x_c) \\ \hat{y} &= y_c + L(r)(y - y_c) \end{aligned} \tag{2.9}$$

The function $L(r)$ is approximated by its Taylor expansion and the requirement $L(0) = 1$.

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \dots \quad , \tag{2.10}$$

where the first coefficients $\kappa_1, kappa_2, kappa_3$ and (x_c, y_c) are usually regarded as part of the internal camera parameters. These parameters are calculated using a scene with known geometry. The deviation of the observed points from the expected position of the points in a linear model is used as cost. The optimal parameters are estimated by minimizing the cost using the Levenberg–Marquardt algorithm, a non-linear least square minimization technique. Once the coefficients are obtained, the undistortion process is completed and the camera can be calibrated.

Calibration is the estimation of intrinsic and extrinsic camera parameters. Extrinsic parameters describe the position of the camera in world coordinates, while intrinsic parameters define the mapping from the camera coordinate system to image coordinate system as defined in Equation 2.1. Calibration is a crucial part for many computer vision applications and different methods for it exist. In this thesis, the method proposed by Zhang [2] is used. It uses a planar pattern with known geometry that is presented with different orientations and distances. Knowing the exact positions of the pattern, we can infer the parameters of the camera matrix by inverting the projective transformation given in 2.6. In practice, the undistortion and the calibration steps are done in parallel using the same pattern (see Figure 2.2).

2.1.3 Epipolar geometry

The previous sections dealt with the geometry and calibration of a single camera. Let us now extend the discussion to a stereo setup by adding a second camera. With the help of a second camera, we can triangulate the distance to objects from corresponding projection points. We require, however, knowledge of the exact geometry between the two cameras. This part of geometry, that deals with the relation of points on the image planes of two cameras, is known as *epipolar geometry*. Figure 2.1 gives an overview.

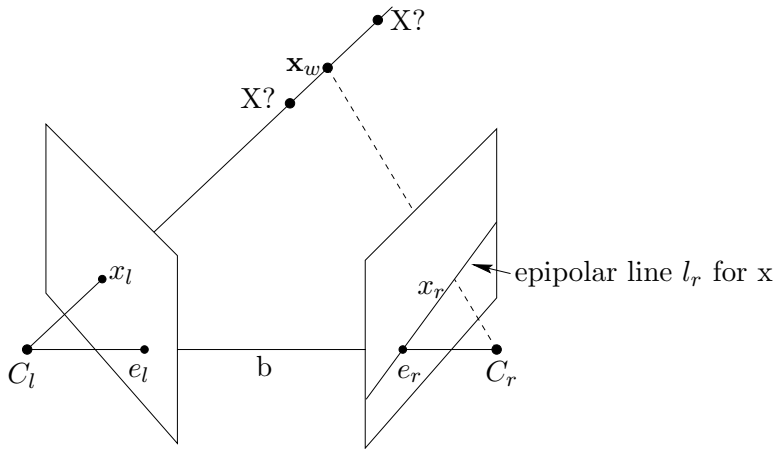


Figure 2.1: Epipolar geometry (Figure adapted from [1])

The two centers of projection C_l and C_r are separated by the baseline b. The respective intersection points of b with the image planes are called *epipoles*, e_l and e_r. A point in the physical world $\mathbf{x}_w$ projects to image points x_l and x_r respectively. These three points lie in a common plane π. This fact can be used to simplify the search for corresponding points: the intersection of π with the image planes is given as lines, called *epipolar lines*, l_l (and l_r respectively). Since x_l, x_r and $\mathbf{x}_w$ will always be coplanar, x_r will always lie on l_r, the epipolar line corresponding to x_L. Thus, knowing the epipolar line simplifies the search for the corresponding point significantly, the search is restricted to a one dimensional line instead of the whole two dimensional image. This reduces computational costs strongly.

The epipolar geometry between x_l and x_r can be algebraically described using the fundamental matrix F that maps every point in one image to its epipolar line in the other image: $Fx_r = l_l$ and vice versa (for a detailed derivation see [1]). It follows that the fundamental matrix satisfies the following equation for any pair of corresponding image points x and x':

$$x_l F x_r = 0 \quad , \tag{2.11}$$

since x_l lies on the epipolar line $l_l = Fx_r$. When the epipolar geometry is known, we can further simplify the matching problem by transforming the image planes, such that the epipoles lie at infinity and the epipolar lines become parallel to each other and to the horizontal axis. Then, finding the epipolar line to a point becomes trivial, it is the horizontal line with the constant vertical coordinate of the point, and computational complexity is reduced. This process is known as rectification and will be explained in the next section.

2.1.4 Stereo calibration and rectification

In practice it is hard to align to cameras perfectly parallel with the optical centers on the same height. There are inevitably small mechanical misalignments, in sensor, lens, casing and the like that cause imperfections. This will render the epipolar lines to not be parallel and complicate the matching problem since the epipolar line for every point that is to be matched has to be computed.Before approaching the matching, we first aim at mathematically transforming the image planes in a way, that they both lie within the same plane. To that end, we have to determine the spatial relationship of the cameras to each other, known as stereo calibration and then rectify the image planes. Stereo calibration estimates the rotation R_{stereo} and translation $\mathbf{t}_{\text{stereo}}$ between the two cameras. If there is

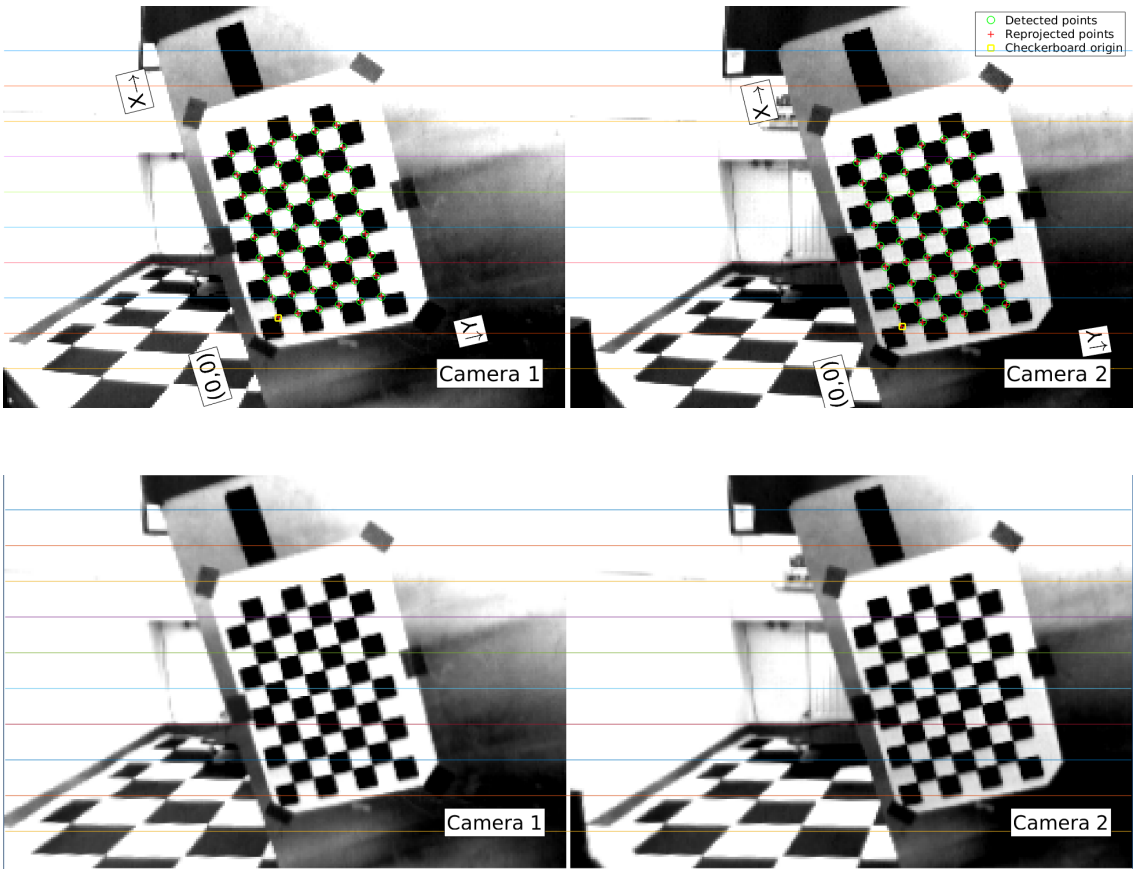


Figure 2.2: Checkerboard pattern with known geometry used for calibration (foreground). Top row: raw views from two DAVIS240C sensors with detected checkerboard corners indicated by green circles. Colored horizontal lines pass through different points in both images. Bottom row: Views after undistortion, calibration and rectification has been applied. The colored horizontal lines now are epipolar lines. Corresponding points lie at same height in image coordinates, hence the search for correspondences is simplified to one dimension. (Figures produced with the MATLAB stereo calibration toolbox[2]

a calibration for every camera individually, the stereo calibration can be computed from the extrinsic parameters. Let us denote the camera-centric coordinates of left and right image plane x_l and x_r. The world point $\mathbf{x}_w$ maps to camera coordinates via

$$\begin{aligned} x_l &= R_l(\mathbf{x}_w - \mathbf{t}_l) \\ x_r &= R_r(\mathbf{x}_w - \mathbf{t}_r) \end{aligned} \tag{2.12}$$

The final step is transforming the images to be suitable for fast stereo matching: align the image planes of both cameras to be in the same plane. The image planes are first rotated around the center of projection such they are aligned with the baseline. Then, the epipolar lines are parallel and corresponding points in both image have the same height. Figure 2.2 shows an example of a calibration pattern before and after calibration, Figure 2.3 details the representation of space extracted by the rectification. Depicted are the positions of the sensors to each other (red and blue) and the position of the calibration patterns in space that were used to compute their relative positions (colored squares in

[2]https://de.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html

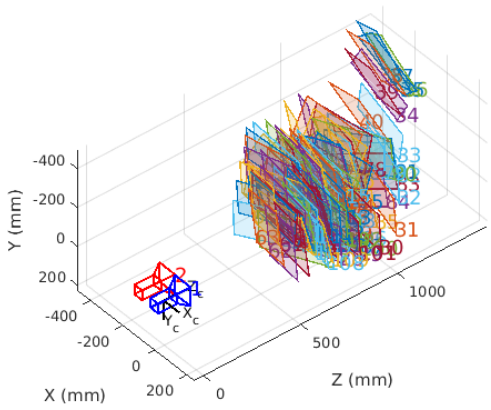


Figure 2.3: Inferred relative position of cameras (bottom left, depicted blue and red) and positions of patterns (squares mid to top right) in relation to cameras. 113 image pairs were used. (Figure produced with the MATLAB stereo calibration toolbox)

the upper right hand side).

Commonly, rectification is applied in a backward way to the images: for every pixel in the rectified target image a virtual pixel at a floating point number is computed in the source image. Then the intensity value of this pixel interpolated by several neighboring source pixels. Figure 2.4 illustrates the process. The problem with forward projecting each pixel of the source image to the rectified target is that it results, in general, in non-integer coordinates for the target. This may cause some pixel to be left blank after rounding the coordinates back to integers (this effect can be seen in Figure 3.8). However, in the frame work of event-based vision, only the forward rectification is applicable. Events are asynchronous binary atomic entities that can not easily be interpolated like synchronous full frame intensity values in conventional cameras. So algorithms must either work with floating point coordinates or deal with pixels that never generate events.

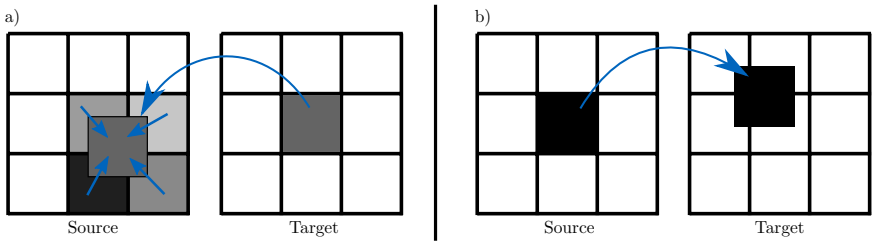


Figure 2.4: a) Backward rectification: the intensity of a pixel in the target image is determined by projecting it back to the source image and computing the weighted average over the neighbors. This way every target image pixel has a defined value. b) Forward rectification: since events are atomic events at different points in time, there is no weighted average of neighbors. Events streams are rectified by forward projecting from source, resulting in floating point coordinate in the target

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = (1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3) K^{-1} R_L \begin{pmatrix} u_L \\ v_L \\ 1 \end{pmatrix} \tag{2.13}$$

2.1.5 Depth reconstruction using point correspondences

The objective of stereo vision is to reconstruct depth of world points using their projections to the image planes. The main bottleneck of stereo algorithms is the *stereo matching problem*. Stereo matching aims at finding corresponding entities between two images, points that are projections of world coordinate (e.g. same object) in the scene. Ways to solve the stereo matching problem will be discussed extensively in the upcoming chapters. First, we will show that solving the stereo matching problem is equivalent to depth reconstruction. Assume correspondences between points on the image planes have been established, then the distance z of a world point can be computed using triangulation. Since we have aligned our image planes with rectification to be parallel (see Section 2.1.4) the depth computation can be simplified.

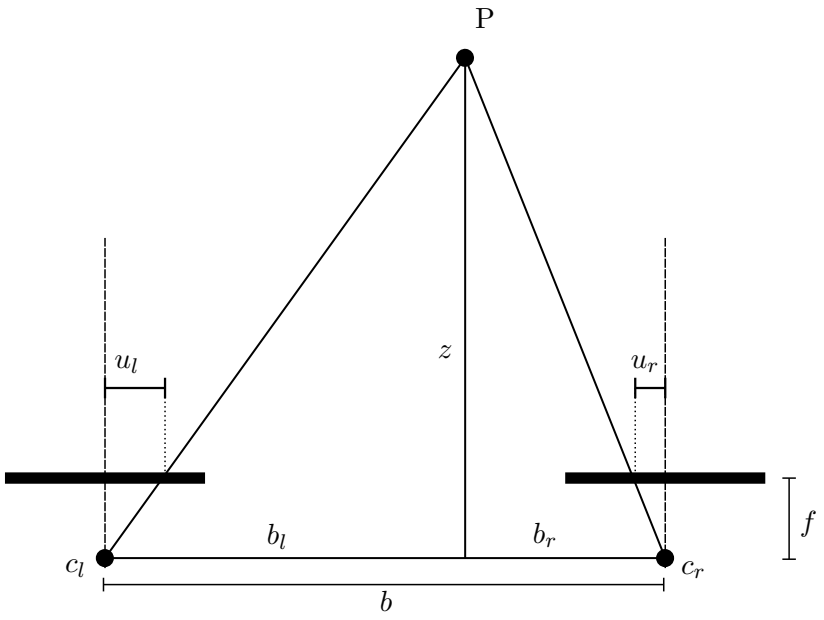


Figure 2.5: Depth triangulation. To find z, the depth of P, it is sufficient to know the baseline distance b and the disparity $d = u_l - u_r$: $z = \frac{bf}{d}$ (see text for details on derivation).

Looking at Figure 2.5, we can see that the following relation holds by similarity of triangles.

$$\begin{aligned} \frac{u_L}{f} &= \frac{b_L}{z} \\ \frac{-u_R}{f} &= \frac{b_R}{z} \end{aligned} \quad , \tag{2.14}$$

where the baseline was split into left and right segments $b = b_l + b_r$. Adding both equations in 2.14 gives the relation between the disparity $d = u_l - u_r$:

$$\begin{aligned} \frac{u_l - u_R}{f} &= \frac{b_L + b_R}{z} \\ \frac{d}{f} &= \frac{b}{z} \\ z &= \frac{bf}{d} \end{aligned} \tag{2.15}$$

Distance is inversely related to disparity, $z \sim d^{-1}$. That means that a given uncertainty in disparity affects depth estimates for far away objects much more than for close object.

d/px	z/m	d/px	z/m
1	24.32	30	0.81
2	12.16	31	0.78
3	8.11	32	0.76
4	6.08	33	0.74
5	4.86	34	0.72
6	4.05	35	0.69
7	3.47	36	0.68
8	3.04	37	0.66
9	2.7	38	0.64
10	2.43	39	0.62

Table 2.1: Disparity-to-depth mapping for a stereo setup with two DAVIS240C sensors, lenses with focal length 4.5 mm and a baseline distance of 10 cm for two selected ranges of pixel disparities. Note the large changes in distance for moderate changes in disparity when disparity is low

Usually, disparity is given in units of pixels. An estimation error of 1 px translates to an depth error of a few centimeters for an object that is less than a meter away and to an error of more than a meter for an object that is a few meters away. Furthermore, for algorithms that do not resolve below the pixel levels, this implies that the depth resolution becomes very coarse for distant objects. The exact relation depends on pixel size; with disparity d' given in units of pixels and the pixel size p, it becomes: $z = \frac{bf}{d'\mathrm{p}}$. Table 2.1 provides a mapping between disparity and depth for the sensor that was used for most experiments in this work.

3 State of the Art

3.1 Conventional Stereo Matching

The main task for stereo matching algorithms is to establish correspondences via points in two rectified images. Once correspondences have been established, the depth of these points can be computed using the epipolar geometry and depth inference describe in the previous sections. In conventional computer vision a multitude of algorithms have been developed to solve the stereo matching problem. This section provides a brief introduction into the fundamentals of computational stereo matching. For a comprehensive summary of algorithms we recommend the reviews found at [3, 4].

Stereo matching algorithms can be classified into two main categories: local and global algorithms. Local algorithms match points between images based on information in the vicinity of the points. They subdivide further into feature-based sparse and area-based dense algorithms. Both classes are interesting for event-based vision as well, so we will briefly outline them here.

Area-based stereo matching

Area-based methods aim at finding a dense disparity map, i.e. they want to map every pixel in the left image a pixel in the right image. The matching is achieved by defining a matching cost function and choosing the point along the epipolar line that minimizes this function. The most common matching cost functions are the absolute difference of pixel intensity of left and right image (I_l and I_r):

$$\mathrm{AD}(x, y, i, j) = |I_l(x, y) - I_r(x + i, y + j)| \tag{3.1}$$

and squared intensity difference:

$$\mathrm{SD}(x, y, i, j) = (I_l(x, y) - I_r(x + i, y + j))^2 \tag{3.2}$$

The underlying assumption is that pixels seeing the same point in the world display the same intensity. To reduce mismatches and suppress physically impossible matches often further constraints are introduced:

- **Uniqueness constraint:** one pixel in the left image has a unique matching pixel in the other image. If multiple matches are found, only one can be valid, all others have to be discarded. It is also possible that no pixel matches, if the point is occluded in the other image.
- **Ordering constraint:** in general, the order of points along a horizontal line in one rectified image is the same as the order of the points in the other image [1]
- **Continuity constraint:** because the world is made from continuous objects the disparity usually varies smoothly along neighboring pixels except for object boundaries, which exhibit disparity discontinuities.

[1] an important exception are points belonging to small objects in the foreground, the "forbidden zone" [5].

Only considering single pixels when computing matching costs is sensitive to noise and often leads to poor results. To improve matching quality, it is common to not only examine the pixel that is to be matched but all pixels $p = (p_x, p_y)^T$ in a window $W(x, y)$ of size $l \times k$ around it (i.e. $W(x, y) = \{p \mid x - l/2 \leq p_x \leq x + l/2 \wedge y - k/2 \leq p_y \leq y + k/2\}$). The cost of all pixels in W are aggregated to obtain a more robust matching result compared to individual pixel cost matching. The most popular cost aggregating functions are the Sum of Absolute Differences (SAD), Sum of Square Differences (SSD) and Normalized Cross Correlation (NCC). Assuming that we have rectified image, the epipolar line is horizontal and the window is shifted only along the x-axis:

$$\mathrm{SAD}(x, y, d) = \sum_{x', y' \in W(x,y)} \left| I_l(x', y') - I_r(x' - d, y') \right| \tag{3.3}$$

$$\mathrm{SSD}(x, y, d) = \sum_{x', y' \in W(x,y)} \left(I_l(x', y') - I_r(x' - d, y') \right)^2 \tag{3.4}$$

$$\mathrm{NCC}(x, y, d) = \frac{\sum_{x', y' \in W(x,y)} \left(I_l(x', y') - \overline{I_l} \right) \cdot \left(I_r(x' - d, y') - \overline{I_r} \right)}{\sqrt{\sum_{x', y' \in W(x,y)} \left(I_l(x', y') - \overline{I_l} \right)^2 \cdot \left(I_r(x' - d, y') - \overline{I_r} \right)^2}} \quad , \tag{3.5}$$

where the sums run over all coordinates within the chosen window. This summation is visualized in Figure 3.1.

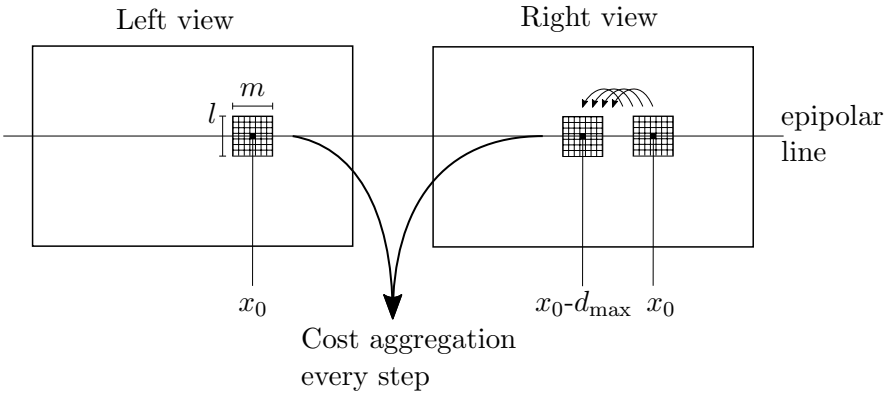


Figure 3.1: Window-based stereo matching. A window of size $m \times l$ is constructed around left and right pixel at (x_0, y_0). Cost aggregation is computed, then the window in the right view is moved along the epipolar line in steps of 1 px. This is repeated until d reaches a predefined disparity $d_{\max}$.

The size of the window is adaptable and depends on the application. It is an important choice because it affects the quality of results. On the one hand, the window needs to be large enough to effectively disambiguate similar regions along the epipolar line by finding unique matches; on the other hand, it must not be too large to be accurate for small objects and at disparity discontinuities. There are also approaches which adapt the window size during the matching process [6]. The cost functions have different advantages and disadvantages. SSD tends to overweight outliers and noise due to the quadratic penalty for intensity difference. The normalization process in NCC subtracts the intensity mean from both images to reduce effects of intensity variations between the two images and divides by the standard deviation to restrict values to the interval $[-1, 1]$. Because of the more complex calculations compared to SAD and SSD, it is the slowest of the three.

Area-based algorithms are challenged by scenes in which the cost function has low values for multiple pixels : in large textureless regions, many pixel neighborhoods look almost identical and the cost function assumes similar values for a large number of contiguous pixels; the same problem arises for horizontally extended patterns (e.g. door and window edges) where the edge is parallel to the epipolar lines; in regions with repetitive patterns multiple spatially separated pixel have the same matching cost. Breaking the tie in these cases is normally not possible without bringing in external, more non-local information.

Feature-based stereo matching

The feature-based stereo matching approach is separated into two steps. First each image is individually processed to extract a range of chosen features. Computer vision literature abounds in algorithms for features extracting of different levels of complexity. Well known examples are the Harris corner detector [7], the Scale-invariant feature transform (SIFT) [8], Speeded-up robust features (SURF) [9], which all extract local point features; extended features images can for instance be constructed on the basis of edges by using the Canny edge detector [10] or the Sobel filter that are designed to find edges. Shape-based feature detectors, such as the Hough transform [11, 12], search for geometric shapes like lines, ellipses and the like. In [13] different features are analyzed and an overview of which features are useful for tracking and matching is given.

After the extraction is done, the correspondences between extracted features are searched in the next step. The search process makes use of the same constraints as the matching in dense methods: all matches must fulfill the epipolar constraint. If there are multiple possibilities uniqueness, ordering and smoothness along extended features are used to resolve conflicts. This way a sparse disparity map is obtained that gives disparity estimates at image positions where features were extracted. Figure 3.2 illustrates the process using the Tsukuba data provided in the Middlebury data set and the Harris corner detector as example. Both images were processed with the feature extractor, resulting corners are shown in red. Now, correspondences have to be found, of which a few are shown using green lines.

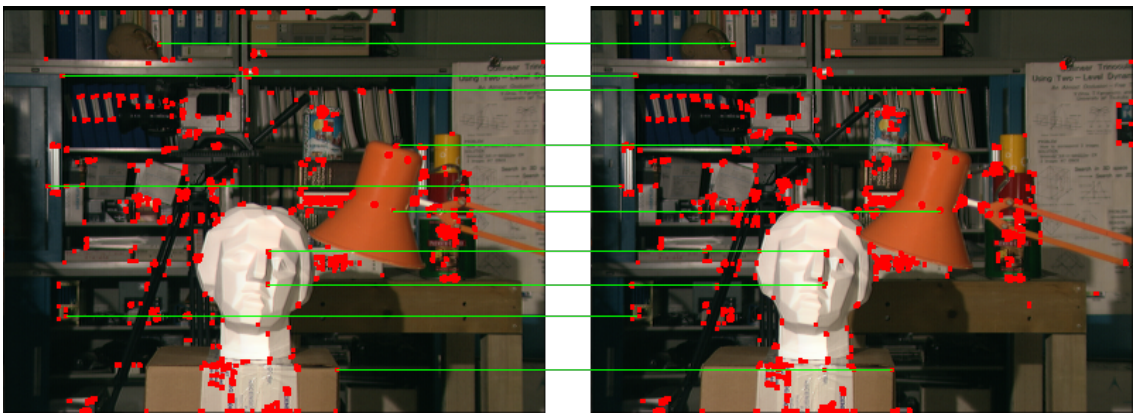


Figure 3.2: Result of the Harris detector on two images from a widely used data. Red dots indicate detected corners, green lines indicate examples of correspondences

In principle any descriptor can be used as basis for stereo matching. In this work, lines are of special interest because, we will later develop a method to extract lines from event streams and match them for depth information (Chapter 5). In the frame-based domain several line-based matching algorithms have been developed, e.g. [14], who proposed a

graph based method, which is the basis for our method,[15] who present a robust descriptor for line matching in images or [16] who recently introduced a 3D reconstruction method based on lines (2017). When applying these to video data an additional inter-frame tracking of lines is desirable to gain temporally coherent depth information. Such methods are described in [17, 18, 19].

Global stereo matching

Global algorithms incorporate dependencies between pixels and their matches and operate on multiple pixels (or even the whole frame) simultaneously. In global methods the matching of one pair of pixels depends not only on their neighbors, but also on the neighbors' matches, because they require the depth map to be smooth. Usually, global methods define a global model and formulate the matching problem as minimization of an energy function. This energy function consists of a term for the local matching costs E_{data} as well as a term that expresses how consistent the matches are (over the epipolar line or over the whole image) E_{smooth}, where the latter term acts as regularization.

$$E(d) = E_{\text{data}}(d) + \lambda E_{\text{smooth}}(d) \tag{3.6}$$

Different methods for global matching have been devised. Some methods separate the global correspondence problem into subproblems and search for consistency along the epipolar lines independently. Dynamic programming [20, 21] is exemplary: it simplifies the problem by disentangling it into smaller parts, thereby, it does neglect the relation between these lines. This can lead to inconsistencies between the lines which can be seen on the left hand side in Fig. 3.3. Different lines may have very different optimal matching solutions resulting in the strip pattern when next to each other. An approach that

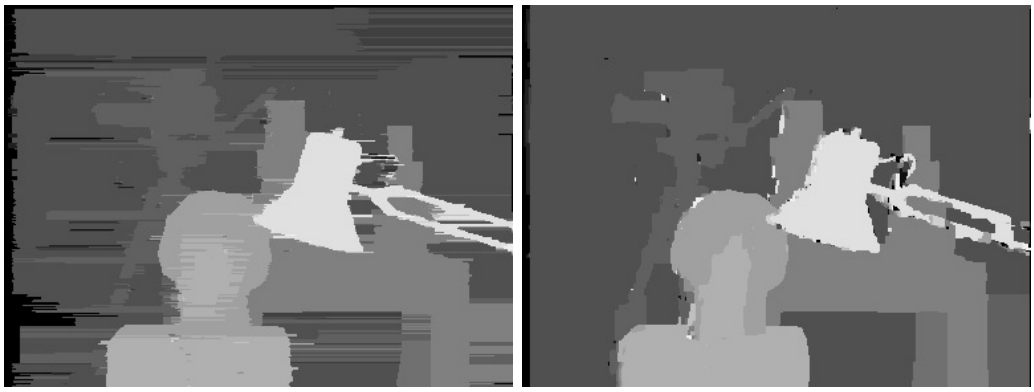


Figure 3.3: Depth map for Tsukuma data set computed with dynamic programming (left) and graph cuts (right). The dynamic programming approach optimizes lines independently, which gives rise to the stripe pattern. Graph cuts also provide consistency in the vertical direction (figures from [3])

includes consistency along the vertical axis is graph cuts [22, 23]. Graph cuts regularize the matching function for local smoothness, also taking into account pixels above and below the epipolar line, such that neighbors in all direction have to be consistent. This generates a smoother depth map as Dynamic Programming (Fig. 3.3).

Concerning computational speed, global methods are, in general, slower than local methods. Global methods try to enforce global consistency, where changing one match has implications for the costs of many other matches which leads to a more complicated

optimization procedure. Local methods typically compute optimal matching costs for every pixel once without using feedback during the matching.

3.2 Dynamic Vision Sensors

One of the main goals of this thesis is to develop a stereo matching algorithm that can be used in real time, has low latency and does not require extensive amount of computing resources. The stereo matching approaches described above use regular cameras as source for visual data which have some disadvantages towards this goal. Regular cameras capture snapshot of the environment at fixed time intervals, typically every 15-40 ms. Every pixel is read out for every new frame, regardless of their values are identical to the prior time step. This generates large quantities of data and requires a high bandwidth for transferal and processing which results in high power demands or long processing delays that come in addition to the data latency of some tens of milliseconds. Latency can be reduced by increasing the frequency of frame readout, but this creates even higher data rates. The relatively low dynamic range [24] can lead to over- or underexposure, even within the same view if the illumination varies spatially, e.g. due to light sources and shadows; this turn is detrimental for machine vision algorithms. All these properties complicate usage of conventional cameras in real-time application.

We tackle the disadvantages of conventional cameras by using dynamic vision sensors for visually recording the environment. Dynamic vision sensors of are a product of neuromorphic engineering. This branch of engineering aims at mimicking sensing and processing architectures of nervous systems in electronic circuits. It uses as inspiration the biological implementation of information perception, representation and processing that enables robustness to changing environmental conditions and system damage, as well as power efficiency. The insights from biology are transferred to create novel hardware and software. One of the results of this endeavor, are *silicon retinae* of which Dynamic Vision Sensor (DVS) are a type. They were firstly built by Mead and Mahowald in the late 1980s [25]. Silicon retinae are inspired by biological retinae of mammals and model their transient responses, as an approach to collecting visual information of the environment that is distinctly different from the approach of conventional digital cameras.

Silicon retinae collect information in a data-driven manner, which gives them properties suitable for e.g. robotic applications, where low latency is desirable and computing resources are limited. The sensor responds to pixel-local relative changes in light intensity and discards the absolute illumination information. Pixels of a silicon retina operate independently and asynchronously. This property allows every pixel to generate an event exactly at the point in time when its illumination changes, leading to a very fast response time; the latency typically lies in the order of tens of microseconds. An increase in brightness triggers a so called ON-event, a decrease an OFF-event. There are no frames, instead a quasi continuous stream of vision *events* is generated. Due to the fast manner in which events are generated, DVS do not suffer from motion blur which occurs in conventional cameras when objects move during the exposure time of one frame. In DVS, there is no fixed exposure time, pixels react as soon as they have measured a change in brightness. In fact, the change of illumination intensity is measured on a logarithmic scale. That way it is possible to achieve a high dynamic range of > 120 dB for modern sensors: in dark environments already small changes in lighting trigger events, while in bright scenes are larger absolute change is necessary. Examples of successful usage of DVS comprise realization of fast reacting robotic systems [26, 27], (real-time) tracking [28, 29, 30], low-latency odometry [31], self localization and mapping [32], neural activity imaging [33]

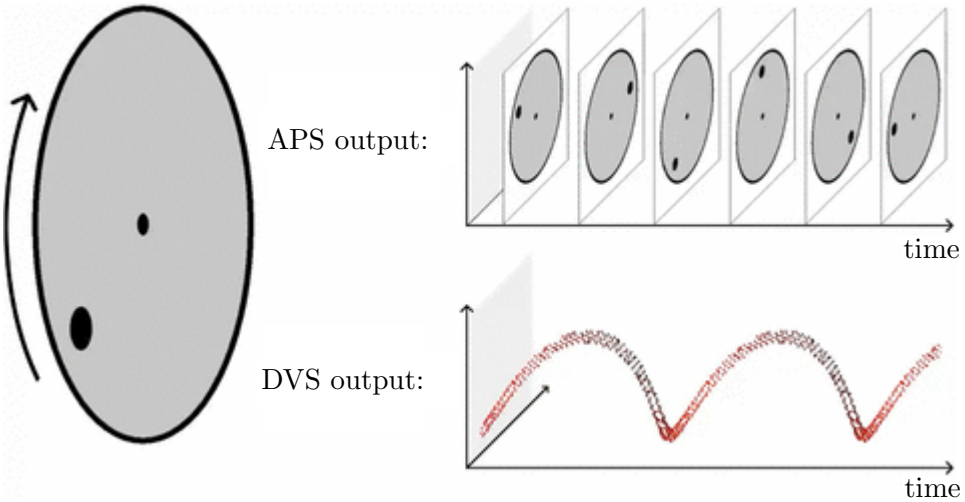


Figure 3.4: Conceptual difference between conventional frame-based cameras and event-based sensors. APS sensors produce a series of frames, while DVS produce a continuous stream of events. (Figure adapted from [34])

Figure 3.4 and 3.5 compare DVS sensor recordings and a comparison with conventional camera. The images show a quickly moving hand with a pen. On the left hand side, recordings with an Active Pixel Sensor (APS)sensor are shown: in the upper image, all pixels were read out simultaneously, the pen and hand are blurred due to their fast movement. In the lower image, a darker scene, the conventional camera is mainly black. On the right hand side, accumulated event streams recorded with the DVS are shown; it is important to note that the pixels were not all read out simultaneously, but at the time they perceived a brightness change. The fainter an event in the figure, the older it is. In the upper image, the hand and pen are visible, because they move, the rest of the scene is static and will not cause the DVS to generate any data. Hand and pen are not blurred, but rather leave continuous traces in the event stream (see Figure 3.4 for a visualization). In the darker environment, hand and pen are still well visible. The noise level is, however, much higher than in the brighter recording, because the pixel sensitivity for illumination changes was increased.

As mentioned before DVS generate a stream of events. These events will be denoted as $ev = (t, x, y, p)$, where t is the time the event was generated, (x, y) the image coordinate where it was created, p the polarity, i.e. if the brightness increased or decreased:

$$ev(x, y, t) = \begin{cases} 1 \text{ , if} \log I(x, y, t + \Delta t) - \log(I(x, y, t) > \Delta I \\ 0 \text{ , if} \log I(x, y, t + \Delta t) - \log(I(x, y, t) < \Delta I \end{cases} \tag{3.7}$$

During the last years, different types of event-based sensors have been developed [35, 36, 37]. The first commercially available DVS has a resolution of 128×128 and produces timestamps with microsecond resolution. Newer sensors combine APS and DVS functionality.

In this thesis, a DAVIS240C sensor is used. This is a hybrid sensor which has dynamic vision sensor functionality as well as active pixel sensing, i.e. conventional frames. The resolution is 240×180 pixels, its dynamic range is about 130 dB. On every pixel DVS and APS circuits operate independently, but they share the same photodiode.

This leads to issues with increased noise in the DVS, if frames are read-out while events are captured: pixels are likely to generate an event, when the APS is read out. This leads to spikes in the event rate and a “sheet” of spurious events in the event stream.

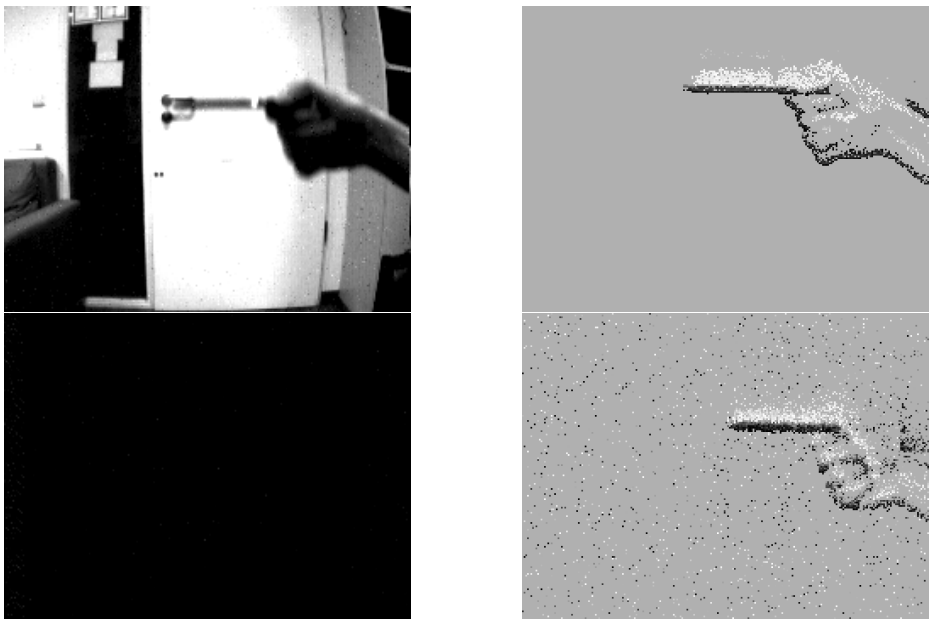


Figure 3.5: Comparison of APS (left column) and DVS (right column). Depicted is a quickly moving pen and hand. Top row: bright light conditions (note the blur on hand and pen), bottom row: dark light conditions (ceiling lights switched off, only light source is laptop)

We aim at developing algorithms that only use event and therefore deactivated the APS most of time. However, when a comparison with conventional algorithms is desired, we captured both data simultaneously, which resulted in deterioration of event stream quality. Additionally, the DVS pixel spontaneously generate ON events due to a leakage current. This is unavoidable and has to be taken care of (Section 4.1 explains how noise was filtered). A more detailed description of the sensor's details can be found in [37].

Summarizing, the main advantages DVS offer over conventional cameras are:

- very high temporal resolution
- low latency
- high dynamic range
- dynamic data rates because only scene dynamic is captured
- no undersampling of moving objects

Since the working principle of DVS is fundamentally different from conventional cameras, algorithms aimed at the latter can not be used on event-based data - new methods have to be developed. The next section will give an overview over the current state-of-the-art in event-based stereo matching.

3.3 Event-based Stereo Matching

During the last years, event-based vision has gained growing research interest. Several approaches towards event-based stereo matching have been explored, albeit not nearly as many as with conventional cameras. The majority of published algorithms can be classified in one of three classes: image reconstruction, time-correlation methods and

neural networks. An early approach is to reconstruct frames from the event stream and use area-based matching patterns like the Census transform to create a disparity map[38], which was later refined by belief propagation and post filtering [39]. Event streams are accumulated to obtain images, like for instance shown in Figure 3.5. Then conventional methods are applied to these images: the authors investigated the accuracy of area-based as well as feature-based methods. Accumulating images, however, loses the high temporal resolution of the stream.

Another class of algorithms processes events individually to gain depth information. In [40], [41] the high temporal resolution of the sensors is used to correlate events based on their timestamps. For each incoming event $ev = (x, y, t, p)$, a set $S_i(t)$ is constructed that contains all events in temporal vicinity δt:

$$S_i(t) = \left\{ ev(x, y, t) \text{ in } \mathcal{R}_i \mid \forall t',\ |t' - t| < \frac{\delta}{2} \right\} \tag{3.8}$$

It is not possible to simply match with the event that lies temporally closest, because timestamps are subject to noise from different sources like electronic jitter, discrete pixel sizes etc.). From $S_i(t)$, a set of possible matches is extracted using the epipolar constraint

$$M(ev(x, y, t)) = \left\{ e(x_k, y_k, t') \in S_j(t) \mid \forall k,\ d(ev, l_{ij}) < \Delta_p \right\} \quad , \tag{3.9}$$

with the Euclidean distance function $d(\cdot, \cdot)$ and a threshold value Δ_p. Additional constraints to further reject events are then applied (uniqueness, ordering, average pixel activity etc.). If M only contains one match this is taken as match, otherwise the events are discarded. So, this method is an event-to-event-matching approach. Differences in event generation characteristics of the sensors' pixels may pose challenges it.

Some extensions to this approach have been suggested since the temporal and geometrical constraints applied alone can not resolve all matching ambiguities. [42] adds Gabor filters to use orientation as additional cue for disambiguating event matches. If the baseline distance of the sensors is small and they are horizontally aligned, then observed object edges have approximately the same orientation. Banks of Gabor filters separate the events based on the edge orientation and match events separately. The base method and the orientation sensitive extension, however, tend to produce a not insignificant number of wrong matches, as analyzed by [43], especially when multiple objects are in the scene. The authors there propose a different extension by using techniques from message passing.A Markov Random Field is constructed with a hidden node for every pixel, which represents the disparity belief of the field at this position. Each node is connected to its four nearest neighbors and to an observation node. The observation node introduces information and is used to calculate the state of the hidden nodes. In addition, the hidden nodes exchange messages about their current belief of the disparity state. The disparity is then inferred by finding a maximum posterior probability over the state of all hidden nodes. Using this probabilistic framework, the authors are able improve matching results compared to prior versions. They show this quantitatively with ground truth data they labeled. The accuracy improvement comes, however, at the cost of increasing computing time by up to two orders of magnitude compared to the base version.

The third class of event-based algorithms is formed by neural networks that implement the cooperative matching approach, first suggested by Marr and Poggio[44] for static images. Different version of such networks for event streams have been proposed[45],[46].

They work in an event-driven fashion, i.e. they process each event immediately when they receive it. The networks are composed of one neuron for all combinations of pixels and allowed disparity. An active neuron stands for the network's belief that the corresponding pixel has the disparity the neuron represents. To arrive at a sensible and correct state, the network architecture implements the uniqueness constraint and continuity constraint by appropriate excitatory and inhibitory connections between the neurons network. Since pixels can only have one disparity value, neurons along the epipolar line inhibit each other. Neighboring neurons on other epipolar lines that represent the same disparity get excitatory input. This implements the continuity constraint, which states that objects are usually smooth and the disparity does not vary (or varies only slowly) along the surface of objects. Figure 3.6 gives a schematic of the network's connections. Some of these net-

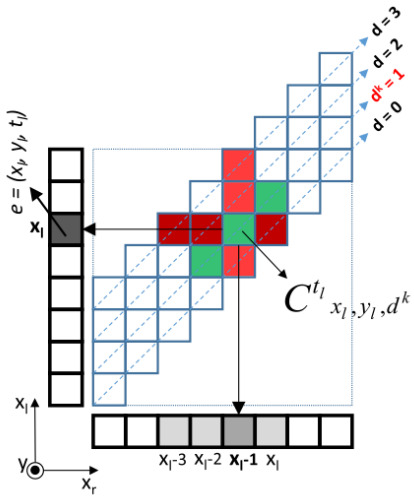


Figure 3.6: Working principle of the cooperative network (Figure from [45]). The center grid is composed of cells representing disparity; on the margin in horizontal direction one pixel row of the right sensor is plotted, in vertical direction one pixel row of the left sensor. An event in the left sensor at x_l is matched with an event at x_{l-1} in the right sensor resulting in disparity d^k. This is represented by activity of cell $C^{t_l}_{x_l,y_l,d^k}$. This cell now excites cells representing the same disparity in the neighborhood (indicated by a green hue) and inhibits cells that represent a different disparity or a different match for the event (indicated by a red hue).

works have been shown to run efficiently on neuromorphic hardware[47, 48]. The data for which they were deployed were, however, simple objects in front of a static setup. Their performance on complex dynamic scenes, e.g. with moving sensors, has yet to be proven.

A stereo setting is not the only way that has been developed to for the task of reconstructing depth from event streams. Rebecq et al. recently proposed a method to estimate depth using a single DVS sensor[34]. The work is based multi-view stereo methods which they made usable for event streams. Using only one sensor frees the approach of the requirements of having a rigid frame for multiple cameras, calibrating the geometry and synchronizing the clocks. Instead the technique requires knowledge of the sensor pose to calculate the depth of events. This information can be gained by an external tracking system or the depth estimation can be combined with a pose estimation algorithm[49]. Events are projected as semi-infinite rays back into space for which the sensor pose is used. When the sensor moves, a point in the world creates multiple events. The corresponding rays intersect in space at the position of the object (illustrated in Figure 3.7). Hence, by

finding the rays' intersection depth of objects can be calculated. The computation load is comparatively light weight and can be done in real-time on a modern CPU.

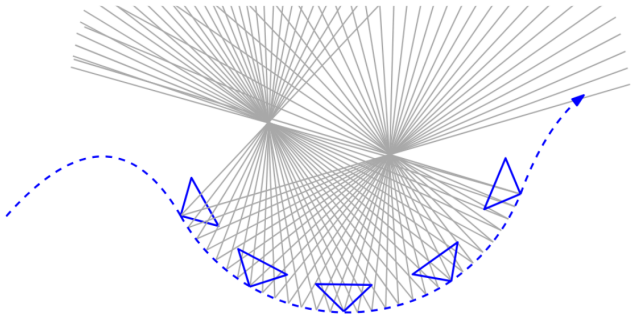


Figure 3.7: Event are propagated back into space using the known sensor orientation (blue triangles). Points of ray interjection correspond to the scene object and allow to calculate the 3-dimensional position (Figure from [34]).

3.4 Event-based feature extraction

Compared to area and correlation-based stereo matching, feature-based stereo matching has gained little attention. It requires algorithms that reliably extract and potentially track features from event streams which can then be used to find correspondences. Several trackers for different shapes have been developed. An early example of this can be found in [50]. Based on this [27] shows how to construct a robotic goalie with fast reaction time of only 3 ms. [26] focuses explicitly on detecting lines from events and describes a pencil balancer which attempts to keep a pencil in an upright position. Pencil movements are compensated by quickly moving the base its standing on using estimates about the pencil position. The estimation is performed in Hough space. In a more recent work, [51] describe a line segment detector to detect multiple lines in arbitrary scenes. They use Sobel operators to find the local orientation of events and cluster events with similar angles to form line segments. Events are stored in a circular buffer of fixed size, so that old events are overwritten when new ones arrive and the position and orientation of lines is updated through this process. Focus lies on line detection, not tracking. There are also increasing efforts to track other basic geometric shapes in event-based systems: corners have been a focus in multiple works as they generate distinct features that do not suffer from the aperture problem, can be tracked fast and find usage in robotic navigation. [52] use a corner matching algorithm based on a combination of geometric constraints to detect events caused by corners and reduce the event stream to a corner event stream. [53] transfer the well-known Harris corner detector ([7]) to the event domain, while [54] present a rapid corner detection method inspired by FAST ([55]), which is capable of processing more than one million events per second. [56] introduces a method to track visual features using different kernels like Gaussians, Gabors or other hand designed kernels. [57] uses a hybrid approach combining frames and event stream. It does not require features to be specified beforehand but extracts them using the greyscale frames. The extracted features are subsequently tracked asynchronously using the stream of events. This permits a smooth tracking through time between two frames.

3.5 Ground Truth Data for Quantitative Evaluation

Evaluation is a crucial part in the development of the vision system and is traditionally done on ground truth labeled data sets. These datasets allow quantitative evaluation of algorithms and provide the community with a metric to compare different proposed methods, extensions and filters. For this reason they are central to the development of new methods. In traditional computer vision there is a variety of accepted benchmark data sets against which algorithms are evaluated. For the problem of stereo correspondence matching, the Middlebury data set [3] has become the de facto standard. The images used for demonstration in Chapter 3 are for instance contained in this set. By now, there have been multiple iterations and extensions of it [58, 59, 60] and this benchmark continues to be used as standard. These dataset are, however, not useable for evaluating the event-based approaches presented in this work because the underlying data representation and working principles differ fundamentally. To this day, no event-based dataset featuring ground truth has established itself as a commonly accepted benchmark in literature. Nevertheless, it is important to quantitatively evaluate methods to measure the impact of parameters and compare different methods. Recently, Xie et al. published a dataset of ground truth labeled event-based recordings [43]. It consists of 5 scenes with simple to moderate complexity. The recorded scenes are the following (illustrated in Figure 3.8):

a) One box: a single box with almost constant disparity moved sideways

b) Two boxes: two boxes at different depths moved sideways in opposite directions at same height but never occluding

c) One person: a person walking parallel to the image planes

d) Two persons: two persons walking parallel to the image plane at different depths in front of each other (with occlusion)

e) One person different depth: one person walking diagonally away from the camera, so the depth changes constantly

In this thesis, these data are used to compare the developed methods with the literature. These scenes have, however, a few shortcomings. They were recorded with static sensors and the recordings are, therefore, very clean and show little noise and no clutter. Calibration is already applied. It was done by rounding the corrected event coordinates to the nearest integer. This leads to a grid like pattern where no events are ever located (see Fig. 3.8 c), d)) where the effect is very visible). Additionally, the original non-integer event locations are lost. The accuracy when analyzing errors of methods with subpixel accurate disparity estimates will therefore distorted.

Next, the settings of the two sensor seem to have been different during recording, this manifests for instance in different event rates even in scenes that show the same stimulus (see Table 3.1). The event streams are quite dissimilar at multiple times during the recordings, which makes matching harder. Especially, the recording 'One Person with Different Depths' is problematic. Figure 3.9 shows the problems. The right stream depicted in the upper row has considerably fewer events, especially ON-events, to the extend that certain features become invisible (e.g. around the head or the left arm). This was likely caused by sensor problem during recording. In the lower row, a similar problem can be seen: the right part of the body is not visible in the right stream.

For these reasons, we also produced an additional ground truth dataset ourselves. We chose a scene of higher complexity and moved the sensors instead of having static sensors

Data set	Total event number	Number events left sensor	Number events right sensor
One Box	97,179	40,059	57,120
One person	441,796	195,873	245,923
Two Boxes	63,544	24,071	38,843
Two persons	636440	272,019	364,421
One person different depth	140,254	87,730	52,524
Table and Cones	3,551,038	1,633,966	1,917,072

Table 3.1: Event distribution over sensors for different data files

observe a dynamic scene. The scene contains is a table, cones and boxes at different depths. Snapshots of the recording can be found in Figure 3.10. Ground truth values were found by manually clustering events belonging to the same edge and aligning the edges. Having all edges labeled independently allows the user to compose the scene in a modular way if a less cluttered scene is desired, it also allows to remove and add sensor noise. The event coordinates are available with unrectified integer coordinates as well as rectified, undistorted floating point number coordinates allowing evaluation on a subpixel accuracy level. This allows a flexible testing and evaluating of novel algorithms.

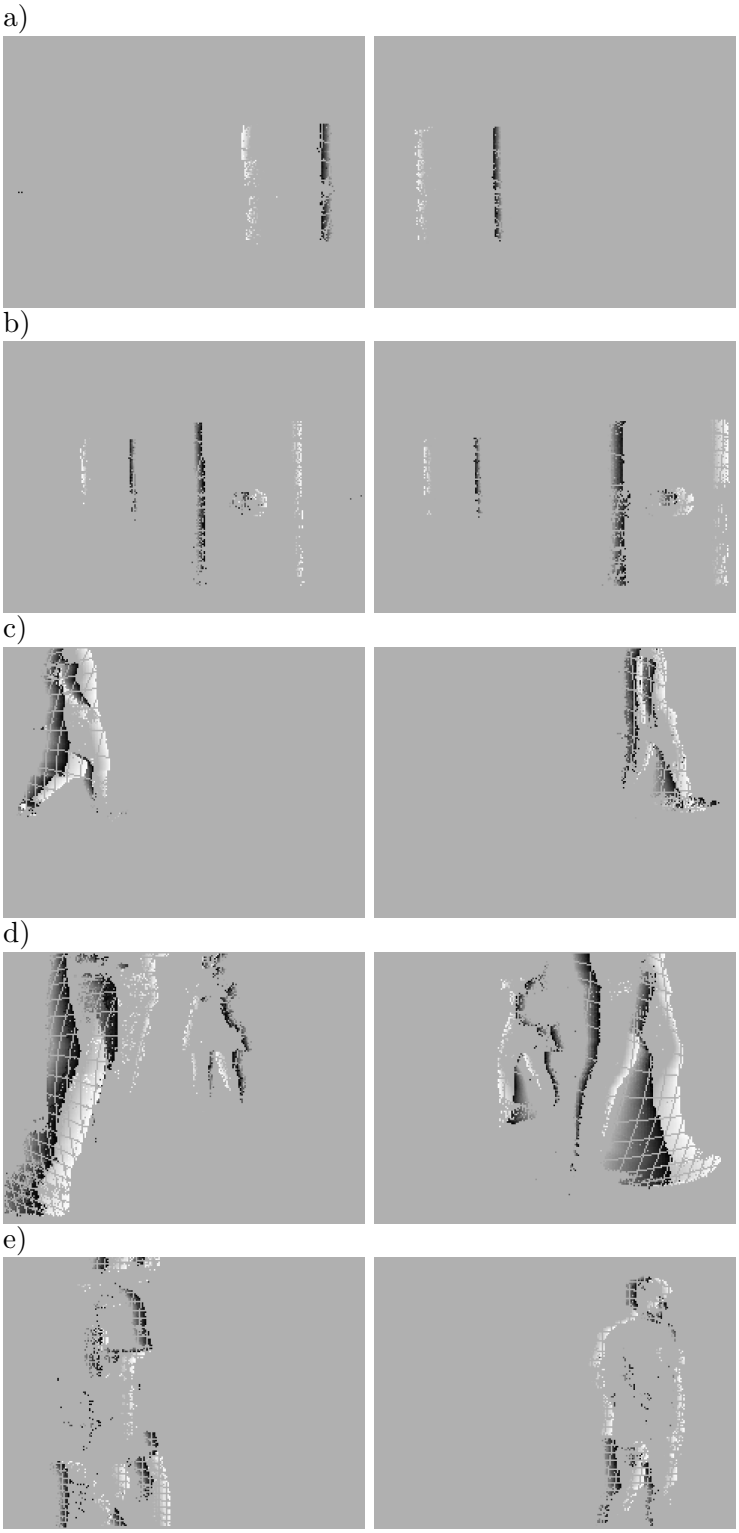


Figure 3.8: Data set provided by Xie et al. (2017) [43]. a) 'One Box', b) 'Two Boxes', c) 'One Person', d) 'Two Persons' e) 'One Person Different Depths'. First column shows snapshot from beginning of respective file, second column from end (event streams were accumulated for 100 ms, fainter color represents older event age, labeled disparity not shown).

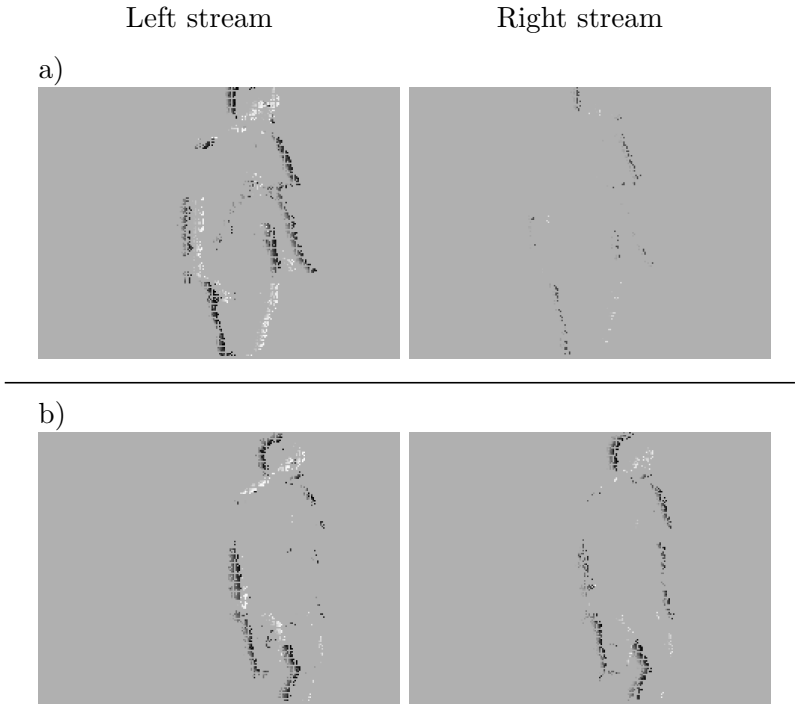


Figure 3.9: Problem with the 'One person different depth' recording. a) and b) are each a corresponding pair of accumulated event streams (100 ms accumulation time). Left and right stream are dissimilar frequently during the recording.

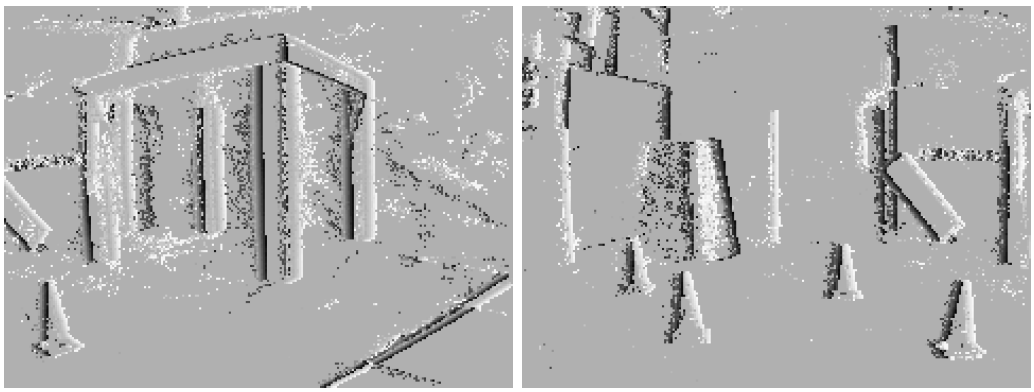


Figure 3.10: Two snapshots from our recorded ground truth dataset. Each edge was separated and labeled manually. The scene can be modularly composed to the users wishes. Left: beginning of recording shows table. Right: Later boxes and cones at different distances become visible.

4 Event-based stereo matching approaches

This chapter describes algorithmic approaches towards event-based stereo vision. In section 3.1, the distinction between stereo matching algorithms using local information and algorithms using global information was made. We will first examine local algorithms, which are more suited towards the nature of event-based data where the information is transmitted as stream of single events and synchronous global information is absent. In principle, it is desirable to process each of the single events directly when it is received in a fast manner using the information that is available at that point in time. Global approaches, on the other hand, require us to wait a certain time interval to collect events and then process all gathered information synchronously. Matching multiple events at the same time, they can impose global consistency and smoothness of the disparity maps. However, one can not make use of one of the major advantages DVS provide: low-latency due to asynchronicity. The second part of this chapter will analyze global matching algorithms.

4.1 Noise filtering

Before taking a look at the matching process, let us first consider the quality of event-based data. The sensors are not perfect, pixel can spontaneously generate events without a corresponding stimulus. These spontaneous events, which we will call *noise*, is cause by thermal fluctuations and junction leakage currents [35]. Leakage current accumulates, causing the pixels to perceive a constant increasing in brightness. The pixels will generate a spontaneous ON-event with a certain regularity. These events deteriorate the quality of the data and create unnecessary data which has to be processed.
The leakage current noise can not be modelled as Poisson process which does not take the accumulation into account. We will use a simple time-interval-based model here to get an approximative assessment of how much noise we have to expect within one time interval. Let $r_{\Delta t}$ be the probability that a pixel generates a spontaneous event in the time interval Δt. We exclude the possibility of two or more spontaneous noise events in this time window since spontaneous activity needs time to accumulate the leakage current that causes it. We assume that noise of all pixels is independent and identically distributed. The probability that n spontaneous events in an rectangular window of $l \times m$ pixel then is binomially distributed:

$$p(\#\text{ of events} \geq n) = \sum_{i}^{l \times m} \binom{l \times m}{i} r_{\Delta t}^{i} (1 - r_{\Delta t})^{l \times m - i} \tag{4.1}$$

We determined an estimate for p by recording a gray surface with a DAVIS240C for 15 minutes and dividing the number of events by time and resolution. The actual amount of noise is heavily temperature dependent. The temperature of the room was approximately 23°C, the temperature reported by the internal thermometer of the sensor was approximately 29°C. Assuming independence and identical distributions for all pixels as stated above, this leads to a spontaneous activity rate, respectively activity probability density $p \approx 0.058$ events/(s*px). To check if an event ev is noise, the nearest neighbor filter

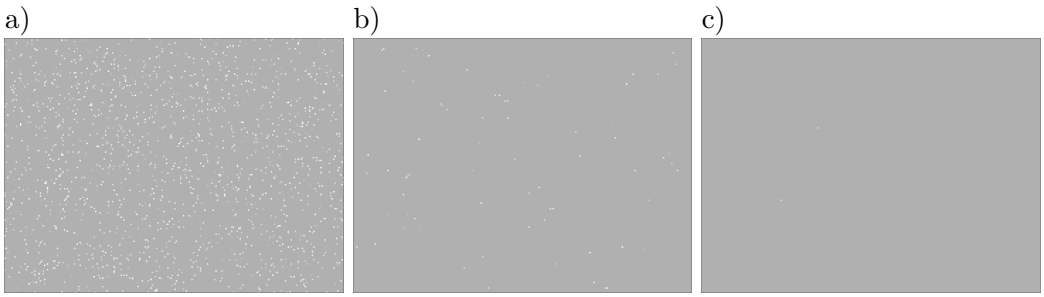


Figure 4.1: Effect of different noise filters: a) Unfiltered accumulated event stream when sensor faces textureless gray surface; b) nearest neighborhood filter; c) 5×5-pixel-window-3-events-threshold filter. $\Delta t = 100$ms for both filters, events accumulated for 1 second for all views. The nearest-neighbors filter lets a significant amount of noise events pass; in contrast the 5×5-pixel-window filter filters out almost all noise

is frequently employed. It checks if there was an event in the eight pixels surrounding ev with age below a threshold Δt. Using the model given in 4.1 with $\Delta t = 100$ms (i.e. $r_{100\text{ms}=0.0058}$ the probability for this criterion to be randomly met is

$$\begin{aligned} p(\geq 1 \text{ event in adjacent pixel}) = \\ 1 - p(\text{no event in adjacent pixel}) = \\ \binom{3\times 3-1}{0} r_{100\text{ms}}^{0}(1-r_{100\text{ms}})^{3\times 3-1} = 4.54\% \end{aligned} \quad , \tag{4.2}$$

where the -1 arises from the fact that the middle pixel, that generated the event, usually is not checked for a prior event. 4.54% false positives is still a relatively high rate of noise events that pass; this is especially important for algorithms that are sensitive to outliers. To suppress more noise, we increased the window size in which it is checked for prior events, and simultaneously increased the number of events we require to find to let an event pass. We used a window size of 5×5 and an event threshold of 3 events:

$$p(\geq 3 \text{ events in } 5\times 5 \text{ window }) = \sum_{i=3}^{5\times 5-1} \binom{5\times 5-1}{i} r_{100\text{ms}}^{i}(1-r_{100\text{ms}})^{5\times 5-1-i} = 0.036\% \tag{4.3}$$

These parameters brought the best trade of between computation time and filtering quality. Also see table 4.1 for more parameters. The filter with a 3×3 window and 2 events threshold was found empirically to filter out to many thin structures and rejected. It is, in principle,important to let the threshold of events be smaller than the window side length, because thin features (e.g. a vertical bar spanning the whole window height but only one pixel in width) are filtered out otherwise.

Additionally, it is worth mentioning that the overall noise level significantly increases, if the APS and DVS operate in parallel. Since the algorithms in this, solely use DVS events (except for calibration with uses APS only), this issue is secondary and we did not deeply investigate how to overcome it.

4.2 Window-based matching

The first approach is based on constructing a small window around incoming events and computing a similarity measure for windows in the other stream. The general idea is to compare the vicinity of an incoming event to same-sized areas along the epipolar line.

Window size	# events	p(false positive)
3×3	1	4.45%
3×3	2	0.092%
5×5	1	13.0%
5×5	2	0.8%
5×5	3	0.036%
7×7	3	0.28%
7×7	4	0.017%

Table 4.1: Comparison of different filter parameters for the described noise filter

The area that minimizes a matching cost function (i.e. is most similar) will be chosen as corresponding area and used to compute disparity. Window-based methods have long been used in conventional computer vision to find dense depth maps(see section 3.1). These methods use pixel intensities as input to the cost function. Event-based sensors do not provide intensity information; the primary information conveyed is event generation time. Therefore, we choose the cost function to be based on the event timing. Generation times of events in corresponding regions of both two views to be very similar since they depict the same physical object. That implies that the time difference of the events within matching windows is on average small what makes time difference of event occurrence a suitable cost function $c(x, y)$. However, we do not know in advance if there recently was an event at a given pixel position which also stands in contrast to conventional cameras where each pixel has an intensity value to read out, if there was no event in either of the both streams during the last τ seconds, we do not consider the position for computing a cost value. These events are considered too old to be correlated with the current scene and are likely to distort the aggregated cost function value. Cost between two events $ev_l = (t_l, x_l, y_l, pol_l)$ and $ev_r = (t_r, x_r, y_r, pol_r)$ at time t_0 is then is computed as:

$$c(ev_l, ev_r) = \begin{cases} |t_l - t_r|, & \text{if } t_0 - t_l < \tau \text{ and } t_0 - t_r < \tau \text{ and } pol_l = pol_r \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{4.4}$$

To find a total matching cost of two windows, a cost aggregation function over the window is required. We chose and compared the three mentioned functions: sum of absolute differences, sum of squared difference and normalized cross correlation (see Section3.1). Both sums are executed only over cost values that are defined, the sums are then normalized by E, the number of pixels considered. This results in an average cost value and make windows with different numbers of valid pixel pairs comparable.

$$C_{\text{SAD}} = \frac{1}{E} \sum_{i=x-\frac{l}{2}}^{x+\frac{l}{2}} \sum_{j=y\frac{l}{2}}^{y+\frac{l}{2}} |c(i,j)| \tag{4.5}$$

$$C_{\text{SSD}} = \frac{1}{E} \sum_{i=x-\frac{l}{2}}^{x+\frac{l}{2}} \sum_{j=y-\frac{l}{2}}^{y+\frac{l}{2}} (c(i,j))^2 \tag{4.6}$$

In the following, we will first present the base algorithm, followed by extensions to enforce consistency using the properties of the event stream. For clarity of presentation, we will assume that we want to find the disparity of events in the left event stream. It is, however, not restricted to the left stream and works identically for the right stream.

The method handles every event as soon as it received. For every incoming event e, we construct a square window of $l \times l$ pixels around the event pixel coordinates x_e and y_e in both retinas. We compute the matching costs between both windows. Then, we move the right window one pixel to the left along the epipolar line. This process is repeated until the window was shifted by a predefined maximal disparity $\Delta_{\max}$. Events with an x-coordinate lower than $\Delta - \max$ are rejected, because not the full range of possible positions can be evaluated and matching would, hence, be biased towards low disparities. The epipolar lines were chosen to be horizontal for all recordings which can be achieved by rectification (see Section 2.1.4). This simplifies computation by turning movement along the epipolar line essentially to incrementing the x coordinate.

The size l of the window and event lifetime τ are parameters of this method. Changing l is a trade-off between the risk of confusing similar textures when the window is to small and their is too little information to disambiguate, on the one hand, and ignoring small structures and requiring more computing time, on the other hand. τ sets how many events are concerned for the matching cost computation. Setting is to a low value may eliminate too many events to achieve meaningful cost values, especially in slower scenes where there are few events. Whereas setting it to a high value risk distorting the cost function with old events from objects that have become unrelated to the currently evaluated location.

4.2.1 Experimental results

First, the performance of the three different cost aggregation functions was evaluated to compare with each other. We tested the algorithm with the ground truth data set described in 3.5. Experiments were conducted on an Intel Core i7 4770K running at 3.5 GHz, the code was implemented in C++.

Figure 4.2 shows a comparison for different values of the parameters. Although NCC is a much more complex function and takes more computation time than the easier SAD and SSD the performance is considerably worse. SAD and SSD perform comparably with SAD being slightly better overall. The quadratic cost term in SSD penalizes the matching of uncorrelated events e.g. through noise heavier than SSD. Therefore, for longer event lifetimes, where there is a higher probability of uncorrelated events spontaneously appearing plus the stronger quadratic penalty. SAD performance is more robust against this effect, performance varies slower with different event life time. Precision increases monotonically with window size, so in terms of precision a larger window is better.

When it comes to computing time, SAD and SSD perform virtually the same, as was expected, and faster than NCC. Larger window sizes require more computing time. For a window size of 5, processing one event takes about 5 us, corresponding to 200.000 events per second. Increasing the window size to 15 already uses 20 us (50.000 events per second). Average computing times are displayed in Figure 4.3.

For the reasons stated above, we chose Sum of Absolute Differences as cost aggregation function and 100ms as event lifetime. Note that event lifetime must most likely be tuned to the velocity expected in the observed environment when this algorithm is deployed.

In comparison with current literature, the method performs comparable with the best published method. Especially for larger window sizes. Table 4.2 provides an overview over the presented window-based method and different other published algorithm. The computation in literature was done using MATLAB, so timing information is not directly comparable. However, considering absolute computing time, we can state that the variant of our algorithm with window size 11×11 can approximately be run in real time, if the event rate does not significantly exceeds 100.000 events per second.

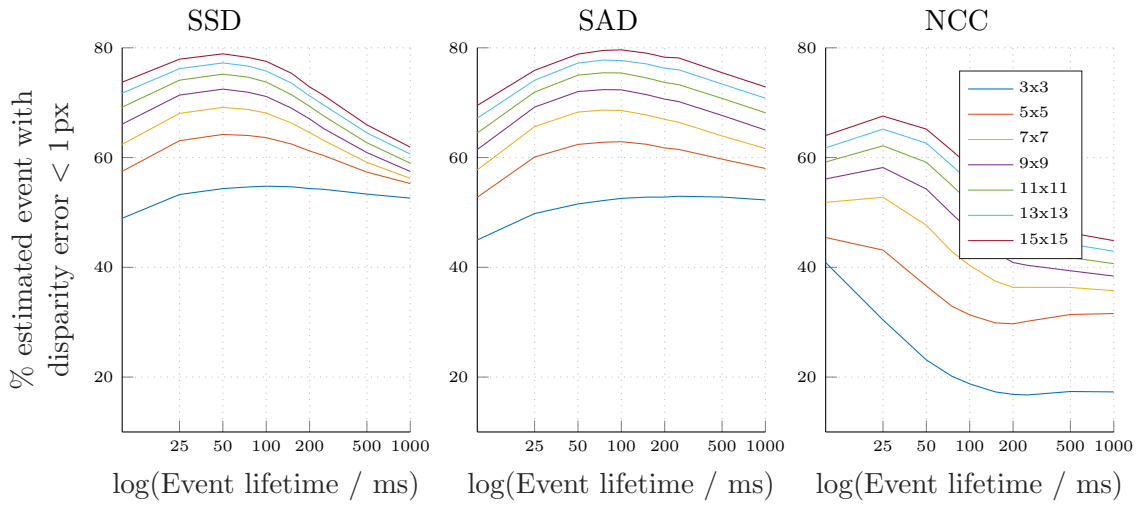


Figure 4.2: Accuracy of the window-based stereo approach in dependence on event lifetime, for different window sizes and cost aggregation functions.

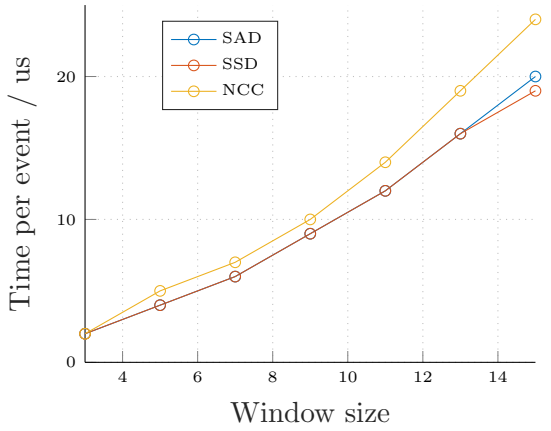


Figure 4.3: Computing time per event for different cost aggregation functions (event lifetime set to 100ms)

A problem that is not reflected in the averaging statistics is fact that falsely matched events tend to cluster. There are regions of consistent false matches. Often they are part of a larger physical object which was partially matched correctly and partly falsely. This can be misleading for an autonomous system using the stereo information. It would perceive two objects at different depth of which one is a phantom object. Because the wrong matches are located close to each other, it is not possible to easily correct them with a window-based smoothing scheme. In the following we will look at a correction mechanism that aims at rejecting wrong matches and generating disparity propagation along edges to improve disparity smoothness over longer distances.

4.2.2 Edge traversing extension

DVS events occur primarily at object boundaries and edges. Along most edges, the disparity varies slowly, if at all. Therefore, we propose a postprocessing step to ensure smooth disparity values along edges. Based on an idea by Witt and Weltin[62] a variable is introduced which reflects the confidence that a match is correct. Confidence is propagated

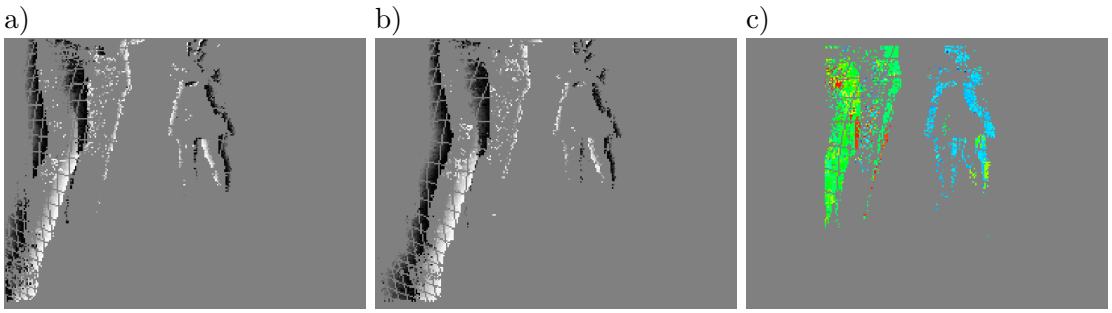


Figure 4.4: Left and right input (a and b) and disparity output (c). Colors in c) represent disparities. The disparity of both person should only vary within a small margin, i.e. both should be shown in a consistent color (green and blue respectively), but there are patches of incorrect disparities. These would appear as phantom objects in a downstream processing system. Note that the left side of the disparity map is empty: in this area not the whole disparity range can be scanned, to prevent biasing it is therefore ignored

along connected parts of the event stream, such that high confidence matches are used to correct low confidence matches and a consistent depth estimate is provided.

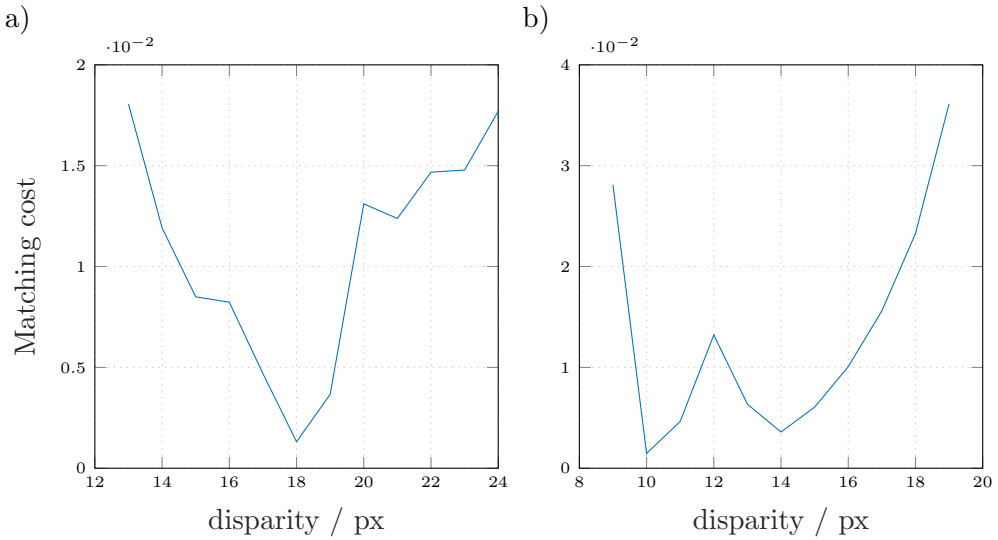


Figure 4.5: Examples for a cost aggregation functions, x shows the disparity of the event, y the associated matching costs for the respective disparity. (a) desirable graph of the cost function with a distinct minimum, (b) an undesirable graph with multiple, here two, minima

Algorithmically, confidence is defined via the ration of lowest and second lowest matching cost. A match should be as unambiguous as possible, the correct position is expected to have significant lower matching cost than all other positions. This is reflected by a distinct minimum in the cost aggregation function. Multiple minima at different positions along the epipolar line, in contrast, suggest multiple fitting positions reducing the certainty of the chosen match to be correct. Figure 4.5 shows examples for both cases. The proposed confidence measure compares the lowest and second lowest cost by taking the quotient $q = \frac{\text{second lowest cost value}}{\text{lowest cost value}}$ of the values and assigning discrete confidence values based on it:

- high confidence, if $q > 3$

- good confidence, else if $q > 2$
- low confidence, else if $q > 1.5$
- no confidence, else $q \leq 1.5$

In addition to high confidence matches, smoothness of the depth map is to be enforced. That implies that neighboring pixels' disparities can not differ by more than one. If they do, either one of the matches is wrong or they see a depth discontinuity and likely do not belong to the same physical edge.

We propose a disparity postprocessing step for every event based on the matching confidence and the smoothness constraint.

After the matching step, every event computes the matching confidence. If the confidence is good or better, the event searches for a two neighboring events, that are consistent and have at least good confidence. This group serves as anchor point for the traversal. Now, first a list of edge events is created, then the found events are refined. Travel along the edge and avoid circular cycling through the same events only neighbors in direction of traversal are regarded. The disparity of newly added events is changed based on its matching confidence. If the matching was done with highest confidence, we do not change it. If the confidence is one category lower, the disparity is set to the mean of the new event and its prior neighbor. If the confidence is two or more categories lower we replace the disparity with the prior event's one.

In the next step, the discovered chain of events is iteratively smoothened. Two adjacent events are compared in terms of matching confidence. If the confidence is the same, nothing is done, if the confidence of the two events differ by one, the disparity of the lower confidence event is set to the mean of both events disparities, if the confidence level difference is more than 1, the lower confidence event's disparity is overwritten by the higher confidence event's disparity. This is done until all events in the chain have been processed. This eliminates low confidence matches from the chain by interpolating them using the higher confidence matches.

Results show no improvement but actually deterioration to the base version of the algorithm, the accuracy is perceivably lower (Table 4.3). A more detailed investigation of this result found two main causes. First, the basis for the cost function is time difference, not intensity as in conventional matching, and the proposed confidence measure is unsuitable for time differences. Event traces are spatially highly correlated: the pattern belonging to an extended object within the comparison window at time t has a very similar looking pattern generated at time $t - \Delta t$, only shifted by one pixel, and again a very similar looking pattern at approximately $t - 2\Delta t$ shifted by two pixels and so on. During the cost aggregation steps all these patterns yield a similar cost value with the additional punishment of $n * \Delta t$, depending on distance. This results in slowly increasing cost aggregation graphs, not single minima spikes as one gets when search for spatial correlation based on intensities. Figure 4.6 shows such a graph. The optimal disparity is very likely correct, but will not be labeled with a high confidence because the neighboring costs are all low as well. High confidence values are a result of low speed of the objects moving, thus effectively increasing Δt and also a result of noise events which break the smooth pattern described above. This causes events to be corrected into the direction of noise, lowering the overall quality of the matching. The ratio of lowest and second lowest cost is, therefore, not a suitable metric to determine unambiguity of matches.

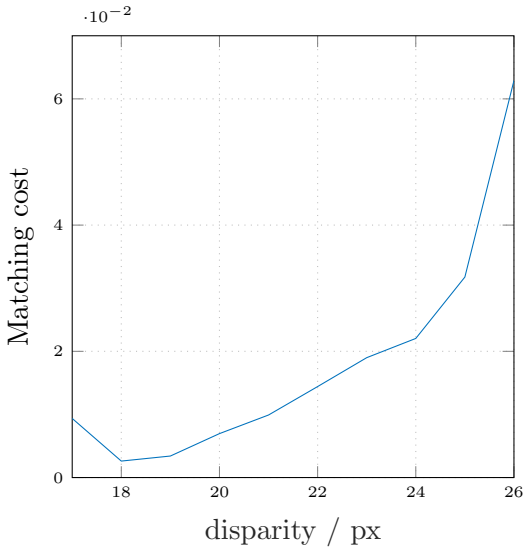


Figure 4.6: Typical cost aggregation graph. Cost increases slowly with disparity.

4.2.3 Discussion

We proposed a window-based stereo matching algorithm based on the local vicinity event timestamp differences. The achieved accuracy of window-based methods is on par or better than that of recently published other methods. In terms of speed the presented algorithm has the potential to processes event rates in the order of a hundred thousand events per second, and hence has the potential to be employed in a real-time setting, especially with an optimized implementation. Efforts to improve the algorithms by postprocessing disparities and incorporating a smoothing scheme failed however. This underlying reason for this was on the one hand, that wrong matches tend to cluster and these clusters have consistent disparity, and on the other hand, that the introduced confidence measure was not suitable for event-based data which is temporally highly correlated along the epipolar line.
A fundamental drawback of the window-based method as presented is that it is restricted to integer disparity value. The epipolar line is scanned in discrete steps of one pixel per step. As we have discussed before small disparity changes can lead to large changes in perceived depth for objects that are medium to far distance (see Table 2.1). It is desirable to have floating point accurate disparity instead of being constraint to integers. This can partly be improved by increasing the resolution of the event-based sensors. However, window-based methods do not scale favorably for higher resolutions. The number of computations per event grows significantly if there are more steps along the epipolar line and window sizes have to be increased to cover the same solid angle. Additionally, sensors with a higher resolution generate more events, so that not only the computations per event, but also the event rate increases. Since the algorithm with the setup as used here is already on the edge of real-time capability, an increased resolution would make real-time hardly achievable. In order to gain the possibility to achieve subpixel accuracy and improve scalability, we now turn to different event processing approaches.

4.3 Aggregate methods

The second class of approach to solve the stereo problem were methods that operate on multiple events at the same time, different to the methods presented in section 4.2.

Event data can be represented as point clouds in three dimensional space: two spatial dimensions and one temporal dimension. In a stereo setting of two event-based cameras with short baseline, the point clouds originating from the same object will be similarly shaped. By aligning two point clouds belonging to the same object we will be able to infer the disparity of the cloud's events.

The problem of aligning two point clouds, is known as point cloud registration or point matching. The objective is finding a transformation that maps on point cloud to another one. The general problem can be formalized as follows: assume we have two finite point sets L and R which contain N and M points from $\mathbb{R}^d$. Find a transformation T that maps R to $T(R) = R'$ such that the sum over the distances of every pair of points from R' and L is minimized:

$$d(L, T(R)) = \sum_{r \in T(R)} \sum_{l \in L} f(r - l) \quad , \tag{4.7}$$

where l(r) is the point in L with the smallest Euclidean distance to r and f is a distance function (e.g. Euclidean distance).

We used point cloud registration to solve the stereo matching problem and analyzed two different approaches which will be introduced in this section: a probabilistic one, based on coherent point drift[63], and a geometric one, based on the iterative closest point algorithm[64].

4.3.1 A geometric approach to registering event streams: Iterative Closest Points

The Iterative Closest Point Algorithm (ICP) is a point-based iterative algorithm to find an optimal alignment between two different point clouds. It is often used to stitch results from surface scans of different sensors together, a task that is very similar to the alignment of event clouds. ICP has already been applied to problems in event-based vision [57, 65] as well as in stereo vision [66]. We will briefly introduce the algorithm here, motivate and derive our adjustments for stereo vision and then present and discuss the results.

4.3.1.1 Algorithm

Initially, one point cloud is fixed and used as reference cloud. The other point cloud (reading cloud) will be iteratively updated to match the reference cloud. In every iteration, the points in the reading cloud are paired with their nearest neighbor in the reference cloud according to Euclidean distance. After having found the pairs, the transformation is chosen such that the sum of squared distances between pairs of points is minimized. Then, the reading cloud is transformed and the next iteration is started with finding new corresponding pairs (see Algorithm 1).

Using the squared distance as error measurement makes the method highly susceptible to outliers. Noise events that lie far away from the actual visual signal will typically have a much larger distance to their nearest event in the other stream. This leads to a high weighting in the squared distance sum. The event correspondence is, however, spurious, since it is matches noise events, and will only distort the result. It is, therefore, important

Data: point cloud R, point cloud L, initial transformation guess T^0
while *not converged and maximum number of iterations not exceeded* **do**
  For every point in L find the closest point in $T^{(k)}(R^{(k)})$;
  Find $T^{(k+1)}$ that minimizes the sum of square distances between point pairs;
  $R^{(k+1)} = T^{(k+1)}\left(R^{(k)}\right)$;
end
Result: Optimal Transformation T^*

Algorithm 1: Iterative closest point algorithm

to preprocess the event clouds using an outlier filter. We used an outlier filter, that rejects a certain percentile of events with respect to the highest distance to neighboring events.

The linear transformation T has the following form:

$$T = \left(\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right) \quad , \tag{4.8}$$

where t is a D dimensional translational vector and R a matrix whose properties can be chosen depending on the desired type of translation. The original ICP is formulated to find a *rigid transformation* between the point sets, i.e. a transformation that is composed purely from rotation and translation and does not change the distance of any two points within a cloud. In this case $R \in SO(d)$. In case of non-rigid transformation R can become an affine matrix that introduces shear terms.

The stereo setup enforces some geometric constraints that simplify the final transformation.

- The vision sensors are oriented in the same direction: the event point clouds are not rotated against each other. R is diagonal.
- The sensors lie in a plane parallel to the pixel's x-axis, objects of the same height will be mapped to the same pixel row: there is no shear along the y-axis, $R_{yy} = 1$
- The sensor clocks are synchronized: there is no shift along the t axis, $q_t = 0$.

We allow R to be an affine transformation along the x axis. This is founded in the fact that the projections of objects who are not parallel to the camera baseline will be have differing lengths along the x axis in image planes. Using these constraints, it is possible to reformulate the problem. The general transformation for an event in homogeneous coordinates $ev = (ev_x, ev_y, ev_t, 1)$ becomes:

$$T(ev) = \begin{pmatrix} R_{xx} & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} ev_x \\ ev_y \\ ev_t \\ 1 \end{pmatrix} = \begin{pmatrix} R_{xx} ev_x + t_x \\ ev_y + t_y \\ ev_t \\ 1 \end{pmatrix} \tag{4.9}$$

Using $c(ev)$ as event in L that lies closest to ev, the quantity to be minimized becomes:

$$\begin{aligned}
&\min_T \sum_{ev \in R} \left(T(ev) - c(ev)\right)^2 = \\
&\min_T \sum_{ev \in R} \left(\begin{pmatrix} R_{xx} ev_x + t_x \\ ev_y + t_y \\ ev_t \\ 1 \end{pmatrix} - \begin{pmatrix} c_x \\ c_y \\ c_y \\ 1 \end{pmatrix} \right)^2 = \\
&\min_T \sum_{ev \in R} \left(R_{xx} * ev_x + t_x - c_x)\right)^2 + \left(ev_y + t_y - c_y\right)^2 + \left(ev_t - c_t\right)^2
\end{aligned} \tag{4.10}$$

The last line in 4.10 is a sum of positive terms (squares of real numbers are larger or equal zero) with independent variables. In order to minimize it, it is sufficient to minimize each summand independently. Let us start with the last term containing the event time. It does not have any free variable, because we neither rotate nor shift the time coordinate. So, there is nothing we can do to optimize it. The middle term containing the y coordinate contains the parameter q_y for which a simple analytic solution is available:

$$\begin{aligned}
\frac{d}{dt_y} \sum_{ev \in R} \left(ev_y + t_y - c_y\right)^2 = &\quad 0 \\
2 \sum_{ev \in R} \left(\Delta y + t_y\right) = &\quad 0 \\
2 \sum_{ev \in R} \Delta y + 2N t_y = &\quad 0 \\
2N\ \overline{\Delta y} + 2N t_y = &\quad 0 \\
t_y = &\quad -\overline{\Delta y} \quad ,
\end{aligned} \tag{4.11}$$

where we used $\Delta y = ev_y - c(ev)_y$ and the average $\overline{\Delta y} = \frac{1}{N} \sum_{ev \in R} \Delta y$. So, the optimal value for the y-shift is the average of the y difference of the corresponding points. The y-shift will not be reflected directly in the computation of event disparity, it influences however the distances of events between the clouds and the pair finding step prior to the optimization step.

After having found a solution the y coordinate, let us look at the x coordinate. The term containing the x coordinate has the form of a problem of linear regression. Formulated as matrix expression the equation that needs to be fulfilled becomes

$$X\beta = \mathbf{c} \quad , \tag{4.12}$$

where we subsumed the data in a matrix X, the parameters in vector β and the comparison values in vector $\mathbf{c}$.

$$\mathbf{X} = \begin{pmatrix} 1 & ev_{x1} \\ 1 & ev_{x2} \\ \vdots & \end{pmatrix}, \beta = \begin{pmatrix} t_x \\ R_{xx} \end{pmatrix}, c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix} \tag{4.13}$$

This is an overdetermined system for which we want to minimize the error. Ordinary least square is applied to compute the minimizing β^*. OLS is a common estimator for observational data, and computationally simple. It minimizes the sum of squared residuals The close form solution for this problem is:

$$\beta^* = (X^T X)^{-1} X^T c \tag{4.14}$$

Note, that this solution may contain the inverse of a matrix, but this matrix is only a square matrix of the same dimensionality as the data, i.e. two in our case, so the inverse can be computed very quickly.

Once we have found β^*, the optimal transformation is applied to the data cloud. Then, the new closest point pair are searched again and a new optimal transformation is computed. This procedure is repeated iteratively until the transformation converges and the iteration is stopped.

4.3.1.2 Evaluation

To evaluate the capabilities of the method, we ran experiments testing different data event cloud sizes and outlier rejection factors on the ground truth data provided by Xie et al. ([43], also see Section 3.5). Because the assumption of approximating the projective transformation with an affine one only holds for scenes that contain only one body, we tested it with the data sets that only contain one object, namely 'One Box', 'One Person' and 'One Person Different Depths' (in scenes with multiple objects, it would be required to segment the recording into single objects first). The objective was to determine the disparity of events in the left stream by finding the transformation that maps them onto the right stream. The number of events in the left cloud per registration attempt N_L was fixed, the reference cloud was constructed by the events in the right retina that occurred during the same time interval as the left cloud events. After registration was completed, the next left event cloud was formed from the subsequent N_L events and the next right cloud from the corresponding right stream events. This was iterated until the whole file was processed. Outlier rejection was varied from keeping all datapoints (100%) to only keeping the 10% datapoints with the smallest distance to other points. Experiments were performed with an Intel Core i7 4770K running at 3.5 GHz. For the registration we used a publicly available library [67] as base and extended it with the presented error minimization and point cloud transformation. The code was written in C++.

In terms of computing time, the registration operates very favorably. Figure 4.7 shows the required computing time per events. Computing time is slightly higher than 1 us for most settings, equaling 900,000 to 1,000,000 events per seconds which is significantly higher that the processing rate of the window-based method. The computing time grows very slowly with the size of the event clouds registered (less than 15% when increasing the clud size by a factor of 10). It also does only weakly dependent on the outlier rejection threshold. As one would expect, rejecting more events is faster, but only slightly so. Consequently, when optimizing the parameters for matching accuracy, we do not have to factor in computing time, since the differences between the different parameter sets are minor.

In terms of accuracy, the method is not as satisfactory as it is in terms of computing time. Figure 4.8 shows the ratio of correctly estimated events to all estimated events. It is clearly favorable to reject a considerable amount of events as outliers. The accuracy was highest when 30%-50% of non-noise input events were rejected. The size of the event cloud has a smaller influence on accuracy, as long as it is over 2,000. Accuracy increase slows down with larger clouds. But since the computing speed does not deteriorate with increasing event cloud size, the size that performed best can be chosen. For the subsequent analyses, the event cloud was set to 6,000 and outlier rejection to 50%. Table 4.4 contains the results achieved with this setting broken down for each data set and compares them with values obtained by other methods in recent literature. Note that, the

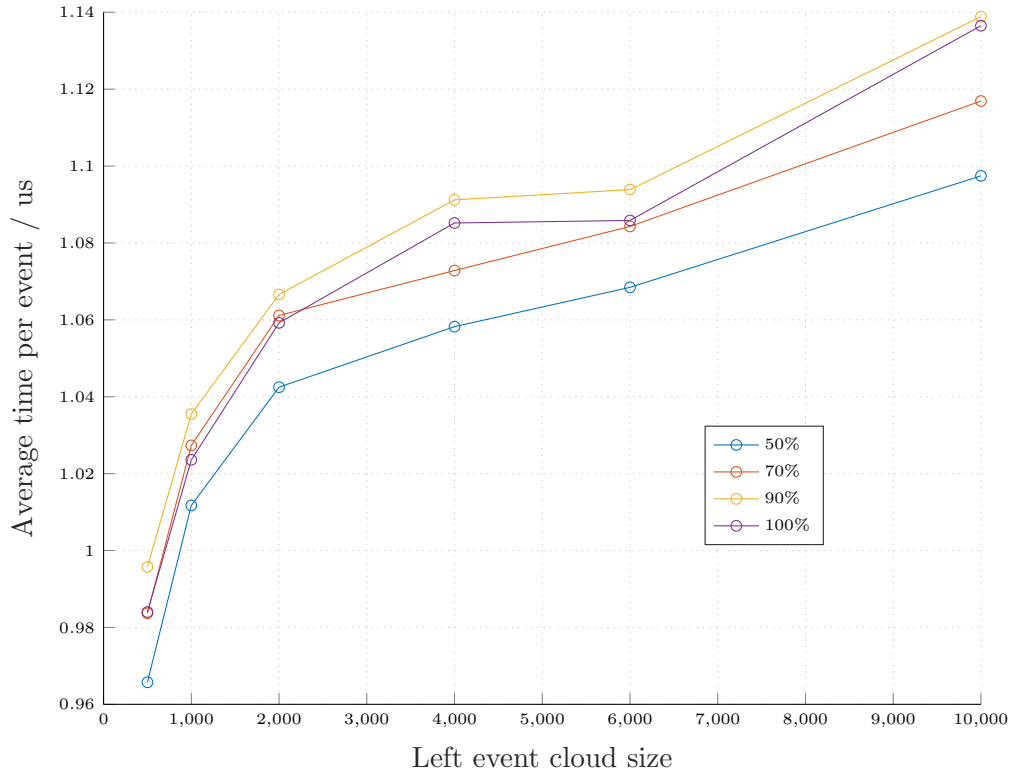


Figure 4.7: Computing time per event in dependence on event cloud sizes for different percentages of inlier events (averaged over 60 on each data set)

presented values do not necessarily reflect the highest accuracy gained for each set, because we evaluated all datasets with the best overall parameters and avoided tweaking parameters to optimize individual set. The different values for the estimation rate come from filtering noise and repetitive events. Every event that was not classified noise was assigned a disparity (with the literature values, it is not clear which events were not labeled.). A disparity was also assigned to events that were not used during the registration processes via the obtained transformation.

Surprisingly, the disparity accuracies this type of registration yielded were only mediocre, more specifically, they were worse than the simple approach of overlaying of the centers of gravity of the event clouds with the same parameters (see Figure 4.9). Interestingly, registration even gave less-than-optimal results for easier stimuli like a box, which basically translate to two lines when viewed with a static DVS. We analyzed the cause of the unexpected result. The analysis revealed a problem with the point matching step of the optimization scheme. As described above, ICP operates by minimizing the sum of squared distances between point pair, where a point in the data cloud is paired with the point in the reference point that lies closest by Euclidean distance. When the data cloud is transformed with an affine transformation the events' spatial distances are not integers in pixel anymore; the grids of reference and data cloud can not be aligned anymore. This leads to the effect that, only considering the x-axis, the closest points can lie on either side of a data point, Figure 4.10 visualizes this. It shows a small slice (5 pixel rows) of the reference event cloud and a optimally transformed data cloud for the box recording projected to the x-t-plane. The transformation we are looking for in the shown case has a larger displacement in the negative direction along the x-axis. However,

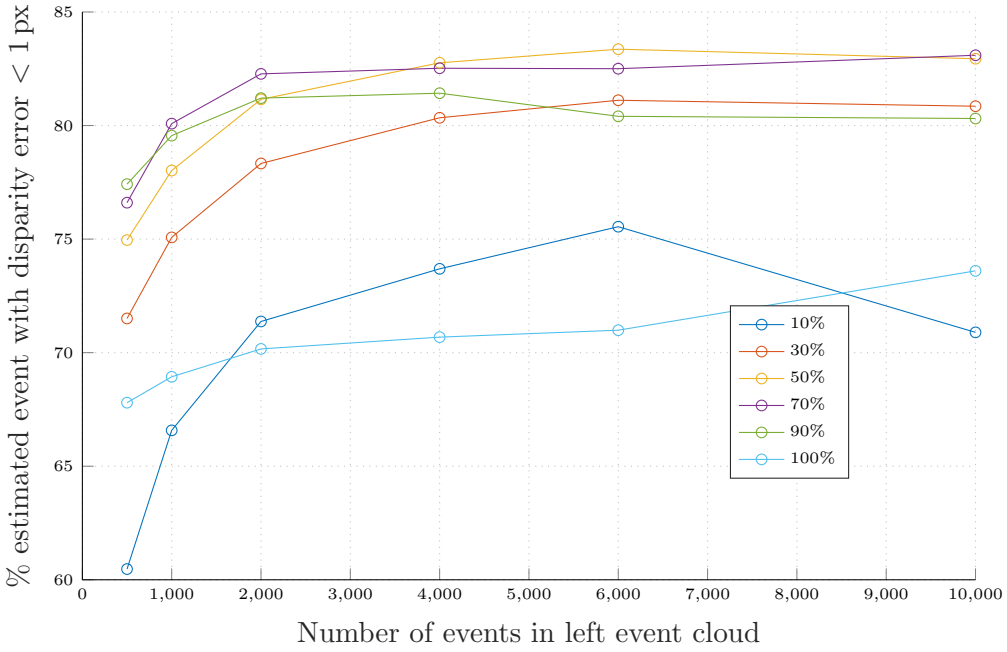


Figure 4.8: Accuracy of geometric registration averaged over all applicable datasets

such a displacement would increase the distances between pairs in which the reference cloud has the larger x-coordinate and is, therefore, not an optimal transformation according to the optimization rule described above. The data cloud is stuck in the reference cloud. The error in the disparity estimate is typically not very large (as it could be in the window-based case). The one person, different depth data set is a very difficult set because at several points during the recording the event clouds are not similarly shaped (cf. Section 3.5). This complicates disparity estimating significantly and explains the poor accuracy. Linear stimuli that caused the described problem are, however, frequent in DVS recordings, so we expect this effect to occur often and there is no simple remedy for it. The pixels are at discrete locations, abandoning the affine transformation renders the method to the center-of-gravity overlay which was also evaluated as comparison, but is certainly no appropriate method for more complicated recordings.

Problems arise as well at the edge of the fields of view. When an object is completely visible in one retina, but only partially in the other one, the event clouds will of course look very different, and registration will fail due to violation of the assumption that the clouds belonging to the same object look similar.

The fact, that this method is already unsuited for easy types of scenes without further tweaking it, and its limitation to rigid objects, that would make non-trivial stream segmentation preprocessing in more complex scenes necessary, led us to the assessment that this path or development will likely not result in a functional and robust stereo matching. Therefore, we continued with a different registration approach which is subject of the next section.

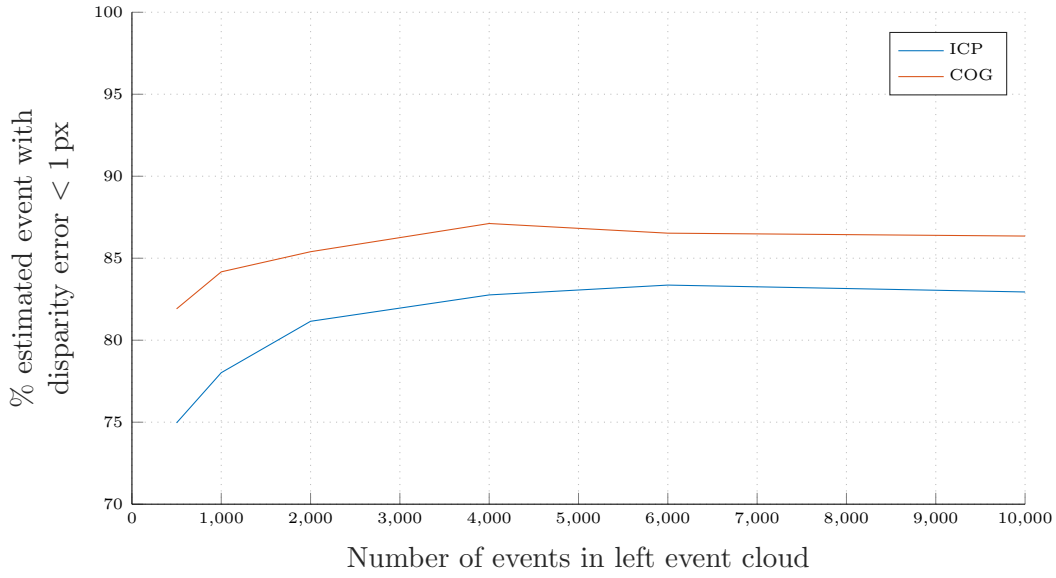


Figure 4.9: Comparison of the registration (ICP) with center-of-gravity overlay (COG). 50% events kept as inlier for both methods.

4.3.2 A probabilistic approach to registering event streams: Coherent Point Drift

The second approach formulates point cloud registration in terms of probability theory. As with ICP the goal is to find a transformation to align two point sets. In Coherent Point Drift (CPD), this is expressed as probability density estimation.

CPD has advantages over ICP. It does not require the transformation to be rigid. So, it can deal with disparity discontinuities in the data. This could not be done with rigid transformations as in ICP. Furthermore, it models the point clouds in terms of probability distributions, so it does not require geometric pairing of point pairs, which lead to problems in ICP. The number of points are not required to be (approximately or even closely) the same. These favorable properties make it applicable to more scenarios in event-based stereo matching.

4.3.2.1 Algorithm

We will now give a brief introduction to the algorithm, which based on [63]. The input consists of two point clouds that we want to map to each other. One point cloud is used to construct a probability density function, the other is regarded as data drawn from the distribution. We are now interested in a transformation on the probability density function that maximizes the likelihood of the data to be drawn.

CPD uses a Gaussian Mixture Model (GMM) to construct the distribution: the points in the cloud are the GMM centroids, the variance of the centroids is assumed to be the same for all centroids. A uniform distribution is added to the model to better deal with outliers and noise. The GMM parameters are iteratively adapted to match the data distribution better based on an Expectation Maximization (EM) optimization scheme.

Let us consider point set L containing N points and the point set R containing M points respectively. The initial probability distribution is

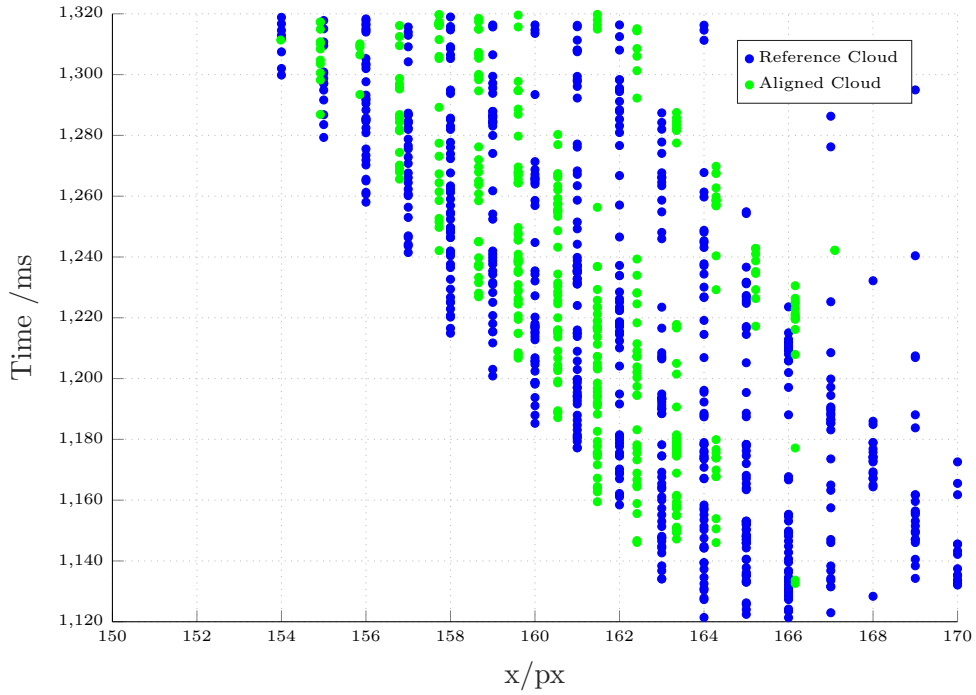


Figure 4.10: Problem with registering event clouds. The affine transformation de-aligns the grids in a way that some closest neighbors are to both sides of events along the x-axis during point pair search. (Depicted is a slice of an event stream of thickness $\Delta y = 5px$ projected to the x-t-plane)

$$p(x) = \sum_{m=1}^{M+1} P(m)p(\mathbf{x}|m) \tag{4.15}$$

with Gaussian distributions $p(\mathbf{x}|m) = \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}_m\|^2\right)$ and uniform distribution $p(\mathbf{x}|M+1) = \frac{1}{N}$. Assigning equal membership probabilities to all Gaussians and introducing a parameter $w, 0 \leq w \leq 1$, that scales the weighting of the uniform distribution, the actual mixture model can be written as:

$$p(x) = w \cdot \frac{1}{N} + (1 - w) \cdot \sum_{m=1}^{M} \frac{1}{M} p(\mathbf{x}|m) \tag{4.16}$$

As commonly done in probability theory, we do not operate directly on this function but reparametrize it in terms of the negative likelihood function $E(\theta, \sigma^2)$ with θ being the positions of the centroids:

$$E(\theta, \sigma^2) = -\sum_{n=1}^{N} \log \sum_{m=1}^{M+1} P(m)p(\mathbf{x}|m) \tag{4.17}$$

Non-rigid registration still is a challenging task in computer vision. Non-rigid transformation is a vast class of transformation. The authors, therefore, pick a regularization framework and define the transformation T that acts on the point cloud R via a displacement function $y(R)$:

$$T(R, v) = Y + v(Y) \tag{4.18}$$

Then a regularization term ϕ is introduced:

$$f(v, \sigma^2) = E(V, \sigma^2) + \frac{\lambda}{2}\phi(v) \tag{4.19}$$

with the log-likelihood function E and a tradeoff parameter λ. The displacement function v is derived via in an intricate manner using variational calculus which can be found in the original work. The final results leads to an iteration rule for the final EM algorithm.

Important parameters of the non-rigid transformation are λ and β. They penalize deviation from a smooth transformation. Especially, λ can be adjusted to trade-off quality of fit versus smoothness of transformation. β defines the width of the regularizer Gaussians.

4.3.2.2 Evaluation

To give an impression of the shape of event clouds, an accumulated event stream (Figure 4.11) and the corresponding three dimensional event cloud from different positions (Figure 4.12) is shown (for clarity, only every fourth event depicted in the cloud picture). Data is taken from the 'Two persons' dataset, which contains a recording of two persons walking at different depths. The right event cloud is shaped similarly, but with events shifted according to their disparity. Figure 4.13 illustrates the result of the registration. In Figure 4.13 a) the two event clouds are shown in the same coordinate frame before registration is applied. The events on the left side, which belong to the person closer to the sensors, have a larger relative displacement than the events on the right side, which belong to the person further away from the sensor. Figure 4.13 b) shows the result after applying CPD: both event clouds have been aligned. There are some obvious errors, e.g. the left (red) events with $x > 100$ and $x < 120$ which were mapped to a location far from any right (blue) event structure, but the overall alignment seems correct. Both clouds were aligned with a non-rigid transformation that was able to find different displacements for different cloud sections, indicating that CPD can indeed be used to match multiple objects at the same time.

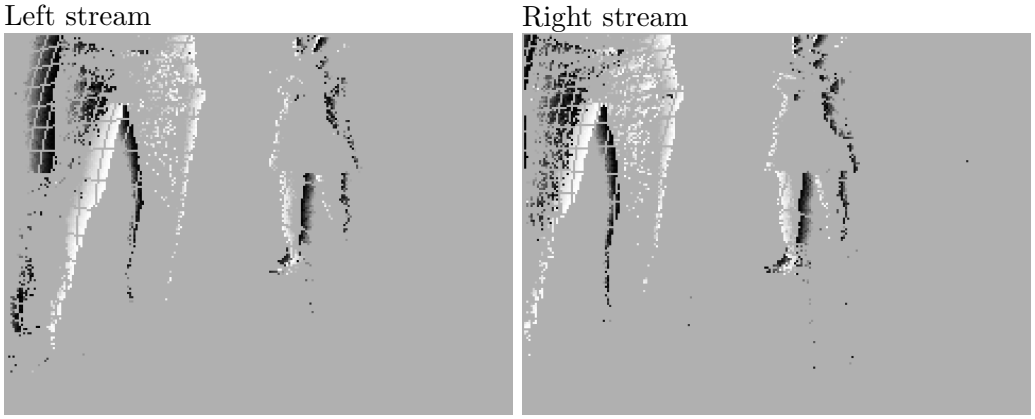
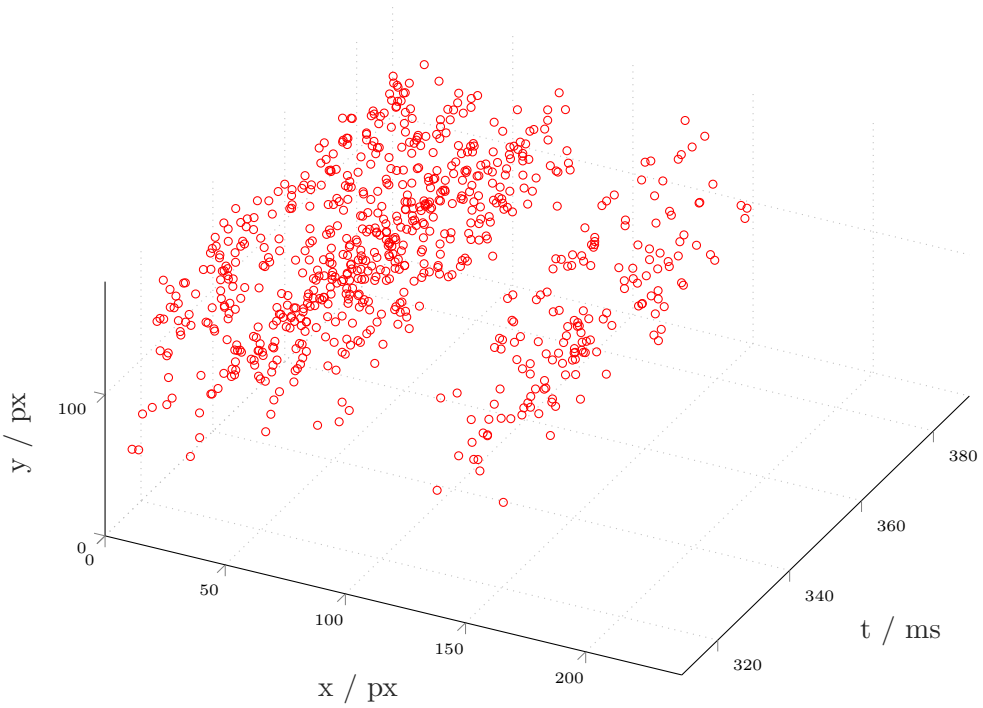


Figure 4.11: Section from 'Two Persons' dataset represented as accumulated event stream.

For a quantitative analysis, we applied CPD to all datasets described in 3.5. The analysis was done using MATLAB and compiling expensive calculations as C-code for acceleration using MATLABs MEX functions. We first performed a grid search over the parameters λ and β on a logarithmic scale. Figure 4.16 visualizes the results. The two parameters share a complex relationship. The mean accuracy over all datasets can both

a)



b)

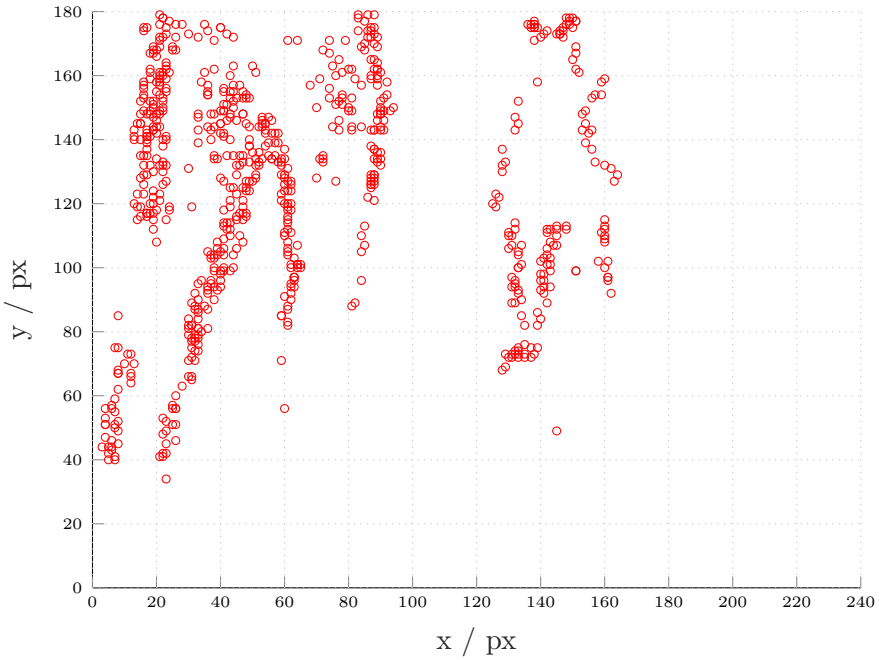
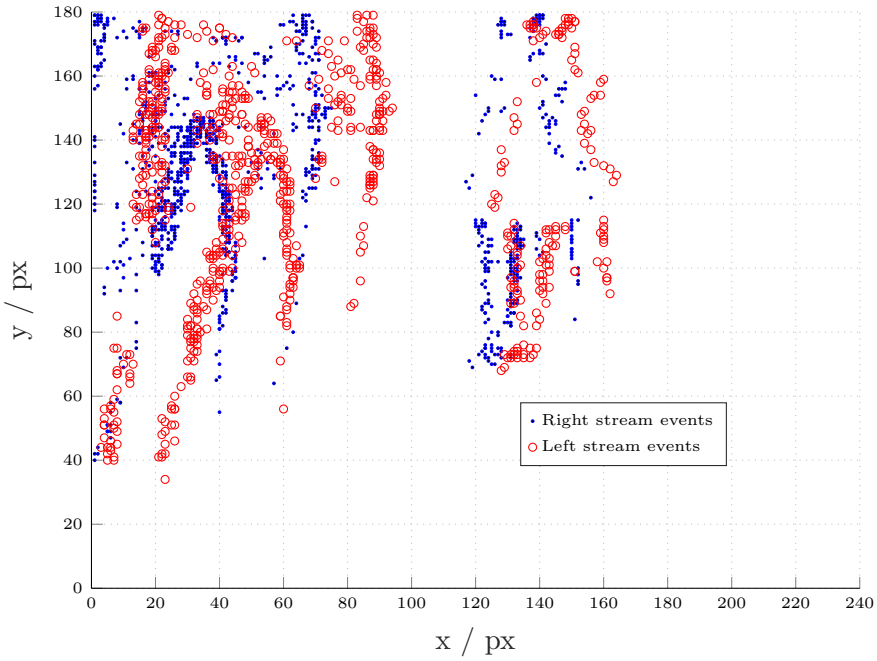


Figure 4.12: Section from 'Two Persons' dataset shown in Figure 4.11 represented as event clouds. a) Oblique view, b) Front view

increase or decrease with increasing λ depending on the value of β, both parameters are coupled and can not be tuned independently; the initial grid search is inconclusive for a definitive tuning of the parameters and must be iterated on a finer scale to obtain optimal values. However, before taking a deeper look at the parameters, there are two things to note: computing time and overall average accuracy.

a)



b)

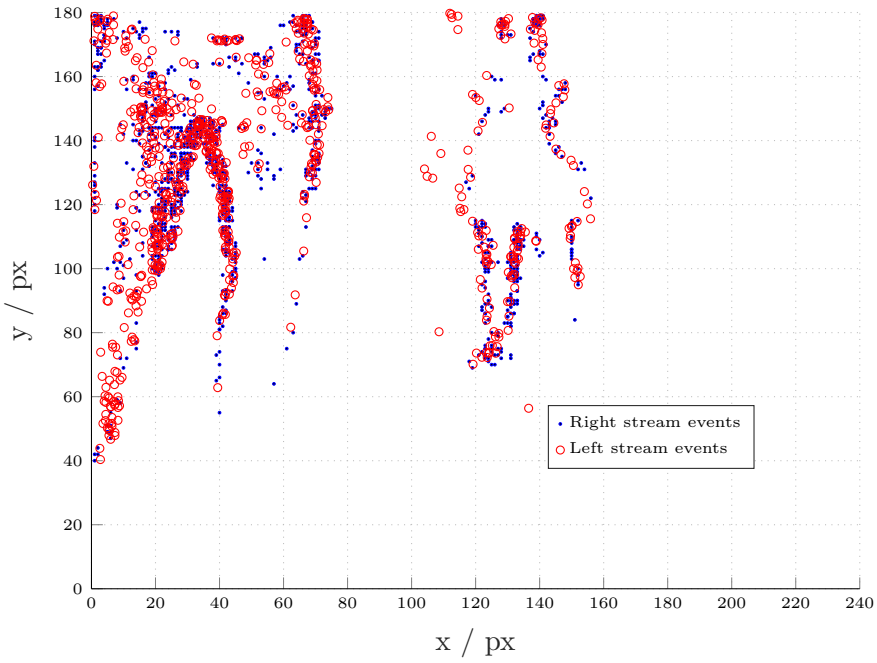


Figure 4.13: Registration process. a) Left and right event cloud in the same coordinate frame before registration, b) After registration

CPD uses an iterative expectation-maximization scheme and, as underlying probability density function, a Gaussian mixture model with a large number of centroids. The optimization equations are very complex compared to ICP and require a lot of computing time. Figure 4.14 shows the average computing time per event for different sizes for event clouds. It processes approximately 100-700 per second. Computing times per event are lower for easier scenes (boxes) and grow with the size of the whole event cloud. But most importantly, the computing times for CPD are orders of magnitude larger than in window-

based stereo matching and iterative closest point registration, both of which are based on much simpler equations and optimization schemes. This can partially be explained by the fact that the computations were partly done in MATLAB. So, there is potential for speeding up the computations using completely compiled code. However, the speed up of several orders of magnitude, that is required to arrive at computing times that can realistically process event-streams in a real-time scenario, seems far from achievable by using only translating the code to a compiled language.
Secondly, the overall average accuracy as assessed by the parameter sweep over wide range of values, lies far below 70%. The combination of requiring intensive computation and yielding suboptimal results, lead us to the conclusion that CPD is not suitable for building a fast and reliable stereo matching algorithm. The matching accuracy compares

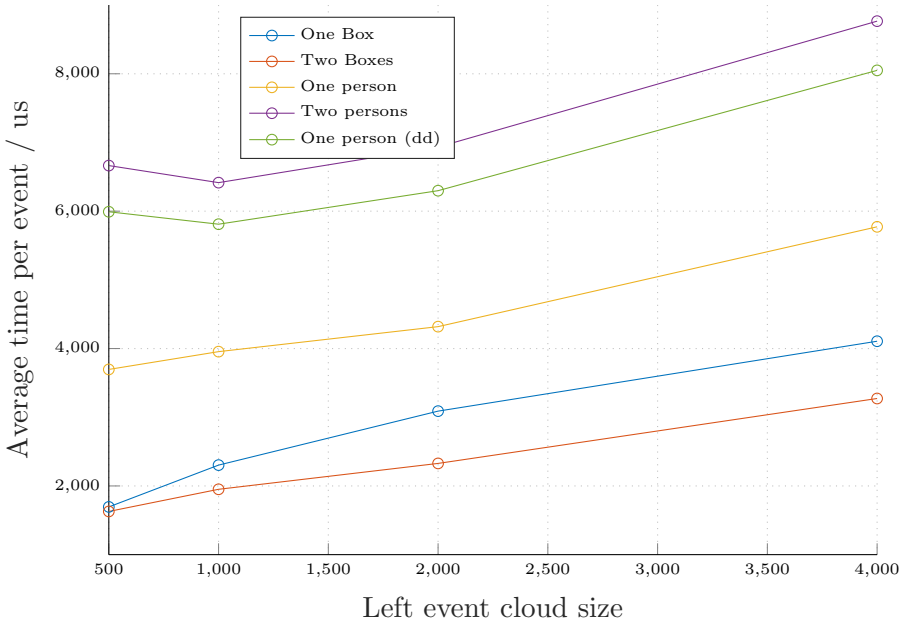


Figure 4.14: Average computing time per event for the different data sets. Geometrically more complex scenes (persons) require longer computing time than simpler ones (boxes). Parameters were chosen as $\lambda = 100$ and $\beta = 1$

unfavorably to the matching with ICP even for the best tested values of the parameters. Compared to the geometric ICP, CPD requires orders of magnitude more computing time. This is shown in Figure 4.14.

4.4 Summary and Discussion

We introduced and analyzed multiple matching methods for event-based stereo. First, a window-based approach, that searches for every event the area that is most similar to the event's vicinity along the epipolar line. It optimizes a cost function based on the timestamps of the neighboring events. We investigated the effects of parameters and found that the window size trades off accuracy and computing speed. Estimating disparity on a ground truth dataset and were able to outperform other algorithm available in the literature on multiple recordings even when tuning parameters towards real-time capability. In general, the approach yielded satisfying results, especially taking its simplicity into account. Drawbacks of the algorithm are on the one hand, that it is fundamentally bound to integer pixel accuracy, since windows are based on pixels. For small disparities this limits

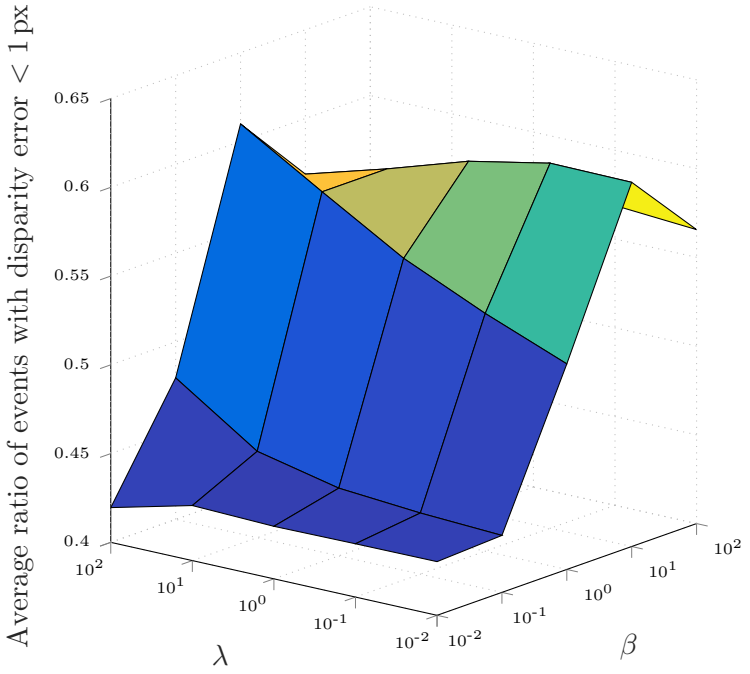


Figure 4.15: Average matching accuracy for different values of λ and β. The parameters are coupled and can not be treated independently.

the depth resolution significantly. On the other hand, there is the problem of having no means to enforce consistency globally or over longer distances. It takes single events and matches them independently from each other. The attempt to introduce a postprocessing step to generate smoothness and consistency based on a confidence measure did not improve the results. This is because wrong matches also often have a high confidence score and are correlated, that is they are spatially close and have the same (wrong) disparity. This causes the postprocessing step to keep the wrong matches; many correlated events seem consistent. The problem with sensing many events at the same area with the same disparity is that it leads to the perception of 'phantom objects', objects that do not exist at the perceived distance.

In order to enforce global consistency, we moved to global matching methods next. The nature of event-based data suggest using point cloud alignment techniques, also known as point cloud registration. We, first, modified a geometric registration algorithm, iterative closest points, to comply with the geometric requirements of a rectified stereo setup. The method is constrained to affine transformation, which limited its use to single objects. Analysis of the properties found that the registration is not robust even in simple scenes (like a single box) due to the grid structure of pixels. This makes the accuracy worse than a simple center-of-gravity overlay. Taking this finding and the fact that ICP is constrained to simple single objects into account, we moved on to a more complex registration technique: coherent point drift. Evaluation showed that this algorithm is capable of matching multiple objects at different depths using a non-rigid transformation. The increased complexity requires, however, much more computing time, rendering the method unusable in a real-time setting. In addition to the accuracy and computation time issues, there is a more fundamental problem when applying any type of registration for event-based stereo vision. The registration can only start as soon as the entirety of the clouds is known, that means, only when the last event arrived so, the event streams have to be accumulated, conceptually comparable to building images. The clouds are processed

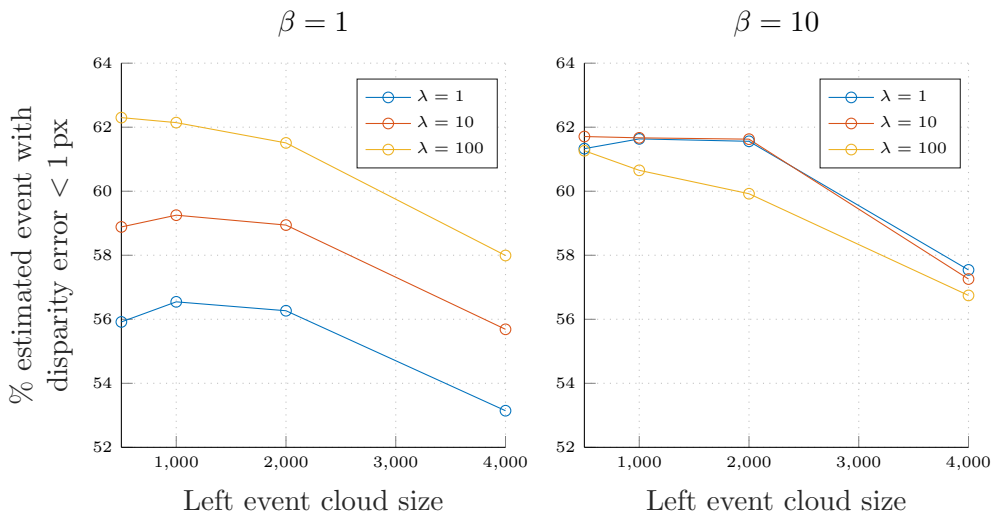


Figure 4.16: Accuracy for different values of β and λ

as batches, resembling frame-based processing style. This loses the major DVS advantage of low latency. So, summarizing, the performance of methods using event aggregations was not satisfactory and we did not achieve our goal of developing a way to enforce global consistency in the matching process. We, therefore, changed focus towards a different class of algorithms: feature-based stereo matching, which is described in the next chapter.

Data set	Method	Estimation rate	Estimation accuracy
One Box	ST	40.74	68.16
	STS	49.16	73.33
	Cop-Net	71.78	75.29
	EMP	82.16	77.15
	WBS(11)	84.45	83.26
	WBS(15)	85.29	82.67
One person	ST	45.43	52.33
	STS	47.87	56.53
	Cop-Net	50.77	74.10
	EMP	94.55	92.00
	WBS(11)	79.26	87.88
	WBS(15)	76.61	90.73
Two Boxes	ST	34.98	54.25
	STS	34.34	62.61
	Cop-Net	61.13	75.29
	EMP	73.64	82.21
	WBS(11)	85.97	73.97
	WBS(15)	83.80	75.00
Two persons	ST	43.06	42.59
	STS	40.89	47.29
	Cop-Net	49.73	67.08
	EMP	92.71	70.64
	WBS(11)	85.85	72.39
	WBS(15)	83.34	76.19
One person different depth	ST	37.86	41.33
	STS	35.42	46.08
	Cop-Net	53.84	40.78
	EMP	58.36	61.14
	WBS(11)	66.76	53.45
	WBS(15)	57.64	58.67

Table 4.2: Comparison of the here presented event time comparing window-based stereo matching with other methods from literature (values and acronyms for other algorithms from [43], ST: space-time correlation method [40], STS: space-time surfaces [61], Cop-Net: cooperative stereo network [45], EMP: event-based message passing [43], WBS(n): window-based stereo with window size n (proposed algorithm))

Data set	Unrefined	Refined
One Box	83.26	79.89
Two Boxes	73.97	73.16
One Person	91.64	87.88
Two Persons	72.38	68.53
One person different depth	53.45	52.79

Table 4.3: Comparison of the matching accuracy for the unrefined and refined versions of the window-based stereo matching algorithm

Data set	Method	Estimation rate	Estimation accuracy
One Box	EMP	82.16	77.15
	ICP	74.89	86.52
	COG	74.89	93.41
One Person	EMP	73.64	82.21
	ICP	54.73	94.29
	COG	54.73	95.94
One person different depth	EMP	58.36	61.14
	ICP	68.39	62.26
	COG	68.39	66.15

Table 4.4: Comparison of the ICP matching results with the method presented in [43] (EMP) and a simple overlay of the center of gravity of both event clouds (COG). This method performed better than methods from literature (compare Table 4.2 for results of more algorithms). However, surprisingly, it performed worse than center-of-gravity alignment.

5 Event-based features and feature matching

This chapter describes the development of a feature extraction algorithm for event streams and a subsequent stereo matching algorithm based on extracted features. Reliably identifying interest points in images has long been a central area of research in computer vision ([68], also see Chapter 3). Here, the need for investigating features arose from the unsatisfying results of depth estimation based on single events or aggregation of events shown in the previous chapter. These methods had undesired properties with regards to computing time, robustness, matching quality and limitations. This chapter will focus on a different approach towards solving the matching problem: feature-based stereo vision. Feature-based stereo vision divides the matching problem into two steps: extraction of features and matching of features. Instead of directly matching single events or event clouds as before, we first aim at finding a robust descriptor for areas in the stream that contain more information than single events, examples of this are for instance corners or lines. Events are labeled as being part of a certain structure. We then perform the matching on features instead of events. This has some conceptual advantages: By matching geometric objects, we maintain the benefit of being able to work with floating point coordinates and the potential to achieve subpixel accuracy. The number of features is very low compared to the number of events. Event rates in streams lie typically in the order of hundred thousand events per second if the sensor is in motion, whereas a typical scene usually contains much fewer than a thousand features like corner and lines. Additionally, features contain much richer information (e.g. size, orientation, shape), so finding correct correspondences is fast and less error prone. Features furthermore offer a possibility to sensibly downsample the event stream by reducing the density of events at feature locations to the amount necessary to keep track of the features and not process the surplus events. This potentially allows more processing time for events in regions where no features where found while simultaneously not discarding information.
The disadvantage of a purely feature-based stereo matching algorithm is that only regions in the stream in which features are found are being matched. Events that do not belong to a feature are discarded and do not get labeled with a disparity.
In this chapter, we will first analyze event-based corner detectors. Subsequently, we will introduce a fast and robust feature extractor based on linear segments in the event stream and compare it with other line detectors. Then, we then submit the linear segments to a matched scheme and use them as landmarks to arrive at a reliable depth estimation for events.

5.1 Event-based Corner Detection

A fundamental structure in visual data are corners. After detecting corners in the event-stream, they can be separated from the remaining events and separately be matched. For the first step, the corner detection, there are algorithms available in the literature. Vasco et al. [53] transferred the Harris corner detector, which is widely used due to its relative numerical simplicity and reliability, to the event-based domain. Mueggler et al. [54] did

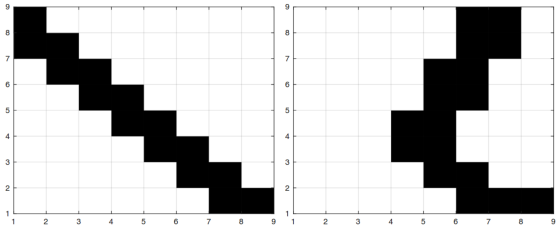


Figure 5.1: Binary surfaces generated from event steam showing an edge (left) and a corner (right). Black corresponds to locations where an event occurred (Figure adapted from [53])

this with the FAST corner detector. These two methods will be examined for their suitability for stereo vision.

The original Harris corner detector measures spatial intensity gradients around a pixel. It is indicative of a corner to have a large gradient in two different directions (compared to a straight edge which has a gradient only in one direction). Mathematically, this can be found by a score value depending on the eigenvalues of the Harris matrix that contains terms of gradients in orthogonal direction. If both eigenvalues are large than the pixel is labeled as corner, otherwise it is not. Since there are no intensities in event-based vision, a binary map of the vicinity of an event is taken as input (Figure 5.1). The main parameter of the method is the threshold value of the score function. We implemented the method as presented in [53]. Figure 5.2 shows results for three scenes: a checkerboard with clear corners, the 'Two persons' scene from ground truth data provided in [43] and the scene with a table, boxes and cones (Section 3.5). Events labeled as corners are drawn in red. In the upper row (a-c), a low threshold was used. Many events are identified as corners, even at locations where there clearly are none, e.g. the borders of the checker board pattern in a) or the foot of the left person in b). So, for the images in the lower row (d-f) the threshold was increased. This reduced the number of corner events significantly and lead to good recognition of the clear corners in d). On the other side, there remained still spurious corner events in e), but more importantly, many of the corners in the more cluttered scene f) were not detected, for instance at the table's top, while wrong detections are still observed, e.g. at the base of the cones or the table's top's sides.

The second corner detector is based on the Features from accelerated segment test (FAST) detector. The original FAST detector labels a pixel as corner if all intensities in a contiguous circle segment around it are all higher or all lower. To adapt it for event-based vision, the intensity was replaced by the event timestamp. The algorithm labels an event as corner event by scanning circles around it for segments of 45° to 90° width with temporally close timestamps. If segments of the appropriate length are found, the event is label as corner. We implemented the event-based FAST detector as well. Figure 5.3 shows results for the three scenes also used for the Harris detector assessment. Compared to the Harris detector, it fails to label all the corner event in the checkerboard scene a), however, it overall finds more corners in the cluttered scene c), especially at the table's legs and cone points. There are still many wrongly label events, for example at the table's top and in the noise parts of the scene.

The many wrong detections in subfigure b) are related to a deeper problem with corner detection which arises specifically in the setting of stereo matching. For proper stereo matching the sensors need to be rectified (Section 2.1.4). Due the forward manner in which rectification is applied events have floating point number coordinates. The pro-

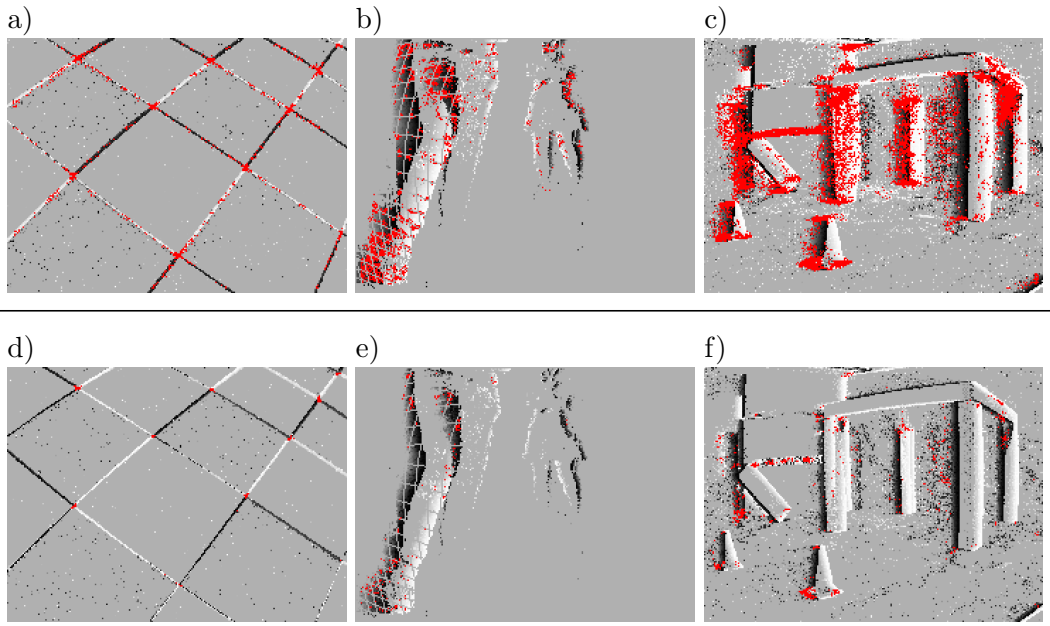


Figure 5.2: Application of the event-based Harris detector to three different scenes. a) - c) low score threshold ($t = 0.04$) to label events as corners, d) - f) high threshold ($t = 0.07$)

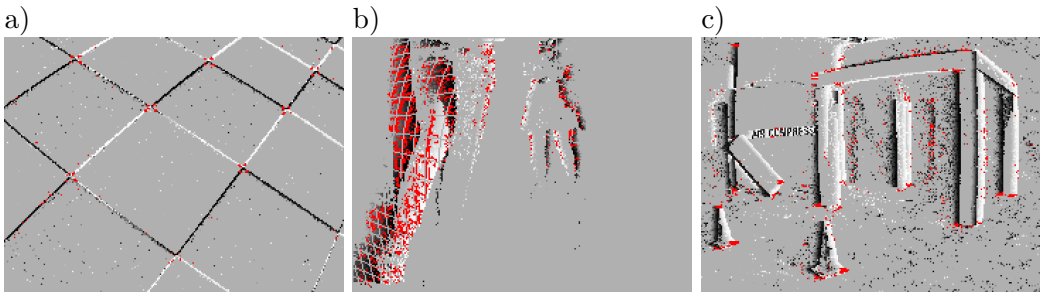


Figure 5.3: Application of the event-based FAST detector

posed corner detectors scan the neighborhood of events based on integer coordinates, floating point values that are not suitable to address positions in the pixel grid. To arrive back at integer values rounding of the floating point is applied; this has been done by Xie et al. with the ground truth data used for evaluation earlier (Section 3.5). Rounding the coordinate values results in pixels never being addressed as can be seen in Figure 5.2 a) and c) and 5.3 a) where pixels in a web like structure do not contain any events. This in turn lets events on the border between addressed and unaddressed pixels have characteristics of a corner and they can be labeled accordingly. The effect is especially visible in Figure 5.3 b) in which many of the events are label wrongly as corners.

Both corner detection methods have undesired properties. Corners are not identified reliably and many falsely labeled corners would generate noise in a potential corner matching scheme. Secondly, although corners are an omnipresent features, they are very localized and formed by only a very small fraction of the stream's events. When reducing streams to corners to much information is neglected to gain a meaningful representation of the observed scene.

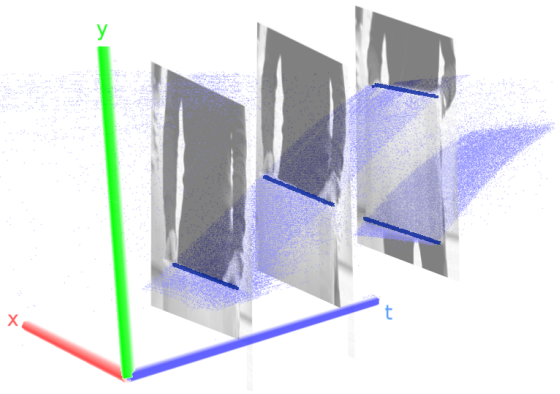


Figure 5.4: Event traces for a box moved from bottom to top through the field of view of a DVS. Visible are dense manifolds of events corresponding to the two edges of the box. Events originating from the movement of the person holding the box are excluded for the sake of clearer visualization. The frames put in the event stream show snapshots of the situation at the time they were triggered. Box edges are indicated by blue bars for better visibility. Note that one axis corresponds to time!

Based on the presented results, we concluded, that corners are not the most suitable feature towards stereo matching. Instead, we focussed on lines, which are a more extended feature and are presented in the next section.

5.2 Line extraction

This section describes a new method to extract lines and linear segments from event-streams, large parts of it have already been published [69]. Lines are an ubiquitous feature in our everyday life. Many structural edges are linear, at least approximately. This can be buildings, door frames, tables, traffic signs, car outlines and the like. They all have shapes that can be decomposed into linear segments. The edges often form a prominent part of DVS steams because, it is at these transitions that brightness difference are large. This makes lines a frequently appearing and interesting feature to detect. Lines serve our ultimate goal to find a fast and robust method for DVS stereopsis well because they are extended features in contrast to corners or other features known from conventional computer vision (like SURF, SIFT etc.). They have a compact symbolic representation, which contains much useful information to disambiguate them, when they are to re-identified in another view on the same scene. They can also build the basis for building visual models of the environment [70]. In the following, we give a explanation the underlying idea of line detection and tracking and a detailed presentation of the algorithm.

5.2.1 Algorithm

The main idea behind the algorithm is to identify planes of events in x-y-t space. On short time scales, straight physical edges move with near constant velocity through the field of view, i.e. acceleration due to physical acceleration or projective transformation can be neglected if the observed time interval is sufficiently small. Therefore, straight edges leave traces of events in x-y-t space that are approximately planar on short time scales. Fig. 5.4 shows event traces of a box that was moved upwards. Note that the duration of this recording would not be considered a short timescale and that the manifolds are slightly curved. Our algorithm aims to identify these manifolds by piecewise

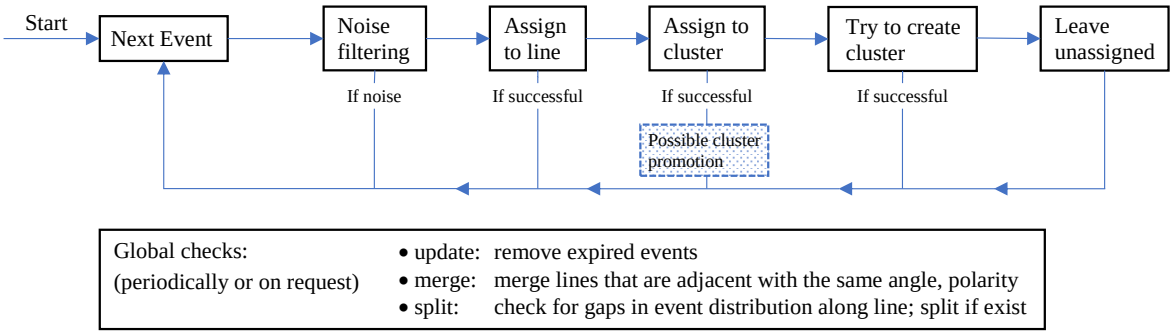


Figure 5.5: Overview over the algorithm. Top: stream part, bottom: batch part running in background

approximation with a chain of planes. Each plane is used to infer the movement of a line in the field of view. Note that this approach does not work with spinning lines and will fail to track these as they do not form planes in x-y-t-space. The detection approach is split into two steps: first, we cluster events that were generated within a small time and space window. These clusters are grown by adding events that are close in x-y-t-space. Secondly, we periodically check if the clusters form planes in x-y-t-space. If they do, we promote the cluster to a line, otherwise we keep them as clusters. Tracking of lines follows by assigning newly incoming events to planes and tracing these planes through time. To avoid that the number of events taken into account in our method is strongly dependent on sensor velocity we use an event buffer with a fixed amount of events at all times and discard old events. Figure 5.5 shows an overview over the complete processing flow that each incoming event will be subject to. Every event is run through a noise filter, then we check if it belongs to a line and, if so, assign it to this line. Else, we check if a cluster is nearby and if that is the case add the event to the cluster. If there is no cluster, nearby events are collected to form a cluster; if we do not find more events than a threshold ν, we leave the event unassigned (ν typically lies in the order of 20-30 events). We will explain every step in detail now.

Event preprocessing First, events are separated by polarity. After separating events, we apply different noise filters: first, we introduce a refractory period per pixel. After a pixel emitted an event we will suppress further events from this pixel within a certain time interval, because pixels sometimes generate additional spurious same-polarity events if they have been triggered before and pixels may emit multiple events of the same polarity if the change in brightness was very strong. Experimentally, we identified 1ms for opposite polarity events and 50ms for same polarity events to work well. All events that are received during this period with respect to their polarity are discarded. Afterwards, an additional filtering step checks for every incoming event if at least 3 same polarity events in a 5x5 pixel window around it have been registered. If not, the event is labeled as noise and not processed further.

In the following, we will continue to explain the algorithm from the end of the processing chain, because it is easier for the reader to follow the whole process starting with the way clusters are initially formed, then promoted to lines, and how these lines are finally transferred through time.

Cluster creation When an event arrives (and could not be assigned to an existing line or a cluster), we use it as seed to search for a chain of adjoining pixels that recently

generated events. First, we search for the youngest event in the ring of the 8 adjacent pixels. If we find no event, we search in the next ring of 16 pixels around the adjacent pixels. If we still find no events, we abort the search. Otherwise, we add the youngest event to our chain and repeat the procedure from the pixel position of the found event. This step is iterated until the chain length crosses a threshold or we do not find any new events.

If we are able to create a chain of events, we cluster the events of the chain, add all events that have been generated by adjacent pixels and store these events as cluster, thereby creating a candidate for a plane.

Cluster growth Moving one step to the left in the process flow (Figure: 5.5): when an event could not be assigned to a line, we look for a cluster to attach it to. To that end, we check if there is an event that belongs to an adjacent cluster. If there is, we assign the event to the cluster. If there is more than one cluster found, we merge these clusters and add the event to the new larger cluster.

Cluster promotion When a cluster has collected enough events (in the order of 20-40 events), we check if its events form a plane in x-y-t-space. As stated above, the underlying assumption is that the velocity of lines on the retina can be approximated as constant on short time scales. Then, non-spinning straight edges in the real world generate flat planes of events. To check if the candidate cluster's events form a line, we compute the principal components of the event coordinates (x, y, t) where we scale the time coordinate with a constant factor. A value of $1000\frac{\mu s}{px}$ was experimentally determined to work well for the robotic platform as well as for handheld movements (in high speed environments this factor may have to adapted):

$$\mathbf{X} = \begin{pmatrix} x_1 & y_1 & t_1 \\ x_2 & y_2 & t_2 \\ \dots & & \end{pmatrix} \tag{5.1}$$

be the matrix, that contains the coordinates of events belonging to the cluster. Assume, that all coordinates (x, y, t) are centered (i.e. $\overline{x} = \sum_i x_i = 0$, etc.). It then holds that,

$$\frac{1}{N}\mathbf{X}^T\mathbf{X} = \frac{1}{N}\begin{pmatrix} \sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i t_i \\ \sum_i x_i y_i & \sum_i y_i^2 & \sum_i y_i t_i \\ \sum_i x_i t_i & \sum_i y_i t_i & \sum_i t_i^2 \end{pmatrix} = \mathbf{U}\mathbf{\Delta}\mathbf{U}^T \quad , \tag{5.2}$$

where i runs over all events and N is the total number of events. For the last step the eigen decomposition was applied. $\mathbf{\Delta}$ is a diagonal matrix containing the eigenvalues of $\mathbf{X}^T\mathbf{X}$, which are guaranteed to be positive real numbers since $\mathbf{X}^T\mathbf{X}$ is real symmetric and positive semidefinite. $\mathbf{U}$ is a orthogonal matrix containing the respective eigenvectors. As a perfect plane forms a linear 2D subspace in the 3D space, one principal value (eigenvalue) vanishes, when we compute the Principal Component Analysis (PCA) on points lying in the same plane. The corresponding eigenvector stands orthogonally on the plane. Event planes in a real world recording, however, will never be perfect since these planes contain noise through various sources like finite size pixels, electronic jitter, delay during event generation, not perfectly constant velocity of observed objects, events stemming from other sources etc. For a planar structure one eigenvalue will nevertheless be very small, proportional to the 'thickness' of the plane. So, assuming that the eigenvalues in $\mathbf{\Delta}$ are ordered in ascending order, we examine $\mathbf{\Delta}_{11}$: if it is greater than a threshold θ, we conclude that the cluster does not form a plane and, therefore, was not caused by an

observed line. We keep on collecting events and trying to promote it until n promotion attempts failed. Then, we drop it (for our experiments, we chose $\theta = 1\text{px}^2$ and $n = 3$).

If the smallest eigenvalue is below the threshold we promote the candidate to a real line (or 3D-plane respectively). The position and orientation of the line at time t_{present} is now inferred by intersecting the approximated event plane with the plane $t = t_{\text{present}}$, further called the *present plane*. The parameters to be determined are a vector $\vec{l}$ pointing along the line, a point $\vec{p}$ which is contained in the line and a the length of the line. We start finding $\vec{l}$ by using the principal component vector belonging to the smallest eigenvalue as normal vector $\vec{n} \in \mathbb{R}^3$. The calculation is straightforward and can be done analytically: we use the fact that a vector $\vec{l}$ pointing along the intersection of two planes must lie within both planes and therefore be orthogonal to both plain normals. A possible choice that fulfills this condition is the cross product of event plane normal $\vec{n}$ and present plane normal e_t:

$$\vec{l} = \vec{n} \times \vec{e_t} = (n_1, n_2, n_3)^T \times (0, 0, 1)^T = (n_2, -n_1, 0)^T \tag{5.3}$$

Having found a vector pointing along the line, the next step is to find a point $\vec{p}$ that is contained by the line. We pick $\vec{p}$, such that it is closest to the events' center of gravity $\vec{c} = (\overline{x}, \overline{y}, \overline{t})$. $\vec{p}$ can then be found using these two observations: first, the vector $\vec{m}$ pointing from $\vec{c}$ to $\vec{p}$ must be contained in the event plane as $\vec{p}$ and $\vec{c}$ are both contained in this plane, i.e. $\vec{m}$ is perpendicular to the plane normal $\vec{n}$. Second, $\vec{m}$ points along the path of shortest distance from $\vec{c}$ to the line, therefore it has to be perpendicular to $\vec{l}$ as well. This makes $\vec{m} = \vec{n} \times \vec{l}$ a natural choice. We then find $\vec{p}$ as intersection of the line $\vec{c} + \alpha\, \vec{m}$ with the plane $t = t_{\text{present}}$ where we determine α using the fact that $\vec{p}_3 = t_{\text{present}}$ and $(\vec{n} \times \vec{l})_3 = -n_1^2 - n_2^2$:

$$\vec{p} = \begin{pmatrix} \overline{x} \\ \overline{y} \\ \overline{t} \end{pmatrix} - \frac{t_{\text{present}} - \overline{t}}{n_1^2 + n_2^2}\, \vec{m} \tag{5.4}$$

Finally, to find the length a of the line we assume that the events are evenly distributed along the whole extension of the line. The length a is then proportional to the standard deviation along the line which we already gained from the principal component analysis. We pick the center of the line as origin and use ξ as a one dimensional coordinate along the line and a constant distribution function $p(\xi)$ over the length of the line a:

$$p(\xi) = \begin{cases} \frac{1}{a}\ , \text{ if } -a/2 \leq \xi \leq a/2 \\ 0\ , \text{ otherwise} \end{cases} \tag{5.5}$$

Then $\langle \xi \rangle = 0$ and

$$\langle (\xi - \langle \xi \rangle)^2 \rangle = \langle \xi^2 \rangle = \int_{-\infty}^{\infty} p(\xi)\, \xi^2 \, \mathrm{d}\xi = \int_{-a/2}^{a/2} \frac{1}{a} \xi^2 \, \mathrm{d}\xi = \frac{1}{12} a^2 \tag{5.6}$$

from which follows that $a = \sqrt{12\, \langle \xi^2 \rangle})$.

We arrived at a line parametrization with midpoint $\vec{p}$, direction $\vec{l}$ and length a. Note that we chose $t = t_{\text{present}}$ arbitrarily! We can use any other time to predict the line parameters in the (near) future without requiring any new input. The corresponding calculations are as outlined only a few three dimensional vector operations and very fast to perform.

Whenever a cluster is promoted to a line we check if the new line's position and slope match the ones of a line that was previously deleted. We require the polarities to be identical, the angular distance to be less than 5° and the midpoints distance to the other line to be less than 2px (although this threshold should be adapted depending on the sensor resolution). If there is a deleted line that matches, we assume that we lost track of the it and recovered it now. In this case we will assign the new line the ID of the deleted line.

Line growth When a new event is received, we check for lines close to the pixel of event generation. We assign the event to the line if it is closer than $\sqrt{\theta}$, with θ the threshold for cluster promotion, to the line inferred at the time of event generation. The threshold could however be fixed independently to make the line collect more or less events generated in its surroundings.

Persistent tracking For larger time spans, the assumption of planarity is violated. This means the principal component analysis breaks down, if events that are too old are used. Therefore, we need to update the inferred planes either periodically or on request, as soon as an accurate estimate is required, by removing events that are older than a certain time or if a line contains many events per length simply by removing the oldest events. After removing these events, orientation of the event plane (and thereby also of the inferred line) will be re-estimated by re-applying PCA and going through all the additional steps described above. Note, furthermore, that this is not an expensive update, since we can store the sum of coordinates for the PCA, and just modify it when adding or removing events from the line. If after an update there remain less than 10 events or the smallest eigenvalue exceeds θ, the line will be deleted. We keep deleted lines' position and orientation in memory in case they are recovered. The following global checks run additionally in the background:

- Clusters are also periodically cleaned by removing old events.
- Lines are checked for coherence: if lines display gaps in the event distribution, they are split into two lines. Gaps are detected by projecting every event position onto the parametrized line, partitioning the line in bins of stepwidth 2px. If two adjacent bins are empty the line is split at this gap.
- Merging of lines: if lines have an angular difference of less than 5° and same polarity, as well as the midpoints' distances to the respective other line are less than 2 px (same values as for recovering deleted lines) and the midpoints' distances to each other are less than the half sum of the lengths, i.e. the lines are adjacent to each other, they are merged to form just one line.

5.3 Experimental results

We performed experiments to evaluate the quality of the matching and tracking, as well as quantifying the latency and computational costs and investigated the robustness. For all experiments we used an Intel Core i7 4770K running at 3.5 GHz. The algorithm was implemented without parallelization in C++.

5.3.1 Quality of matching and tracking

To evaluate the quality, we recorded data sets with the DAVIS240C, capturing both events and frames (frames captured with a rate between 15-20 Hz). To our knowledge, there

exists no data set with ground truth values for event-based line tracking. So, we obtained ground truth values for lines by applying the well-known Canny edge detector ([10]) to every frame. Since there is no definite standard to define and extract line from images, we took the Canny filter output and manually removed bent edges based on human judgment, such that only straight edges (i.e. lines) remained. We then applied linear regression to the coordinates of pixels that belong to lines and arrived at line parametrizations which were used as ground truth. Additionally, we manually labeled the lines of all frames so that corresponding lines got the same ID in every frame to allow to check persistence of tracking of our algorithm.

Our method is able to successfully extract lines from event-based vision streams. Fig. 5.7 displays a snapshot of detected lines in an event stream. Fig. 5.6 compares our results to other methods for line detection on two frames with a 200ms time interval between them using a scene with a staircase which contains many lines. The first row shows the two frames and the corresponding ground truth. The second (resp. third row) show different line detection algorithms applied to the first (resp. second) frame or event stream, depending on algorithm. The leftmost pictures depict the results of the probabilistic Hough transform ([12]) as implemented in OpenCV[1]; the images next to them contain the results of the Line Segment Detection (LSD) ([71]) as provided on the IPOL webpage[2]; followed by ELiSeD ([51]) as implemented in jAER[3] and finally our method (to arrive at the plot for ELiSeD we took all lines that were created from the time the frame was taken up until 10ms earlier). The probabilistic Hough transform (a) is able to detect well contrasted lines, but it runs into problems when estimating the right lengths. The detection images look fairly different from the ground truth. Furthermore, the extracted lines look quite different between the frames what would make inter-frame matching a hard task. In contrast, the line segment detection method (b) yields high quality results. The extracted line segments reflect the ground truth very well. However, longer lines tend to be broken up into segments and would need to be merge before they can be used for tracking in a next step. ELiSeD (c) tends to yield very small segments and to break up longer lines into smaller pieces. That produces lots of very short-lived segments. Meanwhile, the method proposed here (d) is able to successfully extract longer lines and keep track of them. The numbers on the rightmost panels denote IDs we assigned lines after creation. Most of the IDs of corresponding lines in both panels are identical, i.e. they have been successfully tracked. This tracking did not require an additional step, but is gained automatically by applying our algorithm.

To measure the quality, we compared the estimated lines with the labeled ground-truth lines, where we assumed that an estimated line matches a ground-truth line if their difference of angles was less than 5° and the perpendicular distance from the midpoint of the estimated line to the ground-truth line was smaller than 1.5 px. We then obtained difference of angles and lengths for the matched lines: angles of lines are known to be a robust feature to estimate when detecting lines. This holds also for our method where the average absolute angular error over all matched lines was approximately 0.6°, the median absolute angular error approximately 0.4° (on the same data set Hough had mean/median angular error 0.7°/0.4°, LSD had mean/median angular error 0.9°/0.4° and ELiSeD had mean/median angular error of 1.5°/1.1°). In contrast, line length is a rather unstable feature to extract. Using our algorithm we are able to extract lines with a high precision of length. Fig. 5.8 shows a distribution of the relative lengths of the estimated lines to

[1] https://github.com/itseez/opencv
[2] http://www.ipol.im/pub/art/2012/gjmr-lsd/
[3] https://sourceforge.net/projects/jaer/

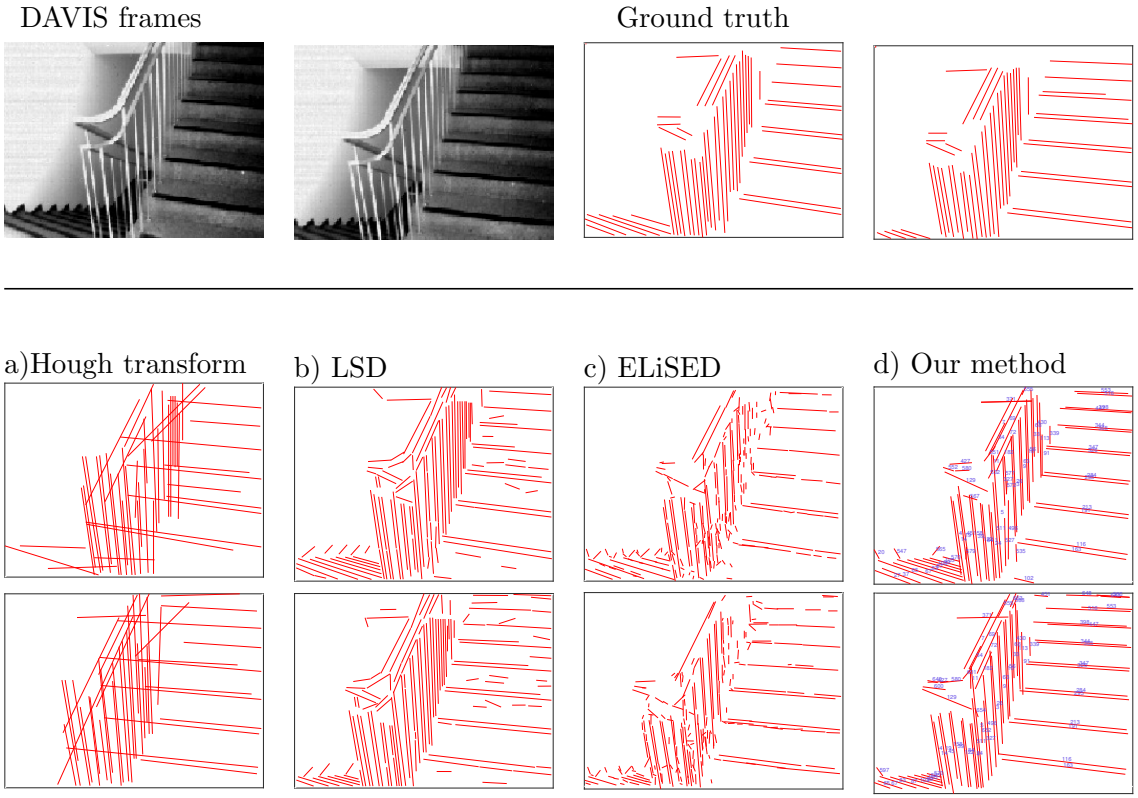


Figure 5.6: Comparison between different approaches. Top row left-to-right: frame taken from a DAVIS240C recording, frame 200ms later, ground truth lines for the first frame, ground truth lines for the second frame. Second row left-to-right: a) Hough transform, b) LSD, c) ELiSeD, d) our method. Third row: same algorithms as above applied to the second frame. In the images of our method lines were additionally assigned an ID to demonstrate the tracking capabilities (cf. text).

the ground truth lines on 52 labeled frames of the staircase data set. For every match of an estimated line with a ground truth line, we calculated the ratio $\frac{\text{length(estimated line)}}{\text{length(ground truth line)}}$ and created a histogram over all matched lines. The distribution for our method peaks comparatively sharply around 100% which means, that the majority of our line estimates are correct in length. LSD also exhibits a peak around 100% but has a tail towards zero, which is caused by estimated lines broken up in segments. ELiSeD produces many small segments leading to a distribution where most matches only cover a small part of the ground truth line and the Hough transformation has big problems estimating lengths, over- and underestimating very often.

5.3.2 Persistence of tracking

The other aspect we aimed towards besides lines detection was tracking, i.e. we should be able to identify every line over the entire time that it is visible with the same ID. In figure 5.6, we attached ID numbers to the lines detected with our algorithm. Most lines that correspond to the same physical lines have the same ID in both images (and at every point in time between the two frames as well (not visible)). To evaluate the persistence quantitatively, we took the difference of the first time a ground-truth line was matched by a specific ID with the last it was matched by this ID and compared it with the duration the ground-truth line was in the field of view. Figure 5.9 shows the distribution of relative

lifetimes. More than half of the lines in our experiments are tracked throughout their whole life time; that means they are found very fast after they enter the field of view and subsequently tracked until they leave it. For a smaller fraction of lines it took longer to identify and track them, so that they are not tracked over their whole visibility, while an even smaller fraction of lines was not found quickly or lost during tracking.

There are certain limitations when working with the DVS. The amount of events that lines generate depends on their angle to the movement direction of the camera. Lines perfectly aligned with the sensor movement direction are invisible to the DVS because only the leading edge presents a luminance change. Therefore, these lines can not be tracked. If a line is being tracked but becomes aligned with the movement after a movement direction change of the sensor it will become invisible, too, and track will be lost. To overcome this, one could include an inertial measurement unit (IMU) and estimate the invisible line's position using acceleration information. However, it turns out that the problem of invisible lines is in practice not severe. Vibrations of the sensor, which can stem for example from a robot's motors or natural tremor in case of handheld DVS, cause the sensor to perform movements in the orthogonal direction of the line which makes them visible to the sensor. We performed an experiment to examine the dependence of the line detection on the angle for which we printed lines with known inclination from 0° to 10° in steps of 2° and recorded the scene using a self-built robotic platform with a mounted DVS that drove in parallel to the stimulus. Figure 5.10 shows the stimulus and the robot we used. Furthermore, it shows the detected lines at the beginning of recording as well as roughly four seconds later at the end of the recording. All lines could be detected, even the line with 0°was tracked because the microvibrations of the robot caused small perpendicular movements which rendered it visible. In fact, these movements generate ON as well as OFF events, leading to a redundant double tracking of the lines due to the fact that we split handling of ON and OFF events. This redundancy can potentially be used to recover from the loss of tracking in one polarity domain. The feasibility of this was not evaluated and will be subject to further investigation.

In the second experiment, we drove with the robot over the seams a of a tiled floor. These irregularities in the surface caused small abrupt movements of the sensor. Figure 5.11 shows tracked lines in such a setting. It contains two snapshots made by a robot driving towards a tiled wall and crossing seams in the floor on its way. Our method is capable of dealing with small irregularities and small amplitude shaking as can be caused by crossing seams. Sudden changes in movements (like large sudden displacements caused by stronger shaking) can, however, not be dealt with and result in the loss of track of lines. The underlying reason for this is that sudden changes will cause a kink in the event trace in x-y-t-space. These kinks can not be modelled well with the chain of planes and our method will fail. As soon as the lines' movement is smooth again, they will typically be regained fast, but with a new ID.

As third experiment, we attached a sensor to a radio controlled model car to evaluate behaviour at high velocity (∼12 km/s). We recorded two different settings and evaluated the results by visual inspection.

In the first experiment, the car started on an checkerboard pattern floor and drove through a door towards another door. Because the floor was smooth, we observed no major disturbances (especially no abrupt changes in motion) and detection and tracking yielded good results. Figure 5.12 presents snapshots from this experiment.

In the second scene, the car drove over uneven floor in a narrow hallway with a comparatively high noise level due to irregular illumination patterns and textures on wall and floor. In this recording detection of lines again yielded good results; tracking, however, was more challenging. While some lines (especially those perpendicular to the car vibrations) could

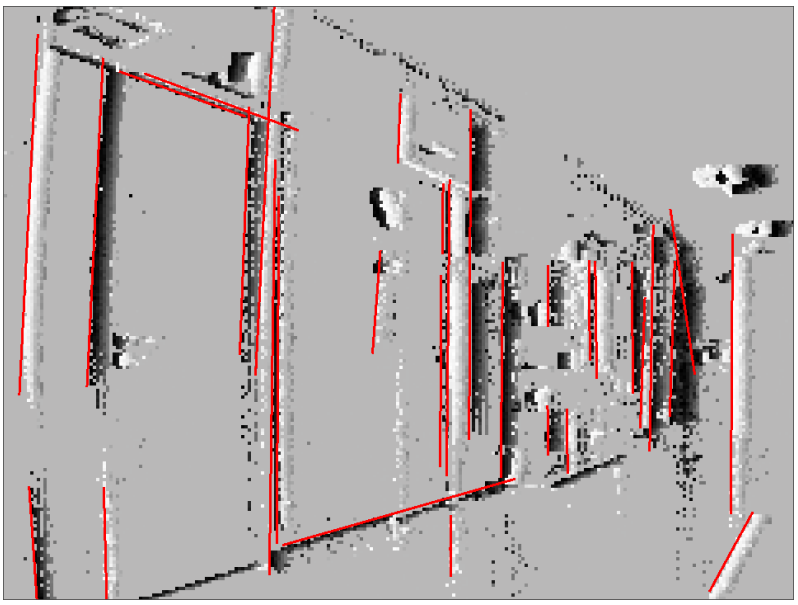


Figure 5.7: Event stream with current position of detected lines (events accumulated for 50ms). Camera rotates clockwise, so lines move to the left and older events trailing to the right of lines are still visible.

be tracked well, others were often lost. This can be explained by the same reason for which we lost track of lines while driving over seams: the car experienced abrupt changes of motion, which lead to kinks in the event plane causing our tracking method to fail.

5.3.3 Latency and computing costs

This section presents an experiment to evaluate the latency. The independent operation of DVS pixels generates a quasi-continuous stream of events. Due to this sensor property the events have already a low delay from illumination change in the scene to reception of the event in a processing device. Each event can be handled individually and is used to update our belief about the current state of the world immediately after arrival. We measured the time it takes to process an incoming event and call the update function that re-estimates the line position using a scene with a swinging pendulum. This gives us a single line traversing the display with predictable translatory speed. To measure the error we obtained ground truth values for the line position in the following way: first, we discarded the OFF events and binned the ON events in slices of 50 milliseconds. We then found the leading edge by picking for every pixel row of the sensor the event that was furthest in movement direction. We used robust linear fitting as built-in in MATLAB to fit a line and reject outliers and inspected each fitted line visually. Fig. 5.13 (left) shows one fitted line graphically.

We compared the position at which our algorithm estimated the line with two different approaches: 1) retrieving the position of the line by interpolating the line movement linearly from the last calculated position of the line until the time of position request and 2) calling the update routine and refitting the line before returning the position estimate. For the case of event processing without line update the average required time was approximately $0.7\mu s$ with an estimation error of 0.50 pixels. When we do an explicit line position re-estimation using PCA and vector recalculation as described in Section 5.2.1, the average time required was approximately 7 μs and the average estimation error was 0.48 pixels. So, the latency between arrival of new information (incoming event) and new

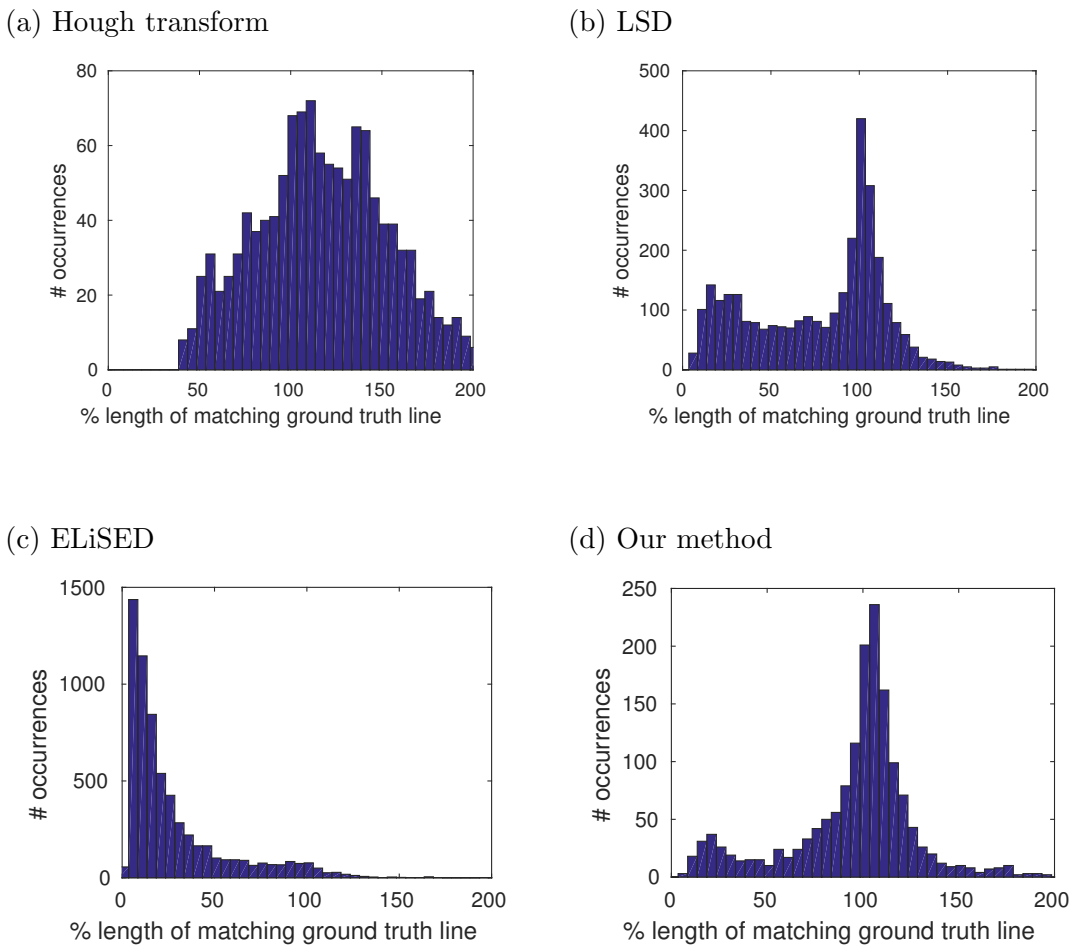


Figure 5.8: Distributions of length ratios between estimated lines and matching ground truth lines for (a) Hough transformation, (b) LSD, (c) ELiSeD and (d) our method in percent

belief (updated line state) lies in the order of a few microseconds. Note, that we normally do not upgrade after each event! The reason why explicit updating does not reduce the error significantly for the pendulum lies in the fact that the line is very well visible and moves very smoothly, so the interpolation of its movement is already quite accurate. Figure 5.13 (right) shows the estimated line position at certain arbitrarily chosen checkpoints versus the true line position for the case with line update for a fixed value of y. The red line shows the position of the line, while the blue crosses indicates the position estimated by our algorithm at the time after the computation (i.e. at the time of the event + the time required to estimate the position). The computing time is sufficiently low to make the estimation error introduced by the delay negligible.

In addition to latency, the overall computation load is another important quantity for judging the usefulness of an algorithm. Figure 5.14 shows the dependence between number of events and required computing time for a number of different recordings for a variety of scenes. The red line gives an estimate for the worst case performance, suggesting that the algorithm can handle at least 400k events per second in the current implementation. On the right hand side, the computing load during one recording with a large number of lines (the staircase scene) is shown. The recording was partitioned in slices of 100ms; the computation time and number of events in every slice was then measured and plotted in

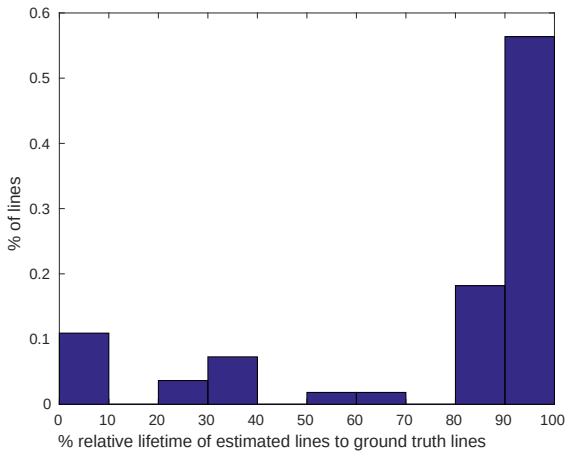
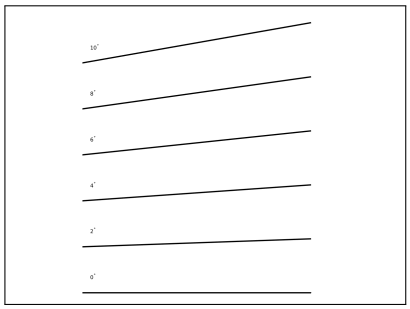


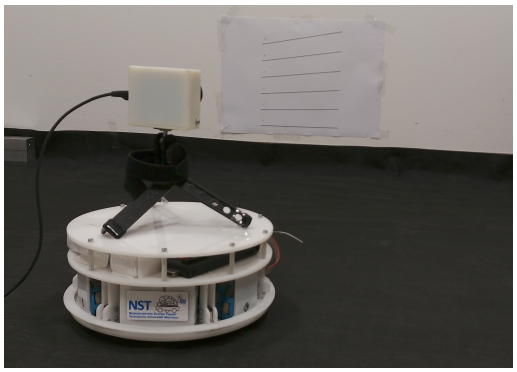
Figure 5.9: Histogram over ratios of lifetimes of estimated lines to life time of ground truth lines in percent

the figure. The graph shows an approximately linear relation between number of events and time required to process them.

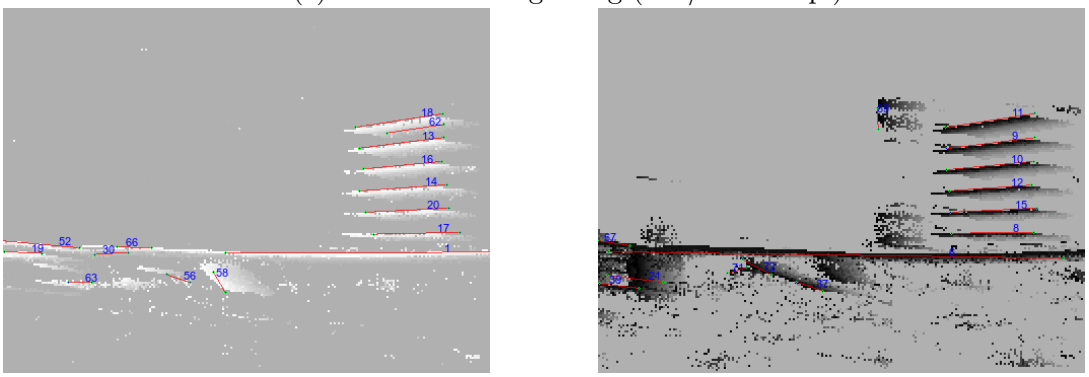
(a) Stimulus image



(b) Experimental Setup



(c) State at the beginning (ON/OFF resp.)



(d) State towards the end (ON/OFF resp.)

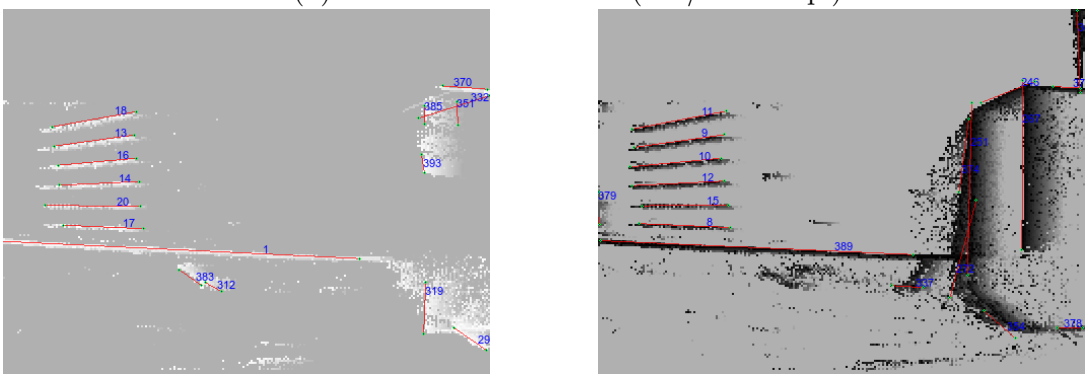


Figure 5.10: Dependence of line tracking on line orientation. Top row: (a) stimulus used: lines with increasing degree versus the camera movement in steps of 2° and (b) camera setup on robotic platform, robot was moving to the right during recording. Second/third row: (c) tracking results at the beginning of the recording for ON/Off events. (d) tracking results towards the end of the recording. Comparing the IDs, that the lines were signed in the images, it can be seen that they were successfully tracked despite being close to parallel to the movement direction of the sensor (lines become visible due to microvibrations of the robot).

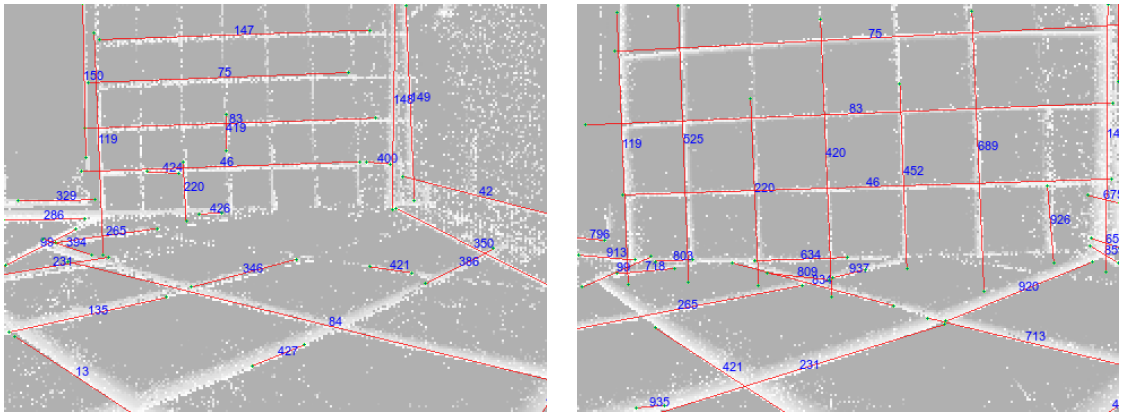


Figure 5.11: Line tracking results for a robot driving over small irregularities caused by a tiled floor. Comparing line IDs shows that lines were tracked even when crossing seams (only ON events)

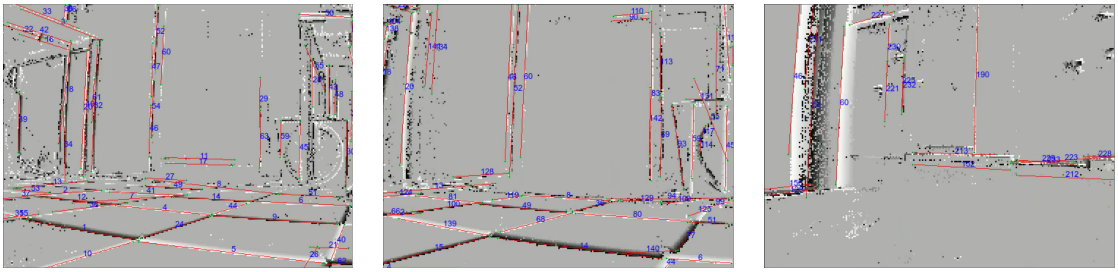


Figure 5.12: Snapshots from a sensor mounted on an RC car driving over even floor through a door (time increases from left to right)

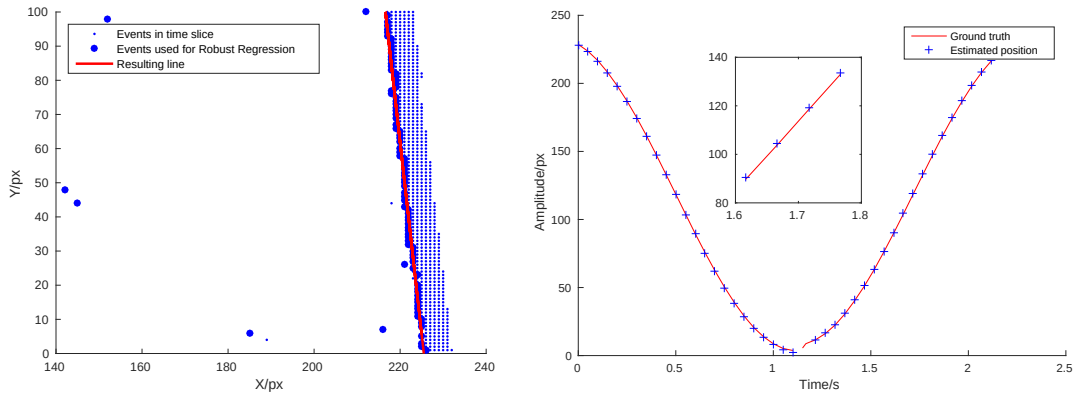


Figure 5.13: Left: Detail from method of ground truth estimation. Right: true line position (red line) and position estimates (blue crosses) at time of availability. Inlay zooms to region between 1.6 and 1.8 seconds. Position estimate overestimates true position by a small margin.

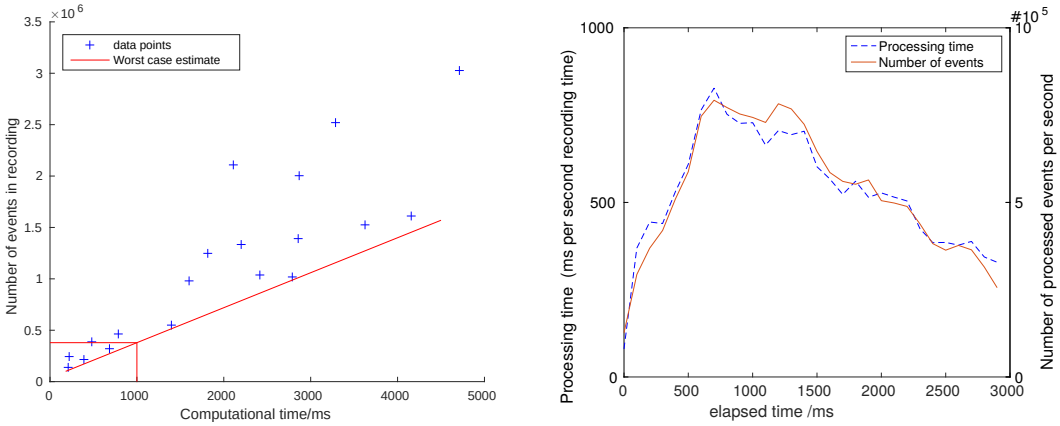


Figure 5.14: Left: Dependence between computing time and number of events for a number of different recordings. Right: processed events per second and required computing time for line tracking in the staircase scene

5.4 Line Matching

Now, that we have developed a feature extraction scheme for event streams, we will put it into use to solve the stereo matching problem. In the following, first, the basic idea of construction a matching graph for line matching across views and our adaptation for event-based visions are explained. Then, we assess the accuracy and required computing time quantitatively and compare it with the values of other algorithms. Finally disparity maps obtained with our method are shown to give an impression of the methods quality.

5.4.1 Match graph

5.4.1.1 Idea

To find correspondences between the lines detected in the two event-streams we advance an idea developed by Ayache and Faverjon[14] to the continuous domain. The original method was developed for line segments in a pair of static images. The basic idea of our algorithm follows the original one, which we will summarize the here in addition to describing where we developed it further for event-based vision.

The way of finding correspondences between lines is based on constructing an undirected matching graph $G = (V, E)$. A vertex V represents a pair of matched lines, whereas edges E link vertices with consistent disparity. The underlying assumption is that most real-world objects consist of a smooth surface and their disparity varies continuously. There are, of course, disparity discontinuities, but we assume that within a subset of the detected lines disparity varies smoothly. G is constructed by exploring potential matches for every line and linking vertices that have a similar disparity. After the graph construction, G consists of several connected components each representing a set of lines that form a smooth surface patch. The matches in disjunct connected components are potentially contradictory, so as last step in the matching process these contradiction are resolved, which concludes the graph construction.

The construction of the graph utilizes only information that is available at a certain time. Because of the continuous influx of events, there will be newly detected as well as disappearing lines in the event streams at unpredictable points in time. The time development is not accounted for when the graph is constructed; therefore, a method to continuously modify a constructed graph is necessary. Lines that are detected are integrated via a voting scheme to ensure they fit in consistently, lines that disappear will trigger vertices of which they are part to be removed from G. In addition, a complete graph will be newly constructed periodically which then overwrites the current matching graph, comparable to key frames in video compression, to correct potential errors having occurred during integration and assure global consistency.

5.4.1.2 Graph construction

G is constructed in a three step process: first, a number of potential correspondences is acquired by randomly selecting lines in the right stream and checking against the lines in the left stream. These tentative matches are used as seeds to grow the graph, so they are required to be of high quality. Strong constraints are imposed to keep the number of wrong matches low. To declare two lines as matching, they need to fulfill the following purely local criteria:

- the right line midpoint must have a homologous point on the other line

- the line geometry must be very similar (angle difference $< 10°$ and length ratio < 1.5 were used for our experiments)
- event polarity of line must be identical
- there can only be one match that fulfills above criteria. If there are more, discard all

In contrast to frame-based methods, we can not use brightness information along or next to lines. The only information related to illumination is the polarity of line events which is incorporated in the method as a matching constraint. Every retrieved match is initialized as a vertex without edges; the vertices are used in the next step to grow the graph.
In the next step, called propagation, we expand G by recursively searching for new matches in the vicinity of already paired lines. We look for lines that belong to the same smooth surface areas, so we require the depth difference between two neighboring lines to be relatively small at the position where they are close to each other. In contrast to the version for static images, we skip building a neighborhood graph. The arrangement of lines constantly changes due the dynamic nature of event-based vision. To establish a neighborhood relation the event streams are separated into a grid of rectangular cells. Lines are members of every cell through which they pass (see Figure. 5.15). Every time a line position is updated, the state of the grid is updated as well. Then the following neighborhood relation is used: lines are considered neighbors, if at least one of the cells they pass through is identical.

To explain how the graph is expanded let us denote the parent vertex that searches new matches with v_{parent}, consisting of a left line p_l, and a right line p_r which form a match. The subscripts indicate in which streams the lines lie (left or right). The graph expansion is started by gathering the set of neighboring lines N_l of the p_l. The lines in N_l are sorted in ascending order of their distance d, where distance is the minimal distance between any two points on p_l and a neighboring line n_l. N_l is pushed onto a stack which contains all lines to be processed. Lines which intersect ($d = 0$) or are very close are processed first. For the first line n_l retrieved we form a set C_r of potentially corresponding candidate lines in the right stream. All lines meeting the following requirements are added to the set:

- the midpoint of n_l must have a homologous point on the candidate line c_r
- the angles of n_l and c_r must be similar (in this phase we chose angle difference $< 17.5°$)
- event polarity must be identical
- disparity must be compatible (see below)
- if c_r has a prior match m_l, n_l and m_l can not overlap

Note that we do not have a length requirement since lines might be broken into segments. Therefore, we also relax the uniqueness constraint. If we encounter a line c_r as potential match that has a prior match m_l we check if this prior match overlaps with n_l. If they do not overlap and all other criteria are met, we add c_r to C_r. (see Fig. 5.15, left stream, leftmost lines for an example). The constraints are chosen less tight compared to the first phase, however, the requirement of compatible disparity is introduced. Although depth may vary widely over the course of lines, it has to be the same at the position where two

lines intersect (or almost the same where they are closest in case of non intersecting lines), if p_l and n_l belong to the same smooth surface.

Because disparity is inversely related to distance, the limits for compatibility are disparity dependent. To determine whether n_l and c_r have compatible disparity, we first determine the point of the intersection i of p_l with n_l (which need not necessarily be element of p_l or n_l). If c_r is the correct match for n_l, then p_r and c_r intersect (or are close) at the y value of i_y and both pairs of lines have similar distance to the sensors at this point.

The distance from the camera D_v of the vertex lines p_l and p_x using the focal length f, the baseline distance b, pixel width w, line disparity $disp$ at height i_y and the stereo geometry is given by:

$$D_v = \frac{f \cdot b}{w \cdot disp} \tag{5.7}$$

The distance D_c of the candidate pair n_l and c_r at height i_y is computed in the same way. If the absolute difference $|D_v - D_c|$ is smaller than a threshold, the lines lie close to each other in the real world and the matches are considered compatible. If C_r contains more than one element, the one where $|D_v - D_c|$ is smallest is chosen. A new vertex v_{child} is initialized and linked to v_{parent}. The procedure is repeated with the winning line c_r from v_{child}. All neighbors are collected and pushed on the stack (i.e. the closest neighbor of c_r is processed next). If we encounter a line l_{st} on the stack which already has a corresponding line (and is therefore part of a vertex v_{stack}), it is checked if their disparity is consistent with the last vertex that put it on the stack v_{calling}. If yes, we set an edge between v_{stack} and v_{calling}. This generates a high level of connectedness, see e.g. Fig. 5.16.

After all seed vertices have been expanded, G contains multiple connected components which represent different assumptions about the structure of the real world. The components potentially contradict each other, i.e. they contain vertices where the left line is identical, but the right line is not (or vice versa). These conflicts have to be resolved using an assumptions about the world. Since we expect that the world is largely continuous, the connected components of G containing the highest number vertices is assumed to contain the correct line correspondences. All vertices in other connected components containing a line that is also part of a vertex in the largest connected component is deleted. Then connected component size is evaluated and the procedure is repeated with the next largest connected component until no further connected components exist. The remaining vertices contain the matching result. This concludes the initialization of the graph based on the version for static images by Ayache and Faverjon.

5.4.1.3 Dynamic graph changes

Event-based vision data is, however, dynamic in nature and we do not have static images. There will constantly be lines appearing and disappearing which need to be integrated into or removed from the matching graph. Therefore, we need to dynamically adapt G to new information. We propose a voting-based approach towards achieving integration. As soon as a new line l_{new} is detected, we add it to the grid of cells to establish the neighborhood relation with the rest of the lines (this is exemplified in Fig. 5.17 where a new line with ID 16 appeared in center of the left stream). Then, l_{new} will perform a search for corresponding lines in the other event stream according to the rules described above except the rule of compatible disparity resulting in a set of candidate lines C. Each neighbor of l_{new} votes which line in C has to most compatible disparity (as described above). The matching combination that gets the most votes will be assumed to be the

correct one. A vertex v_{new} is initialized and all vertices containing lines that voted for the winning line will be linked to v_{new}. Removing lines is an easier process. If a line disappears, we remove all vertices containing the line from the graph.

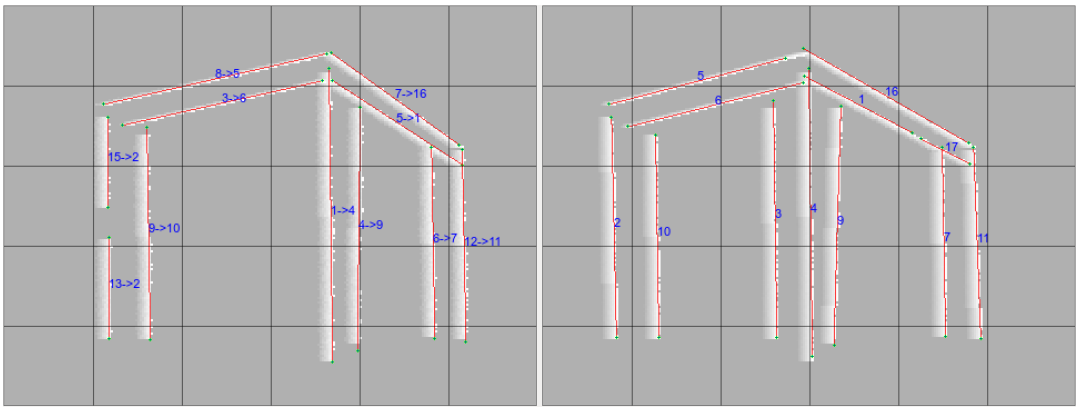


Figure 5.15: Example of matched lines between two event streams (events were artificially created for clarity of scene). Detected lines are indicated with red, cell boundaries with black, line IDs with blue. The lines of the left stream were additionally labeled with their corresponding partner in the right stream.

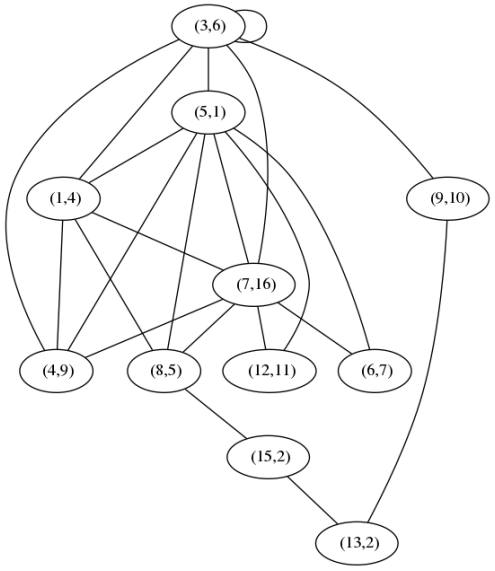


Figure 5.16: Largest connected component of the matching graph belonging to the images in Fig. 5.15. Vertices correspond to identified matches, edges are disparities that are consistent. Note that vertices of diagonal lines in the picture are strongly connected because they have consistent neighbor matches. The line with ID 3 in the right stream does not appear in the graph since it did not find a match.

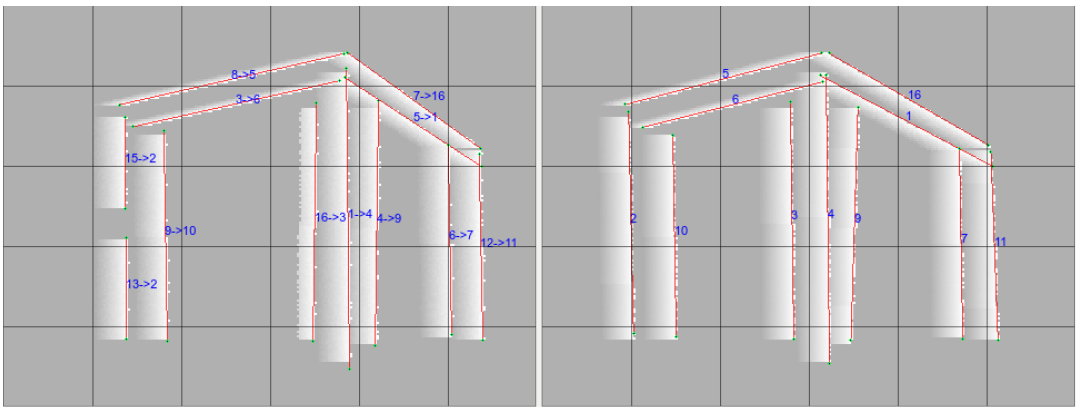


Figure 5.17: The streams depicted in Fig. 5.15 after we inserted a new line in the left stream.

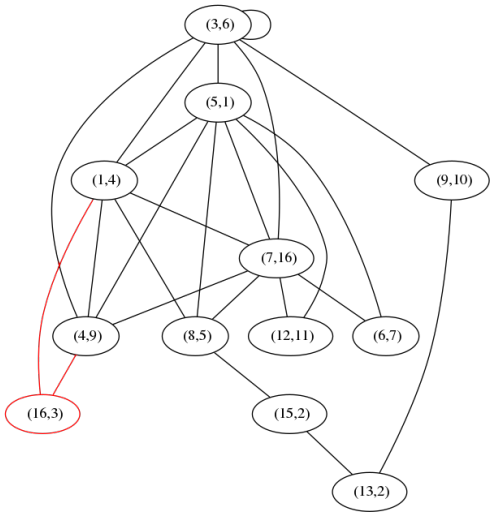


Figure 5.18: Graph from Fig. 5.16 after the new line was integrated. The new vertex and edges are marked in red. Edges were set between vertices who voted for the winning match (here (16,3)).

After having found correspondences between lines, the last step consists in computing the disparity of the events. This can be done in a very fast manner. The line detector provides the index of the line the event belongs to, the line matcher the index of its match in the other stream. The computation of disparity Δ reduces to a subtraction of the lines' x values belonging to the respective events y value.

$$\Delta = \frac{y - t_r}{m_r} - \frac{y - t_l}{m_l} \quad , \tag{5.8}$$

with slope m and intercept t of right and left line. Note, that this approach does not require a one-to-one correspondence on event level. We can assign disparity to events, regardless of the existence of a specifically matching event.

5.5 Results

We conducted experiments in different settings to show the effectiveness of our method. The vision sensor that was used was the DAVIS240C[37] and the computations were performed using an Intel Core i7 4770K running at 3.5 GHz. The algorithm was implemented in a not optimized fashion in C++. For the stereo rig a short camera base line distance of 10 cm was used in all experiments, the focal length was 4.5 mm.

5.5.1 Experiments

The algorithm was evaluated on the data presented in Section 3.5. Figure 5.19 shows snapshots from the matching process. Table 5.1 presents the numeric results. Different methods were compared in terms of accuracy and required computation time. We implemented the event-based message passing algorithm in C++ and ran all experiments on a single core to make time measurements comparable. The columns in the table show the following values: estimation accuracy (Est. acc.) measures the ratio of events, whose disparity estimate lies within 1 px of the ground truth values, to all estimated events. Computation time (C. time) states the required time to process one event in microseconds. Estimation rate is the ratio of events that were labeled with an disparity estimate to all events. The last column gives the ratio of required computing time to length of the recording. To be used in real-time the fraction has to be less than 1.

Although lines are not the optimal descriptor for scenes containing only persons the overall matching accuracy of parts approximated by lines is high. A notable amount of events could not assigned to lines and did not receive a disparity value, but the general coverage of the scenes is quite high as can be seen in Figure 5.19. For the more complex scene containing a table, boxes and cones the line-based matching had much better results than other methods.

In terms of timing the line-based matching approach was superior to the other approaches. It was the only one that achieved real-time for all recorded scenes, even the complex cluttered scene containing a table and cones. In comparison, the window-based method could achieve real time for simple recordings, but needed more than three times the recordings duration to process all events of the most complex recording. The event message passing algorithm's computing time was at least a factor 5 (for a simple recording with low event rate) to a factor of over 200 (for a complex recording with high event rate) higher than the recordings' duration, showing that it can in the current form not be used in a real-time scenario.

To get an estimate of the introduced latency, consider that the detection method used has a very low latency of only a few microseconds per event[69]. After construction of the matching graph, the corresponding lines are held in memory and there is no additionally latency introduced except for a lookup. The computation to obtain the disparity of the events is fairly easy (equation 5.8) and can reasonably assumed to be performable in at most few microseconds on modern hardware. It can be concluded, that the method is able to obtain disparity results with latency in the order of microseconds.

During analysis of the error, a potential problem with the labeling of the data became apparent. The calibration of ground truth labeling system and DVS as described by the authors of the publication may not have been entirely correct. Figure 5.20 shows the histogram of disparity errors obtained with the line-based matching method for one data file (the other data files look similar). The errors follow a Gaussian distribution, however, the mean is not at 0 as one would expect, but at 0.5 px. Since the line detection algorithm operates independently on both streams and disparity is found by subtraction, a systematic error in the feature detection stage would cancel. We, therefore, presume the stated ground truth values are shifted slightly from the real ground truth values. Applying the shift during the evaluation of our algorithm improves the results considerably. Table 5.2 compares the results.

In addition to the disparity labeled scenes for quantitative evaluation, we tested our method in other complex environments: a checkerboard floor and an office scene which

Data set	Method	Est. acc %	C. time $\frac{us}{\text{event}}$	$\frac{\text{c. time}}{\text{rec. time}}$	Est. rate %
One Box	LBS	86.09	1.27	0.026	44.57
	WBS	83.26	13.7	0.28	84.45
	EMP	77.15	298	6.2	82.16
One person	LBS	76.00	1.22	0.16	41.80
	WBS	87.88	15.1	2.04	79.26
	EMP	92.00	384	52	94.55
Two Boxes	LBS	51.56	1.25	0.028	41.80
	WBS	73.97	11.1	0.25	85.97
	EMP	82.21	251	5.6	73.64
Two persons	LBS	59.27	1.81	0.37 45.68	
	WBS	72.38	16.2	3.36	85.85
	EMP	70.64	365	75	92.71
One person different depth	LBS	54.54	1.49	0.047	13.38
	WBS	53.45	21.8	0.69	66.76
	EMP	61.14	392	12	58.36
Table and Cones	LBS	67.08	1.63	0.93	13.27
	WBS	57.53	16.59	9.5	88.11
	EMP	29.05	370	212	88.58

Table 5.1: Comparison of the line-based matching (LBS) results with the window-based method (WBS) presented in Section 4.2 with window size 11 and event-based message passing (EMP,[43]).

Data set	Unshifted	Shifted
One Box	86.09	97.15
One person	76.00	75.73
Two Boxes	51.56	68.27
Two persons	59.27	64.05
One person different depth	54.54	57.22

Table 5.2: Comparison of the line-based matching (LBS) results with results when shifting the ground truth disparities of the dataset from [43] by 0.4 px.

was recorded with a radio-controlled car. Due to lacking ground truth values no numeric evaluation is available for these scenes, but the resulting disparity maps look smooth, consistent and plausible (Figure 5.21).

The first unlabeled complex scene contains a checkerboard pattern (Figure 5.21 c and d). If matching is done based on event time correlation, ambiguities arising due to the repetitive structure have to be solved. This is a non-trivial task and can lead to inconsistent disparity maps. Our method is able to solve these ambiguities due to the graph enforcing global consistency and builds a smooth disparity map. Note that the visible lines have not been matched continuously, but were broken in segments of tile length due to the changing polarity along the checkerboard pattern.

For the other recording, we mounted two sensors onto a RC car and recorded an indoor scene to evaluate the efficacy of our method on a small moving system. Due to the low height of the car, the recording has a steep depth increase from the bottom of the recording upwards. Using our method we could, however, match lines in foreground and background and build a sparse depth map.

5.6 Summary and Discussion

This chapter explored event-based features and their usability in solving the stereo matching problem. First, event-based corner were investigated, specifically two classic corner detectors, Harris detector and FAST detector, that were recently adapted for event-based vision [53, 54], but the conclusion was reached that corners are a suboptimal choice, mainly because they cover only small parts of typical scenes and the extraction is not reliable.

We then proposed a novel method for line extraction from event streams. Lines are a ubiquitous structure and can also be used to approximate non linear edges which makes them a reasonable choice to geometrically describe large parts of event streams. The detector is based on the observation that lines leave planar traces in x-y-t-space. Planes can be identified via dimensionality reduction which is done here via PCA. We showed that the proposed line detection method is accurate, has a latency of only few microseconds and can process at least 400,000 events per second in the worst cases on a standard desktop computer. In addition to line detection, our method can also robustly track lines over an extended period of time which was also demonstrated. Comparison with another event-based corner detector showed the superiority of our method in terms of accuracy and being equally accurate with a conventional method. Conventional methods, however, lack the benefit of having an accurate line position estimate at all times and being able to track the line without a separate processing step.

The developed line detector was used as basis for a feature-based stereo matching algorithm. The fundamental idea of building a match graph [14] was transferred to the continuous event-based domain. The graph ensures global consistency of the line matching and can be constructed in a fast way. Consistency is achieved by a size comparison between connected components, no costly graph algorithms are required. We evaluated the accuracy and computation load. For simple scenes the accuracy was moderately lower than the accuracy achieved with other methods, like window-based matching or event-based message passing. This is partly due to the fact that lines to not approximate the edges perfectly (Figure 5.19, partly by lines being matched wrongly because the actually corresponding edge was not linearly approximated in one stream and is thus not available for matching. The enforcing of global consistency plays only a minor role in the simple scenes with little ambiguity.
For a complex scene the accuracy of line-based matching was much higher than the other methods. This is explained, first, by the line detector's robustness to noise, that allows an accurate estimation of line positions; second, this is a result of the match graph ensuring global consistency. In a cluttered scene there is more ambiguity to resolve when matching single events because there are more edges and, hence, more potential matching partners on the epipolar line. The line-based matching accounts for consistency, the single event based method not. For this reason, the line-detector achieves higher accuracy in complex scenes.

Concerning the required computation time per event, line-based matching was one order of magnitude faster than window-based matching and two orders for event-based message passing. With less than 2 us per event the method proved to capable of estimating disparities in real time. The combination of being able to accurately estimate disparity in complex scenes and being useable in real-time makes line-based matching a promising candidate for deployment on autonomous systems.

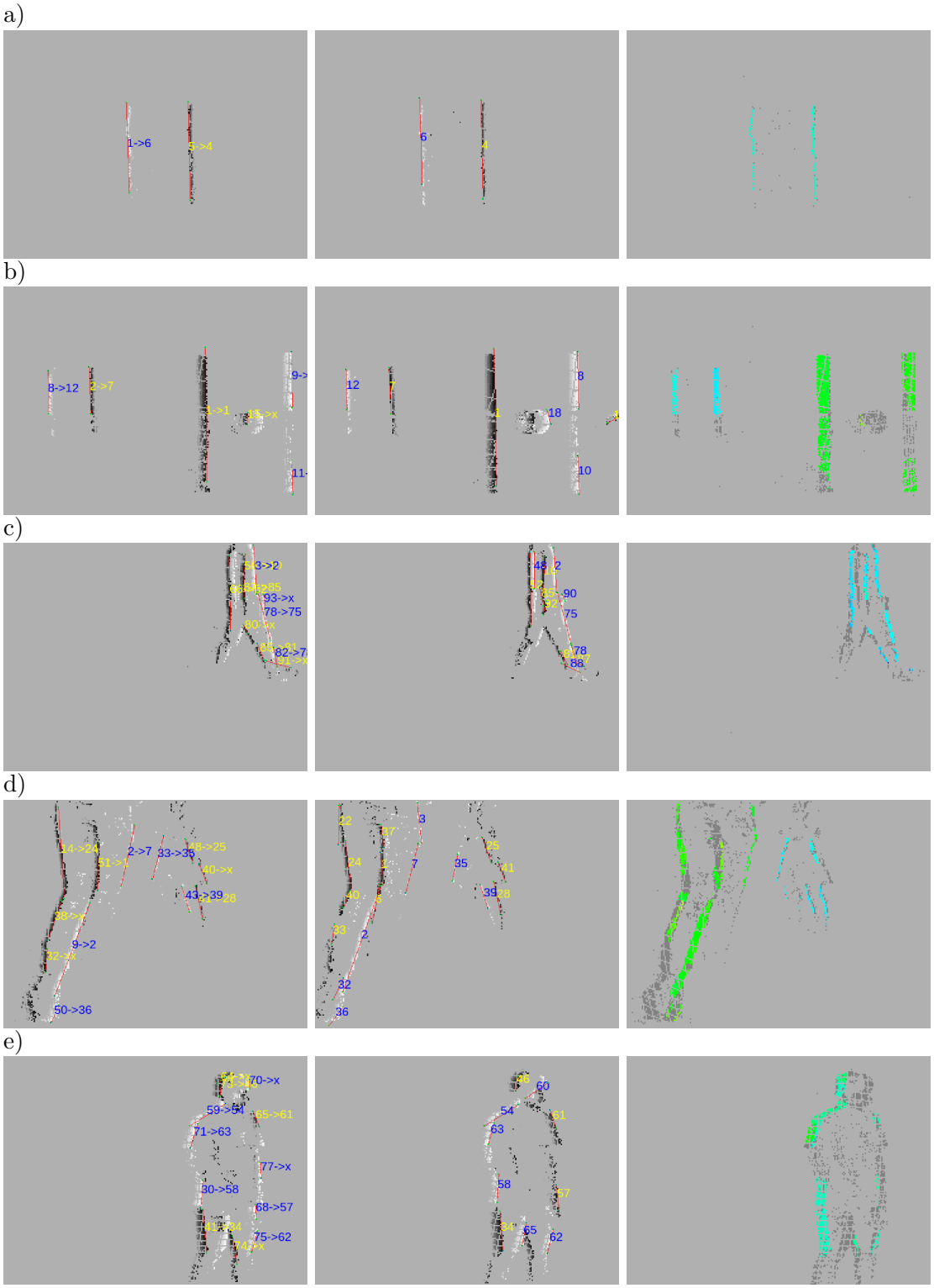


Figure 5.19: Data set provided by Xie et al. (2017) [43]. a) 'One Box', b) 'Two Boxes', c) 'One Person', d) 'Two Persons' e) 'One Person Different Depths'. First column shows snapshot from beginning of respective file, second column from end (event streams were accumulated for 100 ms, fainter color represents older event age).

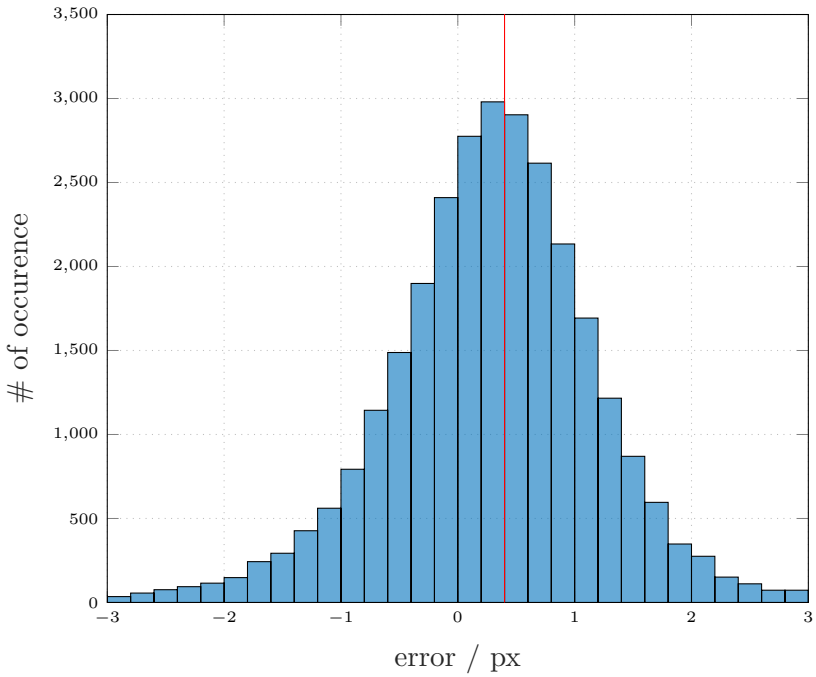
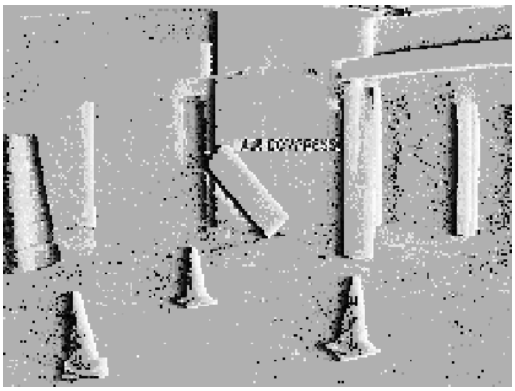
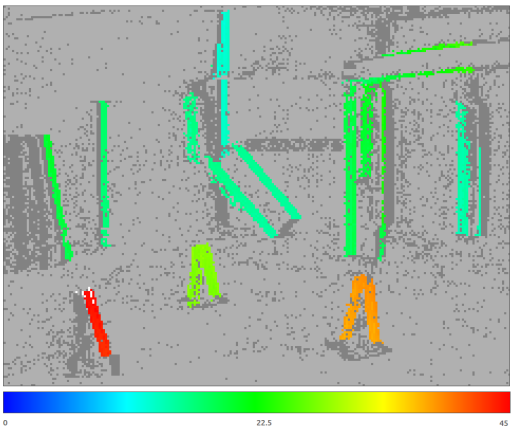


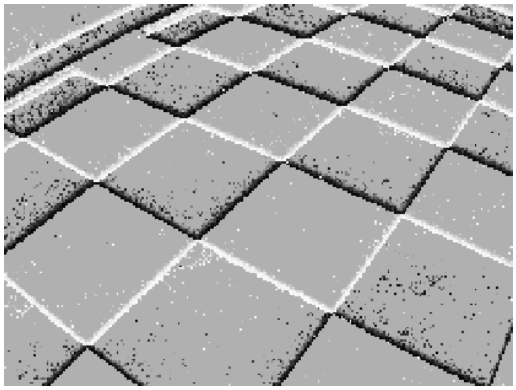
Figure 5.20: Distribution of errors for the 'Two persons' dataset



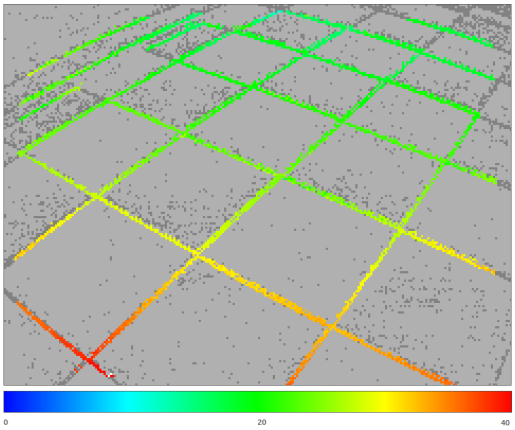
(a) Table, boxes and cones in a noisy recording



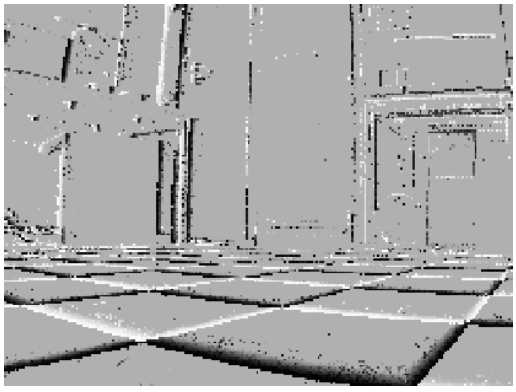
(b) Corresponding disparity map



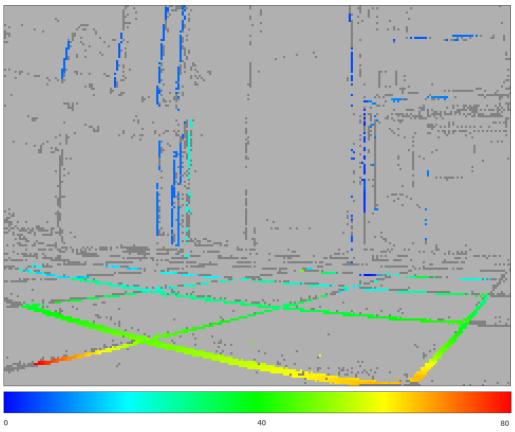
(c) Checkerboard floor



(d) Corresponding disparity map



(e) Office scene recorded with RCcar



(f) Corresponding disparity map

Figure 5.21: (a,c,e) Snapshots of the left event stream for different scenes. ON events are depicted white, OFF events black. (b,d,f) Resulting disparity maps of our method, disparity is given in pixel. Grey events have not been matched. Note the different scales!

6 Societal impact

6.1 A wearable mobility aid

Potential applications of DVS not only lie in the field of robotics; here, we propose a wearable mobility device which is designed to assist blind and visually impaired people with navigation and orientation. We employ two DVS sensors to extract depth information with low latency. The information is translated into virtual 3D sound with the help of individualized head-related transfer functions. This a easy to interpret, very intuitive way of delivering spatial information to the user, especially compared to other Electronic travel aid (ETA)s which use complex encoding like the vOICe [72]. Other ETAs deliver the depth information via different sensor modalities like electric pulses on the tongue [73]. Figure 6.1 shows an overview over the information processing flow. The system discussed here was initially proposed in [74] and then extended in [75]. Large parts of the following chapter are based on the latter.

The system is designed to be wearable head gear. An early prototype is depicted in Figure 6.2). It consists of two DVS with fixed geometry, a resolution of 128×128 pixels each and lenses with focal length of 6mm. Their synchronized vision stream is transmitted to a computing stick on which the depth extraction is performed and events are translated into virtual spatial sounds. The sound signal is sent to the users ear via headphones connected to a USB sound adapter.

The two DVS sensors are equipped with an additional microcontroller and require <=0.3 W each (80 mA at 3.3 V). The processing is currently done on an off-the-shelf compute stick which consumes approximately 10 W (2 A at 5 V). We are working on porting the processing part onto a microcontroller which operates at <= 0.5 W (150 mA at 3 V) and would allow the whole system to operate at ≈1 W.

With a state-of-the-art cell phone battery we aim to achieve battery lifetimes of around 10h when the processing is done on a microcontroller.

In a normal use case the DVS produce on the order of 100 events per millisecond, far more signals than the human auditory system can process [76]. We, therefore, downsample the number of events that will be transmitted to the user by a factor of 1000 to roughly 100 per second. In contrast to the original version, we do not apply clustering strategies as clustering did not prove to increase performance measurably. The distance to the object that generated the event is then extracted using the stereo information of the two DVS stream.

As experiments conducted with an earlier version of this device have shown poor performance of elevation estimation using virtual spatial sounds we developed visual processing as well as sound generation: the user is now provided with a vertical reference point within their field of view. The field of view has been divided into three horizontal stripes of different width. Events are grouped with respect to their specific section. We call the middle stripe focus area. Inspired by the fovea centralis, where human vision is sharpest, we choose to convey a larger amount of information from this area to the user and pick 60% of the sonified events from the focus area. The focus height was chosen to be 4 pixels, corresponding to about 1.5n the field of view; this matches approximately the

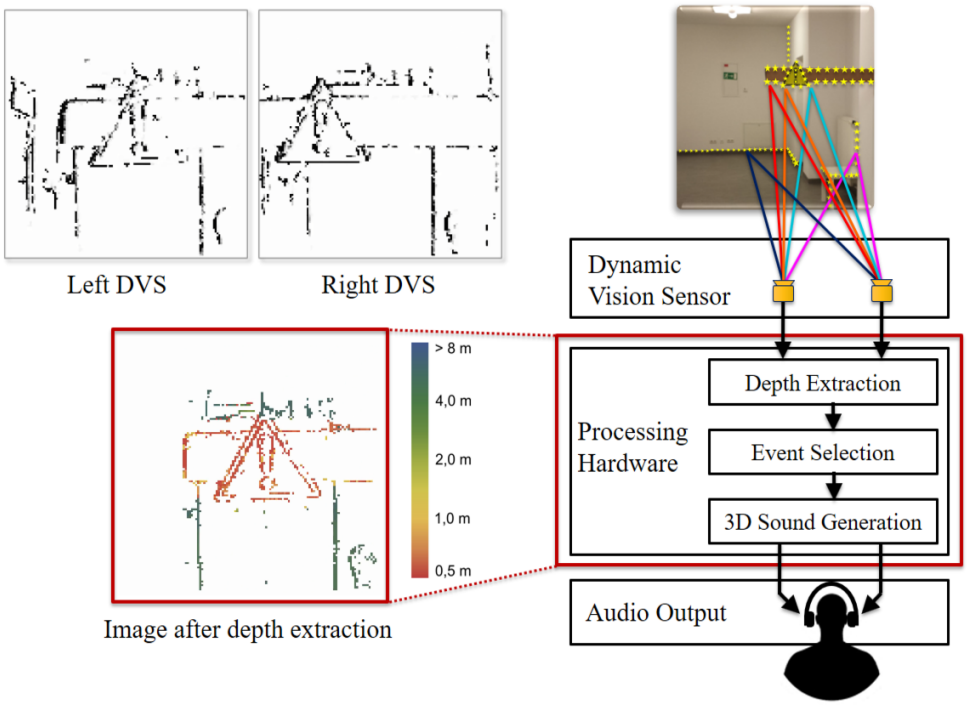


Figure 6.1: Left: DVS events accumulated over 50 ms to form a frame for visualization. Visible is a vertical bar with a warning. Colors correspond to distances of events. White areas have not undergone a change in brightness during the accumulation period. Right: Overview over the proposed system and information flow. (Figure from [75])

minimum audible angle for horizontally separated sounds, i.e. the minimum angle two sound sources must be apart to be perceivable as separate, so that vertical and horizontal resolution become similar. Events from the focal area are furthermore associated with a sound that differs from the sound associated to events from off-focus areas to make them easily distinguishable.

As soon as the position of an event in space is known and it has been selected for sonification it will be translated into a virtual spatial sound, i.e. into a sound whose source appears to lie at the position of the event. This encoding ensures an intuitive presentation of the information to avoid the necessity of long training times before being able to use the device.

Generating Head-related Transfer Function (HRTF)s that are perceived as coming from an external source in space is a complex task. It requires the modulation of sounds via a HRTF that describes how sounds are changed by the perceivers' body, head and pinna before they reach the ear canal. As HRTFs depend on the hearer's anatomy, it is a highly individual function which requires intricate measurements.

To address the need of individual HRTFs for sound elevation resolution, a method for easier personalization of HRTFs from HRTF databases using body measurements of test subjects was developed [77]. It is a trade-off between accuracy and feasibility and achieved a significant improvement in user performance when estimating elevation of sound source.

Two types of clearly distinguishable sounds are implemented to encode events. A 440 Hz tone for events originating in the focus area and a click sound for events from the off-focal area. They are passed through the head-related transfer function which transforms them into binaural sounds that appear to come from the location where the event has been generated when played to the user's ear using stereo headphones.

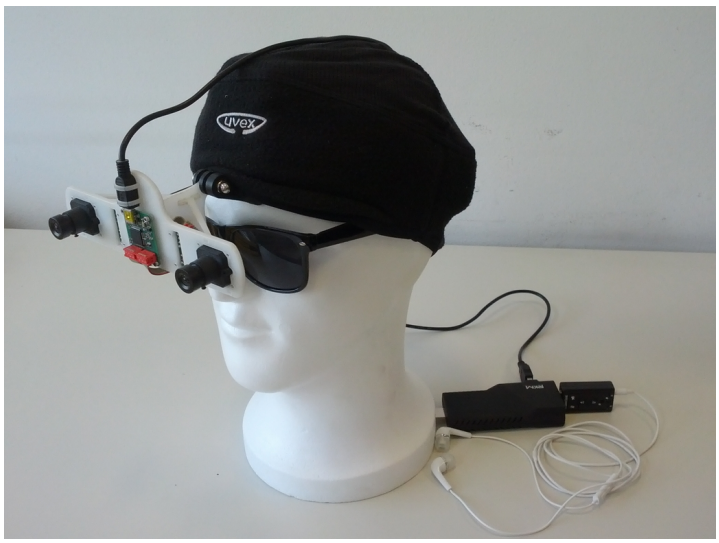


Figure 6.2: A device prototype mounted on a Styrofoam head.

The total latency of the system is approximately 50ms which is mostly due to the way sounds are aggregated before they are replayed. We aim to achieve a latency of 10ms by implementing more sophisticated sound replay algorithms.

6.2 Tests

In order to test the new processing algorithm, two types of experiments were carried out. This section was published prior in [75]. The first is based on previous tests [74] to obtain directly comparable results and consists of object detection, size discrimination, and object localization. The second experiment is a common test for visual acuity using the Landolt C based on free online resources[78]. The visual and acoustic processing during the experiments was done using a laptop which provided better controllability and visualization.

The first test was carried out by 11 subjects (9 male, 2 female, average age $\mu = 28 \pm 5$ years). They were seated in 60 cm distance from a white wall on which circles (black on white paper) of different sizes were fixated. We modified the test in [74] to not use screens to display the circles because many artifact events were generated due to the refresh rate of standard monitors. The subjects wore the device positioned in a way that its field of view was centered in the middle of the test area when they looked straight ahead. For each category (object detection (OD), size discrimination (SD), object localization (OL)) twenty trials were conducted where the truth was randomly, equally distributed. Subjects were given assistance to realign the field of view with the middle of the test area after each trial if required.

The second experiment to test the efficacy of the introduced focal are for vertical resolution consisted of the Landolt-C Test where the task is to discriminate between two possible orientations of an optotype (i.e. a standardized letter for acuity tests), here the letter 'C' (oriented with opening facing right or left). The vertical visual acuity (vVA) is determined by the size of the letter for which a certain percentage of correctly identified orientations is exceeded. The test group consisted of 5 subjects (3 male, 2 female, average age $\mu = 27 \pm 5$ years). Subjects were again seated in front of a white wall on which a

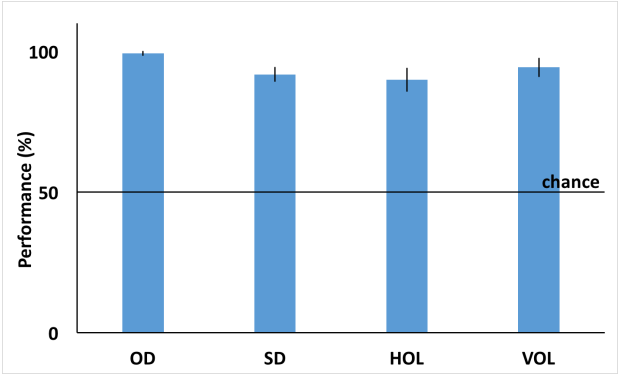


Figure 6.3: Average performance for object detection (OD), size discrimination (SD), horizontal object localization (HOL), vertical object localization (VOL).

printed optotype 'C' was fixated. Due to the fixed absolute size of the printed optotype the distance to the wall was varied to change the relative size. On average the test lasted 30 minutes ($\sigma = 10$ minutes).

6.3 Results

For the first test, we split evaluation of object localization in horizontal position (left, right) and vertical position (up, down) to assess the effects of the new information processing and conveying strategies.

The change in performance for object detection, size discrimination and horizontal object localization is only insignificant compared to the previous version as it was already at a very high level (object detection 99%±1%, size discrimination 96%±5.3%, horizontal localization 90%±8.5%, Fig. 6.3). An interesting observation is that subjects required more time for the size discrimination than before. The reason behind this is not clear yet and needs to be further investigated (Fig. 6.4).
The introduction of a focal area in the system alongside the personalization of the HRTF has, however, helped to overcome the inability to resolve the elevation of the artificial sounds and raised the performance from about chance level to a success rate of approximately 95% (Fig.6.3). In contrast to the prior design subjects explore their environment by moving the field of view by moving their heads, while in prior tests they were asked to keep their heads still. The Landolt-C test for vertical acuity yielded an average visual acuity of 2.23 ± 0.14 on the logMAR scale. The logMAR (Logarithm of the Minimum Angle of Resolution) scale is calculated by the base-10-logarithm of the gap size of an optotype (here the letter 'C') a subject is able to recognize. Lower values correspond to higher visual accuracy. Results are shown in Table 6.1 along with acuity values of other ETAs. Visual acuity is given as fractions: the denominator denotes the maximal distance in feet from which a feature can be recognized with a visual aid while the enumerator denotes the distance in feet from which a normal person can still see the same feature (e.g. $\frac{20}{100}$ means that a visually impaired person can still resolve a feature from 20 feet which a normal sighted person can resolve from 100 feet).

Note: the test setups were not standardized, so comparability of the results is limited and serves only as rough estimate. The vOICe was tested with two different groups of subjects: the first test int Table 6.1 (I) was performed by sighted subjects and consisted of a Snellen-E test [76], the second test (II) by blind users with 50-100h training performing a

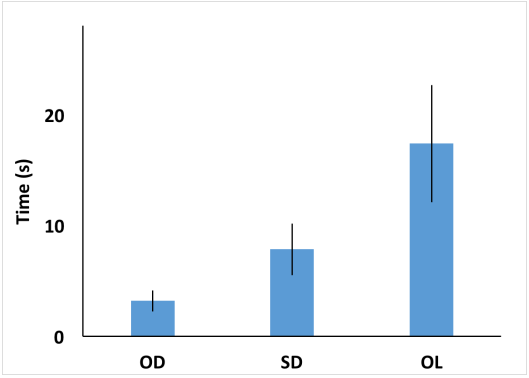


Figure 6.4: Average time per test trial for object detection (OD), size discrimination (SD), object localization (OL, vertical and horizontal was done in one trial). Longer duration correlates with higher difficulty of task

ETA	Acuity	logMAR Value
This device	$\frac{20}{4000} - \frac{20}{5150}$ for vVA	$2.09 - 2.37$ vVA
The vOICe (I)	$\frac{20}{737} - \frac{20}{13965}$	$1.56 - 2.8$
The vOICe (II)	$\frac{20}{200} - \frac{20}{600}$	$1 - 1.47$
Phosphene Prosthesis	$\frac{20}{560} - \frac{20}{1260}$	$1.45 - 1.80$
BrainPort	$\frac{20}{4375} - \frac{20}{5500}$	$2.34 - 2.44$

Vertical acuity for our device; the other devices were tested without discriminating directions. Furthermore, the results were not gained using an identical standardized test, so they can not be directly be compared.
For more details see text.

Table 6.1: Acuity comparison between different devices

Snellen-E test[79], the phosphene prothesis used a Landolt-C test with 8 orientations[80], BrainPort was tested with untrained, mostly visually impaired users[81].

7 Conclusion & Outlook

This thesis has presented different approaches towards depth reconstruction using event-based vision data. Similar to the stereo setup based on conventional cameras, the major challenge is the stereo matching problem, i.e. the identification of corresponding points in two different projections of the same scene. Having established point correspondences, finding the depth is merely a matter of triangulation, hence finding the correspondences is virtually equivalent to depth reconstruction. Solving the correspondence problem in the event-based domain requires different approaches compared to frame-based methods. Event-based sensors convey visual information as sparse, quasi-continuous stream of events and each event contains only little information, namely where and exactly when brightness has changed. The sparse data makes correct matching and scene reconstruction a difficult and complex problem. On the other hand, the sensors are rewarding because they open up the possibility for light-weight algorithms that have the potential to be run in real-time. The high speed, low latency mode of operation of the DVS led to the selection of performance metrics for the presented methods: besides matching accuracy, the focus lay on event processing rate and latency.

During the course of this work different classes of algorithms were developed. The first algorithm was based on correlating matching windows. The asynchronous nature of the data suggested using event timestamps as cue for matching. We evaluated three different cost aggregation functions: sum of squared differences, sum of absolute differences and normalized cross-correlation and showed that sum of absolute differences gives the overall best results.

We compared the proposed method with other published algorithms using the ground truth data and evaluation metrics which were proposed by the authors and are available in the literature [43]. The window-based method yields good results compared other methods [40, 42, 45]. For large window-size the accuracy is comparable to the best available method from literature, event-based message passing [43], on simple data set. On a complex data set the window-based matching outperforms event-based message passing achieving an accuracy of 57.5% compared to 29.1%. Processing speed does not achieve real-time operation for large windows; for a smaller window-size, the algorithm runs in real-time on a standard computer, but this comes at the cost of reduced accuracy. The algorithm allows a trade-off between accuracy and speed, depending on the requirement of the application. The speed stands in stark contrast to the speed of event-based message passing, which is orders of magnitude slower than real-time, making the window-based matching method overall a better choice. The window-based stereo matching has a couple of disadvantages. First, every event is matched independently, there is no notion of global consistency. This leads to depth maps that are not smooth. We observed that mismatches often form clusters of consistent, but wrong disparity leading to the perception of objects at positions where there are none. To improve the accuracy, we reformulated an edge-based refinement scheme, originally designed to increase smoothness and consistency for conventional cameras [62], for event-based data. The results, however, showed no improvement but were worse than without refinement. We investigated the reasons and found that the proposed refinement approach is not compatible with using event-time stamps instead of intensities. Other methods for postprocessing, e.g. on belief propaga-

tion [39, 43], deliver more promising results.
Global methods allow to find a consistent matching and in addition grant access to floating point accuracy. Concurrent matching of multiple events employing point cloud registration techniques was explored. To that end the Iterative Closest Point algorithm was adapted for the use in stereo matching by incorporating the geometric constraints of a stereo setup in the equations. We derived the new optimization equations for the adapted version of ICP. Evaluation showed that the accuracy of the method fell short of expectations and achieved lower accuracy than a simple overlay of the event clouds' centers of gravity (Table 4.4).

The second registration algorithm analyzed was Coherent Point Drift. CPD uses a probabilistic framework and is based on density estimation and expectation maximization. It offers more flexibility than ICP by not being constraint to affine transformations and allows non-rigid transformations. We showed qualitatively that this allows matching of scenes with objects at different depth. Quantitative evaluation over a broad range of parameters revealed that average accuracy is not satisfying and much less than of the window-based stereo method (Figure 4.16). The complex calculations during the iterative expectation maximization require a lot of computing time and render the registration method unfit for real-time tasks. In addition to the relatively poor quantitative results, event-cloud registration has a conceptual disadvantage. Both event clouds have to be known fully before the registration starts, which means events have to be buffered before being processed similar to creating images from events. This loses the low latency, that event-based sensors offer.

Based on the quantitative findings and the conceptual considerations, we changed focus to the third class of algorithms: feature-based methods. There are only few publications concerning event-based features thus far. For generality, a feature that can be found in a wide range of scenes had to be chosen. First, event-based corner detectors [53, 54] were analyzed to assess their usability for stereo matching. We came to the conclusion that corners are not a reasonable base for developing a feature-based stereo algorithm for three reasons: first, the corner detectors do not describe corners as geometric objects, but as event associated labels. They operate rather like a filter. An additional cluster process would be required to identify physical corners, describe them geometrically and track them through time. Second, corners only cover a very small area of typical scenes and they can not be used to approximate extended shapes. Lastly, the currently available corner detectors are prone to both not finding corners, even when they are quite prominent, and to reporting false positives, i.e. they label events as corners, even though they do not lie at actual corners. A feature that covers larger parts of scenes and can robustly be detected should be chosen.

We developed a novel way of detecting and tracking lines in event streams based on the observation that many scenes contain edges which are straight or can be approximated with multiple lines [69]. The evaluation showed that detection and tracking is subpixel accurate. We showed this in the context of tracking lines where we achieved an accuracy of 0.5 px. The method leverages the asynchronous, low-latency mode of operation of the sensors by processing every event independently. This way we could reduce processing time per event down to 0.7 μs. Considering that the timestamp accuracy of events is limited due to electronic jitter and other noise sources to the order of tens of microseconds, the latency of our method is minute. An analysis for the worst case timing scenario showed that our method can process at least 400,000 events per second on a desktop computer making it suitable for usage in a real-time setup. Compared with the other available

event-based line detection method ELiSeD [51] our line estimates proved to be superior in terms of accuracy of angle (average error 0.6° versus 1.5°) and line length where we showed that most of our estimates are within 10% deviation from the ground truth value, whereas ELiSeD mostly matched less than 30% of the line lengths (implying it breaks them down in three or more segments). We additionally showed that our method can robustly track lines, using recordings from a radio-controlled car.

Having developed a way to reliably extract lines, we used them for stereo matching. To that end we advanced a light-weight, graph-based method for matching lines [14] from the frame-based domain to the continuous event-based domain making it suitable for use with continuous data. Lines have an easy symbolic description, but are rich in geometric information. Based on it we construct and continuously update a matching graph that ensures a globally consistent matching of lines. We showed that the line-based stereo matching performs similar to other event-based stereo matching methods. This holds true even in scenes that have no clear straight edges, like walking persons, where edges have to be approximated with multiple line segments. The scenes used for ground truth estimation contain few objects and are relatively simple, so that global consistency ensuring, for which the line-matching was designed, plays only a minor role. In order to better assess the capabilities of the line-based matching, we recorded a more complex scene with multiple objects and a moving sensor and implemented the most recent and thus far best performing stereo matching method we found in literature which is based on belief propagation. We could show that the line matching approach significantly outperformed both the belief propagation method as well as the window-based method in terms of accuracy, achieving 67.1% accuracy, compared to 57.5% for the window-based method and only 29.1% for the belief propagation.

Once lines are initialized and matched, deriving the disparity for an event is very light-weight and can be computed quickly and with low latency. In terms of computing speed, the line-matching algorithm outperforms the compared methods by at least one order of magnitude. It required approximately 1.5 μs per event, while the window-based matching took on the order of 15 us and the belief propagation method 300 us. Hence, the line matching method is the only method that fulfilled the real-time requirement even in complex scenes with high event rates. The evaluation was done on a single CPU core to ensure comparability. The way our algorithm is designed allows it to be easily parallelized. The extraction of lines in every stream in each event stream is independent of each other and can be done concurrently. The line matching can be executed in another parallel process, so there is potential for substantial acceleration. This could be required when event rates reach one million events per second and more, e.g. from next generation dynamic vision sensors with higher resolution than the DAVIS240C.

Conceptually, the feature-based matching approach has some advantages over single event-based and global matching methods. In contrast to the single event matching approaches [40, 42, 43] or the coincidence detector based neural networks [45, 46, 47], the feature based method is not dependent on one-to-one event matching, i.e. an event can be labeled with a disparity regardless of having found an actual matching event in the other stream. The matching is done on an abstracted level, in the feature space and the disparities are inferred by mapping back to the events from the features. Being liberated from the requirement of having a matching event makes the algorithm more robust to differences in sensor settings. Consider the ground truth data streams used in this thesis for instance: the event rates differ considerably and the event streams are very dissimilar at times (Figure 3.9). In contrast to global matching, every event can immediately be

processed, there is no need to collect a number of events before processing can start. This way very low latency can be achieved.

A drawback of the method compared to the semi-dense methods (i.e. methods, that try to match all events) is that not all areas of the stream can be matched, only the ones which can be described by features. As we have shown, lines possess a high descriptive power, and are able to linearly approximate edges of walking persons or to a certain degree even circular features, so they can describe large parts of many scenes, and are not restricted to scenes with many straight edges. Nevertheless, fewer events are labeled with disparity compared to semi-dense methods. One possibility to achieve a better coverage of the scene could be to combine the line-based matching with a secondary algorithm, which processes events that were not assigned to lines, like the window-based matching approach. Another possibility is to incorporate other primitives and shapes as features for matching as soon as new detectors for them become available.

8 List of publications

1. L. Everding and J. Conradt, "Low-latency line tracking using event-based dynamic vision sensors," *Frontiers in Neurorobotics*, vol. 12, p. 4, 2018. doi: 10.3389/fnbot.2018.00004

2. Z. Tayeb, J Fedjaev, N. Ghaboosi, C. Richter, L. Everding, X. Qu, Y. Wu, G. Cheng, J. Conradt " Validating deep neural networks for online decoding of motor imagery movements from EEG signals", submitted

3. C. Widderich, L. Everding, C. Richter, J. Conradt, "A Cooperative Stereo Matching Neural Network on TrueNorth", Bernstein Conference 2017, Göttingen, doi: 10.12751/nncn.bc2017.0101

4. L. Everding, L. Walger, V. S. Ghaderi, and J. Conradt, "A mobility device for the blind with improved vertical resolution using dynamic vision sensors," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–5, 2016. doi: 10.1109/HealthCom.2016.7749459

5. L. Everding, V. S. Ghaderi, M. Mulas, A. Guibourgé, B. Seeber, J. Conradt "A wearable mobility assistant to help the blind navigate autonomously", Bernstein Conference 2016, Berlin, doi: 10.12751/nncn.bc2016.0035

6. V. S. Ghaderi, M. Mulas, V. F. S. Pereira, L. Everding, D. Weikersdorfer, and J. Conradt, "A wearable mobility device for the blind using retina-inspired dynamic vision sensors," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 3371–3374, 2015. doi: 10.1109/EMBC.2015.7319115.

7. V. Stih, L. Everding, N. Waniek, J. Conradt, "Light-weight optic flow calculation on miniature computers using a V1-inspired algorithm", Bernstein Conference 2015, Heidelberg, doi: 10.12751/nncn.bc2015.0183

Acknowledgements

The success and final outcome of this thesis required a lot of assistance from many people and I am very thankful to have got this all along the way of my project.

First of all, I want to cordially thank my supervisor Prof. Dr. Jörg Conradt for the opportunity to work with him and his group and giving me the support and guidance but also the freedom I needed to explore new ideas. My time at Neuroscientific System Theory was an excellent learning opportunity and a truly great experience.

I am much obliged to Lennart Walger and Matthias Emde, who supported me throughout my time at NST with all kinds of things, from experimental setups to data recording and all-night programming and data analysis sessions. Without you, it would have been much harder to complete this thesis. Special thanks also to the people at Neuroscientific System Theory, who made my PhD an exciting and cheerful experience, gave me valuable and constructive suggestions and supported me when I needed it: Viviane Ghaderi, Christoph Richter, Nicolai Waniek, Michael Lutter, Zied Tayeb, Emeç Erçelik, Florian Mirus, Cristian Axenie, Marcello Mulas, Mohsen Firouzi and Susanne Schneider. I had a great time working together with you solving scientific problems, teaching lectures and classes, experimenting, having extensive lunchtime discussions and doing all the other things that life at NST entailed.

Furthermore, I want to express my sincere gratitude to Prof. Dr. Darius Burschka, who agreed to examine my thesis, and to Prof. Dr. Werner Hemmert for chairing the examination committee.

Finally, I wish to gratefully thank my mother, Christine Kalmbach, for always supporting me, no matter what, and doing everything possible to offer me the best prospects in life. I can't say thank you enough for your tremendous support!

Bibliography

[1] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[2] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, p. 1330–1334, December 2000.

[3] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, pp. 7–42, Apr. 2002.

[4] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 993–1008, Aug 2003.

[5] M. Drumheller and T. Poggio, "On parallel stereo," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1439–1448, Apr 1986.

[6] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 920–932, Sept. 1994.

[7] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[10] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, June 1986.

[11] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, pp. 11–15, Jan. 1972.

[12] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic Hough transform," *Comput. Vis. Image Underst.*, vol. 78, pp. 119–137, Apr. 2000.

[13] J. Shi and C. Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, Jun 1994.

[14] N. Ayache and B. Faverjon, "Efficient registration of stereo images by matching graph descriptions of edge segments," *International Journal of Computer Vision*, vol. 1, no. 2, pp. 107–131, 1987.

[15] Z. Wang, F. Wu, and Z. Hu, "Msld: A robust descriptor for line matching," *Pattern Recognition*, vol. 42, no. 5, pp. 941 – 953, 2009.

[16] M. Hofer, M. Maurer, and H. Bischof, "Efficient 3d scene abstraction using line segments," *Computer Vision and Image Understanding*, vol. 157, pp. 167 – 178, 2017. Large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans.

[17] P. Neubert, P. Protzel, T. Vidal-Calleja, and S. Lacroix, "A fast visual line segment tracker," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 353–360, Sept 2008.

[18] K. Hirose and H. Saito, "Fast line description for line-based SLAM," in *BMVC 2012 - Electronic Proceedings of the British Machine Vision Conference 2012*, British Machine Vision Association, BMVA, 2012.

[19] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794 – 805, 2013.

[20] I. J. Cox, S. Hingorani, B. M. Maggs, and S. B. Rao, "Stereo without disparity gradient smoothing: a Bayesian sensor fusion solution," in *BMVC92* (D. Hogg and R. Boyle, eds.), (London), pp. 337–346, Springer London, 1992.

[21] P. N. Belhumeur, "A Bayesian approach to binocular steropsis," *International Journal of Computer Vision*, vol. 19, pp. 237–260, Aug 1996.

[22] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1222–1239, Nov 2001.

[23] H. Ishikawa and D. Geiger, "Occlusions, discontinuities, and epipolar lines in stereo," in *Computer Vision — ECCV'98* (H. Burkhardt and B. Neumann, eds.), (Berlin, Heidelberg), pp. 232–248, Springer Berlin Heidelberg, 1998.

[24] "http://www.outdoorphotoacademy.com/comparing-dynamic-range/."

[25] C. A. Mead and M. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, no. 1, pp. 91 – 97, 1988.

[26] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck, "A pencil balancing robot using a pair of aer dynamic vision sensors," in *2009 IEEE International Symposium on Circuits and Systems, Taipei*, pp. 781–784, May 2009.

[27] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, p. 223, 2013.

[28] S. Schraml, A. N. Belbachir, N. Milosevic, and P. Schön, "Dynamic stereo vision system for real-time tracking," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 1409–1412, May 2010.

[29] Z. Ni, C. Pacoret, R. Benosman, S. Ieng, and S. Régnier, "Asynchronous event-based high speed vision for microparticle tracking," *Journal of Microscopy*, vol. 245, no. 3, pp. 236–244, 2011.

[30] D. Borer, T. Delbruck, and T. Rösgen, "Three-dimensional particle tracking velocimetry using dynamic vision sensors," *Experiments in Fluids*, vol. 58, p. 165, Nov 2017.

[31] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 16–23, Oct 2016.

[32] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 359–364, May 2014.

[33] G. Taverni, D. P. Moeys, F. F. Voigt, C. Li, C. Cavaco, V. Motsnyi, S. Berry, P. Sipilä, D. S. S. Bello, F. Helmchen, and T. Delbruck, "In-vivo imaging of neural activity with dynamic vision sensors," in *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4, Oct 2017.

[34] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time," *International Journal of Computer Vision*, Nov 2017.

[35] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 db 15 us latency asynchronous temporal contrast vision sensor," *Solid-State Circuits, IEEE Journal of*, vol. 43, pp. 566–576, Feb 2008.

[36] C. Posch, D. Matolin, and R. Wohlgenannt, "A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 259–275, Jan 2011.

[37] C. Brändli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 x 180 130 db 3 us latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 2333–2341, Oct 2014.

[38] C. Sulzbachner, C. Zinner, and J. Kogler, "An optimized silicon retina stereo matching algorithm using time-space correlation," in *CVPR 2011 WORKSHOPS*, pp. 1–7, June 2011.

[39] J. Kogler, F. Eibensteiner, M. Humenberger, C. Sulzbachner, M. Gelautz, and J. Scharinger, "Enhancement of sparse silicon retina-based stereo matching using belief propagation and two-stage postfiltering," *Journal of Electronic Imaging*, vol. 23, pp. 23 – 23 – 15, 2014.

[40] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Transactions on Neural Networks and Learning Systems*, 2012.

[41] J. Carneiro, S.-H. Ieng, C. Posch, and R. Benosman, "Event-based 3d reconstruction from neuromorphic retinas," *Neural Networks*, vol. Special Issue, 2013.

[42] L. A. Camunas-Mesa, T. Serrano-Gotarredona, S.-H. Ieng, R. B. Benosman, and B. Linares-Barranco, "On the use of orientation filters for 3d reconstruction in event-driven stereo vision," *Frontiers in Neuroscience*, vol. 8, p. 48, 2014.

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

[44] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, no. 4262, pp. 283–287, 1976.

[45] M. Firouzi and J. Conradt, "Asynchronous event-based cooperative stereo matching using neuromorphic silicon retinas," *Neural Processing Letters*, vol. 43, pp. 311–326, Apr 2016.

[46] E. Piatkowska, J. Kogler, N. Belbachir, and M. Gelautz, "Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 370–377, July 2017.

[47] M. Osswald, S.-H. Ieng, R. Benosman, and G. Indiveri, "A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems," *Scientific Reports*, vol. 7, pp. 40703–, Jan. 2017.

[48] G. Dikov, M. Firouzi, F. Röhrbein, J. Conradt, and C. Richter, "Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware," in *Biomimetic and Biohybrid Systems* (M. Mangan, M. Cutkosky, A. Mura, P. F. Verschure, T. Prescott, and N. Lepora, eds.), (Cham), pp. 119–137, Springer International Publishing, 2017.

[49] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 349–364, Springer International Publishing, 2016.

[50] M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schon, B. Kohn, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *2006 IEEE 12th Digital Signal Processing Workshop 4th IEEE Signal Processing Education Workshop*, pp. 173–178, Sept 2006.

[51] C. Brändli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck, "Elised - an event-based line segment detector," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP), Krakow*, pp. 1–7, June 2016.

[52] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Netw.*, vol. 66, pp. 91–106, June 2015.

[53] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based harris corner detection exploiting the advantages of event-driven cameras," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4144–4149, Oct 2016.

[54] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in *28th British Machine Vision Conference (BMVC)*, September 2017.

[55] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 430–443, Springer Berlin Heidelberg, 2006.

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

[57] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, "Feature detection and tracking with the dynamic and active-pixel vision sensor (davis)," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pp. 1–7, June 2016.

[58] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–195, IEEE, 2003.

[59] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2007.

[60] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Pattern Recognition*, pp. 31–42, Springer, 2014.

[61] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman, "Hots: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2016.

[62] J. Witt and U. Weltin, "Robust real-time stereo edge matching by confidence-based refinement," in *Intelligent Robotics and Applications* (C.-Y. Su, S. Rakheja, and H. Liu, eds.), vol. 7508 of *Lecture Notes in Computer Science*, pp. 512–522, Springer, 2012.

[63] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2262–2275, Dec 2010.

[64] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239–256, Feb. 1992.

[65] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Régnier, "Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics.," *IEEE Trans. Robotics*, vol. 28, no. 5, pp. 1081–1089, 2012.

[66] S. Fox and D. Lyons, "An approach to stereo-point cloud registration using image homographies," vol. 8301, pp. 7–, 01 2012.

[67] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, pp. 133–148, feb 2013.

[68] R. Szeliski, *Computer Vision: Algorithms and Applications*, vol. 5. 01 2011.

[69] L. Everding and J. Conradt, "Low-latency line tracking using event-based dynamic vision sensors," *Frontiers in Neurorobotics*, vol. 12, p. 4, 2018.

[70] D. Burschka, C. Eberst, and C. Robl, "Vision based model generation for indoor environments," in *Proceedings of International Conference on Robotics and Automation*, vol. 3, pp. 1940–1945 vol.3, Apr 1997.

[71] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a Line Segment Detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.

[72] A. Amedi, W. M. Stern, J. A. Camprodon, F. Bermpohl, L. Merabet, S. Rotman, C. Hemond, P. Meijer, and A. Pascual-Leone, "Shape conveyed by visual-to-auditory sensory substitution activates the lateral occipital complex," *Nature Neuroscience*, vol. 10, pp. 687–, May 2007.

[73] P. B. y Rita and S. W. Kercel, "Sensory substitution and the human–machine interface," *Trends in Cognitive Sciences*, vol. 7, no. 12, pp. 541 – 546, 2003.

[74] V. S. Ghaderi, M. Mulas, V. F. S. Pereira, L. Everding, D. Weikersdorfer, and J. Conradt, "A wearable mobility device for the blind using retina-inspired dynamic vision sensors," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 3371–3374, Aug 2015.

[75] L. Everding, L. Walger, V. S. Ghaderi, and J. Conradt, "A mobility device for the blind with improved vertical resolution using dynamic vision sensors," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–5, Sept 2016.

[76] A. Haigh, D. Brown, P. Meijer, and M. Proulx, "How well do you see what you hear? the acuity of visual-to-auditory sensory substitution," vol. 4, p. 330, 06 2013.

[77] A. Guibourgé, V. Ghaderi, J. Conradt, and B. Seeber, "Acoustic spatial encoding in a portable navigation aid for the blind," in *Fortschritte der Akustik – DAGA '16* (M. Vorländer and J. Fels, eds.), (Aachen, Germany), pp. 816–818, Dt. Ges. f. Akustik e.V. (DEGA), Mar 2016.

[78] M. Bach, "The freiburg visual acuity test-variability unchanged by post-hoc reanalysis," *Graefe's Archive for Clinical and Experimental Ophthalmology*, vol. 245, pp. 965–971, Jul 2006.

[79] E. Striem-Amit, M. Guendelman, and A. Amedi, "'visual' acuity of the congenitally blind using visual-to-auditory sensory substitution," *PLoS ONE*, vol. 7, p. e33136, Feb. 2012.

[80] S. Chen, L. Hallum, N. Lovell, and G. Suaning, "Visual acuity measurement of prosthetic vision: a virtual-reality simulation study," vol. 2, pp. S135–45, 04 2005.

[81] A. Nau, M. Bach, and C. Fisher, "Clinical tests of ultra-low vision used to evaluate rudimentary visual perceptions enabled by the brainport vision device," *Translational Vision Science & Technology*, vol. 2, no. 3, p. 1, 2013.