Imperial College London
Department of Computing

Estimating General Motion and Intensity from Event Cameras

Patrick Alexander Bardow

21st October 2018

Supervised by Dr. Stefan Leutenegger and
Prof. Andrew Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Abstract

Robotic vision algorithms have become widely used in many consumer products which enabled technologies such as autonomous vehicles, drones, augmented reality (AR) and virtual reality (VR) devices to name a few. These applications require vision algorithms to work in real-world environments with extreme lighting variations and fast moving objects. However, robotic vision applications rely often on standard video cameras which face severe limitations in fast-moving scenes or by bright light sources which diminish the image quality with artefacts like motion blur or over-saturation.

To address these limitations, the body of work presented here investigates the use of alternative sensor devices which mimic the superior perception properties of human vision. Such silicon retinas were proposed by neuromorphic engineering, and we focus here on one such biologically inspired sensor called the event camera which offers a new camera paradigm for real-time robotic vision. The camera provides a high measurement rate, low latency, high dynamic range, and low data rate. The signal of the camera is composed of a stream of asynchronous events at microsecond resolution. Each event indicates when individual pixels registers a logarithmic intensity changes of a pre-set threshold size. Using this novel signal has proven to be very challenging in most computer vision problems since common vision methods require synchronous absolute intensity information.

In this thesis, we present for the first time a method to reconstruct an image and estimation motion from an event stream without additional sensing or prior knowledge of the scene. This method is based on coupled estimations of both motion and intensity which enables our event-based analysis, which was previously only possible with severe limitations. We also present the first machine learning algorithm for event-based unsupervised intensity reconstruction which does not depend on an explicit motion estimation and reveals finer image details. This learning approach does not rely on event-to-image examples, but learns from standard camera image examples which are not coupled to the event data. In experiments we show that the learned reconstruction improves upon our handcrafted approach. Finally, we combine our learned approach with motion estimation methods and show the improved intensity reconstruction also significantly improves the motion estimation results. We hope our work in this thesis bridges the gap between the event signal and images and that it opens event cameras to practical solutions to overcome the current limitations of frame-based cameras in robotic vision.

Contents

Chapter 1

Introduction

Contents

In the recent decades, video cameras have become ubiquitous in our daily lives, with most people having at least one camera in their phone. The cheap manufacturing costs of these cameras allows them to be employed in many consumer products, like cars, vacuum cleaners, video consoles and so on. The advances in robotic vision and machine learning increase the usefulness of these devices by accessing the rich source of information which the video stream provides. However, it also challenges the developers of those devices to process this information even in extreme use-cases. Creating reliable robotic vision methods is challenging in itself, but it is often further complicated by the limitations of cameras, like their power consumption, motion blur and over- and underexposure. These limitations are inherent to the camera design, which was originally built to capture appealing pictures, and not as a device for computer vision. Robotic vision has different requirements than simply capturing pictures, which inspired the development of devices which mimic biological retinas. These devices have superior properties to standard cameras and motivated the development of silicon retinas like the *event camera*.

In contrast to standard cameras, an event camera does not capture images but only registers changes in light, which creates a sparse measurement signal. The signal drastically reduces the amount of data produced by the camera, but it also increases its power efficiency and lowers the camera response time. The disadvantage, however, is that it produces a very different signal from a standard camera, which requires developing new vision algorithms for event-based cameras.

While the camera potentially offers many benefits for robotic vision, it is still an open question how to correctly process its signal. Many pieces of event camera research, which have been done before this work, have processed the data by using strong assumptions on the environment or the camera motion, but it was long unclear if the event camera signal alone contains enough information for robotic vision.

In this thesis, we estimate motion from the event camera signal in a very general case using *optical flow*. We show that it is not only possible to estimate dense optical flow from the sparse camera stream without any restrictions or additional sensing, but that this also enables us to reconstruct a corresponding intensity image. Our work introduces not only first image reconstruction method from events with no restrictive assumptions, except for smoothness in motion, but also shows that the event stream contains enough

information for image processing. We further investigate the improvements of those reconstructions by employing machine learning methods to combine natural image priors with the event camera signal.

For our approaches, we focus on analysing the event stream to provide a more useful basis for common computer vision applications, which ideally will give the foundation to more efficient and accurate methods based on the minimal event stream.

1.1 Potential Applications

In this section, we further motivate the contributions of this thesis by discussing potential applications for an event camera, if an appropriate event-based algorithm is provided. We assume that such an algorithm preserves four distinct advantages of the event camera over a standard camera: A low-data-rate, low power consumption, low latency and *high-dynamic range* (HDR).

Because of its sparse and frame-less signal, the event camera is maybe not the ideal sensor for taking holiday pictures; however, its properties of low latency and HDR make its application interesting for robotic vision in fast-moving and challenging environments in which standard camera images usually fail. Figure 1.1 shows two examples of common standard camera artefacts: Motion blur and over-saturations. The motion blur is caused by fast motion during the exposure time of the image, while over-saturation occurs when too much light hits the camera sensor for it to quantify the light information accurately. In the example, the standard camera is not able of capturing all parts of the scene, however, an event camera would be able to do so.

A standard camera may compensate for severe lighting conditions by using multiple exposures and merge them into a single HDR image; however, this usually fails when the scene is not still. Such a scenario occurs for example for cameras which are mounted to cars. Here the sun may over-expose any camera which is installed to assist the driver or for self-driving purposes. In this case, the camera cannot correctly perceive the whole environment and may endanger the driver or pedestrians if decisions were made based on those images. In comparison, an event camera offers here many advantageous qualities. For example, the *Dynamic Vision Sensor* (DVS) is a commercially available event camera which has a dynamic range of about 120 dB which can readily handle

(a) Motion-blur (b) Over-exposure

Figure 1.1: Exposure artifacts of frame-based camera. a shows a desk seen which capture while the camera is moved. b shows a street seen, where part of the image is over-exposed by looking into the direction of the sun.

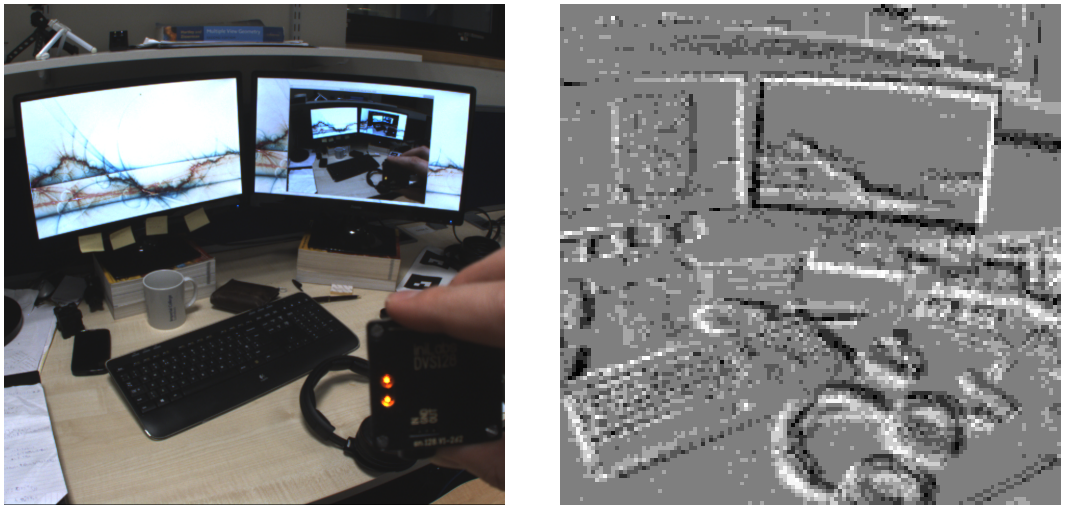
extreme light conditions, while standard cameras usually only have 60 dB of dynamic range [53]. Additionally, event cameras measures changes in light with a very high time resolution on a microsecond scale. In the self-driving car scenario, the combination of the high dynamic range and time resolution allows event cameras to properly sense their environment and provide measurements to the driverless system to take appropriate actions. Here, the low data-rate of event cameras, in comparison to high-speed cameras, is another advantage. Computer vision methods need to process images, which in the case of high-speed cameras, are hundreds of images per second and more. Processing that amount of information alone, even with simple processing algorithms, in real-time is challenging itself. In contrast to this, the event camera produces only hundreds of kilobytes per seconds.

The robust sensing of event cameras is also advantageous for *unmanned aerial vehicles* (UAV) like flying drones, which gained interest in recent years from industry and academia. Most drones are manually controlled, but there is an active field of developments for making drones increasingly autonomous. For the autonomous control, a drone estimates its position and speed with an array of different sensors which allows it to plan and execute manoeuvres. Amongst the sensors, cameras are an essential part of drones, because they relate the drone's position to external landmarks. However, like with cars, the camera suffers from over-exposure and motion-blur which cause situations in which

the drone must ignore the camera input, for example when the drone flies close to the ground. The speed of the drone causes strong motion blur and significant disparities between the camera images, which corrupts the measurements for a pose estimate. An event camera with its frame-less and low-latent signal is still able to operate under these conditions, as well as providing the advantage of low-power consumption, which is an essential aspect of drones as well.

Power consumption concerns mobile platforms like drones, but also robots and mobile phones which must handle different tasks at once and so, spending extra battery life on a sensor is undesirable. In comparison to standard cameras, the event camera consumes very little power. For example, the DVS consumes about $24mW$ of power compared to $1 \sim 2W$ consumed by frame-based cameras [53]. The magnitude of reduced power consumption allows a device to run the camera for long periods of time without recharging the batteries. Head-mounted displays like the HTC Vive [97], Oculus Rift [68] and Microsoft Holo-Lens [59] would benefit from such a the low power consumption as well as from the low data rate of event cameras. For user interaction, headsets must quickly react to head movements and estimate their positions in the world. The low latency requirement means that a lot of the computation must be performed on the devices itself, which increases their power consumption. To reduce this burden external servers are often used, but without a wired connection, bandwidth is limited, which means that sending images or other large amounts of information is not feasible. The event-camera has a natural low bandwidth which eases the burden of transferring the camera data through a wireless connection to perform the more computationally intense processing on a server.

We mentioned in this section just a few potential application which would benefit from event cameras, but this is not an exhaustive list. However, it shows that the event camera offers benefits to a wide range of robotic vision applications. In the long term, the event camera has the potential to be used for all robotic vision applications, since the human visual system, which inspired the camera design, is capable of providing the necessary information for such tasks.



(a) Frame-based Camera (b) Event Camera

Figure 1.2: Example of an event camera capturing a scene. The white pixels in the event camera example indicate increases in brightness which are captured by the camera and black pixel indicate decreased in brightness.

1.2 Event Camera

In the last section, we discussed potential applications of event cameras under the assumption that an appropriate algorithm exists. In this section, we elaborate on the reason why developing such a method is not trivial, which is followed in the next section by a summary of previously proposed event-based methods and their shortcomings, which motivate the approaches presented in this thesis. We begin here with a brief introduction of the biologically inspired mechanics of an event camera and then expand on the challenge of understanding its signal.

1.2.1 Event Camera Design

The event camera design mimics the traits of biological retinas and offers a paradigm shift from the common, frame-based camera. While both standard and event cameras are similar in their optical mechanics of lenses, focal points and so on, they differ in the way they measure light. An example of these different signals is shown in Figure 1.2. Unlike standard cameras, the event camera does not measure absolute intensities, but only changes in light. A change triggers a camera pixel to send a binary signal: 0 for

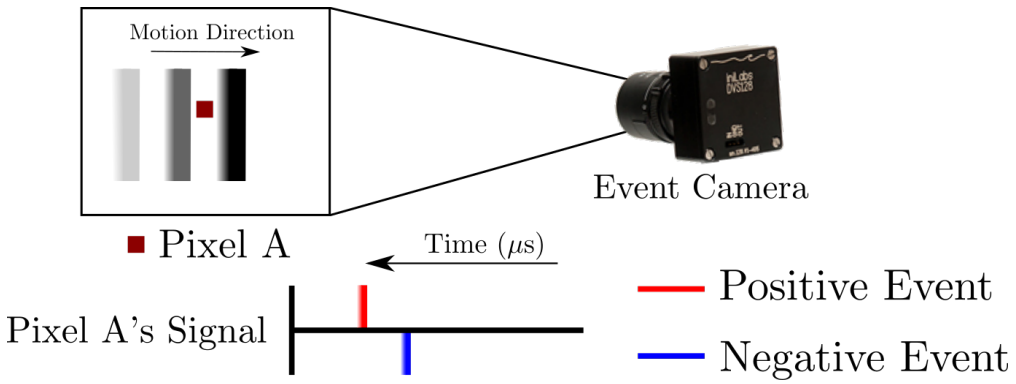


Figure 1.3: Overview of event camera measurement mechanics. If a bar in the figure moves in front of pixel A, then camera triggers a negative event to indicate that the intensity hitting the pixel decreased. When the bar passes over the pixel a positive event is fired which indicates that the intensity increased again.

darker and 1 for brighter, but each pixel does so independently from the others. In other words, the binary signal is sent as soon as the change occurs without waiting for other pixels, which consequently creates a continuous, frame-less signal which is in contrast to the frame-based paradigm of standard cameras. Figure 1.3 provides an overview of event camera mechanics.

The main inspiration for this design is the behaviour of neurons in the retina, which is exemplified by Figure 1.4 by highlighting the correspondences of the camera parts and retinal neurons. As shown in Figure 1.4, each pixel of an event camera has its analogue circuitry with three logical parts: Photoreceptor, Differencing Circuit and Comparators [53]. These parts are responsible for transforming the light into an electrical current, computing the amount of change since the last event and to trigger an event if the change exceeds an internal threshold. In the case an event is fired, the differencing unit resets and starts measuring the difference for the next event.

While the camera design goal it to mirror retinal neurons, it can also be justified if the design goal was to create a video camera without a shutter. A shutter creates a brief period of darkness for the camera sensor to read-out the measurements and resets the sensor values to a global value of uniform darkness. If a measure relative to total darkness (or another absolute value) is missing, then any visual sensor can only perceive relative changes, which is what retinal neurons and event camera pixels do. Both neurons

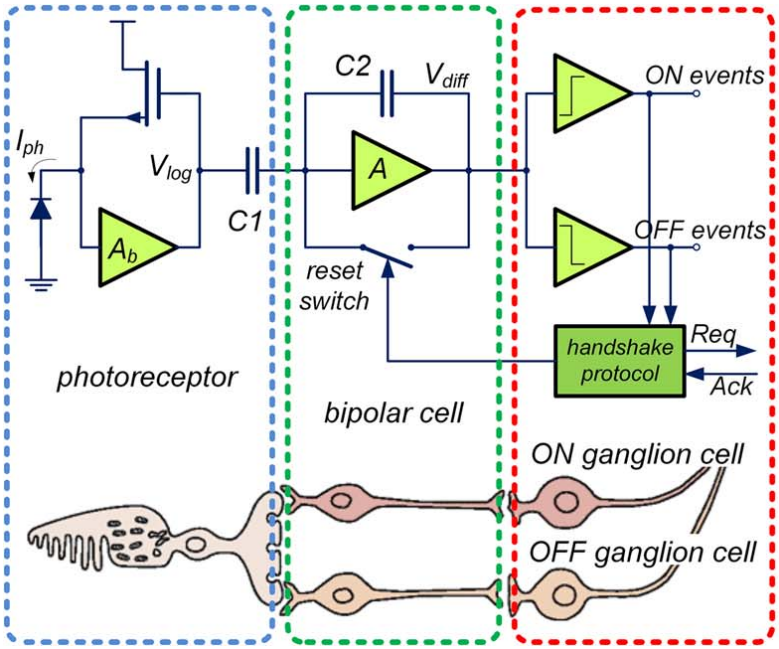


Figure 1.4: Schematics of the event camera circuitry parts and corresponding parts of a biological retina. The image is taken from the publication of Posch *et al.* [76].

and event pixels wait until a change in light appears, then send out their signal and then wait again until the next change relative to the last one happens. By not using a shutter, the event camera avoids problems of standard cameras related to camera exposure like motion-blur and over-exposure. However, this also means that the event camera loses the ability to sense absolute intensities.

Different event camera models have been proposed, but we are mainly focusing in this thesis on the DVS [53] and *Dynamic and Active Vision Senor* (DAVIS) [13]. While the DVS is a pure event-based camera, the DAVIS combines a frame-based camera with an event camera at the sensor level. However, the DAVIS does not synchronise both event and image signal with respect to their light measurements, meaning that the increase of light in the image does not directly translate to fired events. Another event-based sensor which combines an image sensor and event pixel is the *Asynchronous Time-based Image Sensor* (ATIS) [75] which, in contrast to the DAVIS, only trigger an exposure per pixel if an event is fired at that pixel.

Both DAVIS and ATIS, have been proposed since the pure event stream is difficult to

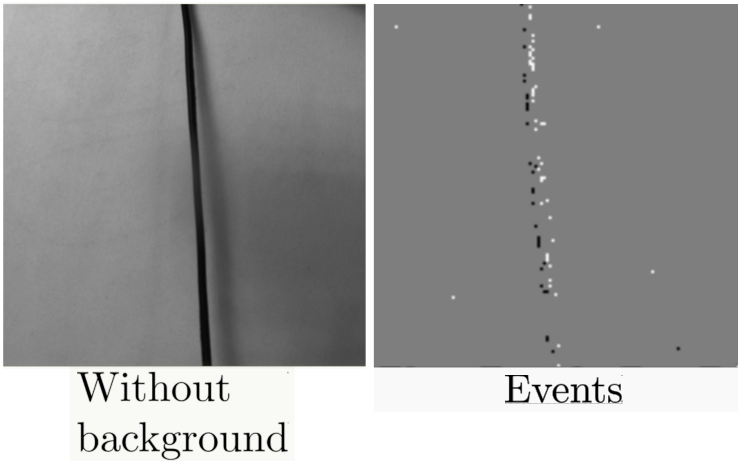
analyse. In the next section, we discuss the reasons for those challenges of understanding event data.

1.2.2 Understanding Event Data

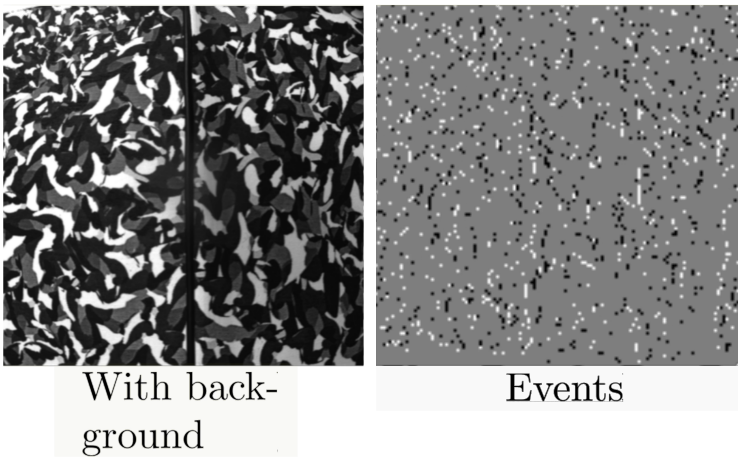
While the design of the event camera offers many benefits, it also creates an entirely different signal from the image signal which we know from frame-based cameras. Frame-based computer vision methods are ill-suited for event cameras because they rely on the information about the relationship between pixels which is missing in the event data. The information gap challenges the design of any event-based method and limits the adoption of event cameras in a wide array of robotic applications. However, before approaching a solution to this problem, we provide here an intuitive example of the information gap and its consequence to frame-based computer vision methods.

As mentioned above an *event* measures an intensity difference over time, which means that if we look at a single event, then we know that the brightness has changed at that pixel, but we do not know how bright that pixel was before or after the event. For the whole event camera sensor this means that the brightness for each pixel is unknown, as well as the brightness difference between pixels which is important for many image processing algorithms like edge detection.

As a practical example for this, we provide in Figure 1.5 two event-based scenes and highlight the problem of identifying a moving object in both scenes. The first scene in Figure 1.5 is simple and shows a black bar moving in front of a plain background. We can easily see the bar in the event signal because the moving object itself causes all the active regions. However, in the second example, we merely add a slightly moving textured background but leave the rest of the scene unchanged. In the standard camera image, this does not change the appearance of the object, but in the event data, the background signal over-shadows the signals from the object, which makes it barely visible. The texture causes the contrast between the object boundary and background to change constantly, which makes the event fire-rate erratic while in the texture-less example the fire-rate is constant. Without understanding the background structure, we have little chance of correctly understanding the erratic firing behaviour of events at the object boundary. This characteristic of the event signal is further studied by the work of Barranco *et al.* [8].



(a) Example with texture-less background



(b) Example with textured background

Figure 1.5: In a the first example is shown, in which we follow an object's movement in front of a texture-less background. The scene is captured by a standard camera as well as by an event camera. b shows the same scene, however a moving textured background it added.

While this dependency on the movement of the object and the texture of the background is problematic for designing event-based methods, there have been a number of methods previously proposed which address this issue in various ways. We provide an overview of those methods in the next section.

1.3 Overview of Event-Camera-Based Methods

The event camera is a relatively recently proposed camera sensor which cannot rely on conventional computer vision algorithms to accommodate its bio-inspired signal and therefore many computer vision algorithms focus on analysing it signal to be used for robotic vision tasks. In this section, we provide an overview of event-based methods which have been proposed before this work to provide context for the contributions of this thesis.

For many methods in this section, it is common to make various assumptions about the environment and the camera motion. These assumptions allow inferring the missing information, which we discuss in the previous section. However, they also limit the application of the event camera, which we highlight during their introductions.

1.3.1 Practical Applications

We begin by introducing methods which highlight the practical benefits of the event camera. Conradt *et al.* [19] introduced one of the first pieces of work in this area, which integrates the event camera into a robotic platform which balances a pencil. In this setup, two event cameras track the position of the pencil, which is determined by fitting the contour lines to the event signal. While balancing a pencil is a very particular task it highlights the low latency capabilities of the event camera.

Muggler *et al.* [60] presented another work which highlights the low latency capacity in a more general setting. The authors mount an event camera onto a quadrocopter which performs high-speed manoeuvres. The event camera provides here low latency feedback about the quadrocopter's position by following a black rectangle on the wall as a reference. Like in the previous method, the event camera tracks the black rectangle by fitting contour lines to the event signal. Muggler *et al.* showed not only the real-time performance of their method but also compared the frame-based and event-based camera and showed that the event camera does indeed not suffer from motion blur while the frame-based camera does. However, their method requires a known object in the world for navigation.

1.3.2 Event-based Estimation

While the above works highlight benefits of event-based cameras, other pieces of work are more focused on analysing the event data alone. Kim *et al.* [39] proposed a panoramic image reconstruction method from an event camera, but they required a static scene and rotation-only movement from the camera.

The method incrementally estimates the panoramic image by alternating between camera tracking and image reconstruction. First, the camera motion is tracked by sampling different possible rotation which are matched against the measured events and the so far estimated image gradients. When the optimal rotation is determined, then the measured events are integrated into the image estimation and extend the panoramic image. By requiring a rotation-only camera motion and a static scene the method does not need to address occlusions and therefore avoids the erratic event fire-rate behaviour which we describe in section 1.2.2, but it also limits the application of the event camera. Despite these restrictions, the quality of the reconstructed image motivates confidence that the event stream contains enough image information for further image processing tasks, but it leaves the question open on whether such information could be obtained without strong assumptions or additional sensors. Prior to Kim *et al.*'s contribution, Cook *et al.* [20] proposed a similar method using the same assumptions about the camera motion and scene but without the panoramic image reconstruction. Their contribution is a modular system based on belief propagation. Each component of the system estimates different quantities like optical flow, image gradients, camera motion and so on. The components build a network in which each component is influenced by the estimation of the others and, through several message-passes, the system converges to a joined estimation. The modular nature of the approach makes it extensible for other computer vision methods as well, but the constraints on the camera motion and the static scene limits the scope of these extensions.

While these pieces of work interpret event data by reconstructing the underlying image signal, other contributions approach the problem by matching events to events. Benosman *et al.* [9] proposed an event-based optical flow algorithm which estimates motion by local plane-fitting. For the plane-fitting, the authors modelled the event data as a 3-D point-cloud with the first coordinates being the image coordinates and the third being the time when the event occurred in the sequence. The plane is then fitted into the

point cloud assuming that the plane normal is perpendicular to the motion vector. The proposed algorithm can be efficiently computed and performs reasonably well in scenes with a texture-less background.

Barranco *et al.*[8] studied in their work the influence of the background on the event signal and proposed a method for object contour tracking to address this problem. The method tracks the object contour between a textured object and background by estimating the correlation between fired events. The assumption is that uncorrelated events are triggered by the contour, while events triggered from the textures on the object or background fire in a highly correlated manner. The approach by Barranco *et al.* provides more precise result than previous event-based methods in textured scenes, but it requires a high event-fire-rate at the object boundary, which depends on the amount of texture in the scene and the speed of motion.

After the release of the first contribution of this these, other pieces of work emerged for structure from motion estimation which showed that event-based signal processing is fully capable of estimating the camera motion with a monocular camera. Kim *et al.* [40] presented the first monocular SLAM systems from an event camera. Their method alternates between the estimation of the 3-D geometry, the camera motion and image reconstruction. The camera is tracked by fitting the 3-D geometry to the image estimation by matching the image gradients against the event measurements. After estimating the camera motion, the image estimation is updated with the measured events, which then allows the extension and improvement of the estimated scene geometry.

Beside this event-based SLAM method, an event-based visual odometry approach is presented by Rebecq *et al.* [78]. Their method estimates a semi-dense geometric map which focuses on the depth estimation of image regions with strong gradients. Like in the previous method, the method estimates the camera motion by projecting the estimated depth map to the image plane and matched it against the measured events, however, unlike the previous method this approach does not require an image reconstruction. Instead of reconstructing an image, the method aggregates a small number of events to approximate an edge map, which is then matched by Lucas-Kanade warping function [5] to other edge maps. However, while the method shows that these event-based edge maps contain much visual information, the authors do not directly address how the method operates with strong background influences as we discussed in Section 1.2.2.

Both structure from motion approaches strongly suggests that event cameras have the same capacity to be used by robotic vision methods as standard cameras, however, the stability of the methods has yet to be proven in practical application. For stable event-based estimations other methods rely on hybrid approaches which combine the event camera with other sensors, which we introduce in the following section.

1.3.3 Multi-Sensor Systems

Other event-based approaches combine the event camera with other sensors to complement the missing information in the event stream, which we describe in Section 1.2.2. Weikerdorfer *et al.* [100] propose an event camera-based SLAM system by combining the camera with a depth-sensor like the Kinect camera [58]. The depth-sensor provides measurement distances between the camera and the scene, which the authors use to estimate the motion of the camera and fuse it with the event data. While this work makes the high-order computer vision methods like SLAM accessible for event cameras, it cannot estimate the camera pose without the additional sensor.

Other works use several sensors, but Rogister *et al.* [84] does so by combining the event camera with another event camera. They propose an event-based stereo method for 3-D reconstruction, which matches events from both cameras by assuming that simultaneously fired events belong to the same point in the world coordinate frame. The authors introduce the novel idea to match image points not by spatial similarities but by temporal correlations. Their method takes advantages of the sparse nature of the event camera, but it is prone to mismatching points in environments with repetitive structure.

1.3.4 Machine Learning

In the area of *machine learning* (ML), there has been recently an increased number of contributions for event-based learning methods. Most of the earlier ML work combine the bio-inspired event camera with biologically feasible neural network models like *spiking neural networks* (SNN). A SNN is a variant of artificial *neural networks* (NN), which uses discrete signals akin to the electrical impulses of a biological NN. Combining a SNN with an event camera is promising because both network and camera use discrete signals, but SNNs are well-known to be difficult to train. More recently, using deep-learning with the event signal has become an active field of research which shows very

promising performance, despite that deep neural networks are not as strongly related to the discrete signal as SNNs.

O'Conner *et al.* [67] presented a SNN training method for recognising numbers from the MNIST database [51]. The authors train the SNN by encoding the image gradient to a discrete signal, which the network can process. The encoding simulates the event signal and is replaced with the actual signal after the network training is completed. The work shows that SNNs are suited for event-based recognition, however generating the necessary training data from actual event data is still a difficult task.

In the work of Osswald *et al.* [70], the authors train a SNN for depth estimation using an event camera stereo setup. For training, the authors render synthetic sequences and convert them into an artificial event signal, which is then, like in the previous work, replaced with the signal of the camera after the training is completed. Rendering sequences circumvents labelling the actual event signal, but it is difficult to generate enough varied and realistic data for a network to generalise well for more demanding scenarios.

As mentioned above, recently other pieces of work emerged which combine event-cameras with differentiable deep neural network, which are easier to train than SNN. Nguyen *et al.* [64] presented a learning-based VO system based on the event signal, which is pre-processed by accumulating events into an image. The authors use the dataset from Mueggler *et al.* [61] for training, which is a benchmarking dataset for event-based VO and SLAM methods. However, the database has a relatively small number of sequences compared to training datasets used for learning-based VO systems with frame-based cameras. Using small training datasets increase the risk for NNs to only perform well on the training dataset and the author did not present results for sequences outside the dataset from Mueggler *et al.*.

1.4 Variational Image Denoising and Optical Flow

The contributions of this thesis differ from previously proposed event-based methods and require additional backgrounds from other computer vision areas. Because of this, we continue in the following sections with an introduction to these areas which form the basis of this work.

We provide in this section a brief introduction to variational methods for image denoising and optical flow estimation. These methods have been successfully applied to a number of image processing tasks and became popular when their computation became very efficient with the introduction of *Graphics Processing Units* (GPU).

For the event camera, we look at variational methods since they fuse statical properties into their estimation, which we will later in Chapter 4 use to analyse event data. To highlight connections between variational estimation and statical inference, we provide in the following a derivation of variational formulations from Bayesian inference.

1.4.1 Bayesian Inference

Bayesian inference has applications in various areas of machine vision. A robotic agent, for example, has often not enough information to understand the environment and so the agent requires extra knowledge. In practice, different approaches are used for providing this knowledge, for example, a 3-D model of a building can provide geometric information to a robotic agent for indoor-navigation. However, such a model needs to be constructed in a precise manner, as well as updated if aspects of the scene changes, which may be time-consuming and difficult to do as well as restricting the agent to the modelled area. If precise knowledge is missing, then we can instead model uncertainty by using probabilistic inference methods. These approaches search the realm of possible solutions which fits the measurements and picks the most probable one. For example, image regions often assume similar pixel values which provide a regularity on the image, which we can exploit for our estimations to counteract measurements which are missing or corrupted.

In computer vision, measurements often lack precision which makes statistical approaches a natural choice. Formally, let us define a model u which describes a hypothesis, like the 3-D shape of an object or the noise-free version of the image, and let us treat u as a sample of distribution of all possible solutions, and the goal is to select the most likely sample given an observation z. This principle is known as the *maximum-a-posteriori* (MAP) estimation [12] and is defined as:

$$\hat{u} = \arg\max_{x} p(u|z), \tag{1.1}$$

where $p(u|z)$ is the posterior probability of u given the observation or measurement z. Usually, the distribution $p(u|z)$ is not known or is hard to describe, but it can be reformulated to:

$$p(u|z) = \frac{p(z|u)p(u)}{p(z)} \tag{1.2}$$

using Bayes theorem, where $p(u)$ is the prior probability of u and $p(z|u)$ is the conditional probability which describes how well the observed data z supports the solution u. $p(z)$ is called evidence and can be viewed as a constant since it is not dependent on u and therefore can be omitted during optimisation. For example, if we assume that u is differentiable and model our distribution as the product of two Gaussian distributions, then the above equation can be defined as:

$$p(z|u)p(u) := \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{|u-z|^2}{2\sigma}} \quad \frac{1}{\sqrt{2\pi\nu^2}} e^{-\frac{|\nabla u|^2}{2\nu}}, \tag{1.3}$$

where σ and ν are the variances of each distribution. The first distribution models the distribution which fits the given measurements z, while the second distribution models the likelihood that u first-order derivate is small. Applying the negative logarithm simplifies the formulation to

$$-\log(p(u|z)p(u)) = \frac{1}{2\sigma}|u-z|^2 + \frac{1}{2\nu}|\nabla u|^2 + \frac{1}{2}\log(2\pi\sigma) + \frac{1}{2}\log(2\pi\nu) \tag{1.4}$$

Since $\frac{1}{2}\log(2\pi\sigma)$ and $\frac{1}{2}\log(2\pi\nu)$ are not dependent on u, we can omit these terms for the optimisation. Multiplying the equation by 2ν gives:

$$E(u) := -2\nu\log(p(z|u)p(u)) = \lambda|u-z|^2 + |\nabla u|^2, \ \lambda = \frac{\nu}{\sigma}, \tag{1.5}$$

which is also known as the Tikhonov model [92]. From this model and Equation (1.2) we can see that the quality of MAP estimation depends on the accuracy of data model $p(z|u)$ and the prior $p(u)$. Unfortunately, the more complex a model is, the harder it becomes to find the optimal MAP solution. In the next section, we discuss one area of application for such models which is image denoising.

1.4.2 Image Denoising

An important area of computer vision, which we introduce in this section, is image denoising which informs our analysis of the event camera signal in later chapters. Image denoising is an interesting field for the event camera since the goal of image denoising is to reveal the true image signal while suppressing the measurement noise. Denoising approaches rely on using image statistics which are integrated into the denoising process. The use of these statistics for analysing the event signal is an important basis to the approaches chosen throughout this thesis. Because of this, we provide here a broader background to image denoising before introducing variational denoising methods. We end this section by briefly addressing machine learned methods for image restoration, which inform our approaches to event-based analysis as well.

Image denoising for standard cameras addresses that the measurements of the emitted light from a scene are altered by different factors, or noise sources, like imprecisions of the photometric sensor, thermal noise, atmospheric influences and so on. Human vision can cope with those influences, which is why a human can recognise the content of an image even in the presence of strong noise.

However, machines cannot discard those influences as easily, and so the ability needs to be emulated by a method which restores the noise-free version of the image. Such a denoising algorithm must be able to separate the noise distribution from the real image distribution while leaving the actual image signal as intact as possible. Over the years different image denoising methods have been proposed, and we give here a brief overview of them, but with the primary focus on variational methods.

Before we introduce denoising methods, we provide here a conventional machine vision model to understand the type of noise that we address with these methods. The model describes the noise of an image as an additive sample of a Gaussian distribution, which is formally expressed as:

$$I(x, y) := \bar{I}(x, y) + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{1.6}$$

where $\mathcal{N}$ is the Normal distribution with 0 as mean and σ^2 as variance with $\bar{I}$ being the true image signal at position x, y. Here we note that ϵ is independently sampled for each

(a) Original Image (b) Corrupted Image

Figure 1.6: Two versions of the same b is the corrupted version of a by augmenting it with Gaussian noise.

pixel. Equation (1.6) is a substantial simplification of the real noise process. However, since we have a complex system of several noise influences, we can justify Equation (1.6) by using the central limit theorem. Figure 1.6 we show an example of such a corrupted image.

Many different approaches have been proposed over the decades to reverse Equation (1.6), however, the most commonly known method is Gaussian smoothing. Smoothing is performed by convolving the image with a Gauss filter and is defined for each x, y position in the image such as:

$$(g * I)(x, y) := \int \int g(u - x, v - y) I(u, v) \, \mathrm{d}x \, \mathrm{d}y, \tag{1.7}$$

where g is a two-dimensional Gaussian probability density function (PDF). There exist different interpretations for Equation (1.7), but it can be viewed as the expected pixel value at x, y over the image, where we assume that nearby pixels are more likely to share the same value than faraway pixels. This interpretation becomes more apparent when we recall the definition of the expected value and compare it with Equation (1.7)

$$\mathbb{E}_{x \sim p}[f(x)] := \int f(x) p(x) \, \mathrm{d}x. \tag{1.8}$$

The Formulation (1.7) can be implemented in a simple and efficient manner, but it also causes the image to lose sharpness, which why the Gauss filter is also referred to as the blur filter. Figure 1.8a shows an example of an image, which has been smoothed with a Gaussian filter in which the loss of sharpness is quite apparent. The sharpness is lost because the Gaussian distribution does not reflect the statistical relationship between nearby pixels in a natural image adequately. Huang *et al.* [33] studied this relationship and showed that a more appropriate approximation is expressed by a generalised Laplace distribution, which is defined as:

$$f(x) := \frac{1}{Z} \exp\left(-\left|\frac{x}{s}\right|^{\alpha}\right), \tag{1.9}$$

where α and s are two parameters which are related to the variance and kurtosis of the distribution and Z is the normalisation factor. Huang *et al.* [33] showed that with $\alpha = 0.55$ Equation (1.9) expresses sufficiently the difference between two neighbouring pixels. Figure 1.7 displays the negative logarithm of Equation (1.9) with different α values. We see that with $\alpha = 0.55$, the minima of the function is at the point when the pixels share the same value. This concurs with the observation that images often contain fairly homogeneous regions like walls, tables and so on, which share similar pixel values. Another observation in Figure 1.7 is that the function with $\alpha = 0.55$ has heavy tails, meaning that at the function decreases slower as it approaches $-\infty$ and ∞ than near the origin. In images, this corresponds to edges and object boundaries in which the value between two pixels suddenly changes by arbitrarily large values, for example in an image with a black square in front of a white background. Edges are often present in an image, but they rarely form the majority of the natural image content. These observations are in contrast to a Gaussian assumption, which is expressed by setting α to 2 which also displayed in Figure 1.7. The Gaussian distribution models large value changes as highly unlikely and, if applied for denoising an image, it causes the image to lose its sharpness.

While Gaussian smoothing is not ideal for denoising, better results are achieved efficiently by using variational methods. Variation in this context of images is the fluctuation or perturbation of pixel values in an image. More formally, for the 1-D function case, it is the measure of the arc-length of the function. To minimise the variation in an image means that the pixel values are brought closer together, which satisfies the first

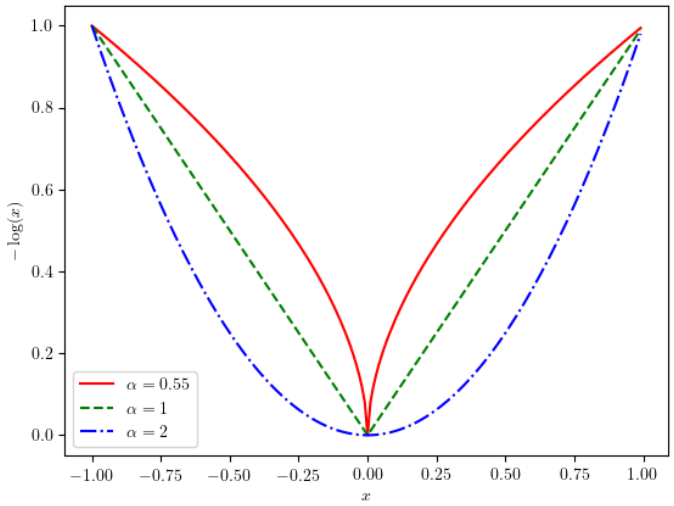


Figure 1.7: Plots of the negative logarithm of the laplace distribution with different α values.

observation from Figure 1.7. First, we formulate a variational minimisation problem which is defined such that:

$$\min_{\hat{I}} \frac{1}{2} \int_{\Omega} |\nabla \hat{I}|^{\alpha} \mathrm{d}\Omega + \frac{1}{2\lambda} \int_{\Omega} (\hat{I} - I)^2 \mathrm{d}\Omega, \tag{1.10}$$

where $\lambda > 0$ is a scalar parameter and α is the parameter of the Laplace distribution, which we choose. If $\alpha = 0.55$ then solving Equation (1.10) becomes non-convex, which makes it much harder to solve since it may create multiple solutions and finding the optimal solution becomes computationally expensive [12]. If α is chosen such that $\alpha \geq 1$, then the problem is convex, for which many efficient methods exist to find the optimal solution. Because of the advantages of convex formulations, a common choice is $\alpha = 1$ since is closet value to the actual statistics while still being convex. Formally, from now on we refer to regularisation term:

$$\text{TV-L}^1(I) := \int_{\Omega} |\nabla I| \mathrm{d}\Omega, \tag{1.11}$$

as *Total Variation* smoothness term, or TV-L^1 smoothness prior. A common model

(a) Gauss Filter (b) TV-L^1

Figure 1.8: Two denoising examples using the same initial corrupted image. a shows the results using Gaussian denoising and b shows the results of TV-L^1 denoising.

which uses TV-L^1 smoothness is the *Total Variation* (TV-L^1) model [65], which is defined such that:

$$\min_{\hat{I}} \int_{\Omega} |\nabla \hat{I}| \mathrm{d}\Omega + \lambda \int_{\Omega} |\hat{I} - I| \mathrm{d}\Omega, \tag{1.12}$$

which is similar to Equation (1.10), except that norm of the data fidelity term is replaced by the absolute norm. The absolute norm is more robust against outlier measurements, which makes the model suitable for practical applications. TV-L^1 is a very versatile formulation which is successfully applied in many computer vision areas, and we discuss it further in detail in the following sections. In Chapter 3, we introduce the primal-dual algorithm which is a fast and efficient method to find the solution of Equation 1.12.

TV-L^1 presents a cost function which is uniquely solvable, but it allows for sudden changes and is robust against mismatches. To highlight the performance of the TV-L^1 smoothness prior for image denoising, we compare in Figure 1.8 a TV-L^1-based solution with another solution based on a Gaussian distribution. In the Figure we see that the Gaussian distribution causes the edges to blur, while TV-L^1 solution is able to restore the image while preserving the sharpness of edges.

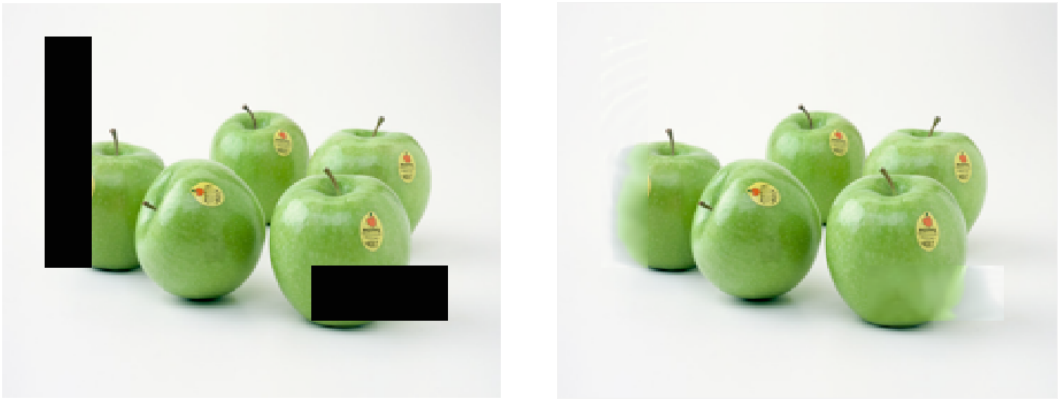
TV-L^1 regularisation preserves edges, however, it reduces variation by enforcing that

local image regions assume the same pixel values, which is not always the case in natural images. For instance, shadows may cause soft gradients which moderately change the pixel value in an image region. In this case, TV-L^1 suppresses the soft-gradients and creates block-artefacts, which is also known as the stair-casing effect. Bredies *et al.* addressed the stair-casing effect by introducing *Total Generalised Variation* (TGV) [14], which proposes not only to regularise the first derivative like TV-L^1 but higher-order derivates as well. Their proposed method significantly reduces the effect, but it also has a slower convergence rate than TV-L^1.

Other denoising methods use statical knowledge gained from samples of other natural images and use it to denoise a given corrupted image. Olshausen and Field [69] proposed using sparse coding methods to learn a set of convolutional filters, which are in the later stage of the algorithm used for denoising the corrupted image. Roth and Black [86] extended this approach by also inferring spatial relationships between image patches. These methods show impressive performance, and that learning image statistic is a promising path to advance denoising methods, which is expanded in the area of *deep learning* (DL).

DL became in recent years a popular approach for image denoising. The advantage of DL methods is that they do not require any handcrafted model of the image or noise distribution. Instead, DL methods use a large dataset of corrupted and corresponding clean images. The correspondences allow for the DL method to learn the properties of the image and noise distribution as well as their separation, which shows state-of-the-art performance but also the capacity to fill in gaps of missing or destroyed image data. For example, the method presented by Fawzi *et al.* [22] trains a network to complement blacked out image regions by basically "hallucinating" the missing content, which is also called inpainting. Figure 1.9 shows an example of this method which highlights its capacity.

The DL inpainting approach highlights the ability of NNs to express natural image statistics and the capacity to use them to restore large image regions. However, applying denoising methods for event-based analysis in only one part of our contributions in this thesis, the other part is to estimate motion using optical flow estimation, which we introduce in the next section.



(a) Masked-Out Image (b) Reconstruction

Figure 1.9: Input and output of the reconstruction network. a shows the corrupted version of the image, which is partically masked with black boxes. The network is tasked to reconstruct these blacked-out regions and b shows the results, which are taken from the results of Fawzi *et al.* [22].

1.4.3 Optical Flow

Perceiving motion is a fundamental ability that humans have to understand the surrounding world, and it allows us to recognise the instantaneous image motion caused by an object's movement and to react accordingly. Devices like robots do not have such "natural" abilities, and so developers must provide methods which mimic such a perception like optical flow. Optical flow is a fundamental form of motion perception, which describes the per-pixel movement between two images. Analysing this type of motion builds the foundation of many computer vision methods for video sequences like action recognition, which seeks to understand human gestures. Optical flow has been a research subject for decades and a range of approaches has developed around this topic. We provide here a summary of some of these methods but focus mainly on the variational approaches, which form some of the basis of this thesis.

Although optical flow is an essential property for motion understanding, it is also an ill-posed inverse problem, because inferring the actual motion of an object by finding pixel correspondences between images is, in general, not uniquely determinable. This indeterminism is also known as the *aperture problem*, which describes the inability to determine the actual motion of an object with only local perception, without additional information or regularisation. Figure 1.10 illustrates this problem.

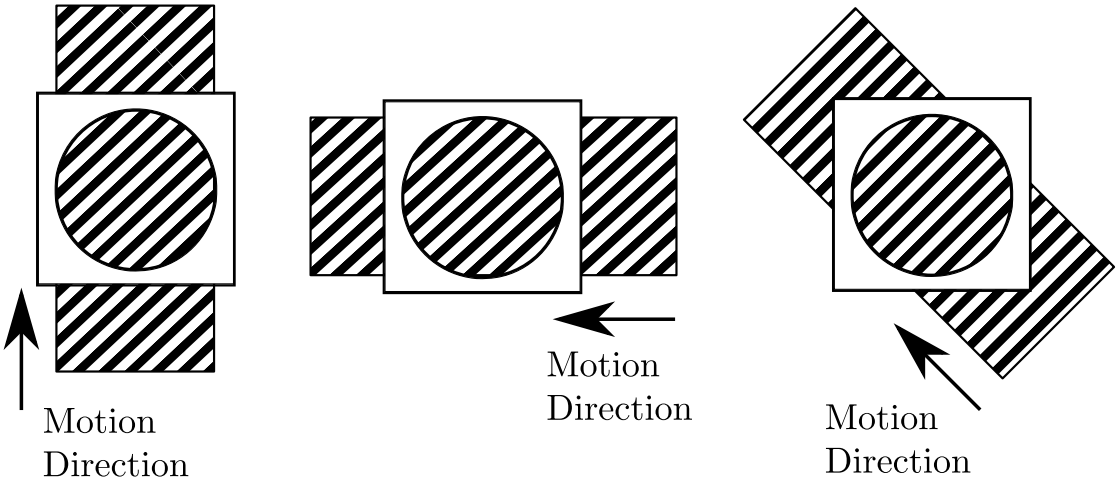


Figure 1.10: Three examples to highlight the aperture problem. The arrows indicate the actual movement direction of the object, but if we only watch through the cut-out hole, then all examples would look like they would move in the same direction.

Nevertheless, even if the actual motion is not determinable, the apparent motion is still a valuable property, and consequently, several approaches have been proposed over decades which apply different constraints and assumptions to uniquely determine optical flow.

Lucas and Kanade [5, 4, 2, 3] proposed one of the first popular optical flow methods, which follows movement by warping and matching local image patches between the images of a video sequence. They assume that all pixel in the local patch move in the same direction as well as that the pixels do not change their colour values, which is also referred to as the *photometric consistency* assumption. The Lucas-Kanade tracking method is quite effective for image regions like corners, but it fails for homogeneous image regions like walls. Because images usually have large areas of similar colour values, this means that for most of the image the method does not work. Because of this, Lucas-Kanade type methods are also referred to as *sparse optical flow* methods.

To estimate the motion for all pixels, Horn and Schunck [31] introduced variational optimisation methods to optical flow estimation. Instead of following the movement of a local patch like Lucas and Kanade's method, Horn and Schunck estimate the per-pixel movement but assume that nearby pixels move in a similar direction. In this thesis, we refer to this assumption as the *smoothness assumption* which allows us to "fill in the

gaps" in the estimation, where measurements are missing or degenerated. Because of the fill-in ability, Horn-Schunk type methods are sometimes referred to as *dense optical flow* methods. Because smoothness and photometric consistency are important properties for the work in this thesis, we provide more details about them here. Formally, the residual of the photometric consistency can be described as

$$\rho(\mathbf{u}) := I'(x, y) - I(x + u, y + v), \tag{1.13}$$

where I, I' are the functions of two consecutive images in a video sequence, which return the intensity values for given x, y positions in the image and u and v are the optical flow estimates. From Equation (1.13) we can see that this equation for unknown u and v is under-determined, which relates to the above-described aperture problem. Horn and Schunk apply the smoothness assumption to constrain Equation (1.13), which uniquely determines the problem. In Figure 1.11b an example of the optical flow results is shown, in which the direction of the motion is visualised by a per-pixel colour-coding.

Formally, Horn and Schunk's approach describes a minimisation problem, but, for convenience, we first define

$$\nabla \mathbf{u} = \nabla(u, v) = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}\right)^T, \tag{1.14}$$

$$|\nabla \mathbf{u}| = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2}, \tag{1.15}$$

where $\mathbf{u} : \Omega \mapsto \mathbb{R}^2$ is our optical flow function, which maps each x and y position of our image domain Ω to the optical flow vector. We can then define the Horn-Schunk minimisation as

$$\min_{\mathbf{u}} \left\{ \frac{1}{2} \int_\Omega |\nabla \mathbf{u}|^2 \mathrm{d}\Omega + \frac{1}{2\lambda} \int_\Omega |\rho(\mathbf{u})|^2 \mathrm{d}\Omega \right\}, \tag{1.16}$$

where ρ is our photometric error as defined by Equation (1.13) and $\lambda > 0$ is a scalar weight. For variational minimisation, Equation (1.16) is similar to the Tikhonov formulation in Equation 1.5. In Figure 1.11b, we see an example of using Equation (1.16).

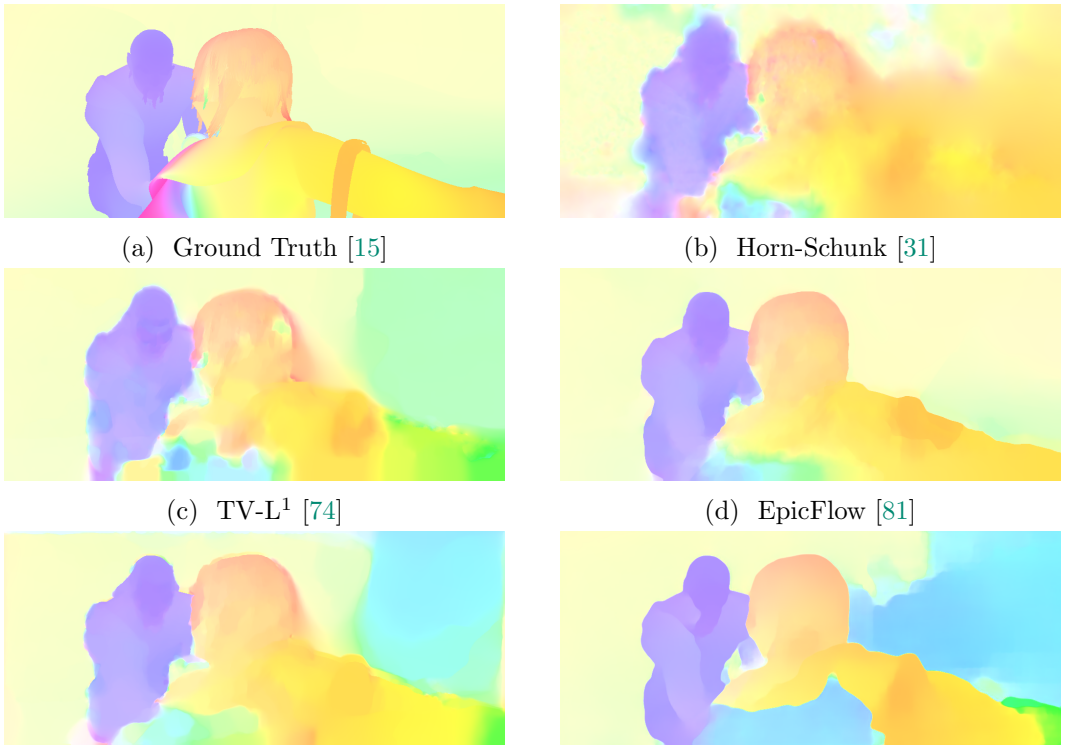
While the solution of Equation (1.16) provides a dense estimate for every element of the image, Horn and Schunk also report that the method fails at object boundaries, which is visible in the comparison in Figure 1.11. In computer vision, the object boundary is the separation between an object to the background or other objects in the image and is the area where the smoothness assumption, in general, does not hold. The assumption is violated at the boundary because objects move independently from each other, which means that, for example, an object may move in the image in the opposite direction than the background. However, Equation (1.16) assumes a Gaussian similarity between the flow vectors, meaning that most flow vectors should move in the same direction as their neighbours. However, at object boundaries, the direction changes may be drastic as well as frequent.

To address this issue, but also not to increase the computational burden, the formulation of Rudin, Osher and Fatemi (ROF) [87] and Total Variational minimization, which we introduce in the previous section, were proposed as alternatives to Equation (1.16). We highlight the performance of TV-L^1 on optical flow estimation can be seen in Figure 1.11.

While variational minimisation is a useful tool for computer vision, it also creates a computationally intense problem because it requires solving a per-pixel minimisation. For this reason, dense optical flow was for a long time only solvable in real-time for small images, which changed with the introduction of *Graphic Processing Units* (GPUs). Pock *et al.* [74] showed in their work that variational problems can be efficiently parallelised on a GPU.

Beyond variational methods, other optical flow approaches estimate object boundaries to account for them during the optical flow estimation. For example, Revaud *et al.* [81] used in their work edge detection methods to preserve the motion discontinuity in those regions. However, often these methods increased significantly the computational burden, which makes them not applicable in real-time scenarios.

In recent years, optical flow estimation also became a subject area for *deep learning* (DL). DL has been successfully applied to a range of problems in computer vision and beat many of the previous non-learning-based methods. Here the focus is less on the design of different cost function, but on the architectural setup and creation of training



(a) Ground Truth [15] (b) Horn-Schunk [31]

(c) TV-L^1 [74] (d) EpicFlow [81]

(e) FlowNet [23] (f) FlowNet2 [34]

Figure 1.11: Comparison of different optical flow methods on the MPI Sintel Flow Dataset benchmark [15]. The movement direction of each pixel is colour coded using the hue wheel as defined by the Sintel dataset.

sets, which allows a network to learn optical flow. Fischer *et al.* [23] and the subsequent work of Ilg *et al.* [34] showed that DL methods are indeed capable of estimating optical flow as well as achieving state-of-the-art performance. However, they do not clearly outperform the previously handcrafted methods, but learning-based optical flow methods are used in various other DL task to support the training. It is still an active research area, and it remains to be shown if DL methods are capable of surpassing the previous handcrafted methods

1.5 Deep Learning for Natural Image Generation

In recent years *deep learning* (DL) has become the method of choice to address many vision problems in computer vision areas like object recognition, semantic segmentation

and many more. These DL methods outperform many of the previously handcrafted state-of-the-art methods. Part of the success of DL methods seems to lie in their ability to express complex image statistics, which is exemplified by the performance of generative DL methods. Here several advances have been made in recent years using *deep neural networks* (DNN), which culminate with *generative adversarial networks* (GAN) [26] being capable of generating compelling looking images. We give here a brief overview of these generative DL methods.

Formally, generative methods model a joint distribution between latent variables and observations, with a DNN learning a mapping function between those variables. Two prominent approaches for DNNs emerged for this: *Variational auto-encoders* (VAE) [42], and the above mentioned GANs.

In the following sections, we introduce the general concept of VAEs and GANs, but in favour of brevity, we are omitting here details about DL and DNNs. We provide details about them in Chapter 5, however, for this section, we treat DNNs as generic function approximators.

Before we introduce VAEs and GANs, we first introduce the machine learning (ML) concept of *training* and *testing*, to provide a better understanding of the ML methodology in general. Training and testing are two phases of an ML method which separate optimising the model parameters from the evaluation of the model. In other words, if we test a DNN, we are fixing the neuron weights and use a dataset which is different from the training set which is used to adapt the weights. The reason for this separation is that we like for the DNN to generalise, which means that we wish that a DNN does not only works on training examples but also on previously unseen tasks. In more condensed form: Training adapts the network weights while testing fixes the weights and we test if the network does what we want.

For VAEs and GANs training and testing are separated by first learning the mapping from the latent variables to a dataset of given observations and then evaluating if the network is capable of generating new observation from samples of random latent variables.

Figure 1.12: Variational autoencoder results trained with the CelebA dataset [54]. The results are taken from the experiments provided by Hou *et al.* [32].

1.5.1 Variational Autoencoders

VAEs and GANs are the most commonly used generative methods for deep learning, and we introduce VAEs here before continuing the discussion about GANs, which we use in later chapters. VAEs have been proposed by Kingma and Welling [42] to learn a parametrised latent distribution from which novel observable variables can be sampled. To learn such a latent distribution, VAEs use a training dataset of observable variables. VAEs can learn, for example, from images of faces a latent distribution which then can be used to improve the feature representation of face recognition and other applications.

A VAE network represents the latent distribution by a set of neurons which represent the distribution parameters, such as mean and variance for a Gaussian distribution. The samples from such a latent distribution can be mapped to observable variables (e.g. images of faces) by a decoder network, which is part of the VAE network.

For VAEs Kingma and Welling proposed using an *autoencoder* (AE) hourglass network structures, which we detail in Section 5.4. An AE is composed of encoder and decoder network, which, in the case of VAEs, map to and from the latent distribution. The encoder generates the latent distribution parameters from which samples can be are drawn, while the decoder maps these samples to observable variables. Formally, we can

express the encoder E and decoder D networks such that:

$$x_i = D(z_i), z_i \sim p(z|E(I_i)), \tag{1.17}$$

where I_i is a sample from the observable distribution (e.g. an image of a face) and $p(z|E(I_i))$ is the latent distribution which is parameterised by the output of E.

A VAE is trained by reconstructing I_i from the latent sample, while also maximising the similarity of latent distribution to a parameterise distribution (e.g. Gaussian distribution). The similarity of the distributions is enforced by maximising the Evidence Lower Bound Loss, which allows using the encoder output as parameters for the latent distribution. The Evidence Lower Bound Loss is detailed in [42].

We show examples of this process in Figure 1.12 which shows examples from the evaluation by Hou *et al.* [32]. The results show the capacity of VAE networks to generate novel faces with a variety of facial features, details of hair and background. However, while the results have a natural appearance, many details are lost because of the lack in sharpness. Nevertheless, VAEs showed that neural networks are capable of generating naturalistic images, but the quality of such generated images was advanced by the GANs, which we present in the next section

1.5.2 Generative Adversarial Networks

Given a jointed distribution of observable variables and latent variables, GANs were proposed by Goodfellow *et al.* [26] to learn the distribution of the observable variables from a given distribution of latent variables and to generate novel observable variables. For a set of images like images of faces, this means that the GANs takes those images as observables variables and learns to generate new faces by learning to generate faces from random variables. Usually, for GAN methods these random variables are drawn from uniform or normal distribution. The core idea of GANs is to use another network, called a *discriminator*, during training to evaluate the generated observations. The discriminator's primary task is to identify the generated observable variables and samples from a real observable distribution correctly. This task provides the generating network, or *generator*, with a measure of similarity between the samples of both distributions and the means to increase the similarity between its samples and samples from the real

(a) GAN results (2016) [77]

(b) GAN results (2018) [37]

Figure 1.13: Improvements of GAN results in recent years. Both method have been trained on the CelebA dataset [54].

distribution. A real-world metaphor for this training is the criminal act of money forgery: A criminal (the generator) creates false currency, while a detective (the discriminator) finds the false money to best the criminal. With practice, the detective gets better at finding the false bills and coins which forces the criminal to create more convincing forgeries. This improvement, in turn, forces the detective to advance his skills, which creates a back and forth between detective and criminal which eventually leads to an equilibrium between them. An equilibrium in this example would be a stable rate of false currency which the detective cannot find, although he is optimally trained, and the criminal has no means to improve the forgeries further. For network training, this

ideal case scenario means that the generator is then capable of creating samples which are indistinguishable from real samples. We provide a more theoretical introduction for GANs in Chapter 5.

GANs gathered much attention from the computer vision community with their capacity of generating natural images. In Figure 1.13, we compare the results of two GAN methods which have been proposed recently to highlight the development in this field. Comparing these results to the performance of VAEs in Figure 1.12 shows that GANs seem to have the capacity to express complex natural statistics. This capacity inspired us to investigate the use of GANs for the event camera in later chapters.

1.6 Contributions

In this thesis, we look for an interpretation of the event stream which is not reliant on restrictions and assumptions to understand the motion or appearance of the scene. Those restrictions prevent the use of event cameras for a wide range of robotic vision tasks, and in this work, we therefore present event-based methods which rely only on the event data for their analysis. We hope that this work will bridge the gap further between event-based and frame-based machine vision and to provide a basis for future event-based vision research.

For this analysis, we estimate the image appearance and motion while preserving the HDR and high-speed aspects of the event camera, while providing the solution as close to real-time as possible. Our methods, which we present in this thesis, utilise the parallel processing capacity GPUs to reduce computation time. In computer vision, the use of GPUs allows real-time processing of large and computationally intensive systems like DTAM [63], which motivates our approaches. We hope our approaches will provide the understanding of the conditions for general interpretations of the event stream and, in future, that they inform the development of other, more power-efficient event-based algorithms.

In the first part of this thesis, we analyse the motion perceived by an event camera, which is fundamental for many structure-from-motion algorithms in computer vision, and it allows us to establish correspondences between images. The goal is to understand the per-pixel motion using optical flow in the same way as this is defined for frame-based

cameras, which is different from previous the event-based approaches, which estimate optical flow by matching events with events. Instead, we utilise the measured intensity differences from the event data as the basis for our flow estimation, which in turn allows us to estimate the image appearance as well. We show that such coupled estimation is not only feasible but that it enables dense optical flow estimation with a complete reconstruction of the image as well. Our approach is the first work which also shows that a monocular event camera contains enough information for robotic vision even in dynamic scenes with moving objects and free camera motion. However, we also discuss the limitations of the methods, especially at object boundaries.

These shortcomings motivate the second part of the thesis which explores machine learning techniques and applies them to the analysis of the event stream. The techniques have been very successful in many computer vision areas, but there is not a clear approach for the application to event cameras, which is caused by the lack of event data training sets. Training sets are ground truth labels per frame for a camera, but the frame-less event signal defining such labels is non-trivial. However, applying machine learning to event cameras is promising because of their capabilities to express complex image statistics. For this reason, we investigate unsupervised methods for image reconstruction and estimation, which allows analysing the event stream with a neural network without the need for labelled ground-truth datasets.

1.7 Publications

The body of work in this thesis is based on two approaches: Variational estimation and deep learning. Our event-based variational method is based on the publication:

P. A. Bardow, A. J. Davison and S. Leutenegger. **Simultaneous Optical Flow and Intensity Estimation from an Event Camera**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Our event-based learned approach is based on the prepared publication which is about to be summited to the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019 under the title “Can Event Data be Interpreted using only Natural Image Priors?” by P. A. Bardow, S. Leutenegger and A. J. Davison.

1.8 Thesis Structure

Here we provide an overview of the chapters in this thesis and their contents. After this chapter, Chapter 2 introduces preliminary mathematical conventions on notations and concepts which are used throughout this thesis. In that chapter, we will also provide a detailed introduction to event cameras and to their bio-inspired design.

In Chapter 3, we introduce the primal-dual algorithm which efficiently solves variational minimisation problem which we later use for our event-based analysis. In that chapter, we provide the theoretical background of the primal-dual algorithm and discuss its application to optical flow and image denoising for standard camera images.

Next in Chapter 4, we introduce our first contribution for event-based motion analysis. Our contribution applies the primal-dual methods to the event-based signal and we will present a series of experiments in which we show that this method is capable of estimating optical flow as well as reconstructing images from the event camera signal.

In Chapter 5, we then introduce generative machine learning methods which are capable of learning complex natural image statistics. In that chapter, we also discuss approaches which combine these generative methods with handcrafted reconstruction losses, which we apply on the event signal in Chapter 6.

In Chapter 6, we present our learning-based contribution for image reconstruction from event data. In contrast to our work in Chapter 4, our learning-based methods does not require a coupled optical flow estimation but is able to reconstruct images alone. In contrast to common deep-learning methods, our method also does require ground-truth data for training, which avoids the cumbersome process of generating such a dataset. In that chapter, we show in a series of experiments the capacity of our method for image reconstruction in different scenarios.

In Chapter 7, we will provide a comparison between our first contribution from Chapter 4 to our learning-based method from Chapter 6. We introduce in that chapter two optical flow estimation methods which utilise our learned reconstruction method to highlight its benefits on the event-based motion analysis. In a series of experiments which highlight the advantages and disadvantages of both contributions.

In the last chapter, we then conclude by laying out potential prospects of future event-

based research based on the insights we gained from the body of work in this thesis.

Chapter 2

Preliminaries

Contents

In this chapter, we introduce theoretical and technical concepts which are used throughout this thesis. In the theoretical part, we present the mathematical notation and methods, while in the technical section we discuss the event camera interface which we use for our work.

2.1 Mathmatical Notation

In this section, we introduce preliminary notation, which we are using throughout this work if not stated otherwise. For scalar variables, we use the convention to indicate them by lowercase letters (a, b, ..., z) and m-dimensional vectors by boldfaced lowercase letters such as:

$$\mathbf{v} = (v_1, v_2, \ldots, v_m)^T, \tag{2.1}$$

where $v_i, i \in \{1, \ldots, m\}$ is the i-th coefficient of the vector. $m \times n$ dimensional matrices are denoated by uppercase letters such as:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \ldots, & A_{1,n} \\ A_{2,1} & A_{2,2} & \ldots, & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \ldots & A_{m,n} \end{pmatrix} \tag{2.2}$$

where $A_{i,j}, i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$ is the coefficient of the matrix at i-th row and j-th colum.

A function $F : \mathbb{R}^n \mapsto \mathbb{R}$ which maps to a scalar is called a *scalar field* and $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ which maps to a vector a *vector field*. The first derivative of a scalar field F is a vector field such as $\nabla F : \mathbb{R}^n \mapsto \mathbb{R}^n$, where ∇ denotes the gradient operator which is defined as:

$$\nabla F(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} F(\mathbf{x}) = f'(\mathbf{x}) = \left(\frac{\partial F(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial F(\mathbf{x})}{\partial x_n} \right). \tag{2.3}$$

Furthermore, the first derivative of a vector field f is called the *Jacobian* which is

defined as:

$$\nabla f(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{2.4}$$

Therefore, the gradient of a scalar field can be seen as a special case of the Jacobian matrix when $m = 1$. The second derivative of the scalar field F maps an n-vector onto a $n \times n$ matrix, which is called the *Hessian* which is defined as:

$$\nabla^2 f(\mathbf{x}) = \frac{\partial^2}{\partial \mathbf{x}^2} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1^2} & \frac{\partial^2 f_1}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 x_n} \\ \frac{\partial^2 f_2}{\partial x_1 x_2} & \frac{\partial^2 f_2}{\partial x_2^2} & \cdots & \frac{\partial^2 f_2}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_n}{\partial x_1 x_n} & \frac{\partial^2 f_n}{\partial x_2 x_n} & \cdots & \frac{\partial^2 f_n}{\partial x_n^2} \end{bmatrix}. \tag{2.5}$$

The second derivative of a vector field f maps to a three-dimensional array or third-order *tensor* of size $n \times m \times n$.

The scalar product or inner product between two vectors is denoted by using the $\langle ., . \rangle$ operator which is also known as the dot product such that:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{m} x_i y_i. \tag{2.6}$$

The divergence operator is defined as:

$$\text{div} u = \nabla \cdot u = \frac{\partial u}{\partial x_1} + \ldots + \frac{\partial u}{\partial x_n}, \tag{2.7}$$

which can be viewed as the scalar product between the vector and the gradient operator ∇.

2.1.1 Taylor Series

The Taylor Series is a commonly used for approximating the neighbourhood of a function $f : \mathbb{R} \mapsto \mathbb{R}$ at point $(a, f(a))$. Assuming f infinitely differentiable then the series

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots = \sum_{k=0}^{\infty} \frac{f^{(n)}(a)}{k!}(x-a)^k \tag{2.8}$$

is called the Taylor series of f at a, where $f^{(n)}$ is the n-th derivative of f. In practice, we use the Taylor series to approximate a function f in the neighbourhood of a linearisation point a using a finite series

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \ldots \frac{f^{(n)}(a)}{n!}(x-a)^n = \sum_{k=0}^{n} \frac{f^{(n)}(a)}{k!}(x-a)^k. \tag{2.9}$$

In different applications, the Taylor series approximation is used because it often allows to simplify the problem rather than using f directly and if the neighbourhood is close enough then the discrepancy between it and the original function is tolerable.

The Taylor series generalises to high-order functions, for instance, the second-order Taylor expansion of a scalar field F is defined as

$$F(\mathbf{a}) + \langle \nabla F, \mathbf{x} - \mathbf{a} \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{a})^T \nabla^2 F(\mathbf{a})(\mathbf{x} - \mathbf{a}). \tag{2.10}$$

2.1.2 Optimization

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a scalar field. In an optimisation problem, our objective is to find the minimum of a function f such as

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \tag{2.11}$$

Assuming that f is differentiable and if $(\mathbf{x}^*, f(\mathbf{x}^*))$ is a local optimum, then $\nabla f(\mathbf{x}^*) = 0$ which is known as the *necessary condition*. For the sufficient condition, if $\nabla f(\mathbf{x}^*) = 0$

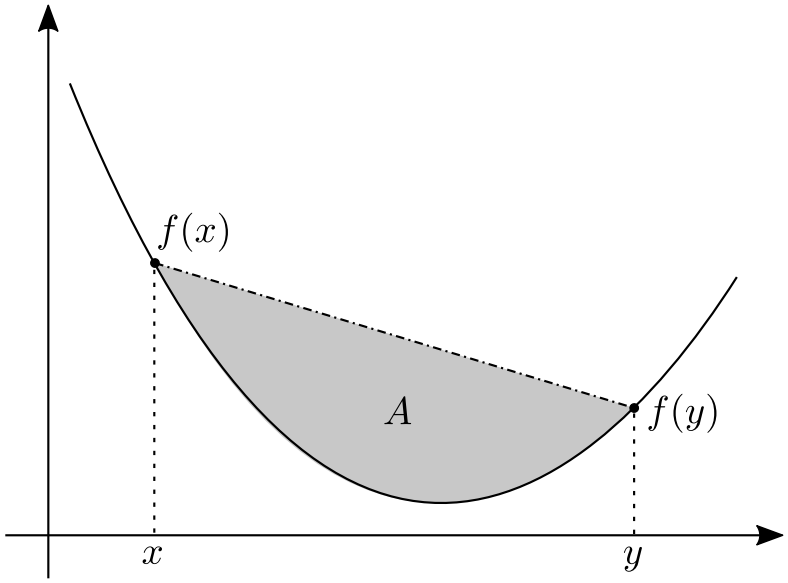


Figure 2.1: Example of a convex function where the area A must lie underneath a line drawn between any selected pair of points x and y.

and $\nabla^2 f(\mathbf{x}^*)$ is *positive definite* such that $\mathbf{y}^T \nabla^2 f(\mathbf{x}^*)\mathbf{y} > 0$, then $\mathbf{x}^*$ is the position of a local minimum.

Convexity

Convexity is an important property in optimization to which we often refer to in this work. A function $f : \mathbb{R} \mapsto \mathbb{R}$ is said to be convex if the following inequality holds

$$f(t\,x + (1-t)y)) \leq f(t\,x) + (1-t)f(y), \forall x, y \in \mathbb{R}, \forall t \in [0,1] \tag{2.12}$$

Furthermore, f is called *strictly convex* if

$$(t\,x + (1-t)y)) < f(t\,x) + (1-t)f(y), \forall x \neq y \in \mathbb{R}, \forall t \in (0,1) \tag{2.13}$$

Figure 2.1 shows an example of an convex function and visualises the condition (2.12). The condition holds if any line drawn from point $(x, f(x))$ to $(y, f(y))$ is above any point of f between x and y.

In optimisation, convexity is an important property which permits optimisation methods to find the globally optimal point from a random initial point in polynomial time using only the local gradient. Without this property, the same methods can only be guaranteed to find a local optimum while finding the global optimum requires more sophisticated methods. However, in this work, we focus on gradient-based methods which we briefly introduce in the following.

Gradient Descent

Different methods have been developed for finding the minimum of a function f; the most commonly used method is gradient descent. Gradient descent is an iterative process which starts from a given point $\mathbf{x}_0$ and from that point, the method steps along the direction of the negative gradient $-\nabla f(\mathbf{x}_0)$ to a point $\mathbf{x}_1$ such that $f(\mathbf{x}_1) < f(\mathbf{x}_0)$. This process is repeated until a lower point of the function cannot be reached. Thus, we can define the update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \tag{2.14}$$

where $k > 0$ is the current iteration step and $\alpha_k > 0$ is the step-size which is chosen such that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$. Since all points of the sequences $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2 \ldots$ are valid elements in the domain of f, gradient descent is an interior point method [12].

Stochastic Gradient Descent

In deep learning, often the computational problem arises that calculating the actual gradient of an artificial neural network is costly in terms of time and memory usage due to the number of training parameters and size of datasets which these methods require. Neural network models usually have a large number of parameters as well as prediction outputs which are matched against the training examples which results in a very large Jacobian matrix. Such a matrix is often too large to be efficiently computed or applied even for gradient descent. However, in practice it has been proven to be feasible to use an approximate method known as *stochastic gradient descent* (SGD) [82] instead. Let us assume that our problem F which we wish to minimise can be separated into n

summands such that:

$$F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} F_i(\mathbf{w}), \tag{2.15}$$

where $\mathbf{w}$ is the network model parameter which we wish to optimise. The individual summands can be viewed as individual training examples. For the network training, we follow the following update-rule:

Stochastic Gradient Descent

- For initialization: Choose parameter ω_1, learning rate $\tau > 0$ and set $k = 1$.
- Repeat until convergence:
  - Randomly shuffle the order of the data w.r.t. the summands of F
  - For $i = 1, 2, \ldots n$ do:
    - $\mathbf{w}_{k+1} = \mathbf{w}_k - \tau \nabla F_i(\mathbf{w}_k)$
    - Set $k := k + 1$

However, since the update of SGD is based on only one example from the training set for each step, convergence can be slow and the gradient unstable, especially in non-linear cases. A compromise between computing the actual gradient and the SGD approach is to compute the gradient with respect to sampled *mini-batches*. The mini-batch approach uses the gradient of the expectation $\mathbb{E}[F]$ rather than of F itself, such that

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \tau \nabla \mathbb{E}[F(\mathbf{w}_k)], \tag{2.16}$$

where $\tau > 0$ is the step size or *learning rate*. The expectation is computed by randomly sampling m examples to form a batch and weighting them by their likelihoods. In practice, the samples are uniformly chosen which makes each sample similarly likely. The gradient is then easily computed by computing and then averaging all gradients with respect to the summands of the mini-batch. Since m is chosen to be much smaller than n, the approximated gradient is more efficiently computed than the true gradient.

For brevity, from now on we refer to mini-batch SGD when we mention SGD since it is the most commonly used network training method in the literature.

Lagrangian Relaxation

Another important concept for optimisation problems is *Lagrange multipliers*. Let $f : \mathbb{R} \mapsto \mathbb{R}$ be a function we wish to minimise and $g_i : \mathbb{R} \mapsto \mathbb{R}$ $i = 1, \ldots, m$ be constraints on the solution such that our optimisation problem is:

$$\begin{aligned} &\min_x f(x) \\ &\text{subject to } g_i(x) = 0, \quad i = 1, \ldots, m, \end{aligned} \tag{2.17}$$

where g_i are called the *equality constraints*. To minimise (2.17), we relax the problem by introducing $\lambda_1, \ldots, \lambda_m$ such that:

$$\max_{\lambda_1, \ldots, \lambda_m} \min_x \mathcal{L}(x, \lambda_1 ..., \lambda_m) := f(x) - \sum_{i=1}^{m} \lambda_i g_i(x) \tag{2.18}$$

A point x^* is a local optimum of if it minimises f as well as satisfies the constraints such that:

$$g_i(x^*) = 0, \text{ for } i = 1, \ldots, m.$$

The Lagrangian links the primal formulation (2.17) and the dual formulation, which we often refer to in this thesis. The solution of a dual problem provides the lower-bound to the solution of the original primal problem. The dual formulation f^* of a primal problem f is defined such that:

$$f^*(\lambda_1, \ldots, \lambda_m) := \inf_x \left(f(x) - \sum_{i=1}^{m} \lambda_i g_i(x) \right). \tag{2.19}$$

As stated above, maximising f^* gives the lower-bound of f, which provides insight into the primal problem and in certain cases, the solution f^* is easier to achieve than the

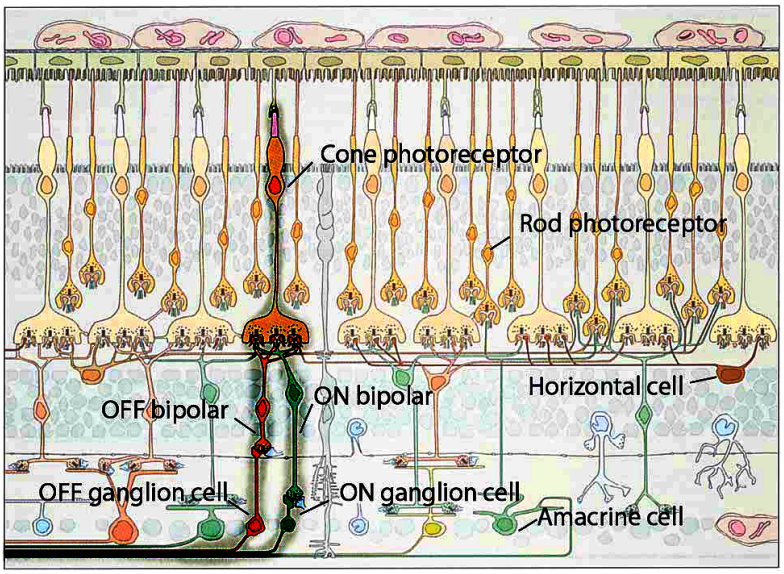


Figure 2.2: Biologial retina neural layer structure which inspired the design of event cameras. The Figure is taken from [76].

primal problem. The easier solution motivates later discussion in this thesis, however, the solution of the primal and dual problem is not necessarily equal. The difference between the solutions is known as the *duality gap*, however, for convex the solutions are equal [12].

2.2 Neuromorphic Silicon Retina

In this section, we are introducing the workings of the event camera. Since it is a relatively new sensor, this section should provide the basis for understanding the mechanics to readers who are unfamiliar with the camera.

In biological organisms, the retina of vertebrates like humans is a thin sheet of tissue on the inner surface of the eye which converts light into an electrical pulse. It is composed of a multilayer neural network which started to evolve about 600 million years ago as light-sensitive neural cells and gained complexity and sophistication during a long evolutionary process. As shown in Figure 2.2 the retina is a layered and complex structure which can be separated into three parts: Photoreceptor layer, outer plexiform layer and the inner plexiform layer [45, 56, 83].

The photoreceptor layer is composed of rod and cones which are neural cells which transform the received light into an electrical pulse which passed on to the horizontal and bipolar cells in the outer plexiform layer. The bipolar cells are divided into two types of cells called *on*- and *off*-cells, which separately encode spatiotemporal brightness changes which indicate if the amount of light is increasing or decreasing. The contrast is measured by comparing the photoreceptor signal to the averages accumulated by the laterally connected layer of horizontal cells. The horizontal cells form an interconnected mesh with each other as well as with bipolar cells and photoreceptors. With the input from the photoreceptors, the cells produce low-pass copies of the input signal which is fed back to the photoreceptors to support their process.

The interconnections of the outer plexiform grow in complexity at the inner plexiform layer which contains many types of amacrine and ganglion cells, which subdivides the neural signal into many different pathways. Understanding these connections and their workings is still ongoing research on the visual perception of the brain and its role in the cognitive process, and even though the reaction of the individual retinal neurons is slower and lacks the precision of the transistors of electronic systems, it still outperforms any artificial system with respect of extracting all essential visual information rapidly and reliably even in difficult light conditions while consuming minuscule amounts of energy.

Such biological retinas have many desirable properties which make them potentially superior to conventional image sensors. For instance, the bio-inspired design offers a redundancy suppressing encoding scheme with a wide dynamic range and a temporal resolution which is far greater than conventional frame-based sensors. These properties inspired the design of silicon retinas, which emulate the receptive properties of their biological counterparts. This development succeeded in 1991 with Mahowald and Mead [55] emulating the first three biological layers in silicon and demonstrate the same output signals observed in biological systems. In their work, each photoreceptor of the silicon retina mimics a cone cell and contains adaptive circuitry which adjusts to changing light levels.

A variety of neuromorphic vision sensors have been developed since then, like the sensor developed by Zaghloul and Boahen which mimics all layers of a retina using a simplified model [103]. While the sensor by Mahowa and Mead models only the cones, horizontal cells and bipolar cells of the outer retinal layers, Zaghloul and Boahen ad-

ditionally implement adaptation to light and contrast as well as adaptive spatial and temporal filtering. The sensor, and others like it, have complex and interesting properties. However, they also contain large pixel mismatches in their response characteristics which makes these cameras difficult to use. Furthermore, the pixel design of those sensors lacks practical technical relevance because of its complex and large circuitry coupled with high levels of noise influences [76]. The sensor complexity is caused by the variety of different cells in the retina, which are have been clustered into 20 to 50 different types of cells in the biology literature [56, 98]. While determining the exact function of these cells is still a research subject, the cells can be roughly categorised by their operations into the following functionalities:

- cells which are sensitive to transients in luminance,
- cells which are sensitive to sustained luminance, which adjusts other cells depending on the ambient light,
- cells sensitive to specific directions of motion,
- cells sensitive to spatial contrast in their local region,
- cells which are sensitive to specific wavelengths of light which provide colour vision.

Many early biological inspired sensor designs focused on mimicking the retina as closely as possible for the purpose of demonstrating neurobiological models and theories without having a real-world application in mind. For more practical applications, since the emergence of these early sensors, other types of silicon retinas have been proposed to reduce the circuitry complexity by only concentrating on some of the above-mentioned cell categories as well as combining them with conventional, frame-based sensors which make the application of traditional machine vision easier. We introduce here the details of some of these sensors, but we refer to the literature for a more thorough literature review [76].

Since the first sensor was developed by Mahowa and Mead, a spectrum of different bio-inspired sensors with varying degrees of biological plausibility was created with different modalities. To increase practicality, some sensors focus on emulating specific elements of the retina; for instance, spatial difference or contrast sensors minimise spatial redundancy

by only encoding intensity differences [38] or ratios [48] with respect to the pixel layout. Another sensor, for example, focuses on temporal filtering cells which register temporal differences with respect to absolute [18] or relative [53] intensity changes over time. This focus on specific retinal elements helps to reduce the circuitry complexity and size of the sensor and makes the behaviour of the sensor more reliable. This specialisation includes high-level abstraction sensors which provide information like image gradients [93] or optical flow [90]. For the purposes of this thesis, we focus on temporal contrast type cameras as they are commercially available and therefore we refer to them when we use the term event cameras which respond to relative intensity changes over time. In the following, we detail these cameras further and compare them against standard frame-based camera sensors.

2.2.1 Comparison of Event and Frame-based Cameras

In order to understand the difference between common frame-based and event-based cameras, we discuss here their different signals which are highlighted in Figure 2.3. A frame-based camera sensor captures light from the scene in fixed intervals using a shutter to control the exposure. Figure 2.3 illustrates a stylised scene captured by a frame-based camera and compares it with the signal of an event camera. Both cameras look at a spinning disk with a black dot near the rim which indicates the movement of the disk. With the frame-based camera, we see a sequence of images with the disk at various positions. However, the frame-based camera does not capture the motion between the frames and we suffer from motion blur if the disk spins too fast.

In contrast to the frame-based signal, the event camera outputs a continuous stream of discrete signals, or events, where each event contains its pixel location, polarity and microsecond-precise timestamp. Each event indicates a change in log-intensity of an amount determined by a pre-set threshold, which is part of the parameter configuration of the sensor. Brighter and darker changes produce *positive* and *negative* events which are also referred to as *on* and *off* events by the manufacturer. By only registering intensity changes, the amount of bandwidth used by the camera is dramatically reduced compared to standard videos, however, the event data excludes an unknown constant which is the first initial frame of the sequence. Figure 2.3 illustrates the signal which observes the same spinning disk but observed by a fixed event camera and the corresponding stream of events. The stream shows the in-between motion which is missing from the frame-based

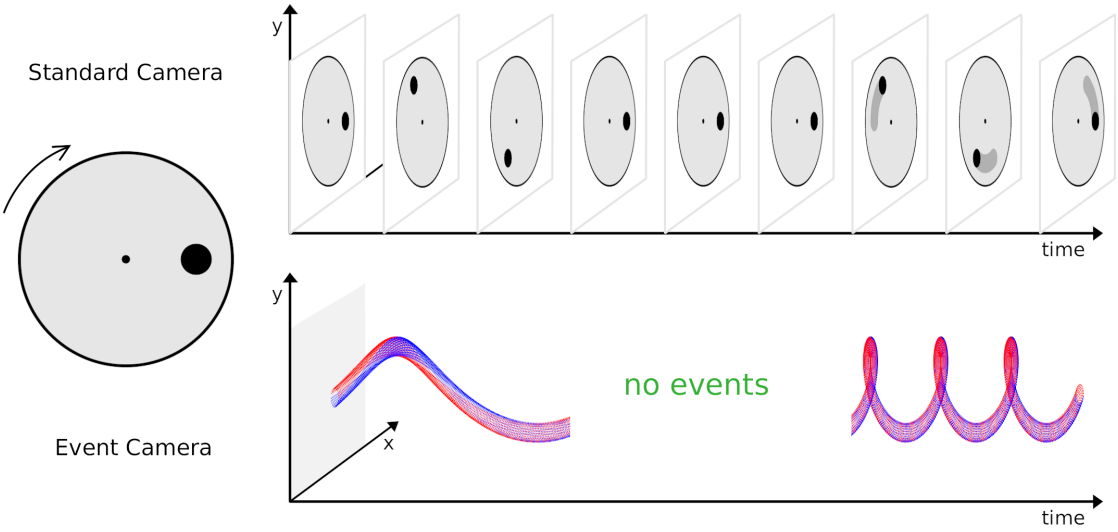


Figure 2.3: Event- and frame-based cameras capturing a spinning disk. The upper row shows the camera frames, where the first three show the spinning motion, in the next three the disk is still and in the last frames the disk is spinning faster which creates motion blur. The lower row shows the event stream capturing the in-between motion, but no signal when the disk is still. The figure is taken from the publication by Kim *et al.* [40].

sequence and illustrates the main property of the event signal: A continuous response to rapid motions which is not data-redundant but depends on the information in the scene. These properties offer the potential to outperform standard cameras and overcome their limitations in real-world applications which suffer from low dynamic range, low frame rate and high power demand. We now introduce details of the event camera interface.

2.3 Event Camera Interface

Although event cameras have become increasingly known within the computer vision community, a unified standard camera interface is up to this point missing, and therefore we introduce in this section the prequisites to communicate with an event camera via its USB interface. We focus here on the *Dynamic Vision Sensor* (DVS) [53] and the *Dynamic and Active Vision Sensor* (DAVIS) [13], which are both commercially available.

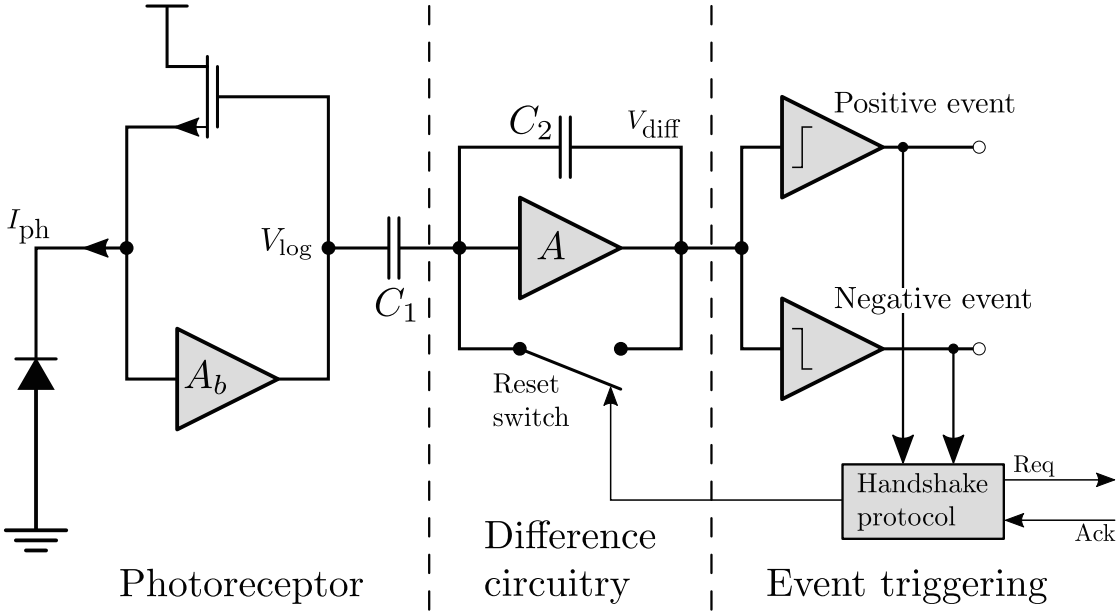


Figure 2.4: DVS circuitry with logical separation into three parts. The light measurement is processed from left to right, where the light is measured by the Photoreceptor and is then processed in the final stage by triggering events. This is recreated from the publication by Posch *et al.* [76].

2.3.1 DVS Pixel

Figure 2.4 shows the logical blocks of a DVS pixel circuit which we detail in the following. The DVS pixel is sectioned into three parts: Photometric measurements, difference circuitry and event triggering. The first stage converts the photocurrent to a voltage proportional to the logarithm of the light intensity:

$$V_{\log} = V_{\text{DC}} + A_v U_T \log I_{\text{ph}}, \tag{2.20}$$

where I_{ph} denotes the photometric current triggered by a light source, and A_v is the voltage gain factor which amplifies the signal. U_T is the thermal voltage which relates the system temperature to the electrical charge of the electrons, which means that noise-level of the photometric measurement is temperature dependent. V_{DC} denotes the direct current, which is a light-independent current which may differ for each pixel.

The second stage of the DVS pixel checks if the change in the current of the logarithmic

light is exceeding the event threshold, which is pre-set in the sensor configuration by the user. For the check, $V_{\log}$ is amplified by C_1/C_2 where the charge integrated at C_2 is reset every time $V_{\text{diff}} = (C_1/C_2)V_{\log}$ exceeds the event threshold $\pm V_\theta$ which is checked by the comparator. In the third DVS pixel stage, if the change in light exceeds the threshold then a signed asynchronous output, or event, is triggered. This condition can be formally expressed by conditioning that an event is triggered at time t_2 when the light changes since time t_1 by $\log I_{\text{ph}}(t_2) - \log I_{\text{ph}}(t_1) = \pm\theta$, with:

$$\theta = \frac{C_2 V_\theta}{C_1 U_T A_v}, \tag{2.21}$$

which can be interpreted as the minimum contrast which can be detected by a DVS pixel. However, in practice, this is limited by the noise of the photoreceptor and the bandwidth of the sensor. The contrast measured by the event camera can be viewed as a normalised temporal difference such as:

$$\theta = \left| \log \frac{I_{\text{ph}}(t_2)}{I_{\text{ph}}(t_1)} \right| \approx \left| \frac{\Delta I_{\text{ph}}}{I_{\text{ph}}} \right|. \tag{2.22}$$

Since $I_{\text{ph}} > 0$, events are triggered by smaller changes in low-light environments than in bright scenes.

2.3.2 DAVIS Pixel

The event camera is a novel sensor with a signal which cannot be trivially used by traditional machine vision algorithms. Vision algorithms rely on rich textural information which is missing in the event data. In order to bridge the gap between event-based and frame-based methods, hybrid sensors like the DAVIS have been developed with combined event-based and exposure-based pixels to provide the best of both sensors. Here we provide a brief overview of the mechanics of DAVIS.

Conceptually, the DAVIS is a straightforward combination of a frame-based and event-based sensor in which each pixel of both sensors shares the same photodiode. Figure 2.5 shows an example of the combined event and image signals of the DAVIS, in which the continuous event data is intersected by a regular camera image. The events shown were triggered between the previous DAVIS frame and the frame shown in the Figure.

Figure 2.5: Example of the combined output of the DAVIS. The events are superimposed over the monochrome camera image as red and green pixels for positive and negative events. The yellow pixels in the image are regions in which positive as well as negative events occured.

By sharing the same photodiode, the DAVIS design avoids the need for elaborate intersensor calibration and pixel alignment, which are usually required in camera stereo-rig setups. The natural sensor alignment allows using the same intrinsic camera parameter for both sensors, which means we can rely on well-proven frame-based calibration methods. Usually, for standard sensors, using the same photodiode for two sensors means that both sensors draw from the photocurrent which correlates the measurements of both sensors with each other. However, the DAVIS does not have this issue since the logarithmic photoreceptor of the DAVIS can continuously measure the photocurrent without consuming it [13]. This means that the photocurrent can be integrated over time and still produce an exposure measurement without interfering with the measurements of the event sensor. The extra readout logic increases the pixel area by 5% but it provides an output which is compatible with traditional machine vision methods. Figure 2.6 shows the logical blocks of the DAVIS pixel, where the event-based parts are similar to the DVS sensor.

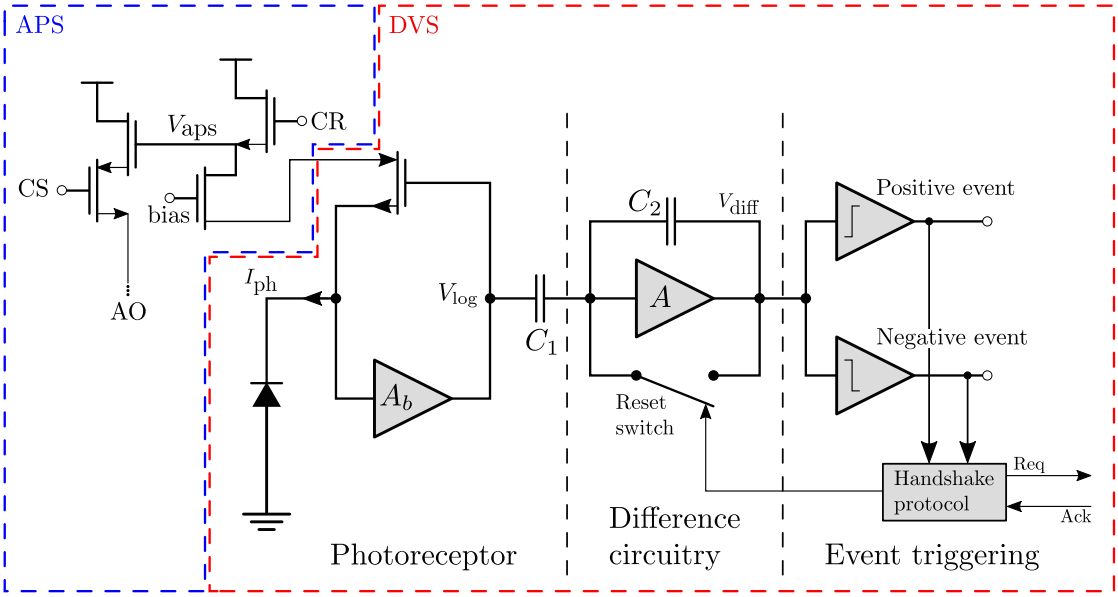


Figure 2.6: DAVIS circuitry with logical separation into the DVS and the APS (standard camera) parts. Note that event camera parts are similar to the DVS camera logic, however the photoreceptor shares the input from the APS section.

2.3.3 Address Event Representation

Before describing the data acquisition from the event camera, we describe first the read-out process of the sensor, which is complex due to the asynchronous and sparse signal. Comparatively, the read-out protocol of frame-based cameras is much simpler because it can exploit the fact that frames arrive at a constant rate, which is not the case for event cameras since their data-rate depends on the scene. If the scene and camera are still then no events are acquired, but if changes occur then they need to be registered as soon as possible before the next change happens. To address the dynamic behaviour of the sensor, a common strategy in event camera-type hardware is to use the *Address Event Representation* (AER) protocol.

The AER protocol was originally proposed by Mead's Caltech research lab [46], which entails a transmitter chip including an array of image pixels or artificial neurons. Each array element is assigned an address corresponding its to x, y-position in the image. The idea is that each array element generates a signal in a random fashion which is transferred on an inter-module high-speed asynchronous AER bus, implementing a strategy where all elements share the same physical bus to transmit their signals, coupled with timing

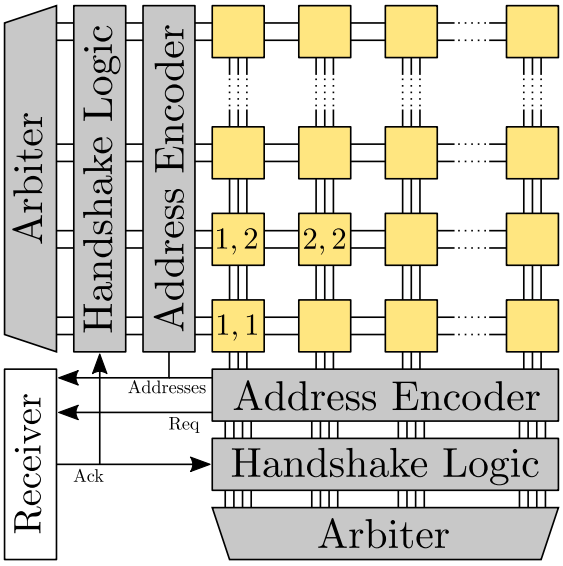


Figure 2.7: Block schematic of the pixel array embedded in the AER periphery. This is recreated from the publication by Posch *et al.* [76].

information. The temporal information corresponds to the arrival of each event and it is explicitly added to the address in the form of a timestamp outside the AER units. The on-chip periphery contains address encoders, arbiters and handshake circuitry, which implements a four-phase handshake with the data receiver. Figure 2.7 shows an overview of the AER system using two control signals *req* and *ack* to synchronise the data transfer through a bus system. The bus system connects a sender and a receiver via a four-phase handshake in which the sender asserts a req signal to notify the receiver that data was written onto the multiline data bus. The receiver then reads out the data and confirms by sending an ack signal which tells both sender and receiver to wait for the next transaction. The AER bus width and protocol varies between different sensor types. For example, the DVS uses a 15-bit bandwidth. However, typical delays for transmitting address events between AER chips range from 30 ns to 1 μs per event. For a more detailed description of the AER protocol, we refer to the literature [76].

2.3.4 DVS and DAVIS USB Interface

Usually, the signal from standard cameras is passed to an onboard processor for further manipulation like colour correction, compression and other tasks, however, for high-level computer vision tasks we usually wish to transfer the image onto a general processing unit

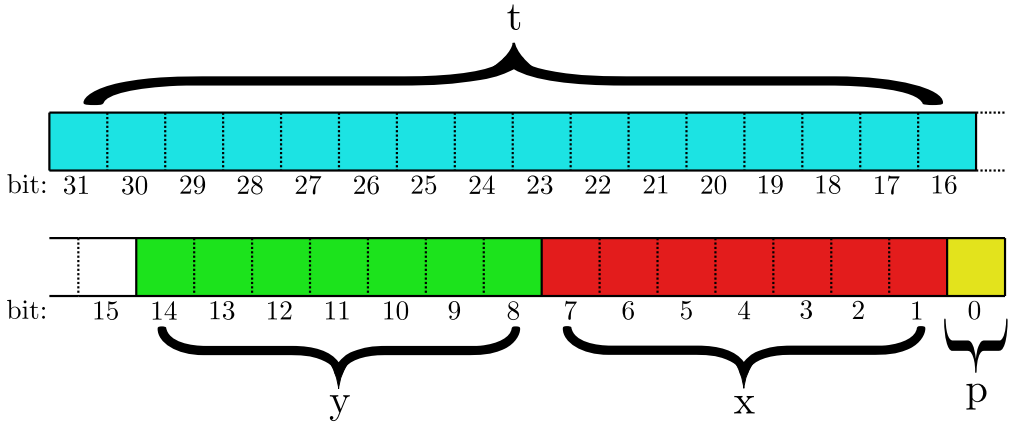


Figure 2.8: Bit layout encoding in a 32-bit array where x, y is the position in the image, p is event polarity and t is the timestamp in microseconds.

like a *personal computer* (PC) and analyse the data there. In this thesis, we transfer the signal from the event camera to a PC via a USB interface. For USB to event camera connections different solutions exist like the open source *jAER* software project which is written mostly in Java. However, since our methods are based on C/C++ and Python, we used initially for our experiments the custom build software from the work of Kim *et al.* [39] for the DVS and in later stages used the C library *libcaer* provided by the manufacturer for the DVS and DAVIS.

In contrast to the native AER event packet, each event transferred through the DVS USB interface is augmented with a microsecond resolution timestamp and encoded in a 4-byte long data packet as shown in Figure 2.8 with the position in the image, the polarity and timestamp of an event. The maximum bandwidth of the DVS128 camera is 1M events-per-second. For a more thorough description of the technical specifications, we refer the reader to the technical manual of the manufacturer.

2.3.5 DVS and DAVIS Biases

The DVS and DAVIS biases are on-chip parameters which determine the behavioural characteristics of the event signal. In hardware terms, the biases are voltages and currents which are kept at a stable level by the analogue circuit design of the sensor. Because of the analogue nature, this process is greatly influenced by noise, electrical leakage and so on, which needs to be considered by the design of a vision algorithm. Both the DVS

DVS Bias	Value	Description
pr	3	Photoreceptor: Controls the response speed to changes in light
cas	54	First stage (photoreceptor) cascode: Controls the speed the stability of the circuit
foll	51	Source follower: Separates the first and second (differential) stages
diff	30153	Differential: Controls the response speed to a change in the light-related signal
diffOn	482443	On event threshold
diffOff	132	Off event threshold
injGnd	1108364	Injected ground: controls the response speed to the acknowledge in order to reset
req	159147	Pull down for passive load inverters in digital AER pixel circuitry
refr	6	Refractory period: Defines the time period during which the pixel will be reset
PuY	16777215	Pull up on request from Y arbiter
PuX	8159221	Pull up on request from X arbiter
reqPd	16777215	Pull down on chip request

Table 2.1: DVS defined bias parameter with default values with default values, which we use throughout this work

and DAVIS define more than a dozen parameters with interdependent behaviour, which makes manipulating parameters and properly relating them to physical property challenging without calibration. We provide a list of the biases for the DVS in Table 2.1 and DAVIS in Table 2.2, which can be dynamically adjusted via the USB interface.

The DAVIS defines a number of experimental bias parameters, which are recommended by the developer to be left alone and we are omitting them here because of that recommendation.

2.3.6 Other Event-based Sensors

In the previous section, we introduced a detailed description of the DVS and DAVIS, however, these are not the only temporal contrast event sensor in this field. Another event-based sensor developed by Leñero-Bardallo *et al.* [48] has a 128×128 resolution

DAVIS Bias	Coarse Current	Fine Current	Description
PrBp	2	58	Photoreceptor: Controls the response speed to changes in light
PrSFBp	1	16	Source follower: Separates the first and second (differential) stages
DiffBn	4	39	Differential: Controls the response speed to a change in the light-related signal
OnBn	5	255	On event threshold
OffBn	4	0	Off event threshold
PixInvBn	5	129	Pull down for passive load inverters in digital AER pixel circuitry
RefrBp	4	25	Refractory period: Defines the time period during which the pixel will be reset
AEPuYBp	7	152	Pull up on request from Y arbiter
AEPuXBp	4	80	Pull up on request from X arbiter
AEPdBn	6	91	Pull down on chip request
ApsCasEpc	5	185	Cascode separating APS and DVS parts of the pixel
DiffCasBnc	5	115	Cascodes in differential comparator
ApsROSFBn	6	219	Source follower for column-parallel APS readout
LcolTimeoutBn	5	49	Timeout after a row event
apsOverFlowLevel	6	253	APS overflow level

Table 2.2: DAVIS defined bias parameters with default values, which we use throughout this thesis.

with a greater sensitivity than the DVS with respect to the detectable minimal contrast and a latency of $3.6\mu s$ while also reducing the pixel size of previously proposed sensors. These technical specifications were further developed Serrano-Gotarredona and Linares-Barranco [88] by improving the contrast sensitivity and lowering the power consumption (4 mW) and latency ($3\mu s$) and small pixel size ($30 \times 31\mu m^2$). The event camera design as also been extended, which adds colour sensitivity to the sensor, which is added to the DAVIS design, as well as a $640x480$ VGA resolution DVS camera with the smallest reported pixel size ($9 \times 9\ mum^2$) [52]. These sensors show that the event-camera design is still an actively developing field and there is not a consensus yet about is the most

	DVS	ATIS	DAVIS
Resolution	128×128	304×240	240×180
Latency (μs)	15	3	3
Dynamic Range (dB)	120	143	130
Min. sensitivity (%)	17	13	11
Power (mW)	24	175	14
Chip size ($\mathbf{mm^2}$)	6.3×6	9.9×8.2	5×5
Pixel size ($\mu\mathbf{m^2}$)	40×40	30×30	18.5×18.5
Supply voltage (V)	3.3	1.8/3.3	1.8/3.3

Table 2.3: Overview and comparison between the DVS [53], ATIS [75] and DAVIS [13] with details of their technical specification.

optimal design for this sensor.

2.4 Summary

In this chapter, we have presented the mathematical notations and conventions which we are using throughout this thesis. We have also given an introduction to the prerequisites required to work with an event camera. We elaborate on the workings of the DVS and DAVIS and their biological justification in their design. In the following chapter, we will present the foundations of our event-based analysis.

Chapter 3

Solving Variational Image Denoising and Optical Flow Models with the Primal Dual Algorithm

Contents

In robotic research, the goal is to advance the capabilities of robots to perform tasks like driving cars or cleaning houses. These menial tasks seem simple from a human's perspective, but a robot misses many abilities which a human has for understanding a particular task as well as the environment. For example, if a human looks at a white wall, then the human infers that he or she might be inside a house, but a robot using cameras has only its sensor feedback, which is the colour white. Humans have the ability of "filling in gaps" to make sense of their sensory stimuli, and this ability is based on knowledge gained from a lifetime of experience. In robotics, providing this kind of knowledge for any problem is a challenging question, but it allows a robot to perform its task even if sensor measurements are missing or incomplete.

The same principle applies to the sparse event camera signal in which filling in the gaps is required to understand its signal properly, and consequently, for processing the signal we must we must find an appropriate method which merges tacit knowledge of images with the signal. In this chapter, we introduce variational methods which fill in gaps by assuming a smooth solution and which have been successfully applied to several machine vision areas like image denoising, super-resolution, optical flow and 3-D reconstruction. Their strength lies in the capacity to impose smoothness globally, which is an essential property of the structure of many computer vision problems. We introduce in this section the primal-dual algorithm which efficiently solves non-smooth variational problems.

3.1 Total Variation Models

Variational methods can solve many ill-posed inverse imaging problems and can be divided into two categories: Convex and non-convex. As discussed in Section 1.4, non-convex formulations fit closer to actual natural image statistics but create multiple solutions as well, which need to be evaluated to find the global optimum which is an NP-hard problem. A local minimum might be acceptable, however, the quality of a local solution is sensitive to initialisation and choice of the optimisation algorithm. Convex problems, on the other hand, can be solved precisely in a reasonable amount of time, independently of the initialisation. Hence, the quality of their solution depends only on the model accuracy of the convex formulation.

For image-based methods, total variation (TV) is a useful formulation because it allows sharp discontinuities in the estimation, which are related to edges in the image and are vital features for identifying object and motion boundaries. However, minimising TV is difficult because of its non-smoothness which spawned several approaches in recent decades to solve TV problems efficiently like the primal-dual algorithm [74]. The algorithm finds the precise solution for a TV-regularised problem and efficiently utilises the GPU for large-scale problems. In the following section, we introduce to the primal-dual algorithm and present example cases of using primal-dual for optical flow estimation and image denoising. However, we first give a brief intuition about how TV allows discontinuous solutions.

For this work, we define the TV norm as:

$$TV(u) := \int_{\Omega} |\nabla u| \mathrm{d}\Omega \tag{3.1}$$

which requires u to have an integrable, first-order derivative. A more general formulation of the TV norm is defined for any real-valued, absolutely integrable function as

$$\mathrm{TV}(u) := \max_{\mathbf{p}} \left\{ \int_{\Omega} u \nabla \cdot \mathbf{p} \, \mathrm{d}\Omega : \|\mathbf{p}\| \leq 1 \right\}, \tag{3.2}$$

which relates to Equation (3.1) if u is differentiable by the Gauss-Green divergence theorem [17]

$$\max_{\mathbf{p}} \left\{ \int_{\Omega} \mathbf{p} \cdot \nabla u \, \mathrm{d}\Omega : \|\mathbf{p}\| \leq 1 \right\} \leq \int_{\Omega} |\nabla u| \, \mathrm{d}\Omega \tag{3.3}$$

The primary property of TV is that it is invariant to discontinuities of the function. Figure 3.1 shows three example functions which share the same TV norm value although the functions have different discontinuities, which is in contrast to other norms like the square norm for which the norm values change to $f_1 : 0.01$, $f_2 : 0.1$ and $f_3 : 1.0$. The norms of the functions are determined by picking 1000 equidistant function points and calculating their differences.

Although the functions in Figure 3.1 are toy examples, they provide an intuition for how TV does not penalise edges in images. However, since the L^1-norm is not differentiable at every point, common gradient-based minimisation schemes like gradient descent are difficult to apply to these types of problems. Since the direct formulation of the TV-L^1 cost function is difficult to minimise, other methods like primal-dual solve an equivalent formulation instead which is easier to solve, which we introduce in the next section.

3.2 Primal-Dual

Pock *et al.* [74] proposed the primal-dual algorithm, which is an indirect, first-order minimisation method for finding the precise solution of a non-smooth, convex problem.

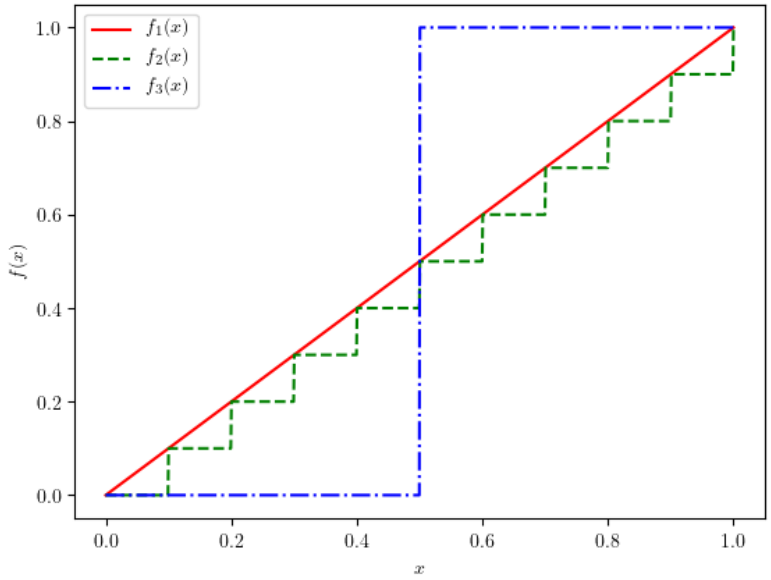


Figure 3.1: Three functions having the same total variation value of 1.

In contrast to previously proposed methods, primal-dual has a super-linear convergence rate and is easily parallelisable on commodity-type GPUs. The convergence rate and parallelizability make the primal-dual method a viable tool for real-time applications, and it has been successfully applied to image denoising, optical flow, super-resolution as well as to geometric models like depth-map estimation, which gave rise to dense SLAM methods [62].

In rough terms, the primal-dual method does not solve our problem directly. Instead, we use the Lagrangian relaxation, as described in Chapter 2.1.2, on the original problem to reformulated it into a *saddle-point* formulation, which has the same solution as our original problem but is easier to solve by an alternating, gradient-based minimisation scheme. Formally we formulate the saddle-point of a minimisation problem like:

$$\min_x f(Kx) + g(x), \tag{3.4}$$

where K is a continuous linear operator and f and g are convex functions. In the case of TV-L^1 problem, the terms in Equation (3.4) correspond $g(x)$ to the data-fidelity term

and $f(Kx)$ to TV-L^1 smoothness term, where K is the derivate operator and f is the L^1-norm. The saddle-point formulation of Equation (3.4) is then defined such that:

$$\min_x \max_y \langle Kx, y\rangle + g(x) - f^*(y), \tag{3.5}$$

where f^* is the convex conjugate of f which is defined such as:

$$f^*(x) := \sup_y \langle x, y\rangle - f(y). \tag{3.6}$$

The convex conjugate has the property that $f = f^{**}$ for any convex function, and applying it twice on Equation (3.4) gives the saddle point formulation (3.5). Another approach for deriving Equation (3.5) from Equation (3.4) is through the Lagrangian, which we introduce to support the understanding of the method. Let us assume that we have a different convex function h and the optimisation objective is such that:

$$\min_{x,z} h(z), \text{ s.t. } z = x,$$

where x is some parameter vector. Relaxing the formulation leads to the Lagrangian formulation leads to:

$$\min_{x,z} \max_y \{h(z) - \langle y, z - x\rangle\}$$

which has the same solution as our original formulation (brackets are added for clarity). Since h is convex we can rearrange the objective:

$$\begin{aligned}&\min_{x,z} \max_y \{\langle y, x\rangle + \min_z \{h(z) - \langle z, y\rangle\}\} \Leftrightarrow \\ &\min_x \max_y \{\langle y, x\rangle - \max_z \{\langle z, y\rangle - h(z)\}\},\end{aligned}$$

where $\max_z\{\langle z, y\rangle - h(z)\}$ is the definition of the convex conjugate, which is also known as the Legendre-Fenchel transformation, and substituting it with $h^*(y)$ gives us the saddle-point formulation:

$$\min_x \max_y \{\langle y, x\rangle - h^*(y)\}.$$

If h^* is easy to minimise, then Equation (3.2) is can be easily solved by alternating the minimisation between x and y, which leads us to the formulation of the primal-dual algorithm for minimising Equation (3.5):

Primal-dual algorithm

- For initialization: Choose $\tau, \sigma > 0, \theta \in [0,1], x^0, y^0$ and set $\bar{x}^0 = x^0$
- For n-th iteration-step: Update $x^n, y^n, \bar{x}^n$ as follows:

$$\begin{cases} y^{n+1} = (I + \sigma\partial f^*)^{-1}(y^n + \sigma K\bar{x}^n) \\ x^{n+1} = (I + \tau\partial g)^{-1}(x^n - \tau K^* y^{n+1}) \\ \bar{x}^{n+1} = x^{n+1} + \theta(x^{n+1} - x^n), \end{cases} \tag{3.7}$$

where τ, σ and θ are the alternating steps sizes, where θ is usually set to 1 and σ and τ are chosen so that $\sigma\tau\|K\|^2 < 1$ holds. For proof of convergence and the conditions of the step sizes, we refer to [16]. $\|K\|$ is the linear operator norm which is defined as:

$$\|K\| = \max\{\|Kx\| : x \in X \text{ with } \|x\| \leq 1\}. \tag{3.8}$$

$(I + \tau\partial f*)^{-1}$ and $(I + \tau\partial g)^{-1}$ are the resolvent operator of the proxmal operator, which is defined for f^* and analogues for g as:

$$x = \arg\min_x \left\{ \frac{1}{2\tau}\|x - y\|^2 + f^*(x) \right\}. \tag{3.9}$$

The proximal relates to the resolvent operator by setting its derivative with respect to x to 0:

$$
\begin{aligned}
0 =& \frac{\partial}{\partial x}\left(\frac{1}{2\tau}\|x-y\|^2 + f^*(x)\right) \Leftrightarrow \\
0 =& \frac{1}{\tau}(x-y) + \partial f^*(x) \Leftrightarrow \\
y =& x + \tau\partial f^*(x) = (I + \tau\partial f^*)(x).
\end{aligned}
$$

For the primal-dual algorithm, we assume that f^* and g are simple in that they have easy to find resolvent operators and that $(I + f^*)$ is invertible, so this gives us the resolvent operator:

$$x = (I + \tau\partial f^*)^{-1}(y).$$

There are different interpretations of the proximal operator, which are outside the scope of this thesis, but we refer to [72] for more details about the proximal operator. For the work of this thesis, the proximal operator can be viewed as a generalised Newton step.

In the next sections, we showcase two example use-cases of the above general formulation of the primal-dual algorithm, which helps to provide a better insight into the algorithm.

3.2.1 TV-L1 Optical Flow

Any convex cost-function can be minimised by the primal-dual algorithm which makes it applicable to a wide range of problems. In the last section, we provided the generic formulation of the algorithm, which we use in this section to derive an example use for optical flow estimation. The derivation provides us with a better understanding of the algorithm as well as the optical flow problem in general, which motivates our the design decision in the next chapter for the event-based signal.

For optical flow estimation, the objective is composed of two terms per pixel: The photometric consistency and smoothness of the motion field, which we introduce in

Chapter 1.4.3. The overall cost over the image domain $\Omega \subset \mathbb{R}^2$ is then defined as :

$$\min_{\mathbf{u}} \int_{\Omega} \|\nabla \mathbf{u}\|_1 + \lambda \|\bar{\rho}(\mathbf{u})\| \mathrm{d}\Omega, \tag{3.10}$$

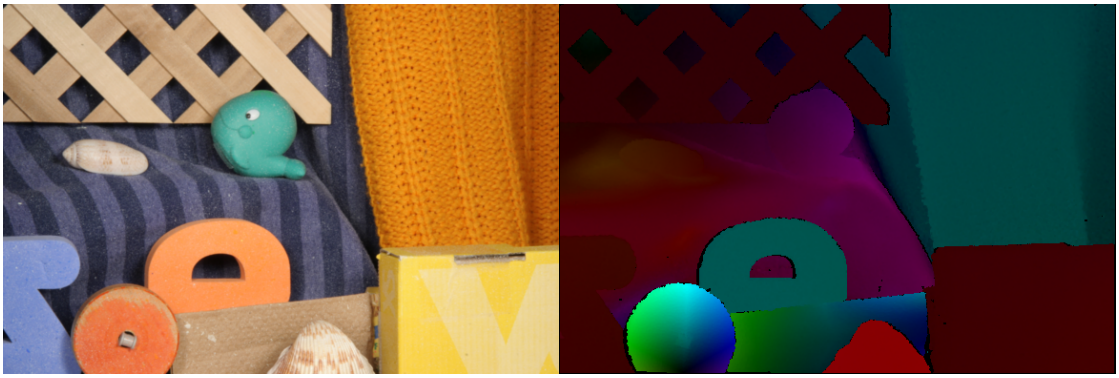
where $\lambda > 0$ is a scalar parameter, $\mathbf{u} = (u, v) : \Omega \mapsto \mathbb{R}^2$ is the motion field and $\bar{\rho}(\mathbf{u}) := I_{t+1} - w(I_t, \mathbf{u})$ is photometric consistency, where w the warping function which displaces each pixel according to $\mathbf{u}$. While Equation (3.10) describes our objective, it is not solvable by the primal-dual algorithm because of the non-convexity of the photometric consistency term. The term is locally convex for a given point, however, there is no guarantee that another point in the image is not more similar than the local neighbourhood. Instead, we circumvent this issue by approximating the photometric consistency term by the first two terms of its Taylor-expansion around the linearisation point $w(I_t, 0)$ which is simplified to I_t since 0 as the optical flow is the identity function. The expansion allows us to define the function:

$$\rho(\mathbf{u}) = I_t + \langle \nabla I_t, \mathbf{u} \rangle. \tag{3.11}$$

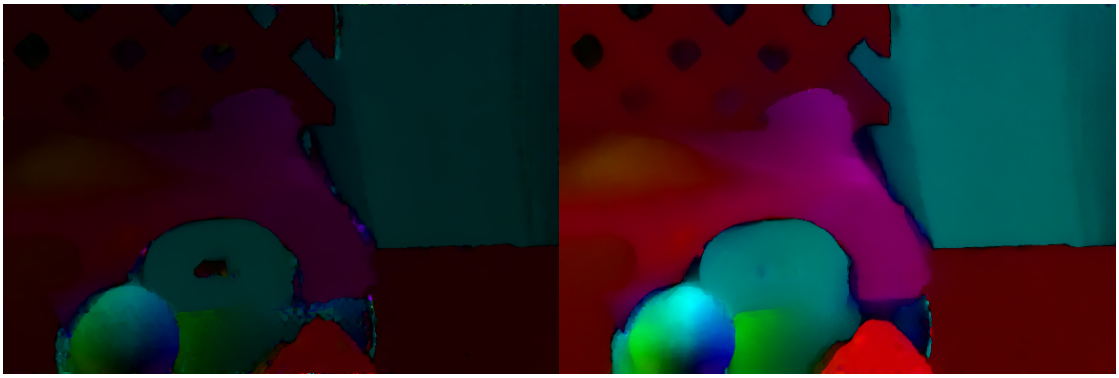
However, the approximation only holds close to the linearisation point, which means that we cannot find the correct solution if the disparity between the two frames is too large, i.e. the pixel moves too far from its original position from one frame to another. For this reason, optical flow is usually estimated for video sequences with a high enough frame rate. Fortunately, the common $30Hz$ signal of frame-based cameras is often high enough and larger disparities can still be addressed by using a coarse-to-fine estimation scheme. With a high frame-rate sequence, we can use the approximation ρ instead of $\bar{\rho}$ to create a convex version of Equation (3.10) which can be solved by the primal-dual algorithm. Before applying the algorithm to an images sequence, we first discretise the new convex version of the cost-function and derive the resolvent operators.

The discretisation addresses the fact that digital camera images have a finite resolution, which requires replacing the continuous operations with discrete counterparts. First, we replace the image domain Ω and define a Cartesian grid of size $M \times N$ instead:

$$\{(ih, jh) : 1 \leq i \leq M, 1 \leq j \leq N\}, \tag{3.12}$$



(a) First Input Image (b) Ground Truth Optical Flow



(c) TV-L^1, $\lambda = 0.5$ (d) TV-L^1, $\lambda = 0.1$

(e) TV-L^1, $\lambda = 0.05$ (f) TV-L^1, $\lambda = 0.01$

Figure 3.2: Optical flow estimation results using the Middlebury flow dataset [6]. We estimate optical with decreasing λ values to highlight the influence of the TV norm on the solution.

---

where h is the size of the pixel spacing in the image, and (i, j) denotes the indices of the locations (ih, jh) in the image domain. The discrete optical flow object is then defined

as:

$$\min_{\mathbf{u}} \sum_{i,j}^{M,N} |D\mathbf{u}_{i,j}| + \lambda \|\rho(\mathbf{u}_{i,j})\|_1, \tag{3.13}$$

and the corresponding saddle-point formulation as:

$$\min_{\mathbf{u}} \max_{\mathbf{p}} \sum_{i,j}^{M,N} \langle D\mathbf{u}_{i,j}, \mathbf{p}_{i,j} \rangle - \delta_P(\mathbf{p}_{i,j}) + \lambda \|\rho(\mathbf{u}_{i,j}))\|_1, \tag{3.14}$$

where $\mathbf{u}_{i,j} = (u_{i,j}, v_{i,j})^T$ and $\mathbf{p}_{i,j} = (p^1_{i,j}, p^2_{i,j}, p^3_{i,j}, p^4_{i,j})^T$ are the motion vector and dual vector at pixel position i, j. D is the gradient operator with respect to $\mathbf{u}$ and is defined by the forward difference with Neumann boundary conditions:

$$D\mathbf{u}_{i,j} = (D_x u_{i,j}, D_y u_{i,j}, D_x v_{i,j}, D_y v_{i,j})^T, \tag{3.15}$$

where D_x and D_y are the directional derivates:

$$D_x u_{i,j} = \begin{cases} \frac{u_{i+1,j} - u_{i,j}}{h} & \text{if } i < M, \\ 0 & \text{if } i = M \end{cases}, \; D_y u_{i,j} = \begin{cases} \frac{u_{i,j+1} - u_{i,j}}{h} & \text{if } j < N, \\ 0 & \text{if } j = N \end{cases}, \tag{3.16}$$

which is analogous defined for $v_{i,j}$. The norm $\|D\mathbf{u}\|_1$ denotes the isotropic total variation norm defined as:

$$\begin{aligned} \|D\mathbf{u}\|_1 &= \sum_{i,j} |D\mathbf{u}_{i,j}|, \\ |D\mathbf{u}_{i,j}| &= \sqrt{(D_x u_{i,j})^2 + (D_y u_{i,j})^2 + (D_x v_{i,j})^2 + (D_y v_{i,j})^2}. \end{aligned} \tag{3.17}$$

In order to apply the primal-dual algorithm, we need to detail the resolvent operators of Equation (3.7) by defining $f(\nabla \mathbf{u}) = \|D\mathbf{u}\|_1$ and $g(\mathbf{u}) = \|\rho(\mathbf{u})\|_1$ and $f^*(\mathbf{p}) = \delta_P(\mathbf{p})$,

where:

$$\delta_P(\mathbf{p}) = \begin{cases} 0 & \text{if } \mathbf{p} \in P \\ +\infty & \text{if } \mathbf{p} \notin P \end{cases}, \tag{3.18}$$

is the indicator function, the convex set P is given by:

$$P = \{\mathbf{p} \in Y : \|\mathbf{p}\|_\infty \leq 1\}, \tag{3.19}$$

and $\|\mathbf{p}\|_\infty$ denotes the discrete maximum norm defined as:

$$\|\mathbf{p}\|_\infty = \max_{i,j} |\mathbf{p}_{i,j}|, \ |\mathbf{p}_{i,j}| = \sqrt{(p^1_{i,j})^2 + (p^2_{i,j})^2 + (p^3_{i,j})^2 + (p^4_{i,j})^2}. \tag{3.20}$$

Since f^* is the indicator function of a convex set, the resolvent operator projects is a pointwise Euclidean projection onto L^2 balls such that:

$$\mathbf{p}_{i,j} = (I + \sigma \partial f^*)^{-1}(\tilde{\mathbf{p}}_{i,j}) = \frac{\tilde{\mathbf{p}}_{i,j}}{\max(1, |\tilde{\mathbf{p}}_{i,j}|)} \tag{3.21}$$

The resolvent operator with respect to g is given by the pointwise shrinkage operations:

$$\mathbf{u}_{i,j} = (I + \tau \partial g)^{-1}(\tilde{\mathbf{u}}_{i,j}) = \tilde{\mathbf{u}}_{i,j} + \begin{cases} \tau\lambda & \text{if } \rho(\tilde{\mathbf{u}}_{i,j}) < -\tau\lambda \\ -\tau\lambda & \text{if } \rho(\tilde{\mathbf{u}}_{i,j}) > \tau\lambda \\ -\frac{\rho(\tilde{\mathbf{u}}_{i,j}) DI_{i,j}}{|DI_{i,j}|^2} & \text{if } |\rho(\tilde{\mathbf{u}}_{i,j})| \leq \tau\lambda \end{cases} \tag{3.22}$$

With the discretisation and the details of the resolvent operator defined, the primal-dual algorithm can be applied to optical flow estimation. However, if M and N are large then it is too computationally expensive to run the algorithm on a single CPU. The primal-dual algorithm addresses this problem by being easily parallelisable, which is advantageous for utilising GPUs which can run thousands of operations at once. The parallelizability is optimal for GPUs if the individual operations are as independent as possible from each other, which is true for the primal-dual algorithm which only requires the neighbourhood pixel values at the beginning of each iteration step.

In the following, we present results for TV-L^1-based optical flow estimation using the primal-dual algorithm. Instead of evaluating the accuracy of those methods, we focus here on qualitative results and their sensitivity to λ in the Equation (3.13). These examples provide an intuition for the properties of the TV norm and its advantages as well as disadvantages, which motivates subsequent work.

The stair-casing effect is apparent in the optical flow estimation which is highlighted in Figure 3.2. The results show similar stair-casing effects, however, we like to note that optical flow estimation often fails at object boundaries, which is caused by the occlusion of corresponding values. The occlusions violate the photometric consistency assumption, which Equation (3.13) is only able to address through the L^1 loss, which is robust against outliers. However, if we wish to explicitly address occlusion more sophisticated methods are required, however, they also increase the complexity of the method as well.

3.2.2 TV-L^1 Image Denoising

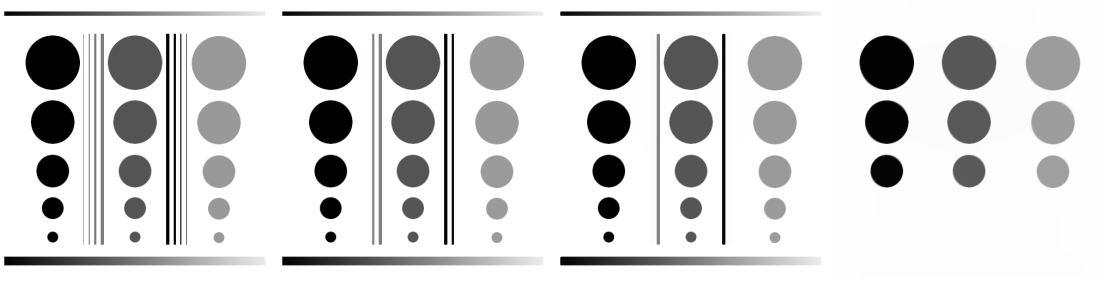
Another example in which variational optimisation is successfully applied is image denoising. Many aspects of optical flow estimation apply to image denoising as well; we simply redefine our data-fidelity term and its resolvent operator. Our discrete image denoising objective is defined as:

$$\min_{\mathbf{v}} \sum_{i,j}^{M,N} |Dv_{i,j}| + \lambda \|v_{i,j} - I_{i,j}\|_1, \tag{3.23}$$

where $v_{i,j}$ pixel value at position (i, j) of our estimated image $\mathbf{v}$ and $I_{i,j}$ is the pixel value of the noise-corrupted image. The norms and the λ are defined similary to Equation (3.13), but since $v_{i,j}$ is a scalar we define the TV-L^1 norm as:

$$\|D\mathbf{v}\|_1 = \sum_{i,j}^{M,N} |Dv_{i,j}|, \;\; |Dv_{i,j}| = \sqrt{(D_x v_{i,j})^2 + (D_y v_{i,j})^2}, \tag{3.24}$$

where D_x and D_y are the directional derivates as defined as in the previous section. For the primal-dual algorithm we define the TV norm as f and $\|v_{i,j} - I_{i,j}\|$ as g. The dual norm f^* is defined as in Section 3.2.1, but defined for two dimensions instead of four.



(a) Input Image (b) $\lambda = 1.0$ (c) $\lambda = 0.5$ (d) $\lambda = 0.1$

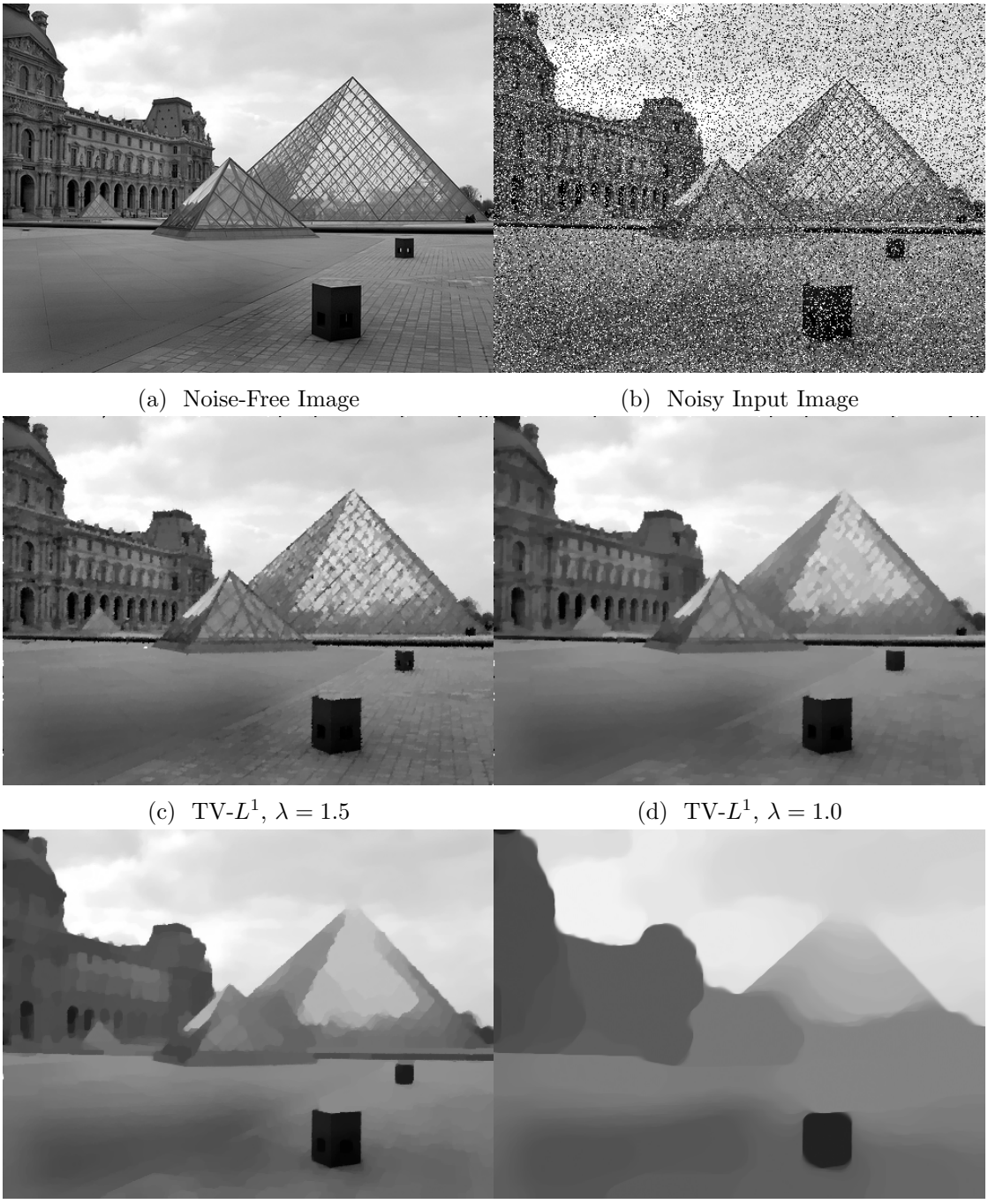
Figure 3.3: Image denoising with different λ values, and its influence on finer image structures.

The resolvent operator of g is then defined as:

$$\mathbf{u}_{i,j} = (I + \tau\partial g)^{-1}(\tilde{\mathbf{u}}_{i,j}) = \begin{cases} \tilde{\mathbf{u}}_{i,j} - \tau\lambda & \text{if } \tilde{\mathbf{u}}_{i,j} - I_{i,j} > \tau\lambda \\ \tilde{\mathbf{u}}_{i,j} + \tau\lambda & \text{if } \tilde{\mathbf{u}}_{i,j} - I_{i,j} < -\tau\lambda \\ I_{i,j} & \text{if } |\tilde{\mathbf{u}}_{i,j} - I_{i,j}| \leq \tau\lambda. \end{cases} \tag{3.25}$$

With the operators defined, the primal-dual algorithm can be applied to image denoising as well as be implemented on a GPU. From the above example, we see that the algorithm is easily adaptable to new problems, as long as the operators are easy to define. However, in order to guarantee convergence the norm or the linear operator $\|D\|$ needs to be determined, which is reasonable for simple linear operators. However, for more complex ones we can use an extension of the primal-dual algorithm, which we discuss in the next section.

Before we continue, we present here some results of the denoising process. The artificial look through the denoising process influence is apparent in Figure 3.4, which shows the denoising results for a natural scene. The algorithm effectively removes the noise influences. However, it suppresses large structures if λ is poorly chosen. In that case, textures are replaced by monochrome regions which are visible in Figure 3.4e and 3.4f. Besides those regions, another effect from TV is visible which is called the stair-casing effect. Star-casing describes discontinuities within regions, which do not correspond to real structures. The effect stems from the over-regularised areas and the TV invariance towards discontinuities. Unfortunately, these effects are unavoidable using purely TV-L^1



(a) Noise-Free Image (b) Noisy Input Image

(c) TV-L^1, $\lambda = 1.5$ (d) TV-L^1, $\lambda = 1.0$

(e) TV-L^1, $\lambda = 0.5$ (f) TV-L^1, $\lambda = 0.1$

Figure 3.4: Denoising results of a natural image with decreasing λ values. The each results used the noisy version as the input image.

and thus it becomes important to carefully chose an appropriate λ parameter.

The effect of over-smoothing is shown in Figure 3.3 with an image of an abstract scene and the results after denoising by minimising Equation (3.24) with different λ values. Lower λ values decrease the influence of the data term and subsequently increase the TV norm. With a weaker data term, finer image structures are overwritten by the smoothing effect of the TV norm. This effect explains why TV is able to remove noise, however, it suppresses finer details as well which results in an artificial appearance in the image.

3.2.3 Preconditioned Primal-Dual

An important extension to the primal-dual algorithm (3.7) is preconditioning, which was proposed by Pock and Chambolle [73]. The primal-dual algorithm converges as long as $\theta = 1$ and $\tau\sigma\|D\|^2 \leq 1$, which requires determining $\|D\|$ in an efficient manner. For the discrete gradient operator the tight upper-bound $\|D\|^2 \leq 8/h^2$ holds, which allows us to choose a τ and derive σ and vice versa [16]. However, if D is replaced by a more complex and large $m \times n$ matrix K then computing the matrix norm $\|K\|$ might be too computationally expensive to process which negates the efficiency of the primal-dual algorithm.

Preconditioning allows circumventing this problem by projecting the linear operator K such that it fulfils the condition $\tau\sigma\|K\|^2 \leq 1$ even if we choose τ and σ to be constant. The preconditioned primal-dual algorithm is thus defined as:

$$\begin{cases} \mathbf{p}^{k+1} = (I + \Sigma\partial f^*)^{-1}(\mathbf{p}^k + \Sigma K\bar{\mathbf{u}}^k) \\ \mathbf{u}^{k+1} = (I + T\partial g)^{-1}(\mathbf{u}^k - TK^T\mathbf{p}^{k+1}) \\ \bar{\mathbf{u}}^{k+1} = 2\mathbf{u}^{k+1} - \mathbf{u}^k, \end{cases} \tag{3.26}$$

where the step-sizes τ and σ from the Algorithm (3.7) are replaced by the matrices T and Σ. There are different ways how T and Σ Different preconditioning methods exist for a variety of optimization algorithm and are often used for numerical stability, but one of the most straight-forward ones is the diagonal preconditioning which multiplies the T and Σ are chosen to be diagonal matrices, such that $T = \text{diag}(\tau)$, where $\tau = (\tau_1, \ldots, \tau_m)^T$

and $\Sigma = \text{diag}(\sigma)$, where $\sigma = (\sigma_1, \ldots, \sigma_n)^T$. Each element of τ and σ is then:

$$\tau_j = \frac{1}{\sum_{i=1}^{m} |K_{i,j}|^{2-\alpha}}, \ \sigma_i = \frac{1}{\sum_{j=1}^{n} |K_{i,j}|^{\alpha}}, \tag{3.27}$$

for any $\alpha \in [0, 2]$. Using Σ and T in Equation (3.26) allows using the condition $\|\Sigma^{\frac{1}{2}} K T^{\frac{1}{2}}\| \leq 1$. The diagonal preconditioning matrices can be efficiently computed and applied to the primal-dual minimisation scheme and guarantee convergence even when if the norm of the matrix is unknown.

We also note that if K is the gradient operator and the preconditioning is applied then the upper-bound $\|D\|^2 \leq 8/h^2$ hold as well.

Because Σ and T are easy to compute and to apply, the preconditioned primal-dual algorithm can be leveraged for less structured problems, which is the case for event camera signal where we show in the next chapter that the differences of our estimation are determined by the events in the sequence instead of an equidistance differences as with the gradient operator. We conclude this chapter with results of the optical flow estimation and image denoising, and provide an intuition of the properties and influences of the TV-norm in the solution.

3.3 Conclusion

In this sections, we introduce the primal-dual algorithm; a fast and efficient variational estimation method which provides precise solutions of TV-L^1 problems in real-time. However, the algorithm has a number of drawbacks which are batch-type nature, its memory consumption and its restriction to convex problems. Further, if parameters are chosen poorly then finer structures are removed and stair-casing artefacts are introduced into the solutions.

These artefacts are created because the TV norm only approximates the actual image statistics. With this, the solution cannot express the actual nature of the image statistics, which causes the suppression of finer image details. However, using a more accurate representation has several drawbacks. First, the statistic of natural images is complex and hard to describe, which is why TV norm restrict itself to the relationship between

neighbouring pixels. Another issue is that the actual image statistics require a non-convex formulation, which creates a hard-to-solve problem. The limitation to convex problems is a necessary restriction for computational efficiency, which is detrimental since we often face non-convex problems in machine vision.

However, TV minimisation has many beneficial properties for computer vision and its approximate image statistic allows us to apply prior information to our solution, which is interesting if such information is missing in our estimation. For example, DTAM [63] uses this information to infer 3-D structures, which are not revealed by photometric measurements. In the same manner, we could use the estimation for inferring information in relation to our event camera measurements and fill in the missing information. In the next chapter, we introduce our work which combines the event camera with variational estimation.

Chapter 4

Simultaneous Optical Flow and Intensity Estimation from an Event Camera

Contents

In Chapters 1 and 2, we introduced the bio-inspired vision event camera sensor which mimics the retina to measure per-pixel intensity changes rather than outputting an

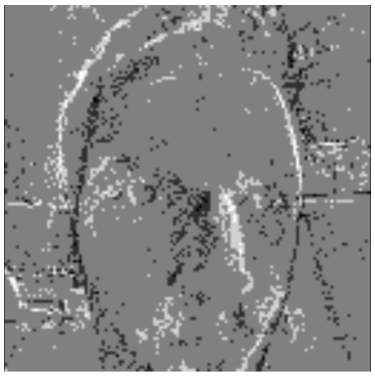
actual intensity image. The camera proposes a paradigm shift away from traditional frame-based cameras which offers significant potential advantages: namely avoiding high data rates, dynamic range limitations and motion blur. Unfortunately, as discussed in Chapter 1, established computer vision algorithms may not at all be applied directly to event cameras. Previously proposed event-based methods to reconstruct images, estimate optical flow, track camera motion include severe restrictions on the environment or on the motion of the camera, e.g. allowing only rotation.

In this chapter, we are introducing the first contribution of this thesis, which is the first algorithm to simultaneously recover the motion field and brightness image from an event camera, while the camera undergoes a generic motion through any scene. The approach employs minimisation of a cost function that contains the asynchronous event data as well as spatial and temporal regularisation within a sliding window time interval. The implementation of the algorithm relies on GPU optimisation and runs in real-time. At the end of this chapter, we show a series of examples which demonstrate the successful operation of our framework, including in situations where conventional cameras suffer from dynamic range limitations and motion blur. In Chapter 7, we will provide additional evaluations, in which give an overview and comparison of the contribution of this thesis. This chapter is mainly based on Bardow etal [7].

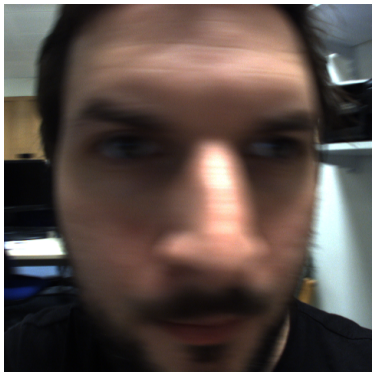
4.1 Introduction

We know very clearly from systems like LSD-SLAM [21] that pixels, where edges move, are the only ones which give information useful for tracking and reconstruction, and it is precisely these locations which are highlighted in hardware by an event camera. We can see those edge regions distinctly in the event camera signal shown in Figure 6.5.

Our contribution provides a significant step towards the general potential of event cameras for motion and structure estimation by presenting the first algorithm which simultaneously estimates scene intensity and motion with minimal assumptions about the type of scene and motion. In the following, we introduce the technical details of our method, which is based on a sliding-window variational optimisation approach, where the events within a certain time period are used to estimate motion and intensity at image resolution and a user-defined time quantisation. The optimisation combines in its cost function the event measurements with an optical flow-like brightness constancy term

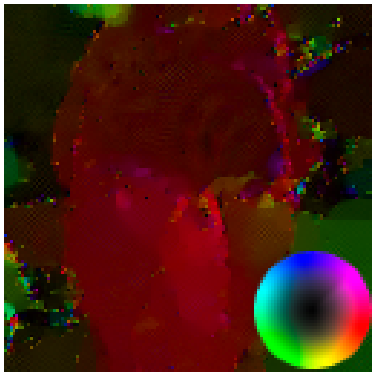


(a) Raw event camera output



(b) Standard camera image

(c) Intensity estimate from events



(d) Optical flow from events

Figure 4.1: Results from our reconstruction method: a raw integrated output from a DVS128 event camera; b the same scene from a standard camera. c and d show a snapshots of the intensity and velocity fields we estimate jointly only from event data. A color-wheel is used to show the velocity per pixel.

and terms which enforce spatial and temporal smoothness on the estimated intensity and motion fields.

4.2 Combining Optical Flow, Intensity and Event Data

'Optical Flow' is in the title of this chapter, because a key goal is to recover a generic motion field from camera data, but it is important to clarify the difference between the common understanding of this term in computer vision and the rather different type of estimation our algorithm achieves. Optical flow is normally understood as the correspondence field between two temporally close intensity images — an estimation of two parameters per pixel, the horizontal and vertical displacement between one frame

and the next. In our case, we use the time stream of event data to estimate a continuously varying motion field at any time resolution. Therefore, it is more precise to say that we estimate a continuously time-varying *velocity field* in image coordinates.

Measuring velocity as a camera observes an arbitrary moving scene requires and implies knowledge of the correspondence between entities at different points in time, and this is where we face a particular challenge when the only input is event data which does not directly record image intensity or even oriented edges. We can only interpret an event as a measurement of motion if we know about the intensity gradient in the scene; but on the contrary we can only interpret it as a measurement of intensity gradient if we know the motion. Therefore, we must formulate a joint estimation problem to recover both motion and intensity together. We will show that weak assumptions about regularity in the overall solutions for motion and intensity are enough to allow this.

Our algorithm is formulated as sliding window variational optimisation and has much in common with well known variational methods for estimating two-view optical flow from standard video [31, 102]. We pre-define an optimisation time window T, and within this a fine time discretisation δ_t. We take all of the events in time window T as input and solve jointly for the velocity field $\mathbf{u}$ and log intensity L at all cells in the associated spatio-temporal volume. We then slide the optimisation forward to a highly overlapping position, initialising the values of all cells to either previous estimates or predictions, andmp solve again.

4.2.1 Event Data Loss

As described in Chapter 2, each pixel of an event camera independently measures the intensity and reports an event when that intensity changes by a pre-defined threshold amount relative to a saved value. The pixels operate asynchronously but each event is time-stamped relative to a global clock. The electronics of the camera gather events from all pixels and transmit them to a computer as a serial stream.

With this in mind, we define an event for this work as a tuple $e_i = (x_i, y_i, t_i, \rho_i)^T$, where $\mathbf{x}_i = (x_i, y_i) \in \Omega$ is the position of the event in the image domain, t_i is its time-stamp to microsecond resolution and $\rho_i = \pm 1$ is its polarity (sign of the brightness change). Referring to the event response function (2.22), an event is fired when a change in that

log intensity exceeds threshold θ:

$$|L(x, y, t) - L(x, y, t_p(x, y, t))| \geq \theta, \tag{4.1}$$

where $L(x, y, t)$ is the log intensity at pixel (x, y) at time t and $t_p(x, y, t)$ is the time when the previous event occurred. For this work, we assume that each event is triggered as soon as θ is met, which is valid since the latency and refractory period of an event camera pixel last only a few microseconds. The assumption then allows us to define the reconstruction loss as

$$\sum_{i=1}^{N} |L(x_i, y_i, t_i) - L(x_i, y_i, t_p(x_i, y_i, t_i)) - \rho_i \theta|, \tag{4.2}$$

where N is the number of events in the event camera stream. With the reconstruction loss defined, we introduce next the optical flow loss.

4.2.2 Optical Flow Loss

We aim to estimate continuously varying image velocity $\mathbf{u}$ and log intensity L at all image pixels over the duration of our input event sequence. The log intensity is related to image intensity as follows: $L := \log(I + b)$, where b is a positive offset constant. Since event cameras do not come with any notion of frames, we note that $\mathbf{u} = (u, v)^T$ is in velocity units of pixels/second rather than a frame-to-frame displacement. For brevity we will write partial derivatives as $\mathbf{u_x} := \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$.

As described in Section 1.4.3, well-known two-view methods for optical flow estimation [31, 102] use the key assumption of brightness constancy, which asserts that the brightness value of a moving pixel is unchanged. In the context of the logarithmic intensity of the event camera we use the brightness constancy such that:

$$L(x + \delta_t u, y + \delta_t v, t + \delta_t) = L(x, y, t). \tag{4.3}$$

Using the first-order Taylor-expansion by applying Equation (2.9) to the brightness constancy, we can define the optical flow loss based on the logarithmic intensity such

that:

$$|\langle \nabla L_x, \mathbf{u} \rangle + L_t|. \tag{4.4}$$

We detailed in Section 1.4.3 that on a per-pixel basis, the brightness consistency equation is under-determined, which requires us, like many other optical flow methods [31, 102], to introduce regularisation and perform global optimisation in order to achieve a well-defined solution across the whole image domain simultaneously. For the regularisation of the intensity and the optical flow, we use the same formulations as for the task of image denoising and optical flow estimation, which we introduced in the Sections 3.2.2 and 3.2.1.

But in the case of the input data from an event camera, Equation (4.3) cannot be directly applied since event measurements do not provide absolute intensity information but only differences. This means that we must estimate the intensity information to calculate the optical flow in the scene properly. To proceed we formulate our problem as a simultaneous estimation of both intensity and velocity. The details of our approach follow in the next section.

4.2.3 Variational Formulation

As mentioned above, we add regularisers to Equation (4.3) to determine the system and to handle the sparse measurements from the event camera. These are, in essence, smoothness priors which we discussed in Chapter 1 in context of image processing applications for standard cameras, such as optical flow [99], image denoising [87] and SLAM [63]. In Chapter 3, we introduced variational methods which have been very successful to include smoothness priors such as TV-L^1 [74], which approximates natural image statistics [80], while leaving the optimisation problem convex.

With event cameras, smoothness priors allow us to estimate image regions in between events — both spatially and temporally. In other words, we have sensor regions where events are firing and giving information about gradients, and we have regions with no data and no events firing, but we can assume smoothness in the absence of events.

Since an event relates a temporal intensity difference between a previous event and the

current one, we integrate the event measurements to the spatio-temporal smoothness and photometric consistency (optical flow constraint) within a time window. As opposed to traditional optical flow estimation, intensities are unknown, and so is optical flow. Since both quantities are coupled by the optical flow constraint (4.3), we need to estimate both *jointly*. We assume that the intensity change at a pixel is induced only by optical flow.

We therefore propose the following minimisation:

$$\begin{aligned}\min_{\mathbf{u},L} \int_\Omega \int_T \Big(&\lambda_1 \|\mathbf{u}_\mathbf{x}\|_1 + \lambda_2 \|\mathbf{u}_t\|_1 + \lambda_3 \|L_\mathbf{x}\|_1 + \\ &\lambda_4 \|\langle L_\mathbf{x}, \delta_t \mathbf{u}\rangle + L_t\|_1 + \lambda_5\, h_\theta(L - L(t_p))\Big) \mathrm{d}t\, \mathrm{d}\mathbf{x} \\ &+ \int_\Omega \sum_{i=2}^{|P(\mathbf{x})|} \|L(t_i) - L(t_{i-1}) - \theta\rho_i\|_1 \mathrm{d}\mathbf{x},\end{aligned} \tag{4.5}$$

where the individual λs are positive scalar weights. For brevity, we omit the parameters $\mathbf{x} = (x, y)$ and t. The first three terms regularise the smoothness of the flow (both spatially and temporally) and the smoothness of the intensities. The fourth term is the first order Taylor approximation of the optical flow constraint (4.3). The regularisers ensure a unique solution, and are necessary due to to the sparseness of our data term. This means that the regularisers allow us to find a solution even when no data is present, which is necessary due to the nature of the event camera.

The last two terms of (4.5) are the data terms of the event camera: The *event data term* and the *no-event data term*. The event data term is derived from Equation (4.1), where $P(\mathbf{x})$ is the set of all events fired at $\mathbf{x} = (x, y)$, with t_i and ρ_i being the time-stamp and polarity of the i-th element in $P(\vec{x})$. While this term models events which have been fired, the no-event data term models the case of no events occurring at a certain pixel: the absence of events gives us the information that the log intensity, after the last event, has not changed more than the given threshold θ. Therefore, we constrain the intensity between two events with an L^1-norm cost term containing a dead-zone, which is denoted

by h_θ. h_θ is defined as

$$h_\theta(x) = \begin{cases} |x| - \theta, & \text{if } |x| > \theta \\ 0, & \text{otherwise,} \end{cases} \tag{4.6}$$

and takes as input the difference between L and $L(t_p)$, which is the log intensity at last event relative to L. By using the dead zone, the term does not add any cost when the difference is within the bounds of $[-\theta, \theta]$, but penalises deviation beyond. We use an L^1-norm, in both terms, as we anticipate outliers. Events may be missed by the chip specifically in a short period just after an event has fired — and we anticipate randomly firing events (background noise), which occur in the sensor due to leakage [53], which we described in Chapter 2. Next, we will describe the discretisation and minimisation of Equation (4.5).

4.2.4 Discretisation

As described in the previous section, we estimate L and $\mathbf{u}$ over a time period T and over the image domain Ω. For the minimisation, we discretise Ω into a regular pixel grid of size $M \times N$, and T into K cells each of length δ_t microseconds, which forms a spatio-temporal volume. For each element in the volume created we estimate L and the motion $\mathbf{u}$ by minimising the discretised version of Equation (4.5). After the minimisation, we then slide the window in time by δ_t and minimise again. Figure 4.2 visualises this scheme.

Such a sliding window scheme bounds the temporal domain and allows us to propagate previous estimates, allowing the method to work with constant computation time on arbitrarily long sequences. Important here is the choice of δ_t, since a larger choice of δ_t allows to estimate slower motions, while a smaller value is good for fast motions. We use a constant value for δ_t, but in future work this choice may be automated by adopting it to the rate of incoming events.

Ideally, δ_t would match the digital time-stamp resolution of the sensor (i.e. microsecond-scale), however, if the number of cells in the windows is too small, then this could cause that only a few cells correspond to event measurements. Having too few event measurements in the windows means that not enough information is contained in the estimate for a proper estimation. This issue could be addressed by increasing the number of cells

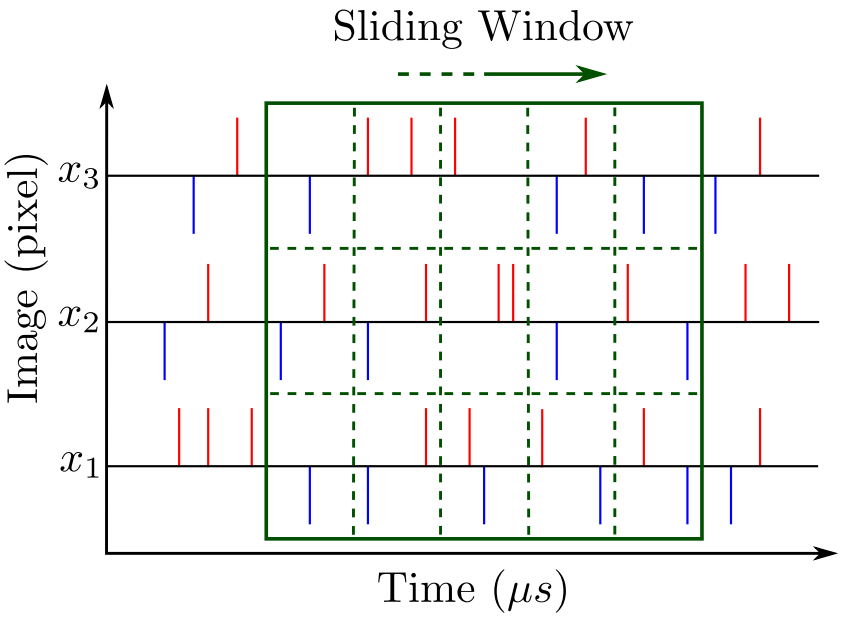


Figure 4.2: The sliding window (green box) bins the incoming positive events (red bars) and negative events (blue bars) into a regular grid (dashed lines). When the minimisation converges, the window shifts to the right. For clarity, we only depict only a signal pixel row of the image instead of the full 2-D image.

and therefore the time interval of the window, however, this exceeds the memory capacity of most commercially available computers even for short periods of the sequence. Because of the limited memory capacity, we choose δ_t to be larger than the actual digital time-stamp resolution. However, choosing larger values for δ_t can be justified by the limited bandwidth of the camera pixels themselves as well as limited rates of change of intensities in an actual scene. To adapt our event data term to a lower temporal resolution we linearly interpolate the intensity at the time of each event, as described in Figure 4.3.

After shifting the sliding window, the oldest estimates and related event data terms drop out while we add new incoming events to the window. These new elements in our volume are initialised by assuming a constant motion from our previous estimate. For this, we "copy" the previous estimates of $\mathbf{u}$ into the new grid cells and use the newly "copied" motion vectors $\mathbf{u}$ to bilinearly interpolate from the previous log intensity estimate.

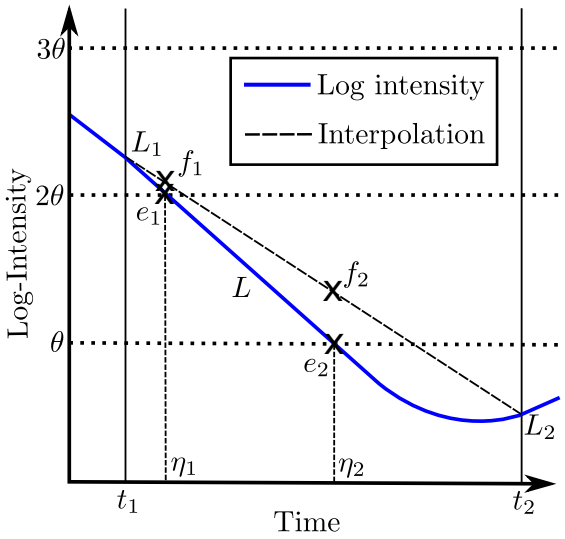


Figure 4.3: Approximation of the intensity for two given events e_1 and e_2 between two intensity estimates L_1 and L_2. For the discrete data term, we use the linear approximations f_1 and f_2 at the time of each event η_1 and η_2.

Unfortunately, by deleting estimates from the oldest band of grid cells, we lose information about that period. To compensate for this loss, we constrain the oldest estimates in our sliding window with a *prior image* and penalise deviations from those values in the next minimisation. The idea is that consecutive estimates by the sliding window should be discouraged to discard previous estimates, since they are based on measurements which are not included anymore. To include those priors, we add $\lambda_6 \|L(\vec{x}, t_1) - \hat{L}(\vec{x})\|_2^2$ to (4.5), where $\hat{L} \in \Omega$ is the prior image and t_1 is the first event time-stamp at $\vec{x}$ in our sliding window. If there is no such t_1 at $\vec{x}$, then we use instead of t_1 the minimum of T. Before each minimization, we update $\hat{L}(\vec{x})$ by copying the values of $L(\vec{x}, t_1)$ for all pixels and if there are no events at $\vec{x}$, because they dropped out of the sliding window, we leave the value in $\hat{L}$ unchanged. By using this prior image scheme, we can mitigate the loss of information to a certain extent.

We also want to point out that the prior image scheme would allow us to include intensity *measurements* from other sources, *e.g.* with DAVIS240 [13], which provides a standard camera image besides an event stream. However, in this work we focus on exclusively analysing the event stream.

For the discretisation of the spatio-temporal sliding window, we extend here the TV-L1 smoothness losses for image denoising (3.24) to the temporal domain such that:

$$\|D_{\mathbf{x}}L\|_1 = \sum_{i,j,k} |D_{\mathbf{x}}L_{i,j,k}|,$$

$$|D_{\mathbf{x}}L_{i,j,k}| = \sqrt{(D_x L_{i,j,k})^2 + (D_y L_{i,j,k})^2}.$$

We apply temporal smoothness to the optical flow smoothness of (3.17) as well, which is then defined as:

$$\begin{aligned} \|D_{\mathbf{x}}\mathbf{u}\|_1 &= \sum_{i,j,k} |D_{\mathbf{x}}\mathbf{u}_{i,j,k}|, \\ |D_{\mathbf{x}}\mathbf{u}_{i,j,k}| &= \sqrt{(D_x u_{i,j,k})^2 + (D_y u_{i,j,k})^2 + (D_x v_{i,j,k})^2 + (D_y v_{i,j,k})^2} \end{aligned}$$

Both smoothness priors are analogously defined for the temporal derivative operator D_t as well.

4.2.5 Optimisation

To minimise Equation (4.5) we use the preconditioned primal-dual algorithm [73], which we described in detail in Chapter 3. As stated in that chapter, the primal-dual algorithm has the advantage that it has optimal convergence and that it is easily parallelisable, which permits real-time implementation of the algorithm. To use the primal-dual algorithm we use the duality principle and replace individual L^1-norms of (4.5) by their conjugate using the Legendre-Fenchel transform defined by Equation (3.6) [28]:

$$\begin{aligned} \min_{\mathbf{u},L} \max_{\substack{|\mathbf{a}|_\infty \le \lambda_1 \\ |\mathbf{b}|_\infty \le \lambda_2 \\ |\mathbf{c}|_\infty \le \lambda_3 \\ |\mathbf{d}|_\infty \le \lambda_4 \\ |\mathbf{y}|_\infty \le 1}} & \langle \mathbf{D}_{\mathbf{x}}\mathbf{u}, \mathbf{a}\rangle - \delta_{\lambda_1 A}(\mathbf{a}) + \langle \mathbf{D}_t\mathbf{u}, \mathbf{b}\rangle - \delta_{\lambda_2 B}(\mathbf{b}) + \langle \mathbf{D}'_{\mathbf{x}}L, \mathbf{c}\rangle - \\ & \delta_{\lambda_3 C}(\mathbf{c}) + \langle \langle \mathbf{D}'_{\mathbf{x}}L, \mathbf{u}\,\delta_t\rangle + \mathbf{D}'_t L, \mathbf{d}\rangle - \delta_{\lambda_4 D}(\mathbf{d}) + \\ & \lambda_6 h_\theta(L - L(t_p)) + \langle \mathbf{E}L - \mathbf{z}, \mathbf{y}\rangle - \delta_Y(\mathbf{y}), \end{aligned} \tag{4.7}$$

where $\mathbf{D}_{\mathbf{x}}$, $\mathbf{D}'_{\mathbf{x}}$ represent the finite difference matrices with respect to $\mathbf{x}$ and $\mathbf{D}_t$, $\mathbf{D}'_t$ are the difference matrices with respect to t. The individual $\delta(.)$ terms are the indicator

functions which are defined by (3.18) regards to the dual variables of the individual norms. Note that we reformulate the event data term to the matrix expression $\mathbf{E}L - \mathbf{z}$, where $\mathbf{z}$ is our measurement vector containing the signs of all observed events scaled by θ and $\mathbf{E}$ is the *event matrix* which transforms the intensity estimate to pairwise differences of linearly interpolated intensities of the observed events.

The optical flow term in Equation (4.7) is biconvex, due to the inner product of $L_{\mathbf{x}}$ and $\mathbf{u}$. We follow here the minimisation strategy for a biconvex function in [25] by minimising this term by alternating between the estimation for L and $\mathbf{u}$. This gives us the following minimisation scheme of (4.7):

$$
\begin{cases}
L^{n+1} = (\mathbf{I} + \mathbf{T_1}\lambda_6 \partial h_\theta)^{-1}(\bar{L}^n - \mathbf{T_1}(\mathbf{D'_x}^T \mathbf{c}^n - \mathbf{D'_x}^T(\bar{\mathbf{u}}^n\, \mathbf{d}^n) - \mathbf{D'_t}^T \mathbf{d}^n - \mathbf{E}^T \mathbf{y}^n)) \\
\bar{L}^{n+1} = 2L^{n+1} - \bar{L}^n \\
\mathbf{u}^{n+1} = (\mathbf{I} + \mathbf{T_2}\lambda_4 \partial G)^{-1}(\bar{\mathbf{u}}^n - \mathbf{T_2}(\mathbf{D_x}^T \mathbf{a}^n - \mathbf{D}_t^T \mathbf{b}^n))) \\
\bar{\mathbf{u}}^{n+1} = 2\mathbf{u}^{n+1} - \bar{\mathbf{u}}^n \\
\mathbf{a}^{n+1} = (\mathbf{I} + \mathbf{\Sigma_1} \partial F_1^*)^{-1}(\mathbf{a}^n + \mathbf{\Sigma_1 D_x} \bar{\mathbf{u}}^{n+1}) \\
\mathbf{b}^{n+1} = (\mathbf{I} + \mathbf{\Sigma_2} \partial F_2^*)^{-1}(\mathbf{b}^n + \mathbf{\Sigma_2 D}_t \bar{\mathbf{u}}^{n+1}) \\
\mathbf{c}^{n+1} = (\mathbf{I} + \mathbf{\Sigma_3} \partial F_3^*)^{-1}(\mathbf{c}^n + \mathbf{\Sigma_3 D'_x} \bar{L}^{n+1}) \\
\mathbf{d}^{n+1} = (\mathbf{I} + \mathbf{\Sigma_4} \partial F_4^*)^{-1}(\mathbf{d}^n + \mathbf{\Sigma_4}(\langle \mathbf{D'_x} \bar{L}^{n+1}, \bar{\mathbf{u}}^{n+1} \rangle + \mathbf{D'_t} \bar{L}^{n+1})) \\
\mathbf{y}^{n+1} = (\mathbf{I} + \mathbf{\Sigma_5} \partial F_5^*)(\mathbf{d}^n + \mathbf{\Sigma_5}(\mathbf{E}\bar{L}^{n+1} - \mathbf{z})),
\end{cases}
$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{\Sigma_i}$ and $\mathbf{T_i}$ are diagonal pre-conditioning matrices as described in Section 3.2.3 [73]. The preconditing helps to define a optimal step-size for the irregular event matrix $\mathbf{E}$ without determining its norm, which otherwise would significantly increase the computational cost of the algorithm.

Following the notation in [16], F_i^* represents the indicator functions and G is the optical flow term. Their respective resolvent operators are defined by Equation (3.21). $(\mathbf{I} + \mathbf{T_1}\partial h_\theta)^{-1}$ is the resolvent operator with respect to h_θ, which can be solved by a

soft-thresholding scheme for each log intensity estimate L_i:

$$(\mathbf{I} + \lambda_6 \tau_i h_\theta)^{-1}(\tilde{L}_i) = \tilde{L}_i + \begin{cases} -\lambda_6 \tau_i, & \text{if } (\tilde{L}_i - L(t_p)) > \theta + \lambda_6 \tau_i \\ \lambda_6 \tau_i, & \text{if } (\tilde{L}_i - L(t_p)) < -\theta - \lambda_6 \tau_i \\ 0, & \text{otherwise,} \end{cases} \tag{4.8}$$

where we fix $L(t_p)$ during the minimisation step. A fixed $L(t_p)$ can slow down the convergence of the algorithm, but in practice we have not experienced such behaviour. Next, we discuss the result of this minimisation scheme.

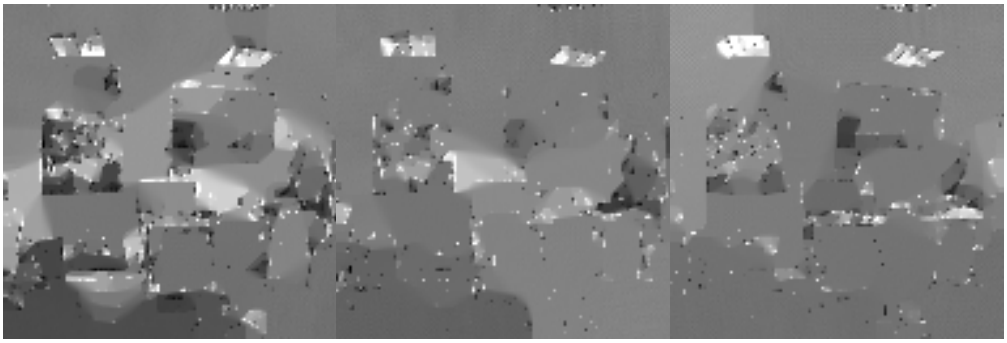
4.3 Experiments

We present the capabilities of the method in a series of experiments with the DVS128 camera [53]. For these experiments, we choose a spatial discretisation of 128×128, which corresponds to the resolution of the actual camera. To highlight the results, we show log-intensity and velocity field estimates at certain time-stamps to highlight how they resemble images and optical flow fields from standard cameras. owever, we believe that these results are best viewed in the accompanying video on our project webpage [1], where also show that we can estimate super-resolution log-intensity and velocity, via a simple extension to our formulation based on the work of Unger *et al.* for standard cameras [96]. allows us to perform intensity and velocity estimation at sub-pixel resolution.

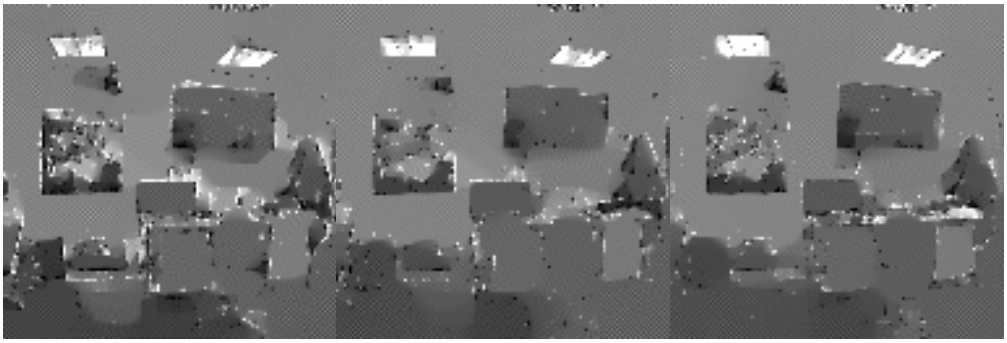
For the following experiments, we set the sliding window depth K to 128, which we have found to be an appropriate choice for most sequences to capture a large amount of events. A smaller number of events in the sliding window is sufficient, but this reduces detail in the intensity estimate. If not specified otherwise, δ_t is set to 15 milliseconds. During experiments, we observed that 15 milliseconds gives a good result for most sequences, but for more rapid motions smaller values of δ_t are required. For all sequences we set $\theta = 0.22$, $\lambda_1 = 0.02$, $\lambda_2 = 0.05$, $\lambda_3 = 0.02$, $\lambda_4 = 0.2$, $\lambda_5 = 0.1$ and $\lambda_6 = 1.0$. All sequences are initialised by assuming no initial motion and only a uniform greyscale intensity distribution, which includes the prior image as well. For comparison, we mounted the DVS128 next to a standard frame-based camera with standard 640×480 resolution, 30 fps and global shutter settings for all sequences.

[1] http://www.imperial.ac.uk/dyson-robotics-lab/

(a) Camera image



(b) Without flow 1 (c) Without flow 2 (d) Without flow 3



(e) With flow 1 (f) With flow 2 (g) With flow 3

Figure 4.4: Comparison of intensity estimation without (b, c, d) and with (e, f, g) the optical flow term. In a we show an image from a standard camera for reference.

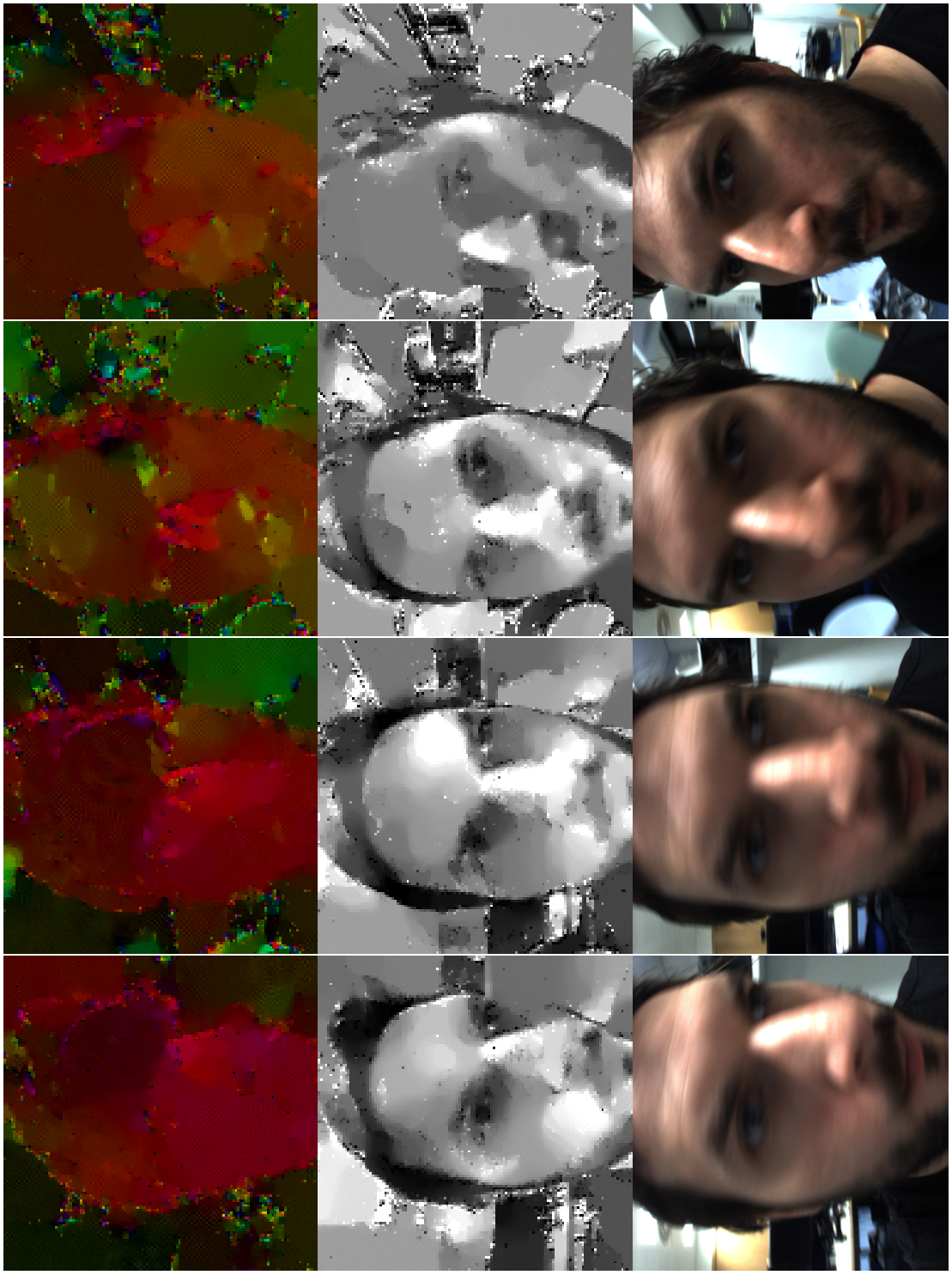


Figure 4.5: Hand-held event camera moving around a person's head. Here the face is moving from left to right, while the background is moving in the opposite direction, which can be seen in the estimated velocity field (top row). The middle row shows high quality and consistent reconstructed intensity. Bottom row: standard video for comparison.

4.3.1 The Benefits of Simultaneous Estimation

We begin with an experiment to argue for the simultaneous estimation of intensities and optical flow with event cameras. In this sequence, we compare the intensity estimate of our method with and without using the optical flow term as defined in Equation (4.5) (Figure 4.4).

From these results, we can see that our method without the optical flow is still able to estimate the intensities well in regions with strong gradient, but between those regions artefacts occur which do not correspond with the real intensities in the scene. Also, we see that the term enforces temporal consistency and without it other image areas become brighter or darker from frame to frame. In contrast to this, the sequence with the term enabled is much more consistent.

4.3.2 Face Sequence

In this sequence, we show intensity and velocity reconstruction of a moving face while the camera is in motion as well. We demonstrate the ability of our method to handle motion discontinuities, while also recovering consistent intensities. The results are shown in Figure 4.5.

At the beginning of the sequence (left), the intensities have not been properly estimated yet, because only a few events have been captured. But even without these events, stronger gradients are recovered and provided an estimate of the motion. We see that more details become visible in the following frames as more events are processed. Velocity visualisation shows clear motion boundaries between the head and the background, proving that our approach can handle motion discontinuities. However, both the intensities and optical flow show noise, presumably caused by outlier events and/or missing data.

4.3.3 High Dynamic Range Scene

In this example, we show a comparison between our method and a traditional camera in a high dynamic range scene. In our experiment, we point the cameras out of a window from a dim room, which is a challenging case for a traditional camera, as can be seen in Figure 4.6.

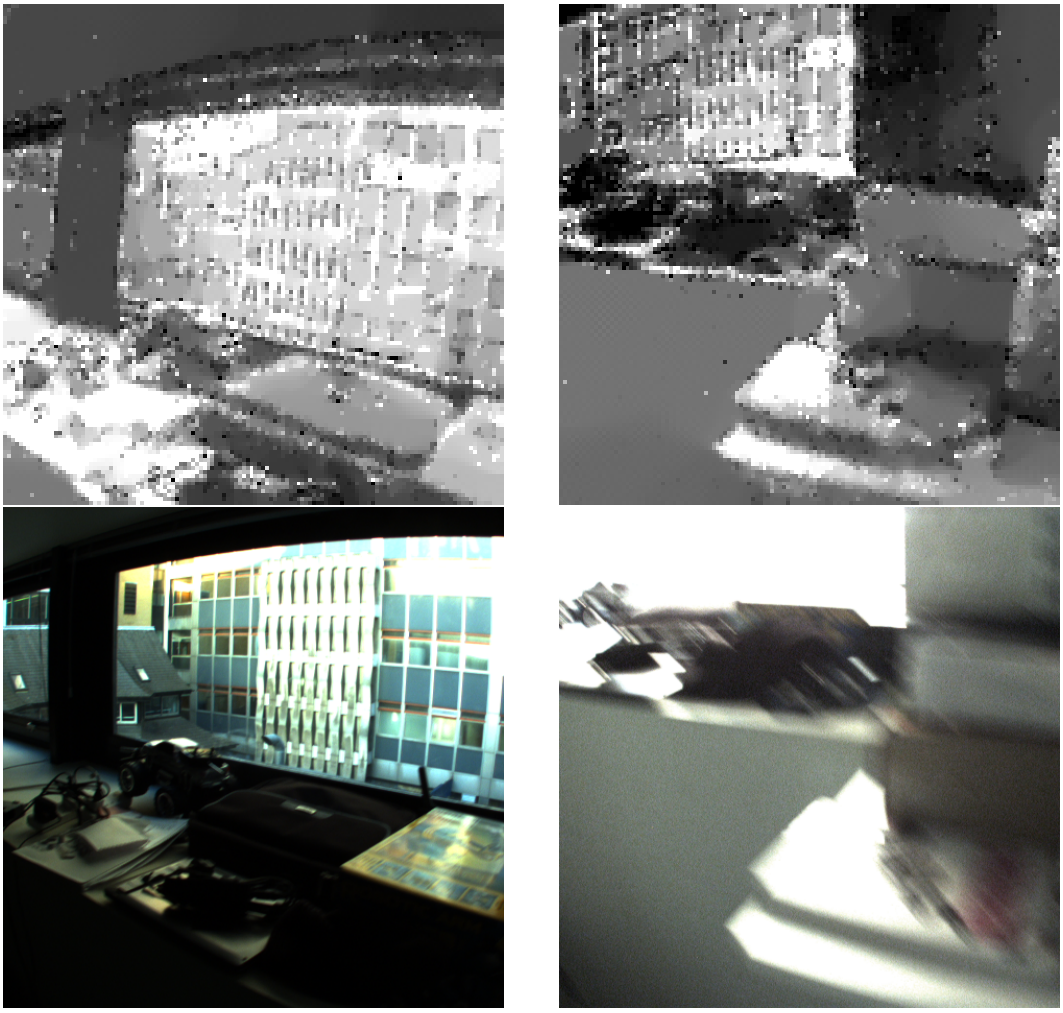


Figure 4.6: High dynamic range scene. Intensity reconstructions from events (left) contrasted with standard video images (right). The camera moves from observing the bright outside scene to point towards a shelf inside the dim room.

We see that our method recovers details both inside the room and outside the window, while the traditional camera, because of its low dynamic range, can only show either the room or the outside at one time. However, some finer details are not visible due to low resolution.

Figure 4.7: Ball thrown across a desktop scene. In the estimated velocity field (top) we see a clear segmentation of the ball, while video from a standard camera (bottom) is heavily blurred. In the intensity reconstruction (middle) we observe good reconstruction of the monitor which was traversed by the ball, causing many events.

4.3.4 Rapid Motion

Here we present the capability of our method to estimate fast motion in front of a cluttered background. We throw a ball in front of a desktop scene, while the camera is also in motion (Figure 4.7).

For this sequence, we set δ_t to a smaller value of 4 milliseconds. We see how the traditional camera is affected by motion blur, while our method is able to recover clear motion boundaries. However, due to the small δ_t, intensity details are not estimated as well as in the previous examples. This gives the impression that the ball is transparent.

4.3.5 Full Body Motion

In our last example, we show a person performing jumping jacks. For this sequence, we set δ_t to 7 ms and reduce λ_1 and λ_3 to 0.01, which preserves smaller regions from being smoothed out in the optical flow and intensity estimate. In Figure 4.8, we see that our method can estimate arm motion well, even though it occupies a small image region. However, with the decreased smoothness weight, the influence of noisy events is stronger, which becomes visible in the motion field.

4.4 Conclusions

We have shown that event data from a standard DVS128 sensor can be used to reconstruct high dynamic range intensity frames jointly with a dense optical flow field. As demonstrated experimentally, our sliding window optimisation based method does impose any restrictions on camera motion nor scene content. In particular, we have shown tracking and reconstruction of extreme, rapid motion and high dynamic range scenes which are beyond the capabilities of frame-based cameras. We thus believe that this work is important in supporting the claim that event cameras will play a major role in real-time geometric vision — the information in a low bit-rate event stream really does contain all of the content of a continuous video and more.

We also believe that this work strengthens the argument that research on embodied real-time vision needs to look at all of the components of a hardware/software vision system together (sensors, algorithms and ultimately also processors) in order to reach for maximum performance. Progress in the specific technology of event cameras must be

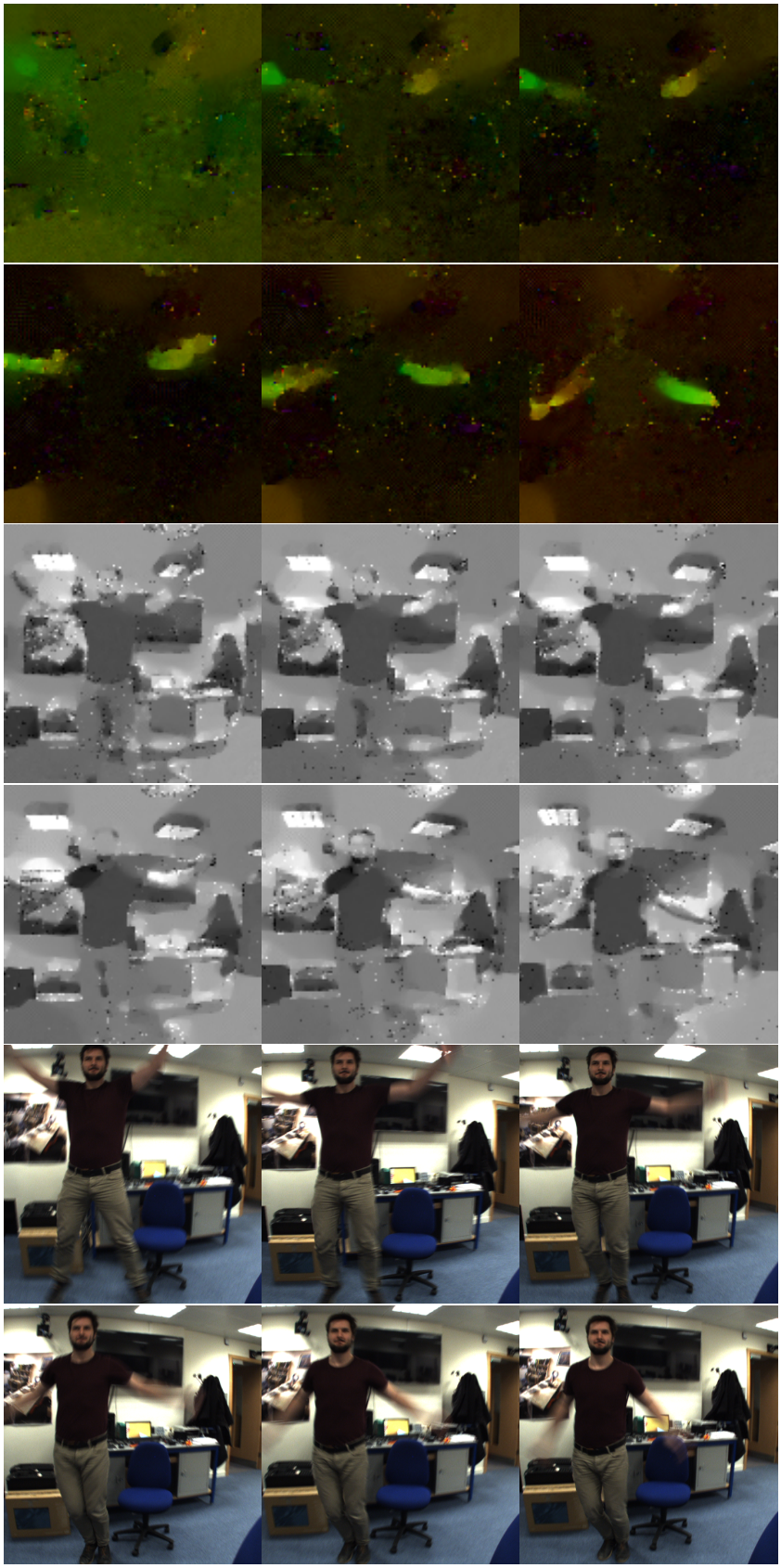


Figure 4.8: Jumping jacks. In the sequence from a standard camera the arms are blurred, while our reconstruction from events allows clear delineation of the arms in both the intensity and velocity fields.

compared with alternatives, always with an eye on the rapidly improving and strongly market-driven performance of standard frame-based cameras.

Chapter 5

Deep Learning for Image Generation

Contents

Artificial neural networks (ANN) have been a research subject for many decades but only in recent years has their progress started to outperform previous state-of-the-art methods in many areas, including computer vision. In computer vision, ANNs the current method of choice for object recognition, semantic segmentation, image restoration and so on, in which ANNs demonstrate an ability to handle the complex appearance variations of objects. The success of ANNs suggests that these networks can learn an image representation which expresses the variations in objects and image appearances. The representation suggests that ANNs are suitable to express the complex image statistic

of natural images better than traditional, hand-crafted methods. If such a representation could be transferred to event cameras, it would allow us to complete the missing information in the event stream and give any application a better understanding of the event signal. In this chapter, we give a detailed introduction to generative adversarial networks and a background into their architecture design as well as method to train them. At the end of the chapter, we give an introduction to adversarial losses, which combine the principle of GAN training methods with a handcrafted loss function, which we use in the next chapter. However, before we approach the combination of a neural network and event camera, let us first introduce the general concepts of ANNs in this section.

5.1 Introduction to Neural Networks

The interactions between cells in a neural network are a complex, biochemical process which ANNs are modelled after, but with strong simplifications. These simplifications are necessary for ANNs to be trained with well-established optimisation methods, but they also weaken the biological justification of ANNs. Despite these differences, ANNs can be successfully trained in practice and have shown remarkable performance in recent years. In this section, we introduce formally the concept of artificial neurons and network architectures which we will use in the later chapters.

5.1.1 Neurons, Connections and Activations

Both artificial and biological neurons react to external stimuli and send out signals if the stimulation reaches a certain amount. The stimuli often originate from other neurons and the outgoing signal is sent to other neurons as well. All of these neural connections form a complex network of neural interaction. The difference between biological and artificial neurons is that the biological signal is a discrete electrical impulse, while the artificial signal is usually a continuous function. This continuity allows the neural function to be differentiable and therefore trainable by gradient-based methods.

Formally, the goal of a neural network is to approximate a function $\bar{F}$ which maps from a domain X to the desired domain Y such that $\bar{F}(\mathbf{x}) = \bar{\mathbf{y}}, (\mathbf{x}, \bar{\mathbf{y}}) \in X \times Y$. The

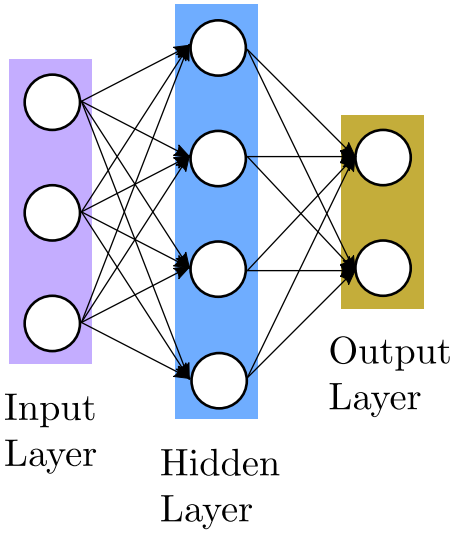


Figure 5.1: Structure of a fully connected artifical neural network. The circles indicates neurons and their signal directions are illustrated as arrows.

network approximates $\bar{F}$ such that

$$\mathbf{y} = F(\mathbf{x}; \theta), \tag{5.1}$$

where F is our network function which is parameterised by vector θ, which is network parameters which provide the best approximation of $\bar{F}$ such that $\mathbf{y} \approx \bar{\mathbf{y}}$. Usually, the goal in machine learning is to learn the optimal parameters θ using only a small subset $S \subset X \times Y$, which is called the *training dataset* and we refer to $\bar{\mathbf{y}}$ from now on as ground truth labels [50].

The network function F describes our neural network as a whole, but a network is normally segregated into several layers. Each layer consists of neurons which are not connected to each other but share the same input and output layers. The most common form to structure the layers of an ANN is called *feedforward neural networks*. A feedforward network connects the layers such that they form recursive connections. As an example for a feedforward network, let F_1, F_2, and F_3 be the functions corresponding to the layers of a three-layered neural network such that the composite $F(\mathbf{x}) := F_3(F_2(F_1(\mathbf{x})))$ is the complete network function, where the output of the first layer feeds into the second layer and the second layer into the third layer. These chain structures are the most

commonly used architectures in ANNs. The number of layers is considered to be the *depth* of a neural network and deep learning studies network architectures with several layers. Figure 5.1 shows an example of such a neural network. There is a distinction in the terminology for the first and last layer of a neural network, which are the *input layer* which receives the input of the neural network and the *output layer* which maps to the desired output. The layers between input and output layers are often referred to as hidden layers [50].

Up to this point, we have discussed the overall structure of an ANN to provides an overview, but in detail, the networks are *neural* since they take inspiration from biological neuron cells. As mentioned above, this inspiration is weakened by simplifications, but the working principles are similar; a neuron takes an incoming signal and maps it to a non-linear function by either amplifying it or suppressing it. Formally, we express the incoming signals to a neuron as a vector $\mathbf{x}$ and the neuron model is then defined as

$$f(\mathbf{x}; W, \mathbf{b}) := g(W\,\mathbf{x} + \mathbf{b}), \tag{5.2}$$

where W is the weight matrix, $\mathbf{b}$ is the bias vector and g is the activation function. The weight matrix can be viewed as modelling the weighted influences of the incoming signals and the bias as the sensitivity of the neuron toward the incoming signals. From Equation (5.2) we can see that the expressiveness of an ANNs is given by the choice of its activation functions. For instance, if g is chosen to be the identity function such that $g(x) := x$, then the layer f defines an affine transformation and therefore the network can only approximate an affine transformation as well. In machine learning, a number of different activation functions have been studied such as

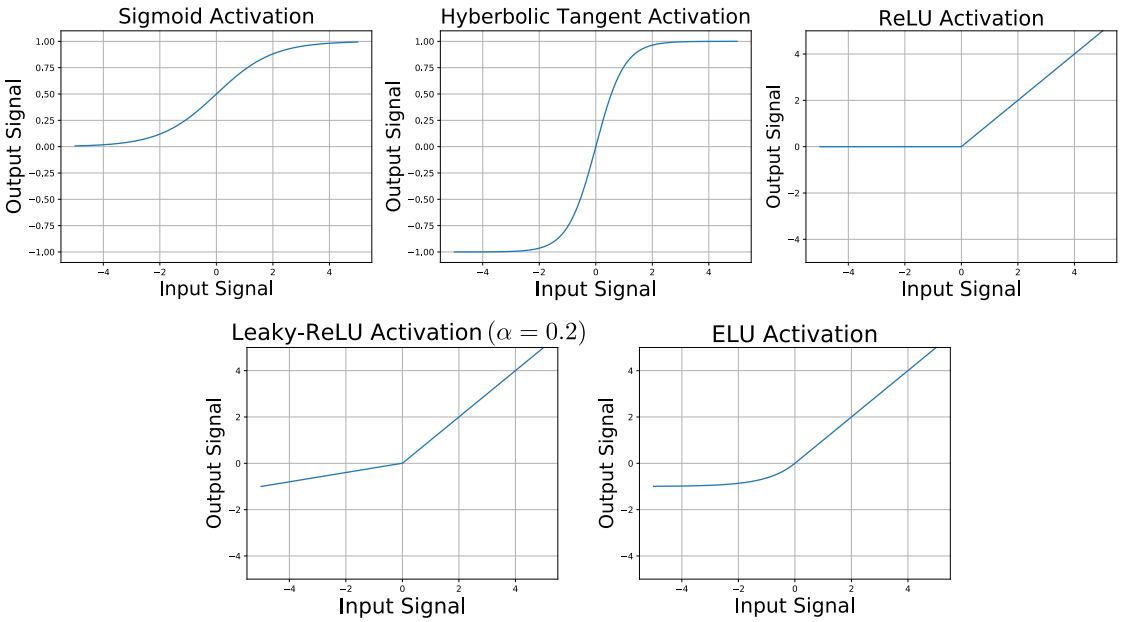


Figure 5.2: Typical activation functions used in neural networks.

$$\text{Sigmoid: } g_1(x) := \frac{1}{1+e^{-x}} \tag{5.3}$$

$$\tanh\text{: } g_2(x) := \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5.4}$$

$$\text{ReLU: } g_3(x) := \max(x, 0) \tag{5.5}$$

$$\text{Leaky-ReLU: } g_4(x) := \max(\alpha\, x, x),\ \alpha \in (0, 1] \tag{5.6}$$

$$\text{ELU: } g_5(x) := \begin{cases} x & \text{if } x \geq 0 \\ e^x - 1 & \text{if } x < 0. \end{cases} \tag{5.7}$$

These five examples are the commonly used activation functions for ANNs, though there any many more choices than these four functions. Figure 5.2 shows the plots of the five functions.

In recent years, the *rectified linear unit* (ReLU) function has become commonly used as the activation function in much machine learning work, and seems to improve performance over sigmoid and tanh functions. The sigmoid function suffers, like the tanh function, from *vanishing gradients*, while the ReLU function does not [50]. Vanishing

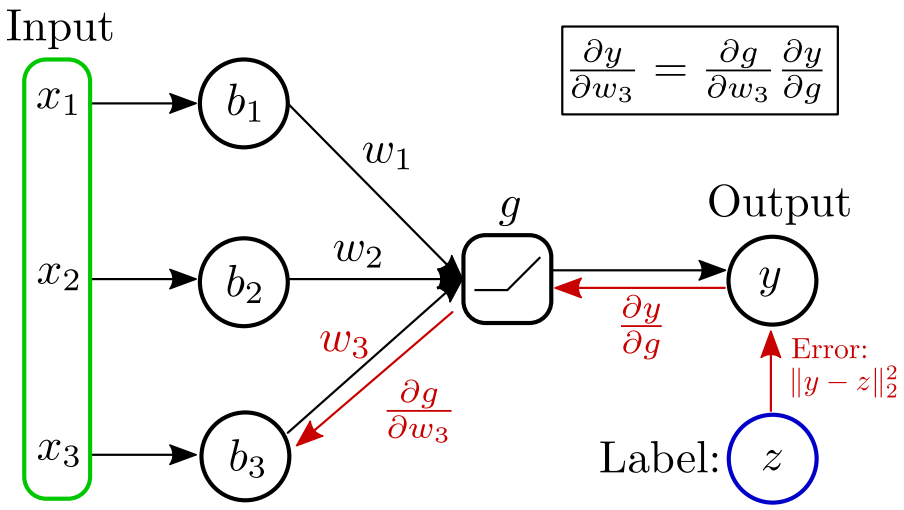


Figure 5.3: Back-propagation with the forward pass through the activation function g marked as black arrows and the backward pass as red arrows. Backward pass shows the update direction for network parameter w_3.

gradients is a numerical problem in optimisation when the input of activation approaches $-\infty$ or ∞ and the gradient becomes close to 0, while for the ReLU gradient is constant if the input > 0. ReLU also relates more closely to biological neurons, which suppress the incoming signal until the neural threshold is exceeded. However, for optimisation purposes, this means that the gradient of a suppressed signal is zero, which in turns means that the parameters of that signal are not changed in the optimisation step. For that reasons, the Leaky-ReLU function was proposed which downscales the signal by a chosen α. These developments of activation functions are still ongoing research, which driven by empirical evaluations.

5.2 Network Training

Typical artificial neural networks for deep-learning are composed of millions of parameters which are initialised before training with sampled values from a random distribution or with values which have been selected by other means (e.g. through pre-training) [50]. In general, these initial values need to be trained (i.e. optimised) to perform optimally for a given network objective. In this section, we introduce the fundamentals of network training methods. For more a thorough review of these methods, we refer to the deep-learning literature [50].

Traditional optimisation and network training methods optimise the loss of an optimisation problem but with different goals. For traditional optimisation methods, the goal is the optimisation of the loss function itself. However, for network training methods the goal is to optimise a measure of generalisability which is different from loss function. The idea is to train the network by using examples from the training dataset, but then test the performance of the network with unseen examples. By using examples which are not included in the training dataset, the evaluation tells us how well the network learned the desired function, and it indicates if the network starts over-fitting to the training examples [50]. Over-fitting describes the undesired behaviour of the neural network during training when the network performs well only training examples but fails on novel examples, which indicates that the network fails to generalise. With the network objective described, we continue with the training method in the following section.

5.2.1 Backpropagation

The challenge for neural network training is the large number of network parameters and the non-linearity of the activation function which makes the optimisation task challenging. However, in practice, gradually adapting the network parameter through *backpropagation* has proven to be a successful strategy [50].

Backpropagation describes an iterative optimisation process which updates the network parameters in a similarly to gradient descent, which we described in Section 2.1.2. Each backpropagation step is sub-dived into three steps: A forward pass which determines the network error, a backward pass which produces the network gradients and weight update which adapt the network parameters. During training, these steps are repeated until convergences. An overview of the forward and backward pass is given in Figure 5.3. The figure shows the calculation of the gradient of a weight parameter which is used to update the weight parameter, which we describe in the following.

Formally, let us assume that F_i is the i-th layer of a N-layered neural network which is define by Equation (5.2). We simplify the layer equation for clarity by omitting the bias parameter such that:

$$F_i(x_{i-1}; W_i) := g(W_i x_{i-1}) \tag{5.8}$$

where x_i is the output of i-th network layer with x_0 being the network input, W_i is the network parameters of the i-th layer and g is the activation function. Next, let us assume that our training objective is defined such that:

$$\mathcal{L} := \min_{W_1,\ldots,W_N} \frac{1}{2} \sum_{j=1}^{M} \|F(y_j; W_1, \ldots, W_N) - z_j\|_2^2, \tag{5.9}$$

where F is network function of all concatenated network layers F_i, z_j is the j-th ground-truth label from the training dataset which corresponds to the network input y_j. M is the number of training pairs in our training dataset.

Now, with the training objective $\mathcal{L}$ defined, we can derive the gradients for the backward pass for each network layer. The gradient of the output network layers is defined such that:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W_N} &:= \sum_{j=1}^{M} (x_N^j - z_j) \frac{\partial x_N^j}{\partial W_N} \\
&= \sum_{j=1}^{M} [(x_N^j - z_j) \circ F_N'(x_{N-1}^j; W_N)] \frac{\partial W_N x_{N-1}^j}{\partial W_N} \\
&= \sum_{j=1}^{M} [(x_N^j - z_j) \circ F_N'(x_{N-1}^j; W_N)] (x_{N-1}^j)^\top,
\end{aligned} \tag{5.10}$$

where $\circ$ is the Hadamard product, or the element-wise multiplication operator, and x_i^j is the output of the i-th layer given y_j as the network input. The other network layers are derived analogously such that:

$$\frac{\partial \mathcal{L}}{\partial W_i} := \sum_{j=1}^{M} [W_{i+1}^\top \delta_{i+1}^j \circ F_i'(x_{i-1}; W_i)] (x_{i-1}^j)^T, \tag{5.11}$$

with δ_i^j defined for each layer of the N-layered neural network such that:

$$\delta_i^j := \begin{cases} W_{i+1}^T \delta_{i+1}^j \circ F_i'(x_{i-1}^j; W_i), & \text{if } i < N \\ (x_N^j - z_j) \circ F_N'(x_{N-1}^j; W_N), & \text{if } i = N \end{cases} \tag{5.12}$$

With the gradient for each network layers defined, we can then formulate the update rule the network parameter of the i-th layer W_i such that:

$$W_i \leftarrow W_i - \lambda \delta_i \frac{\partial \mathcal{L}}{\partial W_i} \tag{5.13}$$

where $\lambda > 0$ is the step size or *learning rate*. We like to note, that for the update of the W_i network parameter, we assume that all other layers are constant for the update step.

For training neural networks large, training datasets are often required with millions of training examples. Computing the network gradients as defined by Equation (5.11) with respect to all training examples is often too computationally expensive to be calculated for every training step. To lower the computational burden, it common to compute the gradient with respect to the expected training loss which is called *Mini-Batch Stochastic Gradient Descent* (SGD), which we described in Section 2.1.2. In this work, we refer from now on to SGD when we describe network training approaches.

With the general network structure and training method described, we next introduce convolutional neural networks which have been particularly successfully applied in computer vision.

5.3 Introduction to Convolutional Neural Networks

Arguably, the *convolutional neural network* (CNN) [49] is so far the most influential network structure introduced in machine learning, and has formed the basis of machine learning development in recent years. The CNN design is inspired by the visual cortex of the brain and has been used in machine learning for decades [51], but it was not until the works of Krizhevsky *et al.* [44] in 2012 that CNNs received wider attention from the research community and from computer vision in particular. In their work, the authors took advantage of the computational power of GPUs to reduce the training time from days to hours while allowing an increase in the number of network layers, which spawned the area of deep learning. Before their contribution, CNNs were limited in size and training time, taking days for even moderately sized networks due to computational limitations at the time.

Ulyanov *et al.* [95] recently suggested that a CNN's structure itself is particularly well-

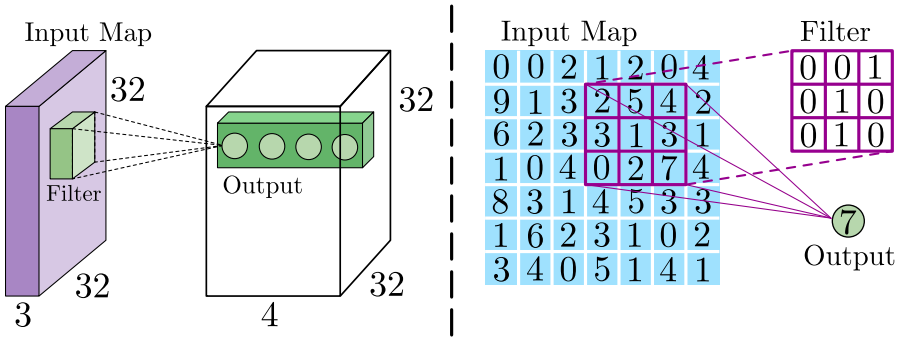


Figure 5.4: Example of convolutional neural network architecture. The left side is a structural overview, where the input map is convolved by the filter and provides 4 output signals per convolution step. The right side shows an example of the convolution process. For simplicity, we show only a 2-D input map and a scalar output.

suited for describing the image statistics. They demonstrated this by training a CNN for image denoising and inpainting using a single image for training without labels. The results of this training suggest that the self-similarity of images is utilised by the CNN structure. Because of this ability and the performance in various vision-based applications, we rely on CNNs for the work of this thesis and detail their architecture in the following.

The CNN approach does not fundamentally change the neuronal function, but it drastically reduces the number of neurons needed and their receptive fields by sharing neuron weights across the input which effectively reduces the number of training parameters. In CNNs, the neuron weights of a layer are defined as a special kind of linear operation: A convolution which replaces the weight matrix in Equation (5.2). Figure 5.4 shows an overview of CNNs. The idea is that the weights are applied to a local patch of the input, shifted and then applied again until they have been applied at every position of the image. The results of the convolutional layer are stored in a so-called *feature map* which forms the layer output and the input for any following layer.

A discrete 1-D convolution is defined as:

$$f(u) := (\mathbf{x} * \mathbf{w})(t) := \sum_{i=-\infty}^{\infty} x_{u-i} w_i, \tag{5.14}$$

where u is the position of the input $\mathbf{x}$ which should be convolved and $\mathbf{w}$ is a kernel with infinite size, which means that the convolution incorporates all elements of $\mathbf{x}$. CNNs however often use a filter kernel much smaller than the input, which essentially limits the receptive field of the neurons. For a kernel of size $2n+1$, Equation (5.14) changes to $f(u) := \sum_{i=-n}^{n} x_{u-i} w_i$. The one-dimensional convolutional case is often used in applications like audio processing, but convolutions can be applied to higher dimensional inputs like images or videos. The 2-D case is defined for kernel size $(2m+1, 2n+1)$ as:

$$f(u,v) := (I * W)(u,v) := \sum_{i=-m}^{m} \sum_{j=-n}^{n} I_{u-i,v-j}\, w_{i,j}, \tag{5.15}$$

where I is the 2-D input (e.g. image) and W is the kernel weight matrix. The kernel size limits the receptive field of a neuron to its neighbourhood region, which in turns requires from the input that relevant information is spatially close, which is the case for images. For example, the features of a face, like the eyes and nose, are close to each other and thus a face detection network needs only their spatial relationship to detect the face, while the rest of the image is irrelevant. Because the weights of the same neurons are applied across the image the network weights are trained to be spatially invariant, which means that the location in the image, for instance, does not affect the detection of an object. In general, the first layers of a CNN learn to perceive local features of an image like edges and combine them in high-level features in later layers for object descriptions. This intuition is supported by the work of Zeiler *et al.* [104], who visualised the different feature layers of the network, showing this hierarchical behaviour.

With the network layers described, we next provide an example in which layers are organised into a complete neural network. We provide this example in the context for one of the most common and simple training tasks, which is known as autoencoders network training.

5.4 Autoencoder Networks

Neural networks have the ability to express complex image statistics in a compact manner which is can be practically demonstrated by the *autoencoder* (AE) neural networks [11, 30]. An AE is a special type of neural network which learns the identity function for

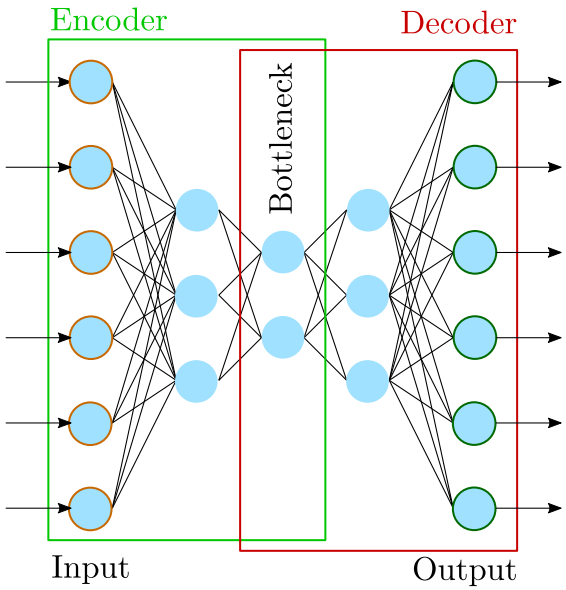


Figure 5.5: Structure of a fully connected neural network. The network has a hour-glass structure which reduces the number of neurons until the bottleneck in the encoder part. In the decoder part the number again expands.

a given dataset to learn expressive features from that dataset. The networks are often augmented with constraints during training which encourage the networks to learn such features. Since AEs only reproduce their input signal they can be trained without labelled data, which makes them useful for pre-training purposes for other learning tasks.

Pre-training is a processing step used in machine learning before the actual training which helps in speeding up convergence and in avoiding the risk for the network to fall into a local minimum. For pre-training, a network is trained first for an another, often simpler task and then the trained weights, or a subset of them, are further trained for the actual task. AEs are a popular pre-training approach since they require no labelled data and help to express the dataset with meaningful features.

A common constraint for AEs to learn these features is to use an hourglass-like structure for the network, which is depicted in Figure 5.5 [50]. The hourglass structure is split into two parts: The *encoder* network which reduces the input data to a lower-dimensional space, and the *decoder* network which reconstructs the input signal from this lower-dimension space. The projection of the encoder network to a fewer number

of neurons can be interpreted as a compression task. The decoder network then decompresses the information to the original signal. By using a lower-dimensional space between the encoder and decoder network, the encoder needs to learn the most essential features for the projection for a successful reconstruction by the decoder network.

Because of the compression, the hourglass structure is a commonly used network structure in machine learning and various variants of it have been proposed. For instance, a common choice for images are *convolutional autoencoders* (AE-CNN) which use convolutional layers instead of fully connected network layers [50]. Figure 5.6 shows an example AE-CNN. The convolutional layers reduce the number of parameters in the network and allow the network to exploit the self-similarity which often occurs in images. An AE-CNN reduces the resolution in the different network layers by increasing the step-size of the convolutional kernel. For example, if the kernel step-size is set to 2, then the convolutional kernel skips every other pixel location in the image, which results in an output feature map which half the size of the input image. The resolution is increased again in the decoder part of the network by using transposed convolutional layers, which are commonly used for upsampling network signals [66]. These layers are similar to convolutional layers, except that they map a single pixel region into a larger image region of the size of the convolutional kernel.

With typical AE network structures described, we discuss AE training next. Formally, let us define an autoencoder function A which parameterised by θ which is trained by minimising the loss function:

$$\min_{\theta} \|A(x;\theta) - x\|_2^2, \tag{5.16}$$

where x is the network input. For the reconstruction loss, the squared norm is often used, however, different norms can be used as well.

Beside learning identity function, AE can also be used for learning invariant features to certain data augmentations or transformations. For example, adding noise to the input is a common augmentation to train the AE to be robust against those influences and recover a noise-free version, which is often referred to as *Denoising Autoencoders* (DAE) [50]. Similarly to DAEs, other augmentations can be applied like random transformations. For instance, if a dataset is available with centred objects then training the AE to revert

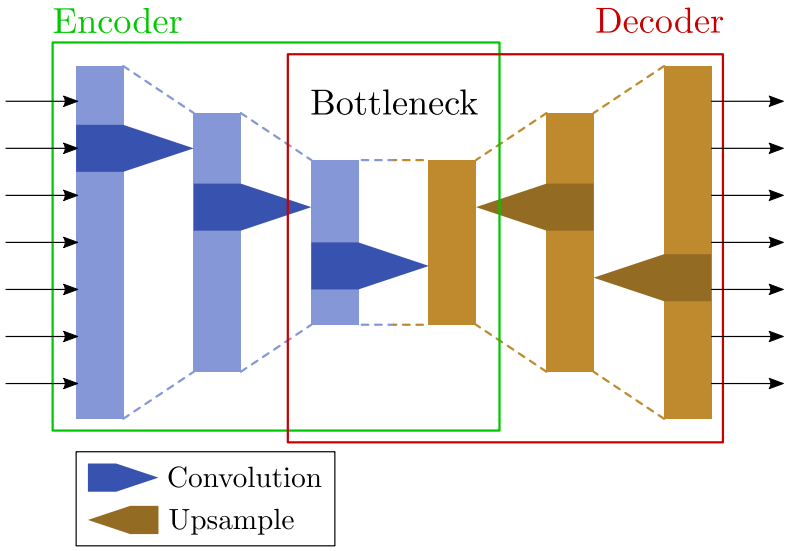


Figure 5.6: Structure of a convolutional autoencoder network which follows an hour-glass structure. The resolution of the network is reduced until the bottleneck in the encoder part and expands again in the decoder part.

random shifts learns a canonical representation which has an object at its centre.

We introduce with AEs a comparatively simple network learning tasks, however, for our event-based analysis in later chapters we are using a more advanced learning method of using adversarial networks.

5.5 Adversarial Neural Network

Using adversarial neural networks is a recently proposed technique for training generative neural networks (GAN) [26], which briefly introduced in Section 1.5. For the GAN training, the adversarial network, also called a *discriminator*, is trained together with a generating network (generator) and is tasked differentiate between network generated samples and real examples from a provided dataset. By classifying real and fake examples, the discriminator provides the generator network with information on how to generate more convincing samples at the same time, which is the core idea of GANs.

Adversarial network training can be related to a real-world example of the way humans learn certain tasks. For example, if a person is playing chess, he or she gets better by playing against other competitors. A player usually chooses opponents, who have a

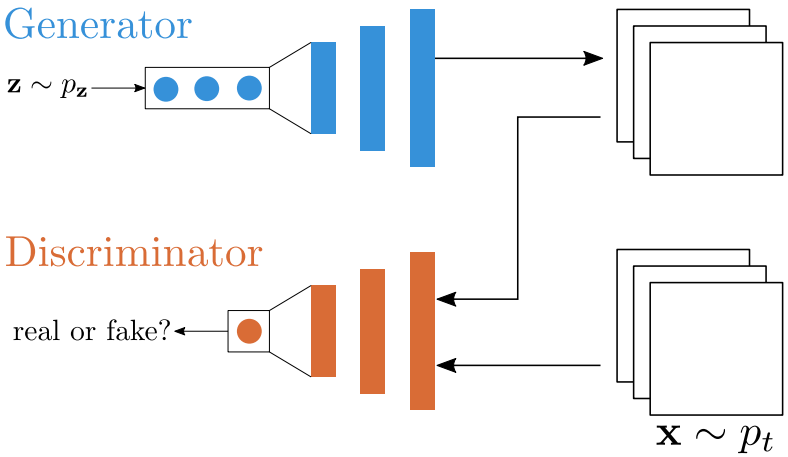


Figure 5.7: Overview of GAN training. The Generator takes randomly sampled values $\mathbf{z}$ from $p_{\mathbf{z}}$ and maps from a fully connected layer and a series of convolutional layers to generate the samples. The discriminator takes the generated and real samples from p_t and learns to determine which is real and which is the fake example.

similar level of expertise. As their skill improves, stronger and stronger opponents are challenged. This principle of improving by competing against an increasingly stronger opponent is the basis for GANs.

The power of GAN training is that it is able to learn a distribution and to generate convincing samples from it. In various applications, GANs demonstrated an impressive capacity to learn the complex distribution of natural images by generating faces [77] and changing the natural appearance of images [35]. The generator network learns these appearances by learning to mimic a given target distribution. How well the generator can mimic the distribution is determined by the discriminator which labels the generated samples and the samples from the target distribution. The labelling provides the generator with a distance to the target distribution, which the generator tries to diminish by creating more and more realistic looking samples while the discriminator is working against the generator, which leads to a min-max optimisation scheme. At the end of training, the generator and discriminator reach a stable equilibrium between generator and discriminator, in which the generator creates samples which are hard to distinguish from real samples.

Formally, we define the above scheme with two neural networks D and G, which are the discriminator and generator and p_t is the target distribution. D maps its input to

a single neuron, which is usually truncated by a sigmoid function and is interpreted as a probability since the sigmoid is bounded between 0 and 1. The generator G maps a sampled input $z \sim p_z$ to the domain of D, where p_z is a random distribution, which is usually chosen to be a uniform random distribution. The objective of GAN training is then defined as:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{x \sim p_t}[\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z; \theta_G); \theta_D))], \tag{5.17}$$

where θ_G and θ_D are the network parameters. The Objective (5.17) is based on the Jensen-Shannon distance, which is a symmetric version of the Kullback-Leibler divergence which is defined for two probability distributions p and q:

$$\mathrm{JSD}(p||q) := \int p(x) \log \frac{2p(x)}{p(x) + q(x)} + q(x) \log \frac{2q(x)}{p(x) + q(x)} \mathrm{d}x. \tag{5.18}$$

Setting $p(x) := D(x; \theta_D)$ and $q(x) := 1 - D(G(x; \theta_G); \theta_D)$ into Equation (5.18) simplifies the expression to Equation (5.17) [26]. Forming the gradient of Equation (5.17), we can see that the gradient for the generator updates includes the discriminator, which indicates how the discriminator provides information to the generator to improve its perfromance. To explicitly show this, we separate Equation (5.17) into the two objectives of the generator and the discriminator such as:

$$\begin{cases} \mathcal{L}_D := \mathbb{E}_{x \sim p_t}[\log(D(x; \theta_D))] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z; \theta_G); \theta_D))] & \text{for } \theta_D \\ \mathcal{L}_G := \mathbb{E}_{z \sim p_z}[\log D(G(z; \theta_G); \theta_D)] & \text{for } \theta_G. \end{cases} \tag{5.19}$$

The gradient of the generator loss $\mathcal{L}_G$ is then defined as

$$\nabla \mathcal{L}_G = \mathbb{E}\left[\frac{\nabla D \nabla G}{D(G)}\right], \tag{5.20}$$

where we omit the network parameters for brevity. However, we can see that the gradient of the generator ∇G is projected onto ∇D which gives the update of G the information to improve its performance. In other words, the generator utilises the gradient information from the discriminator during training to improve its results and produce more

convincing samples. The success of this training scheme depends on the capacity and design of the discriminator. However, the discriminator network provides no guarantees about its gradient which might cause numeric problems during the minimisation, like vanishing or exploding gradients, which imply gradient values outside the bound of machine-precision. Another issue is the balance between the discriminator and the generator. If the discriminator performs too well, then the generator samples will always perfectly rejected and the generator will be unable to improve its results. Different research work emerged around the stability of GAN training, but generative networks still remain difficult to be trained [43].

Another approach to stabilising the GAN training is changing the distance function itself. The Jensen-Shannon distance, which was used in the initial work of Goodfellow *et al.* [26], is not the only valid metric for GANs. In recent years, the Wasserstein-1 distance, also known as the *Earth-Mover* distance (EMD), became more and more commonly used in the literature. We use the distance in later chapters of this thesis and introduce it in the following section.

5.5.1 Wasserstein-1 distance

Wasserstein-GAN is a variant of the adversarial network training which is based on the earth-mover distance (EMD), instead of the Jensen-Shannon distance which was originally used by Goodfellow *et al.* [26]. The original version of GANs is known to be difficult to train due to their numerical instability during the training process. Arjovsky *et al.* [1] proposed Wasserstein-GANs to increase stability and performance of GAN training, which motivates the theoretical design of Wasserstein-GANs.

The Wasserstein-1 metric can be interpreted by viewing the two distribution as two piles of earth. The distance is the expected minimum amount of work to transform one pile to the other one. Work here refers to the distance between every element moved multiplied by its probability, which is formally defined as

$$\mathrm{EMD}(p, q) := \inf_{\gamma \in \Pi} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] = \inf_{\gamma \in \Pi} \iint \|x - y\| \gamma(x, y)\, \mathrm{d}x\mathrm{d}y, \tag{5.21}$$

where $\Pi(p, q)$ is the set of all joint probabilities of p and q, in which γ has p and q as marginals respectively, such that $\int \gamma(x, y)\, \mathrm{d}x = q(y)$ and $\int \gamma(x, y)\, \mathrm{d}y = q(x)$.

Optimising the EMD directly is difficult since the formulation itself is a minimisation problem of finding the optimal joint probability distribution γ. However, finding such a distribution is not part of the original GAN objective and is otherwise not relevant for the generative model. To avoid solving this additional problem, Arjovsky *et al.* [1] proposed using the Kantorovich-Rubinstein duality which creates the dual problem of Equation (5.21) without the estimation of the joined distribution. The dual is defined as:

$$\mathrm{EMD}^*(p, q) := \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)], \tag{5.22}$$

where f, in the context of GANs, is the discriminator and L is its Lipschitz constant. The Lipschitz constant is a constraint on f such that $|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$ for all x_1 and x_2 in the function domain. In contrast to the original GAN objective, the formulation (5.22) offers a more stable training process since it has a continuous gradient while the Jensen-Shannon distance (5.17) is not differentiable if both distributions are disjointed.

The challenge of using Equation (5.22) is to ensure the constraint of the Lipschitz constant of the discriminator, which means that the norm of the gradient must be ≤ 1. Arjovsky *et al.* [1] proposed in their original paper to clip the weights of the discriminator to a small values, which guarantees that L stays small. However, simply clipping values by a constant value is not a proper approach to enforce the Lipschitz constraint and the authors themselves stated that the clipping is not ideal. An improved strategy was later proposed by Gulrajani *et al.* [27] using a gradient penalty strategy, where the gradient of the network is explicitly regularised by adding the penalty to Equation (5.22) such that:

$$\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{z\sim p_z}[f(g(z))] + \underbrace{\lambda \mathbb{E}_{\hat{x}\sim \hat{p}}[(\|\nabla f(\hat{x})\| - 1)^2]}_{\text{gradient penalty}}, \tag{5.23}$$

where $\lambda > 0$ is a scalar value, $\hat{x} = (1 - \alpha)\, x + \alpha\, g(z)$, $\alpha \in [0, 1]$ and alpha is randomly picked at each training step. By penalising the gradient at the interpolated samples, the penalty creates a Lipschitz continuous gradient between the generated samples and the

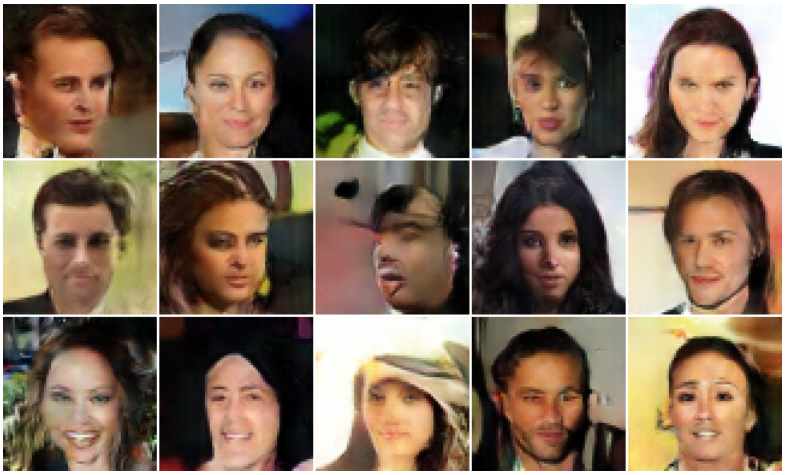


Figure 5.8: Generated samples from an autoencoder trained generative network. [105]

target distribution.

5.5.2 Autoencoder Discriminators

GAN training, during the duration of this thesis and beyond, is an active field of research which continuously investigates the optimal configuration of different network architectures and cost functions to improve the results of GANs. These investigations include different forms of discriminator networks as well, which have much influence on the training results and stability of the training.

In the original GAN paper [26], the discriminator has a single output neuron to indicate if the input is a sample from the real dataset or is a generated sample, which is inspired by object detection networks which use the output of N neurons to detect N categories of objects, which is in the case of GANs is $N = 1$ (i.e. real or fake sample). However, it is still an open question whether this is the optimal configuration. For instance, PatchGANs [35] have been proposed for larger images which are based on pure CNN structures. The PatchGAN discriminator has $M \times N$ output neurons, which can be viewed as single-neuron discriminators for $M \times N$ patches from th input image.

Zhao *et al.* [105] proposed using the reconstruction loss of an autoencoder network as the discriminator signal instead of a single output neuron. AEs have interesting properties, which makes using them as discriminators promising since they are, as discussed

above, trained to project their input signal to a lower dimensional representation while maintaining as much information as possible for a successful reconstruction. The representation contains meaningful features which distinguish different aspects of samples of the dataset, which can be viewed as a compact description of the data. Zhao *et al.* argue that this description is also helpful in determining the difference between real data and the generated samples, and in Figure 5.8 results of their approach are shown.

An improved version of the AE-based approach is proposed by Berthelot *et al.* [10] in which the authors argue that the AE reconstruction objective allows the discriminator to be validly trained even in the absence of generated samples, while single output discriminators always require both positive and negative samples in order to provide a meaningful output. Berthelot *et al.* use this property of AEs to introduce methods of control theory to the GAN training scheme, which dynamically throttles the influence of the discriminator during training. The idea is that the discriminator dynamically discriminates less against the generated samples if the generator performs poorly and vice versa, which allows the generator to "catch up" and therefore avoid that the discriminator over-powers the generator. The authors report more stable behaviour during training and presented results superior to previous methods as shown in Figure 5.9.

The optimisation scheme is defined by Berthelot *et al.* [10] as followed for each iteration step t:

$$\begin{cases} \mathcal{L}_D = D(x) - k_t D(G(z)) & \text{for } \theta_D \\ \mathcal{L}_G = D(G(z)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma D(x) - D(G(z))) & \text{for step t,} \end{cases} \tag{5.24}$$

where $k_0 = 0$ and $\gamma > 0$ is a control parameter. The first two steps are very similarly defined as in the WGAN optimisation scheme. However, a scaling factor k_t is added which is bound to $[0, 1]$. k_t controls the influence of the discriminator on the generator and is updated in the third iteration step of the Scheme (5.24) with λ_k as the step size. The idea is that k_t is set to 1 when the generator performs well), but k_t assumes 0 if the generator performs poorly, which turns $\mathcal{L}_D$ into a simple autoencoder training. γ sets the desired ratio between the generator and discriminator, which the Equation (5.24) achieves when the optimisation reaches an equilibrium. In their work, the authors state

Figure 5.9: Training examples of the GAN training by the method of Berthelot *et al.* [10].

that a low γ value causes the samples from the generator to be less diverse but better in quality. In summary, the added control scheme in Equation (5.24) stops the discriminator becoming being too strong relative to the generator which adds a stabilising effect to the training. Because of these advantages and the qualitative comparison between the results seen in Figure 5.8 and Figure 5.9, we are using the work of Berthelot *et al.* as the basis of the adversarial training in this thesis.

However, before we apply this scheme to the event camera signal, we need to introduce methods which combine the adversarial network approach with an estimation task for the purpose of reconstruction.

5.5.3 Combining Adversarial Network with Measurement Losses

As discussed above, the adversarial training scheme has been proposed to train generative models which relate latent distributions to distributions of observations which also allow the generation of new observations. However, in many computer vision applications we are given specific measurements and we seek the best estimation to explain them. Those tasks are typical in applications like 3-D reconstruction, optical flow and image restoration. However, conventional reconstruction methods are often focused on matching the observation as closely as possible rather than on the structure of the estimates. In 3-D reconstruction, for instance, we are interested that the 3-D geometry is consistent with the measurements, but less emphasis is paid to the feasibility of the

object (i.e. enforcing that walls are flat and so on). This kind of knowledge requires a non-local understanding of the scene which is often very difficult to define properly and simpler definitions are too weak or too specific for a type of scene. In these cases, adversarial networks offer an alternative approach to learn these features in a scene and reinforce them during training.

In the previous sections, we highlighted the capacity of generative adversarial networks to improve the results of generator networks and, conceptually, the same principle can be applied to reconstruction methods as well. This idea inspired many machine learning researchers to introduce adversarial losses during training. Tung *et al.* [94] propose using adversarial neural networks in combination with other networks, which are designed to produce an estimation given measurements as inputs. They showed that this combination helps in a variety of tasks like image inpainting, monocular depth estimation and so on. The results of their inpainting approach are shown in Figure 5.10.

Formally, we can describe their approach of minimising the following cost function for a given set of measurements X:

$$\min_G \max_D \mathbb{E}_{x \in X} \underbrace{\|P(G(x)) - x)\|_2}_{\text{reconstruction loss}} + \beta \sum_{i=1}^{N} \underbrace{\log D_i(M_i) + \log(1 - D_i(G(x)))}_{\text{adversarial loss}}, \tag{5.25}$$

where $\beta > 0$ is a scalar parameter, G and D are the generator and discriminator networks and M_i are samples from our example dataset. The function P is the *differentiable renderer* which maps the output of G back to the input domain. By optimising Equation (5.25), the network G learns to create an estimate which agrees with the measurements x but is also part of the same distribution as the examples M_i. The appeal of the adversarial loss approach is that the network learns a transformation without labelled training data. We can view the adversarial loss in Equation (5.25) as the prior of our estimate, which becomes apparent when we compare to MAP estimation:

$$p(y|x) \propto p(x|y)p(y), \tag{5.26}$$

where x is the measurement and y is the output of the network. $p(x|y)$ models our reconstruction loss which describes how well the estimate explains the measurements

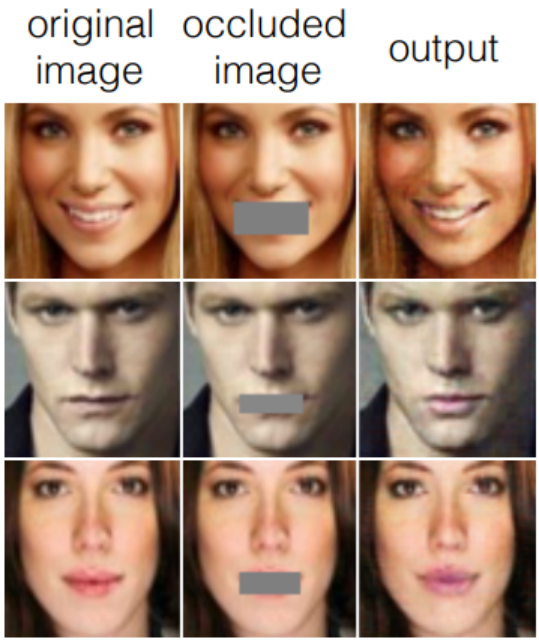


Figure 5.10: Inpainting results in which the network is trained to fill-in the occluded mouth region. The network is trained with an adversarial loss using a face dataset for training. The results have been originally published by Tung *et al.* [94].

and $p(y)$ corresponds to to the adversarial loss since $p(y)$ describes the likelihood that y belongs to the model distribution. The prior of a MAP estimation is often the most difficult property of a problem to describe since it is an internal property which is true even in the absence of any measurements. An adversarial network can learn such aspects of the model, but we lose control over the exact properties which are taken into account for the prior. On the other hand, the adversarial loss requires only an example set which does not need to correlate with the measurements. In other words, a network can learn a transformation without labelled training data, unlike many deep learning methods. This unsupervised training is interesting for problems in which such training data is difficult to acquire or not available at all.

Tung *et al.* [94] demonstrated that Equation (5.25) can be applied to different tasks, however, using adversarial networks in for this approach is still notoriously difficult to train as well since the results are sensitive to weight parameter β in (5.25).

In the previous chapter, we demonstrated that the event camera signal can be translated to an image signal with the TV-L^1 smoothness prior, which is very simplified image

prior and which has undesired effects of suppressing finer details in the image. In the next chapter, we will investigate the combination of a learned prior with an adversarial network and the event camera signal. We will see that adversarial training allows reconstructing the image signal while only using standard camera images for training which are not synchronised to the event signal.

5.6 Conclusion

In this chapter, we have introduced some of the fundamentals of artificial neural networks and how to train them. We delved into some more advanced concepts of neural network design with the focus on GANs, which are generative methods to learn joint probability distributions of the observable and latent variables. These methods are very successful in generating naturalistic looking images which makes them very promising for learning natural images statistic. In the next chapter, we will use these techniques to reconstruct intensities from event camera signals.

Chapter 6

Adversarial Networks to Recover Intensity from Events using only Natural Image Priors

Contents

We introduced in Chapter 4 our first contribution for recovering generally useful information from event data by jointly estimating intensity and motion in a variational framework. To get complete reconstructions from the sparse event data, we introduced smoothness priors on intensity and optical flow in the TV-L^1 optimisation problem. However, in experiments we showed that the results of this approach suffers from artefacts at the motion boundaries in the intensity reconstruction. However, the issue for our method is not only the boundaries but the image prior itself. The event-based optical flow estimation relies on the image reconstruction, which in turn relies on the precision of the flow field. This interconnection requires both estimations of intensity and motion to be optimal since if one estimation is sub-optimal it disturbs the other. If the image reconstruction is sub-optimal then the flow estimation can only be, at most, sub-optimal as well. To improve the estimation process more appropriate priors are needed which do not suppress details of the scene, like the TV-L^1 smoothness regularisation, but encourage finer details and sharp object boundaries.

In the previous chapter, we introduce the machine learning methods which we will use in this chapter to learn more complex image priors and apply them to the event data in order to improve our image reconstructions. Our contribution here is inspired by the works of Reinbacher *et al.* [79], who showed that an image reconstruction can be achieved from events without optical flow by using a sophisticated spatio-temporal image prior. Their prior was hand-crafted, but unfortunately, it causes artefacts in their reconstructions as well. However, although the prior include an implicit motion prior as well, the authors' arguments gave us reasons to believe that the event signal can be interpreted by using natural image priors alone.

Accessing the information hidden in the event stream is difficult, but providing an easy image reconstruction which is usable by other applications in robotic vision and beyond would make the event cameras easier to employ in those applications. For doing so, we propose to directly interpret the event stream by reconstructing a sequence of image frames using an unsupervised deep-learning approach which requires only a dataset of natural images as additional training data. Our proposed approach is based on the adversarial loss, which we introduced in Section 5.5.3. For training, we let a

network generate a sequence of images which respect the event-data as well as satisfy an adversarial network, which is trained at the same time. For adversarial network training, we use a dataset of natural image examples, which are captured by a standard frame-based camera. At the end of this chapter, we present reconstruction results which reveal substantially more details in moving scenes than previously handcrafted methods for intensity reconstruction from event data.

Since our method does not require specialised datasets, it allows deep learning for event cameras to be much more accessible, opening up event cameras to the wide range of algorithms for learned object detection, optical flow and visual odometry.

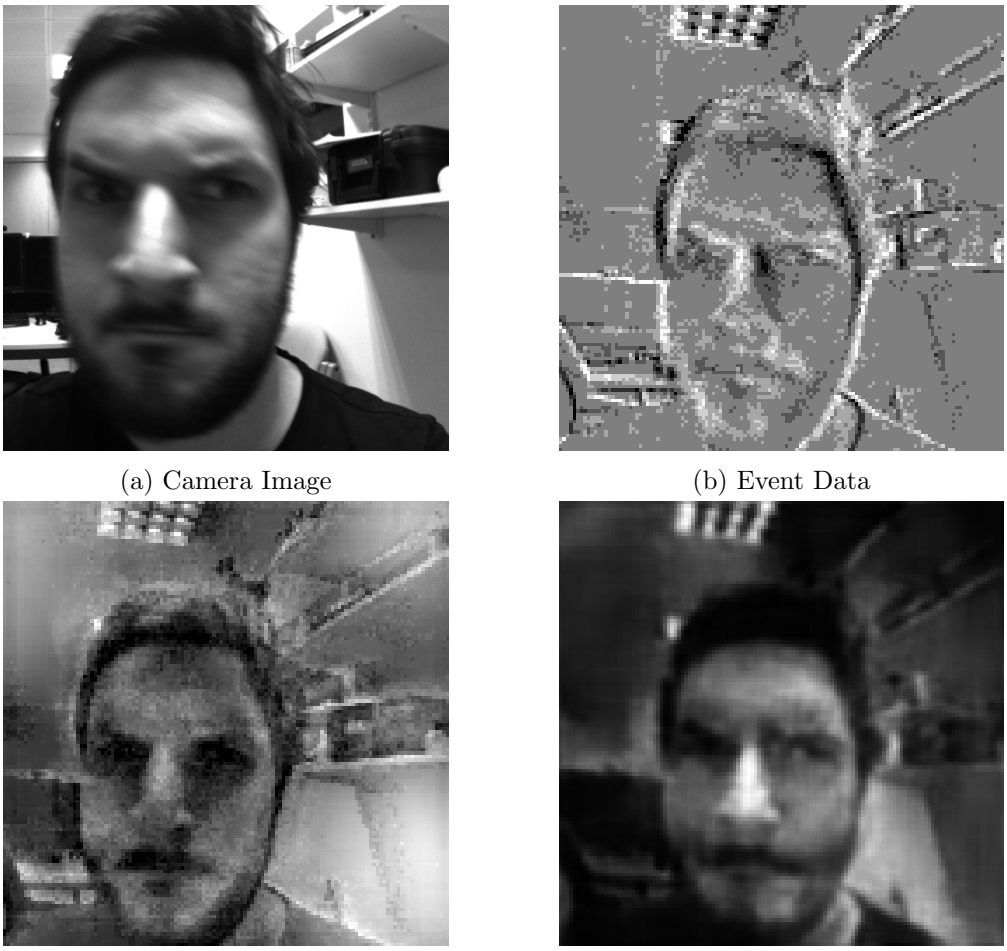
For this work, we decided to use the adversarial loss approach, although using a ground truth dataset for training is a more straightforward approach. Such a dataset would be composed of pairs of synchronised events and standard images and in fact the DAVIS camera from iniLabs [13] is able to capture intensity frames alongside events from a single sensor. However, such datasets would have significant disadvantages:

- The dataset would fix the training data to the standard camera frame-rate and it would suffer from motion blur and low-dynamic range.

- Training sets for standard cameras are often task dependent and creating them again for event cameras is undesirable.

- An event camera's response bias settings alter its sensitivity, which would require creating datasets for different bias-settings.

Despite these disadvantages, a number of event-based deep-learning approaches have emerged recently [64] [107], which circumvent labelling events by using the corresponding standard camera images instead. Standard camera images allow common image processing algorithms to be used to generate ground truth labels like image correspondences or object boundaries. However, we argue that while using standard frames images supports training it also introduces the drawbacks from a standard camera which we wish to overcome with the event camera like motion blur.



(a) Camera Image (b) Event Data

(c) Unsupervised (d) Semi-Supervised

Figure 6.1: Input and output of our method. The upper left image shows a scene captured by a standard camera and the upper right image shows the raw events which are collected into a integrated image over a small time window. The two lower images show reconstructed intensity frames from our two learned approaches, which have been trained in an unsupervised manner and a semi-supervised manner, where partial DAVIS frames were used during training.

Our training data consists of event data from moving scenes, and unconnected natural images. These are taken from an SLR camera in static scenes, so they have high image quality and no motion blur.

We believe that our work is at the forefront of deep learning-based analysis for event data. It makes very weak assumptions and generates results for image reconstruction which are comparable to hand-designed algorithms. The contributions in this chapter are based on a prepared submission to Conference on Computer Vision and Pattern Recognition, 2019 under the title "Can Event Data be Interpreted using only Natural Image Priors?" by P. A. Bardow, S. Leutenegger and A. J. Davison.

6.1 Unsupervised Image Reconstruction

An event camera measures only per-pixel temporal differences in logarithmic intensity, as defined by Equation (2.22), and thus the absolute pixel intensity of the image is not retrievable. If we integrate events at their pixel location we could not retrieve the actual image signal from it, unless we knew the starting frame of the sequence.

Theoretically speaking, if we knew the first frame of the sequence and had an event camera sensor which operated noiselessly, then we could reconstruct the intensity at any point in the future using the event signal. Formally, we can express this by first defining an event like in Chapter 4 as a tuple $e_i = (x_i, y_i, t_i, \rho_i)$ where $(x_i, y_i) \in \Omega$ is defined as the position in image coordinates, t_i as the timestamp and $\rho_i \in \{-1, 1\}$ as the polarity. We can then compute the intensity of the i-th event by using the formulation

$$I_i(x_i, y_i) := \exp(\theta \rho_i) I_{i-1}(x_i, y_i), \tag{6.1}$$

where θ is the event threshold and I_0 is the starting frame of the event sequence. For Equation (6.1), we make the same assumptions as for our previous contribution, that an event is triggered as soon as the logarithmic intensity meets θ.

In reality, of course, the intensity of I_0 is not available using only the event stream. Additionally, any event-based reconstruction method needs to handle sensor noise in the event stream, which is a strong influence due to the analogue circuitry, as we discussed

in Chapter 2. Still, the formulation of Equation (6.1) provides insight into the problems our learning task must solve: Reconstruct I_0 and denoise the event signal.

In the next section, we describe our learning approach to address those problems and to train a neural network given only a stream of events. The training entails a reconstruction loss which depends on how well the produced image fits the given data, as well as an adversarial loss which provides image prior information in order to generate a realistic looking image.

6.1.1 Event Data as Network Input

Before we detail the training and structure of the reconstruction network, we need to define the form of the event data to allow the network to process it. The challenge of using events as a network input is the temporal domain of the signal. ANNs are well designed for images with a few colour channels per pixel. However, using image channels to encode individual timestamps and polarities from the event data would create images with millions of image channels which would consume too much of the memory capacity of commonly used computers.

In order to compress the signal, we take another look at Equation (6.1). One of the main goals for our training is to estimate a starting frame I_0 which is the unknown constant of the integral of events. As we see from Equation (6.1), the event update does not require timestamp information, which means that for reconstruction purposes the sum of the events per pixel should provide sufficient information. Formally, we define the integral of the events into a 2-D image as the sum the over first k events of the sequences as:

$$E_k = \sum_{i=1}^{k} \mathbf{u}_{y_i} \mathbf{u}_{x_i}^T \rho_i \theta, \tag{6.2}$$

where $\mathbf{u}_j \in \mathbb{R}^M$ is a one-hot M-dimensional vector with all entries being 0, except for the j-th coefficient being 1 and $M \times M$ being the event camera resolution. Since we often use only an interval of the integral, we introduce a shorthand notation for an event

interval from the j-th to k-th event such as

$$E_{j:k} := E_k - E_j. \tag{6.3}$$

Using Equation (6.1), we can define the reconstruction of image I_k which corresponds to the k-th event as:

$$I_k = \exp(E_{j:k} + \log I_{j-1}), \tag{6.4}$$

where apply the exponential and the logarithmic function in a pixel-wise manner.

We use the event integral for visualisation as well, which is seen for instance in Figure 6.1. This visualisation is commonly seen in the event camera literature [53, 79], since it allows us to view the signal in an image-like way. The network is then tasked with completing the event integral in order to reconstruct the image, which we detail in the next section.

6.1.2 Event Data Reconstruction Loss for Learning

As mentioned above, our learning approach follows the scheme introduced in Section 5.5.3 by combining a reconstruction loss and an adversarial loss. However, it is not possible for single image reconstruction to match the integrated events $E_{j:k}$ since event data expresses the logarithmic difference between images. Because of the differential nature, it is not enough for our network to reconstruct a single image, but an image sequence of at least two images. We, therefore, break the event integral $E_{j:k}$ into N integrals and task the network to create corresponding N images. Formally, we define the new network input as $X_i \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}\times\mathrm{N}}$ for each image pixel position (x, y) and channel z such that:

$$X_j(x, y, z) := E_{j:k}(x, y), \;\; k = zK, \tag{6.5}$$

where $K > 0$ is a scalar which describes the increment of events between the N integrals. With the input defined, we can formulate the reconstruction loss of our network. Let G be the reconstruction network which is parameterised by ω_G and which produces N images,

where each image channel contains the intensity of one the images corresponding to the input X_j, then the reconstruction loss is defined as:

$$\mathcal{L}_R := \mathbb{E}_{X_j \sim p_E}[\|D \log G(X_j; \omega_G) - DX_j\|_\theta], \tag{6.6}$$

where p_E is our event data distribution, and D is the linear difference operator, which maps to the difference between the channels such that

$$(DX_j)(x, y, z) := X_j(x, y, z + 1) - X_j(x, y, z), \tag{6.7}$$

for each $1 \leq x \leq M, 1 \leq y \leq M$ and $1 \leq z \leq N - 1$. For Equation (6.6), we use the Huber norm $\|.\|_\theta$ which uses the event threshold θ as its threshold such that

$$\|x\|_\theta := \begin{cases} \frac{1}{2}x^2, & \text{if } |x| \leq \theta \\ \theta|x| - \frac{1}{2}\theta^2, & \text{otherwise.} \end{cases} \tag{6.8}$$

Unlike our variational reconstruction method in Chapter 4, we use here for the reconstruction loss the Huber-norm, which creates here an approximate combination of the event and no-event data term. If $\theta \leq 1$ for the Huber-norm, then any absolute residual beyond θ is penalised by a L^1-norm, which is robust against outliers, but for residuals below θ the penalty is smaller than a L^1-penalty. Therefore, the Huber-loss allows for more deviations in the error within the bounds of the event-threshold, which models the uncertainty of the intensity in the interval between two events. In our experiments, we choose θ to be less than 1, so the above argument holds. However, for event thresholds greater than 1 this requires rescaling of the residuals. The robustness of Equation (6.6) is an important feature which allows the network to learn to denoise and filter-out events for the reconstruction.

The reconstruction loss forces the network to create an output which agrees with the event measurements, but it alone does not incentivise realistic reconstruction results, which is the reason that we include the adversarial loss in our approach as well. In the next section, we detail the adversarial loss and its incorporation into our training scheme.

6.1.3 Adversarial Loss for the Event Camera

In general, it is difficult to quantify images by their natural appearance, because *natural* is an ill-defined term. In Chapter 1, we introduced the TV-L^1 prior as an approximation to natural image statistics which allows for efficient minimisation. However, in Chapter 4 we discussed that the TV-L^1 regulariser is too simple to allow high quality reconstructions. Therefore, we need to introduce a more expressive prior in order to improve the reconstruction. In Chapter 5, we introduced GANs and the adversarial loss which are capable of capturing the structure of complex image statistics. In this section, we introduce this scheme to the event camera image reconstruction.

In the previous section, we defined the output of the network as a sequence of images which gives us two choices with respect the adversarial network: Discriminate the output by treating it as a video sequence or by treating it as a series of single images. At first, the former option is the most natural choice since the output is similar to a short video sequence from a frame-based camera. However, the adversarial network then requires real examples from video cameras which have properties in comparison to event cameras which are undesirable. Their signal has a fixed frame-rate, which requires us to fix the time-interval for our event signal as well. However, if there no motion in the image then the event signal is empty and no meaningful reconstruction can occur. Another problem with video cameras is motion blur which is, in general, difficult to avoid, and the event camera reconstruction should be discouraged from recreating those artefacts.

The other option is to treat the reconstructions as independent images, which then allows us to use still images from a standard camera as training data. Although movement artefacts can also occur in single images, they are easier to avoid. Additionally, a single image dataset allows us to use HDR images which encourages the network to map the event camera signal to a HDR domain as well. Following the above arguments, we decide in this work to use the single image strategy for adversarial training.

With the network output strategy decided, we next make the choice of discriminator structure and the GAN distance function. As we described in Chapter 5, a GAN training scheme requires two neural networks: the generator and discriminator. The role of the generator is taken by our reconstruction network, which leaves the choice of the discriminator open. In Chapter 5, we discussed different discriminator structures, and

here we decide to use an *autoencoder discriminator* [105, 10] which defines its output signal as the autoencoder reconstruction loss such as:

$$F(x;\omega_F) := \mathbb{E}[\|A(x;\omega_F) - x\|_1], \tag{6.9}$$

where A is the autoencoder network function which is parameterised by the network parameter ω_F. By using Equation (6.9) as the discriminator signal, the network A learns to reproduce real images while discarding generated samples. We follow here the argument from Zhao *et al.* [105] that autoencoders learn a meaningful representation of the training data, even without negative or fake examples, which the generator can exploit during training.

The choice of using an autoencoder discriminator allows us also to adapt the training scheme proposed by Berthelot *et al.* [10], which is described in Section 5.5.2. The training approach incorporates control theory as well as an approximation to the stable Wasserstein-1 distance. We believe that these aspects of the training scheme are appropriate for our work to address the well-known instability of GANs [43]. Using this training, we can define the adversarial loss such that:

$$\mathcal{L}_A := \mathbb{E}_{Z\sim p_Z}[F(Z;\omega_F)] - \mathbb{E}_{X_i\sim p_E}[F(G(X_i;\omega_G),\omega_F)], \tag{6.10}$$

where p_Z is the distribution of natural images and G is our reconstruction network function which is parameterised by ω_G.

Optionally, synchronized frames from a standard camera may be used to support training. The frame should be aligned with the event signal, which is the case for the DAVIS [13]. However, since these images suffer from over- and underexposed pixels, we propose using them as partial labels, which turns the training scheme into a semi-supervised training scheme. If DAVIS frames are available, we incorporate them by adding the following loss:

$$\mathcal{L}_c := \mathbb{E}_{(X_i,Y_i)\sim p_D}[M\|G(X_i;\omega_G) - Y_i)\|_1], \tag{6.11}$$

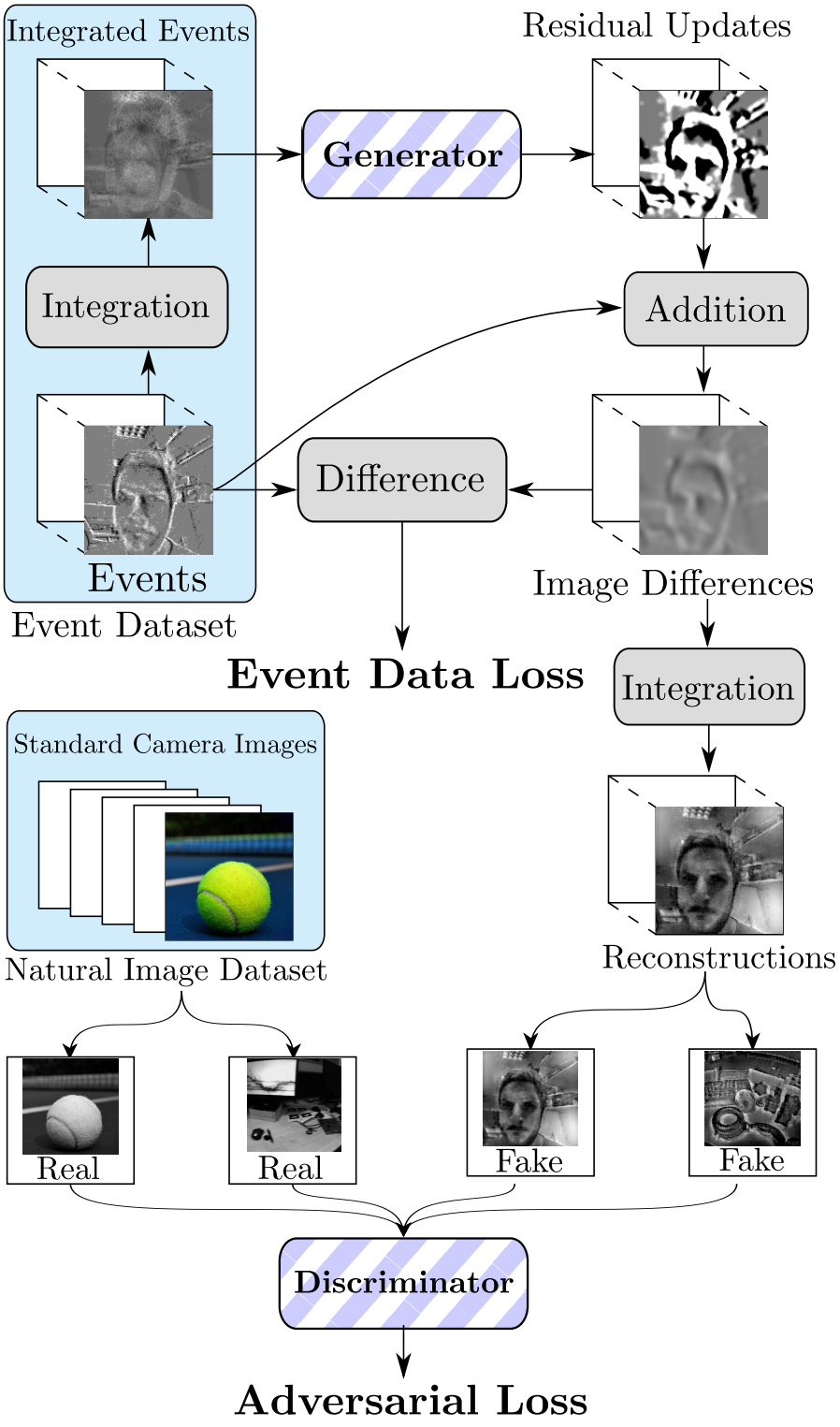


Figure 6.2: Overview of our approach and training framework. Trainable parameters are in striped boxes and non-learned operations are marked as grey boxes. The events from the camera are integrated and presented to the generator network, which creates residual updates. We add these updates to the events to create difference images, or refined events, and we ensure that they stay close to the events by computing the event data loss. For the final reconstruction, we integrate the difference to a sequence of images and present them as single images to the discriminator. The discriminator is the adversarial network, which classifies between the reconstruction and real images from a natural image dataset.

where p_D is the joint distribution of synchronised events and images and M is a binary matrix whose elements are 0 if a corresponding pixel in the frame Y_i is over- or underexposed, otherwise 1. We identify the over- and underexposed pixels by thresholding. The Equation 6.11 therefore matches the reconstruction to the DAVIS frames except for the over- and underexposed areas. Later in this chapter, we show that the network learns to generate artefacts if these pixels are not excluded by M.

With our training objective defined, we continue by describing our training scheme and network implementation.

6.1.4 Training Scheme

Before we describe the details of the training, we first add some details about the network output. At this stage, we would have the option to let the reconstruction network to simply generate N images. However, we like to support the reconstruction as much as possible by handcrafting some of the operations, which we expect the network to learn regardless. For the output definition, we revisit Equation (6.1) and the goals of reconstructing the starting frame and denoising. From (6.1), we can see that the starting frame is a summand of the events, which means that the network output can be added to the events before applying the exponential function. Formally, we define this using the output of the network R_i which has the same dimensions as the input of the event integral X_i. We then reconstruct the pixel intensity at position (x, y) of the k-th image of the sequence such that:

$$I_i(x, y, k) := \exp\left(X_i(x, y, k) + \sum_{j=1}^{k} R_i(x, y, j)\right). \tag{6.12}$$

We sum the channels of R_i for the reconstruction so the network learns to generate R_i, and so should it only contain the updates between the frames of the sequence. We can see that this reconstruction is modelled after the event update function (6.4). This scheme is also inspired by the *residual-network* architecture which is designed to avoid copying the input of a network, and which is commonly used in deep-learning [29].

For overview purposes, the complete training scheme of our approach is shown in Figure 6.2. Next, we formally describe the minimisation scheme.

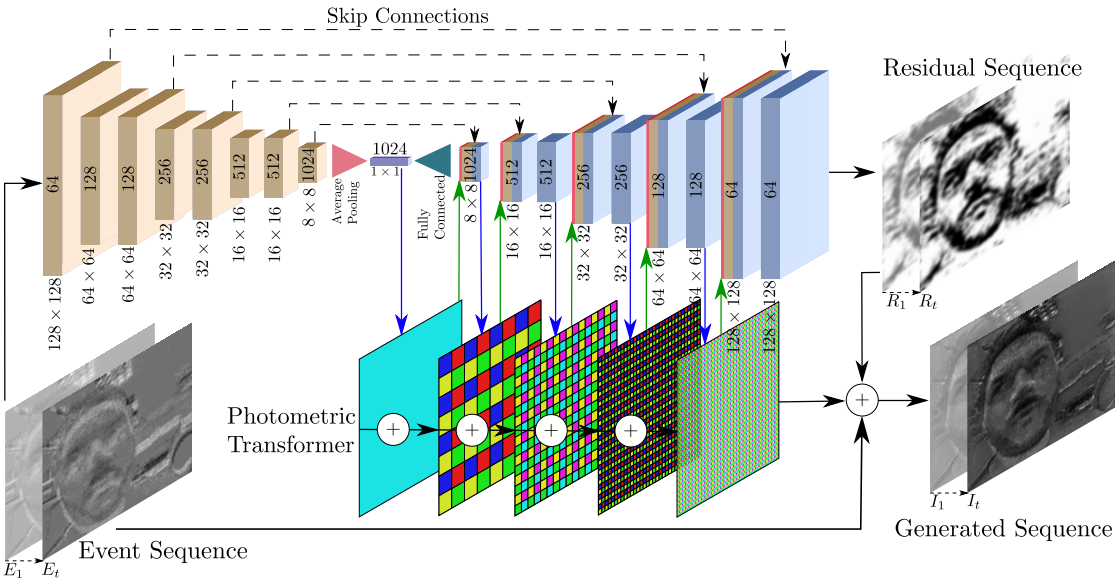


Figure 6.3: The reconstruction network takes as input a sequence of t 128×128 integral event images and returns a sequence of t residual updates, which are integrated and added to the input together with the photometric transformer result. The transformer performs a linear convolution for each network resolution (blue arrows), which outputs a 1-channel image. The image is up-scaled, using nearest-neighbour interpolation, and accumulated with the previous transformer images. The final result is concatenated to the input of the next convolutional layer (green arrows). On each convolution layer, we note the resolution and feature map size, which refers only to decoder weights at skip layers.

As mentioned above, our training approach adapts the feedback loop proposed by Berthelot *et al.* [10] by adding an influence-factor k_t to the discriminator objective for the t step of the training process and which is limited to a value range such that $k_t \in [k_{\min}, k_{\max}], k_{\min} \in \mathbb{R}, k_{\max} \in \mathbb{R}, k_{\min} < k_{\max}$. We then define our training scheme for the network paramters ω_F^t and ω_G^t for the t-th training step such that:

$$\begin{cases} \mathcal{L}_F^{t+1} := \mathbb{E}_{Z \sim p_Z}[F(Z; \omega_F^t)] - k_t \mathbb{E}_{X_i \sim p_E}[F(G(X_i; \omega_G^t); \omega_F^t)] & \text{for } \omega_D^t \\ \mathcal{L}_G^{t+1} := \lambda \mathcal{L}_R^t + \mathbb{E}_{X_i \sim p_E}[F(G(X_i; \omega_G^t); \omega_F^{t+1})] & \text{for } \omega_G^t \\ k_{t+1} \leftarrow \min(\max(k_t + \gamma \mathcal{L}_A^{t+1}, k_{\min}), k_{\max}) \end{cases} \tag{6.13}$$

where γ is the step-size of the influence factor and λ is a scalar which weights the

influence of the reconstruction with respect to the adversarial loss. At the beginning of training, we initialise ω_G^0 and ω_F^0 with random values, which is common practice in machine learning and we set $k^0 = k_{\min}$. We describe the implementation details of the network and training scheme in the next section.

6.1.5 Network Implementation

Both the generator and discriminator follow a U-Net like structure [85], which is an hourglass, autoencoder-like type structure as discussed in Chapter 5. Figure 6.3 shows the schematics of the generator structure, which doubles the number of feature maps in the encoder part whenever the resolution is reduced and then halves them again in the corresponding decoder part which increases the resolution. We augment the hourglass structure by using skip connections, as used in U-Net [85] which concatenate the output of a corresponding layer with the input to another layer. We follow here the argument of U-Net to use skip connections to improve finer details of the network output.

For the convolutional layers of the generator network, we use a kernel size of 5×5 and for the discriminator a kernel size of 3×3. At the coarsest level of both networks, we average the all the entries, which is called *average pooling*, with respect to each feature map. After the pooling operation, we increase the resolution again by using a fully connected layer. The networks reduce the resolution by using a convolutional layer with a stride factor 2 and for increasing the resolution we use nearest-neighbour interpolation followed by another convolutional layer. For the activation function, we are using *exponential linear units* (ELU) which are also used in the work of Berthelot *et al.* [10]. ELUs are a variant of ReLUs, and are defined as follows:

$$\mathrm{ElU}(x) := \begin{cases} x, & \text{for } x > 0, \\ \exp(x) - 1, & \text{otherwise.} \end{cases} \tag{6.14}$$

All activation functions of the networks are ELUs, except for their output layer, which is a linear function for the generator and a hyperbolic tangent function for autoencoder discriminator.

Additionally, we add extra outputs to the generator to support the reconstruction, which we call a *photometric transformer*, which is shown in Figure 6.3. The photometric

transformer creates predictions of the image intensities at different resolutions and combines them by upscaling their resolution and adding them together, which is inspired by the approach of Lai *et al.* [47]. By doing so, the transformer provides the generator with a coarse-to-fine scheme to influence the reconstruction at different frequency-levels in the image. In early experiments without the transformer, we observed that the generated images have uneven brightness in uniform regions.

6.2 Evaluation

For the evaluation, we use the DVS dataset which we used for the evaluation of our method in Chapter 4 and compare the results in a series of qualitative experiments with that method and the event-based image reconstruction method by Reinbacher *et al.* [79]. We show in those experiments that our method retrieves more details of the scene while avoiding artefacts which the other methods suffer from. We especially like to point to details in the scene like moving objects and high-dynamic lighting. We also like to note that these sequences are not part of our training set.

Our event data-set is composed of 41 sequences captures by the DVS128 [53] of an office scene, containing people and different camera motions. For training, the network input X_j as defined by Equation (6.5) is sampled from the sequences by uniformly randomly selecting an event and using its index for j as the starting point of the integral. We choose the difference in the number of events between the input channels such as $K = 10k$. Also, we found that adding events, which occurs before the j-th event helps the reconstruction, and therefore, we add additional $100k$ to the input, which occurred before the j-th event, but which do not influence the reconstruction loss.

For the natural image training set, we captured 1000 HDR images with a Canon EOS 5D DSLR camera and with all images captured from the same office environment as the event dataset. The HDR-mode of the camera creates HDR-images by using 3 exposures of different durations from its lower dynamic sensor. As we discussed for the adversarial loss, we use the HDR dataset to encourage the network to map the reconstruction into an HDR domain and discourage it from creating over- and underexposured artefacts. Since the image resolution is larger than then event camera sensor resolution, we create a sampling batch by downscaling the resolution of each image by a random factor and then cropping the image to a random 128×128 sub-region. Additionally, we augment the

samples from the event and image datasets by applying random mirroring and rotational transformations before the input to both networks.

For implementation and training, we use the Tensorflow library and follow the implementation of Brethelot *et al.* [10] by using the ADAM optimiser [41] with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with a learning rate of 0.0001. We continuously decrease the learning-rate, which halves every $10k$ iterations. For training, we use a batch size of 128, which we sub-divided into 16 sequences of 8 images for the reconstruction network. Each reconstructed image from the generator is treated as a single sample for the mini-batch for the discriminator training.

For the training scheme (6.13), we set the bounds of the influence factor to $k_{\text{min}} = 0$ and $k_{\text{max}} = 0.25$. At the beginning of our training, we set $k_0 = k_{\text{min}}$, which means that the generator can first minimise the reconstruction loss $\mathcal{L}_{\mathcal{R}}$ and the adversarial loss before the discriminator starts to maximise the generator loss. Setting upper-bound of the influence factor $k_{\text{max}} = 0.25$ showed good results during our experiments, while we observed that higher bounds cause strong changes in the reconstruction during training, which neglect the loss of $\mathcal{L}_R$.

For our comparisons to other reconstruction methods, we used the default parameters of the implementation provided by Reinbacher *et al.* [79] and for our previous method, which uses a defined frame-rate. We choose a duration which contains roughly $170k$ events which match the number of events for our learned estimation, assuming that a DVS collects around $300k$ events per second [40].

6.2.1 Face Sequence

In the first sequence, the event camera captures a moving human head, with the camera in motion as well. We evaluate here the consistency of our reconstruction and the level of detail and compare it to our previous method. In Figure 6.4, we show different stages of the sequence and the reconstruction results.

We see that our method can consistently retrieve many details which are missing in the results of the previous approach, which uses TV-L^1 as the image prior, which causes many details to vanish, particularly in the background. Also, we see here the HDR capability of event cameras, which are able to perceive details from the ceiling

Figure 6.4: Sequence of a moving head, captured by standard camera (first column) and reconstructed by previously proposed method from Chapter 4 (second column) and by our proposed semi-supervised method (third column). Our method can reconstruct much finer details in the background such as the ceiling light which is not visible in the second column and is overexposed in the first column.

Standard Camera

Variational Method (Chapter 4)

Reinbacher *et al.*

Our Method

Figure 6.5: Hand-held event camera observing objects on a desk.

light, while the standard camera images are overexposed. However, our method does not reach the same level of contrast.

6.2.2 Desk Sequence

This sequence shows an office desk with different items scattered around, which is shown in Figure 6.5 with a comparison of all three reconstruction methods. Our learning-based method captures more details of the scene than our previous handcrafted method and has a comparable level of details to Reinbacher *et al.*'s method.

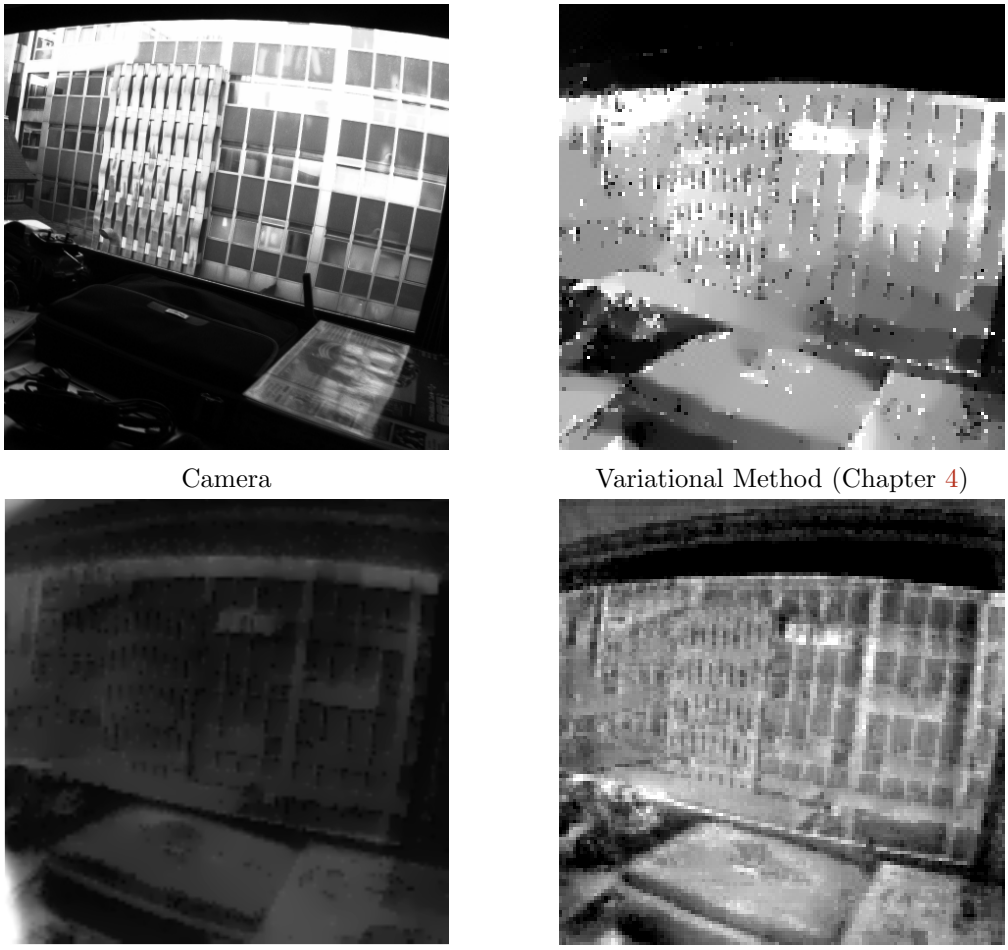
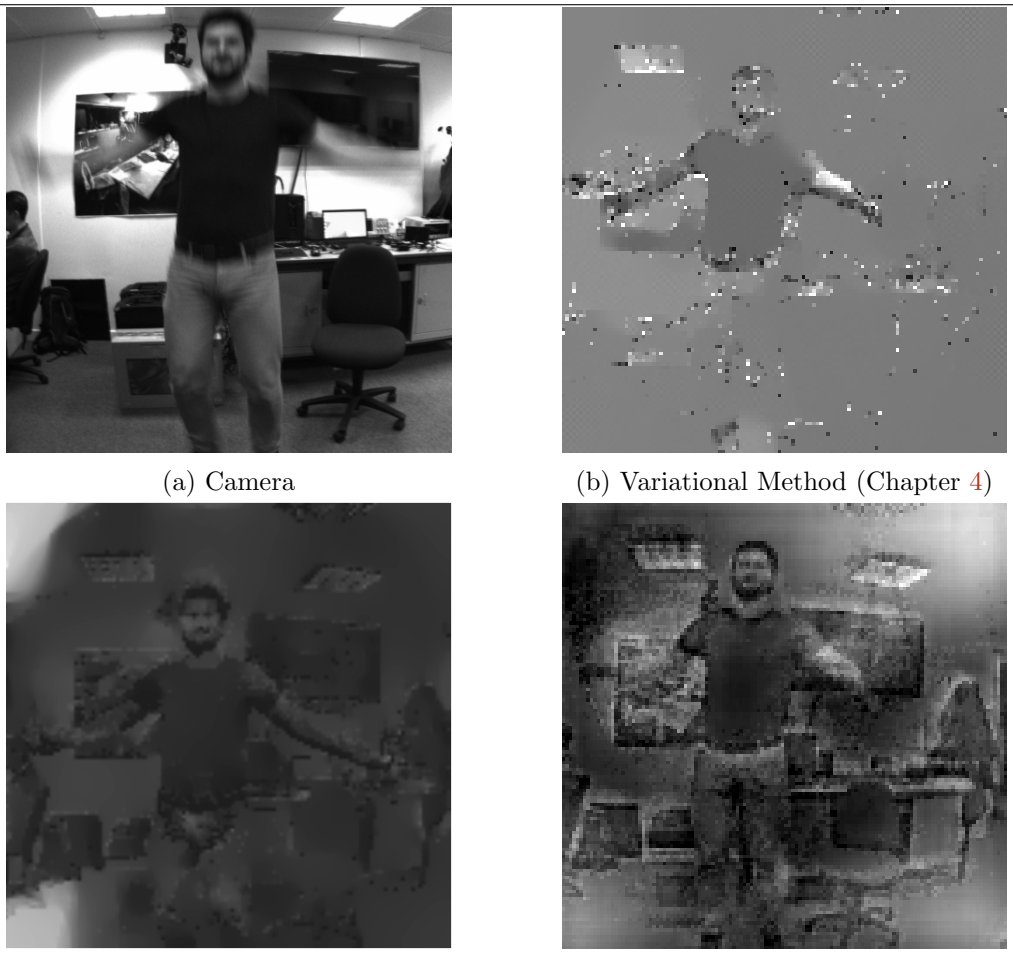


Figure 6.6: Camera pointing through a window from a dark room. The standard camera is over-and under-exposed.

We can see that the object boundaries with the learning-based method seem cleaner and finer details around the headphones appear. However, Reinbacher *et al.*'s method has a more consistent global brightness than our reconstruction, which is darker between the objects. We observe this behaviour in multiple sequences, which suggests that the reconstruction network is more focused during training on minimising the reconstruction loss in local areas than on being globally consistent.



(a) Camera (b) Variational Method (Chapter 4)

(c) Reinbacher *et al.* (d) Our Method

Figure 6.7: Sequence of a jumping man, where the standard camera suffers from motion blur.

In comparison, our previous handcrafted method has trouble at object boundaries of the face and the background, as well as reconstruction many of the details due to the TV-L^1 smoothing. The Reinbacher *et al.*'s method reveals more details of the scene, however, it creates artefacts along the direction of motion of the face.

6.2.3 Window Sequence

In Figure 6.6, the camera is pointed towards a bright window from inside a darker room, which creates a wide range of lighting conditions in the image. The frame-based camera

in the Figure suffers from over- and under-exposure, while the event data does not have this problem. Like in the previous sequence, our learning-based method shows more detail, while having cleaner object boundaries. Our handcrafted approach shows problems with properly reconstructing details of the opposite buildings, which is apparent in Reinbacher *et al.*'s method as well. However, the learning-based method shows the window details clearly, suggesting that the prior is indeed better at revealing those details.

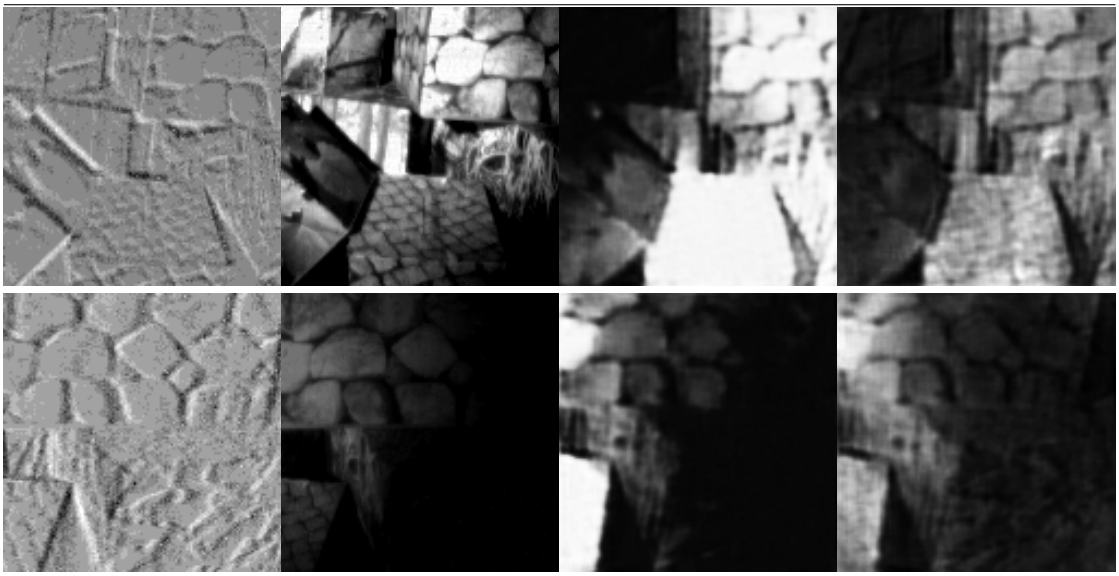
6.2.4 Jumping Sequence

In this experiment, we show the capability of our method to handle fast object-motion. As mentioned above, the reconstruction network learns the image prior from single images, which means that there is not a learned bias towards a particular speed of motion. In Figure 6.7 we see that the method is indeed able to retrieve the arm position like the other methods, while the standard camera image suffers from motion blur. It also retrieves details on the clothing and background objects, like the chair. However, is not as consistent with regard to global brightness as Reinbacher *et al.*'s method.

6.2.5 DAVIS Sequences

In this section, we evaluate our proposed optional loss which uses DAVIS frames as partial labels. For this experiment, we use a sequence from the DAVIS dataset provided by Mueggler *et al.* [61]. Figure 6.8 compares the learning based results, which are trained by using (6.11) for under-exposed areas, and a network with the same architecture but trained using the whole DAVIS frame and without the adversarial loss for supervised training. For reference, Figure 6.8 also shows the corresponding events and DAVIS frame We see that the supervised network learns to create over- and under-exposed image areas, since they are present in the DAVIS training frames, even though event signals are present in those regions. On the other hand, the semi-supervised network is able to reconstruct most image areas without artefacts, which suggests the adversarial loss is more suitable to learn the image prior for image reconstruction.

In our experiments, we observe that the less we rely on labelled data the more the network is able to reconstruct finer image details, which is highlighted in Figure 6.1 by comparison between the semi- supervised and the unsupervised approach. While the semi-supervised network is able to consistently reconstruct homogeneous areas in the



(a) Events (b) DAVIS (c) Supervised (d) Semi Supervised

Figure 6.8: HDR scene with stacked boxes captured by DAVIS. The reconstruction from the supervised network cannot properly reconstruct all areas of the image, while the semi-supervised network is able to do so.

image like the table and walls, the unsupervised approach reveals more details of the head and background. The semi-supervised network also creates more blurred images, which is cause by the blur present in DAVIS training sequences.

6.3 Reconstruction Problems

The learning-based approach which we introduced in this chapter revealed an image reconstruction results with fine detail, but unfortunately, these results appear not as naturalistic as the original GAN results shown in Chapter 5. In the previous section, we highlighted, in several experiments, that our learning-based approach, in comparison to handcrafted methods, seems to reconstruct details well, but the contrast and global brightness seem to fit only locally, which gives the image a noisy and patch-like appearance.

In this section, we explore some of the reasons for this behaviour and try to apply some techniques to stabilise the training and report the results.

6.3.1 Gamma Correction and Event Reconstruction Loss

As mentioned above, the results of our learning-based method are inconsistent in terms of contrast and brightness throughout the image sequence. Indeed, we observed during training with different values for the event threshold θ that the reconstruction is either too low or too high in contrast which creates regions of inconsistent contrast and brightness.

We observed during our experiments that early training results resemble the behaviour of a gamma correction function for standard frame-based camera images and, indeed, we can relate the gamma correction and the event measurement function. We recall that a gamma corrected intensity I' relates to an uncorrected intensity I such that

$$I' = sI^{\gamma} \tag{6.15}$$

where $s > 0$ is a normalising factor and γ is the correction factor. If $\gamma > 1$, then bright image regions are mapped to higher values and darker regions to lower. Vice versa, if $\gamma < 1$ then bright and darker appear to get closer to each other. For more on gamma correction, we refer to the literature [91].

To establish the connection between gamma correction and the event measurement function, as defined by Equation (2.22), we look at the logarithmic difference of two gamma corrected intensity values I'_2 and I'_1. We can express the difference as:

$$\log I'_2 - \log I'_1 = \log(sI_2^{\gamma}) - \log(sI_1^{\gamma}) = \gamma(\log I_2 - \log I_1), \tag{6.16}$$

which strongly resembles the event measurement function but scaled by γ. The relation between γ and θ means that the choice of the event threshold strongly influences the relationship between dark and bright image regions in the reconstruction. This relation leads us to conclude that a sub-optimally chosen θ might be one cause which creates these inconsistencies in the contrast and brightness of the image.

However, there is no established method for finding the optimal event threshold with respect to an arbitrary set of standard camera images. Despite this, we still performed a verification that the event threshold strongly influences the reconstruction results. We

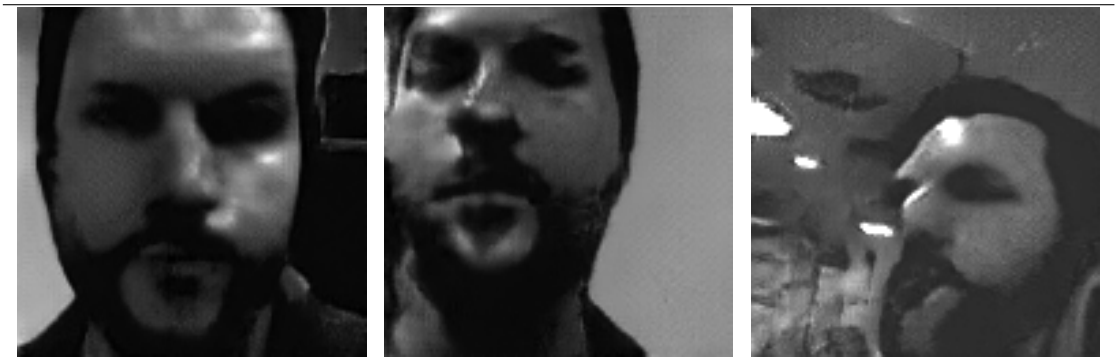


Figure 6.9: Reconstruction sequence using DAVIS frame of Face sequences.

conducted some initial experiments using the already aligned frames of the DAVIS. For these experiments, we use 4 DAVIS sequences capturing a person and optimise the event threshold to those sequences using (6.1). After obtaining θ through this process, we use it for training the generator network but replaced the natural dataset with the DAVIS sequences. In Figure 6.9, we show the results of the initial experiment.

The initial results show more consistent reconstructions and a more naturalistic appearance than the previous results using a sub-optimal event threshold. However, the DAVIS dataset and the event data share a very narrow domain, which makes it uncertain if the results are applicable in a more general setting. Using the DAVIS optimised event threshold with the previously used natural image dataset proved to be unsuccessful, which is to be expected due to qualitative differences in the images and the DAVIS, whose examples frames are given in Figure 6.8.

During different training sessions, we found that due to the non-linear nature of the optimisation, empirically finding an event threshold is challenging. Instead, we propose using an adaptive approach, which we incorporate during the training. The idea is based on the previously established connection between the event threshold and the gamma correction function, which we use to allow the reconstruction network to perform a gamma correction before the results are passed to the discriminator network. The correction allows the reconstruction network to compensate for poorly chosen event thresholds while still maintaining the same reconstruction loss.

Formally, we describe this by changing the adversarial loss (6.10) by adding a variable γ

such that:

$$\mathbb{E}_{Z\ \sim p_Z}[F(Z;\omega_F)] + \mathbb{E}_{X_i \sim p_E}\left[F\left(\frac{2}{a^\gamma} I_i^\gamma - 1;\omega_F\right)\right],\ I_i = G(X_i;\omega_G) + b, \tag{6.17}$$

where b is a scalar which is added to the pixel values of the generator output to ensure that its values are all positive and a is the maximum intensity value I_i can achieve. For the minimisation, γ is jointly minimised with ω_G, which means that γ is adjusted to support the reconstruction effort of the network.

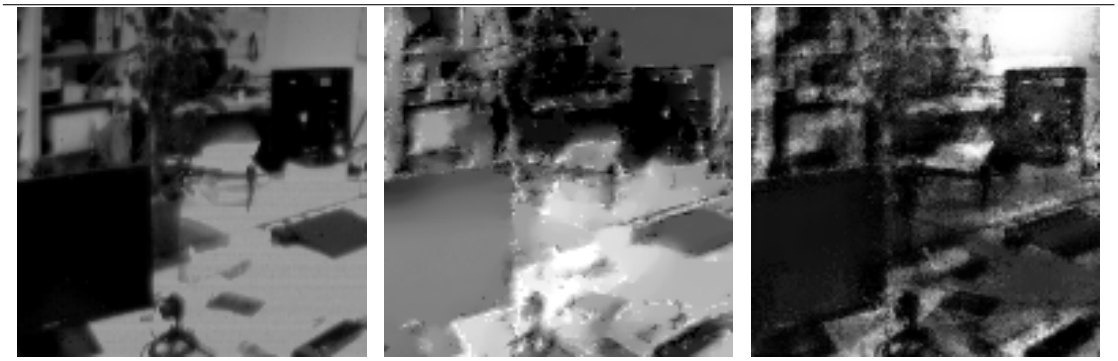
For the reconstruction loss, we changed the output of the network by removing the handcrafted reconstruction function (6.12) and instead let the network create the images directly after passing through a tanh activation layer. Counter to our previous expectation, we found that directly creating the images has a stabilising effect. We then redefine the reconstruction loss (6.12) such that:

$$\mathcal{L}_R := \|D \log I_i - DX_i\|_\theta \,. \tag{6.18}$$

By using the uncorrected image for the reconstruction loss, the generator network is free to optimise γ to satisfy the discriminator network without affecting the reconstruction loss. This scheme allows us to implicitly change the event threshold for a better reconstruction result.

6.3.2 Network and Training Changes

For further improvements, we looked into the discriminator network structure since it affects the stability of the GAN training, as discussed in Chapter 5. For this, we experimented with different distance functions for GAN training, but we found that the Wasserstein-1 distance with the gradient penalty was performing best since it explicitly defines the network gradient. However, we found that two aspects of the autoencoder discriminator network were conflicting with the gradient penalty and stopped the adversarial loss from operating correctly: The autoencoder reconstruction loss and the ELU activation functions. Up to this point, we do not have a clear answer on the cause of the conflict, but we removed them and replaced them with Leaky-ReLU activation



(a) Davis frame (b) Variational (Chapter 4) (c) Learned reconstruction

Figure 6.10: Comparison between the variational method from Chapter 4 and our learned reconstruction using the learned gamma correction. The image showed an office scene captured in the "Office Zigzag" sequence from the event data benchmark dataset provided by Mueggler *et al.* [61].

and the standard single neuron output discriminator, which is more commonly used in the GAN literature [43, 37].

We also changed the discriminator training by only selecting 1 random reconstructed image per sequence for the discriminator mini-batch, to guarantee the independence of the samples for the Wasserstein-1 distance. By not using the whole sequence we strengthen the statistical soundness of the minimisation scheme. However, we still use the whole sequence for the reconstruction minimisation, since this task relies not on the distance itself but on the information of the discriminator for the reconstruction.

Another change is the removal of the photometric transformer from the reconstruction network since it provided fewer benefits in combination with the other changes.

6.3.3 Gamma Corrected Results

In this section, we highlight our results with introduced learned gamma correction and compare them against the corresponding DAVIS frame results. For a fair comparison, we use the same training dataset as our previously trained results without gamma correction.

Figure 6.11: Sequence of a moving head, captured by standard camera (first column) and reconstructed by previously proposed method from Chapter 4 (second column) and by our proposed semi-supervised method (third column).

Office Zigzag Sequence

In Figure 6.10 we show a direct comparison of the reconstruction results with the previous variational approach from Chapter 4. The scene was captured by a moving DAVIS panning over an office scene with various objects in the frame. In the front are parts of a computer monitor visible on a desk with various items next to it. In the background is another monitor on a desk with a houseplant next to it and bookshelf. In the DAVIS frame in Figure 6.10a, over- and underexposure artefacts are visible, which are apparent when compared to the learned reconstruction results in Figure 6.10c. The reconstruction results contain many of the details which are visible in corresponding to the DAVIS frame and even reveal details on the desk, which are not visible in the overexposed region in the DAVIS frame. The variational method in Figure 6.10b shows brightness artefacts in the centre of the image, but is able to reveal many of the details in the scene as well. In comparison to our previous results without the gamma correction, the reconstruction results are consistent with a consistent brightness level. However, the overall reconstruction has some fine artefacts, which give the image a noisy appearance.

Person Sequence

In this sequence, we sample several images from a DAVIS sequence capturing a close-up of a person. In Figure 6.11, the results of our learned methods reveal many of the details which correspond to the DAVIS frames. All frames show good reconstruction results, however, artefacts are present in the regions of the face. The results of the variational method suppress many of the details, which are especially apparent in the background.

Our learned intensity reconstruction shows a more consistent reconstruction in larger image regions and shows good performance in revealing smaller details like fabrics of the person's cloth and details of the beard. The reconstructions show strong similarities to the corresponding DAVIS frames, however, artefacts on the face show that the method does not yet perfectly reconstruction the image. However, these qualitative improvements of the reconstruction using gamma correction encourage us that using this learned correction scheme in network training provides the foundation for further development in learned event-based reconstruction methods.

Finally, we also tested the method using images from the ImageNet benchmark dataset [44] instead of our HDR dataset to weaken the correlation between the event data and

Figure 6.12: Training results using gamma correction during training.

the natural images. Figure 6.12 presents the reconstruction results of the training with ImageNet data which shows a more naturalistic appearance as well. However, we also observed that the results contained more regions with inconsistent brightness and contrast values than in the results which are trained with the HDR dataset. The ImageNet dataset is composed of images from different sources with different gamma values, which makes learning a specific gamma value for our reconstruction method harder. Because of this, we believe that using an image dataset in which each image has the same gamma value is important for successful reconstruction.

6.4 Conclusion

In this chapter, we presented the first unsupervised deep-learning algorithm for image reconstruction from an event camera and demonstrated its performance compared to previous non-learned methods. The method relies only on unsupervised event data and an unrelated natural image dataset so is very generally applicable to different sorts of event data and variations in camera settings. We also showed that the training can be further supported by using corresponding DAVIS images as partial labels. We also introduced a learned gamma correction scheme into the network training, which revealed more consistent results but still containing image artefacts.

While our proposed learning-based method is able to retrieve more details of the scene than previous methods and preserves object boundaries, it has issues with reconstructing global brightness. This leads to areas of the image seeming darker than others which our photometric transformer is unable to mitigate. We hope in future work to add more

constraints which could potentially fix this issue.

Looking further ahead, it is clear that our unsupervised network's ability to reconstruct blur-free images indicates that it has a strong implicit understanding of temporal event correspondence, and therefore it is plausible that with a relatively small change it could also be set up to output optical flow, hopefully with better properties than previous learned methods which struggle for instance with estimating motion accurately at occlusion boundaries.

Finally, this method currently works in a batch fashion, and in the future, we hope a recurrent approach could enable sequentially operation. In the longer term, a Spiking Neural Network would be a perfect fit with the asynchronous input event data and might enable high efficiency on neuromorphic processing hardware.

Chapter 7

Event-Based Motion Estimation with Natural Image Priors

Contents

In Chapter 4, we showed that event-based intensity and motion estimation are coupled problems. By using priors on both, we were able to jointly estimate both using combined variational optimisation. However, while this approach proved to be successful it created artefacts in the intensity and motion results as well.

To improve the intensity reconstruction results, we showed in Chapter 6 how deep learning methods can achieve better reconstruction performance by learning a natural image prior from standard camera image examples. By using an adversarial network during training while enforcing event data-fidelity, we were able to reconstruct intensities

without using supervised training data. The results successfully revealed finer details which were suppressed in the results from our variational approach from Chapter 4.

Although in Chapter 6, we did not attempt motion estimation, our knowledge that events coupled information about intensity and motioned leads us to believe that our reconstruction network provides the basis for motion estimation as well.

In this chapter, we investigate combinations of motion estimation and our learning based reconstruction method. We compare the results of the combinations with the variational method from Chapter 4.

7.1 Related Work

Besides this work, other learning-based event camera work has recently emerged which uses deep-learning to estimate motion from event data. However, in contrast to our efforts, these methods rely on synchronised frame-based cameras such as the DAVIS to provide labelled data. For instance, Zhu *et al.* [107] proposed using DAVIS-frames to calculate optical flow by a hand-crafted method and use the result as labelled data to train a neural network to predict optical flow given event data. The authors present good qualitative optical flow results, which make this a very promising approach. However, the quality of the estimation highly depends on the quality of the DAVIS frames and the frame-rate of the camera, which are the arguments which motivated our unsupervised learning-based approach.

Subsequent work was presented by Shedligeri *et al.* [89], which used DAVIS frames in a more indirect manner. The authors assume a static environment and incorporate 3 separate neural networks. The first network uses the DAVIS frames to predict a depth map, the second network maps events to an initial low-quality image estimation and the third network warps and merges the predicted image to higher quality versions. This is a very different approach for image reconstruction from event data, using depth estimation and warping, instead of optical flow. However, not using optical flow restricts the estimation to a static environment, and also the depth estimation requires synchronised frames from a standard camera.

Using a frame-based camera to support event-based estimation is, of course, a valid approach, but it limits the estimation done by with the event data because of the inherent

limitations of standard cameras, which motivated the design of event camera in the first place. These works are a valuable contribution to the development of event-based algorithms, but we still believe that the goal should be to perform estimation based on the events alone as much as possible.

7.2 Combined Learning-Based Image Priors and Optical Flow

We present here two optical flow estimation approaches which utilise the learned reconstruction, which we will then compare with our previous variational method.

7.2.1 Handcrafted Optical Flow

Our first approach for recovering motion from our learning-based intensity reconstruction method is straightforward. We run our method from Chapter 6 as before to reconstruct an intensity sequence at a chose frame-rate from a set of events. We then apply a standard image-based optical flow estimation method to the reconstructed frames. Our insight is that the better quality of intensity reconstruction, the more effective this simple motion estimation method will be

For optical flow estimation from reconstructed frames, we use a similar variational formulation as we introduced in Chapter 3. Formally, we estimate an optical flow field $\mathbf{u}$ by minimising:

$$\min_{\mathbf{u}} \int_{\Omega} \|\nabla \mathbf{u}\|_1 + \lambda \|\langle \nabla Z_i, \mathbf{u} \rangle + Z_i - Z_j\|_1 \mathrm{dx}, \tag{7.1}$$

where $\lambda > 0$ is a scalar weight and Z_i and Z_j are a pair of consecutive pair of reconstructed images.

We improve the above minimisation method with coarse-to-fine estimation. As discussed in Chapter 1, optical flow estimation is based on a local approximation of the photometric consistency function, which is not able to express larger displacements between the images. To address such displacements, a common strategy to use coarse-to-fine estimation, which describes a multi-resolution solution. First, a low-resolution estimation is created and subsequently, that solution is used as initialisation of higher resolution

estimates. This process is repeated until the full resolution problem is solved. The strategy helps to cover larger displacements in the solution since the distance shrinks in the lower resolution version, which then covers more space by the local approximation.

To increase the resolution of the estimate, we rely on subsampling the domain of the bilinear interpolation function which we apply separately to the optical flow coefficients. The bilinear function is defined for an image function I such that:

$$\begin{aligned} B(x,y) :=& (1-b)[(1-a)I(i,j) + a\,I(i+1,j)] + \\ & b\,[(1-a)I(i,j+1) + a\,I(i+1,j+1)], i = \lfloor x \rfloor, j = \lfloor y \rfloor, \end{aligned}$$

where $a = x - i$ and $b = y - j$ are the bilinear coefficients which weight and interpolate the four pixel values.

7.2.2 Learned Optical Flow

The second approach we evaluate is a new combined learning approach for image reconstruction and optical flow estimation. The combination can be viewed as a fully learned version of our variational method from Chapter 4. Estimation of the optical flow and image reconstruction influence each other during training, which creates a joint estimation problem. For the estimation, the network uses two separate outputs: One for the image reconstruction and the other for optical flow predictions. The network output for image reconstruction is defined as in the previous chapter (using learned gamma correction), where the image is given out directly with a tanh activation, and its flow output is defined as a two-channel image per reconstructed image pairs with a linear activation function. In other words, the network produces for N event-based image reconstructions $N - 1$ optical flow estimates which describe the motion between the images.

The learned image reconstruction is regularised by an adversarial loss, as described in the previous chapter. However, for the optical flow estimation, we use the handcrafted loss from Equation (7.1), which is used in unsupervised optical flow training as well [101]. We could also train optical flow with supervised data [23]. However, this would require labelled event data, which challenging to create or would be synthetic in nature.

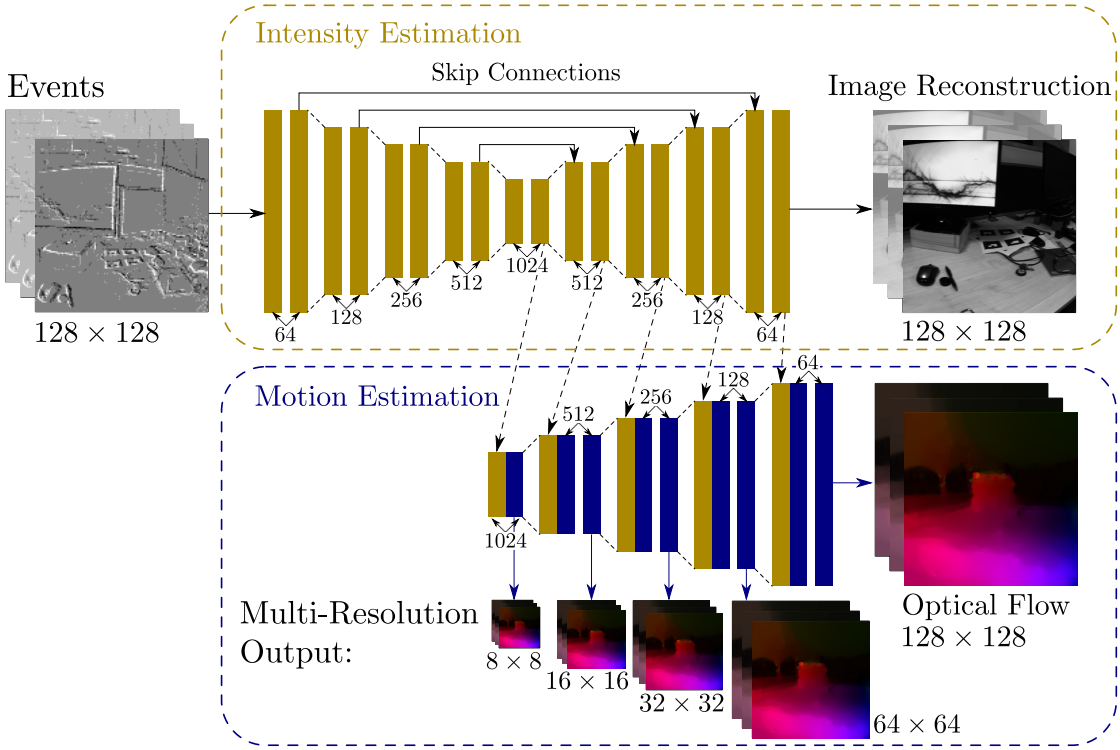


Figure 7.1: Structure of our event-based optical flow network. The network has to separate output for the image reconstruction and the optical flow.

Formally, using the notation of the previous chapter, we incorporate the optical flow estimation into our training scheme by adding to the generator objective the photometric consistency loss which is defined here as:

$$\mathcal{L}_{\text{Flow}} := \mathbb{E}_{X_i \sim p_E}[\|W(I,U) - I\|_1], I = G(X_i;\omega_G), U = H(X_i;\omega_G), \tag{7.2}$$

where G is the reconstruction network with ω_G as the network parameters and H is the optical flow network which shares its parameter with G. W is a bilinear warping function, which warps the image with the corresponding flow field. For the flow estimation, we use TV-L^1 regularisation which is defined by Equation (3.17) and which has been successfully applied in other unsupervised optical flow training methods [36, 57]. Formally, we define

the regularisation such that:

$$\mathcal{L}_{\text{Flow-TV-L}^1} := \mathbb{E}_{X_i \sim p_E}[\|\nabla U\|_1], U = H(X_i; \omega_G), \tag{7.3}$$

where ∇H is the flow gradient along the image axis, which we define as the mixture-norm as in Equation (3.17). With the optical flow loss and its smoothness regularisation, the total loss is then defined such that

$$\mathcal{L}_G := \lambda_1 \mathcal{L}_R + \mathcal{L}_A + \lambda_2 \mathcal{L}_{\text{Flow}} + \lambda_3 \mathcal{L}_{\text{Flow-TV-L}^1} \tag{7.4}$$

where $\lambda_1 > 0, \lambda_2 > 0$ and λ_3 are scalar parameters which weight the influence of the individual terms to the overall generator loss. $\mathcal{L}_R$ and $\mathcal{L}_A$ are the event-data reconstruction loss and the adversarial loss which we introduced in Chapter 6. For training, we use the learned gamma-correction since is showed more consistent results in the intensity reconstruction.

We require for the learning-based optical flow estimation a similar coarse-to-fine strategy as for the handcrafted estimation which supports the network in learning larger displacements in the image. However, in the previous approach, the iterative nature of the estimation allowed for a sequential refinement of the estimation from a coarse to a finer resolution, while feed-forward networks create a "one-shot" estimation. The network output is the final estimation which is not further improved, except during training.

To adapt the network to the coarse-to-fine scheme, we augment it at different layers with additional outputs which create different resolution estimates of the optical flow. The augmented network structure is depicted in Figure 7.1. Multi-output networks are commonly used for unsupervised estimation [24, 106], which provides guidance for estimation.

However, using the optical flow loss for different resolutions means that we require a multi-resolution image reconstruction. For the estimation, we simply downsample the image reconstructions, which does not require any alteration of the original loss.

We proceed with a comparison between the proposed optical methods to understand

each method's advantages and potential drawbacks, which we discuss at the end of this chapter.

7.3 Comparisons

In this section, we present our comparative results between the methods described in this chapter and our previous variational-based estimation from Chapter 4. For the comparison, we use several sequences from the event benchmark dataset provided by Mueggler *et al.* [61] which consists of events and camera image from a DAVIS. We use several scenes from the dataset with free camera motion (6 degrees of freedom motion (DoF)) in different scenes with outdoor, indoor, and textured planar environments. Also for comparison, we also use two sequences capturing the same scene ("office spiral" and "office zigzag") but with smooth and rough camera motions, which provide insights into the sensitivity of the estimation methods to sudden motion changes.

For the comparison, we first provide a quantitive evaluation, where we compare our estimation against corresponding DAVIS frames and optical flow estimation results based on the DAVIS sequences. We then show a qualitative comparison between the different methods.

7.3.1 Quantitive Comparsion

Here we compare the performance of the intensity reconstruction and the optical flow estimation. To evaluate the reconstruction, we match the estimation against corresponding DAVIS frames to measure how well the methods retrieve the image structure. However, the exposure of DAVIS frames influences the brightness and contrast in the image, which we compensate for in the evaluation by normalising the frames and the reconstructions before matching, which we detail below.

For the optical flow evaluation, we compare the motion estimation results against optical flow results based on the DAVIS frames with a comparable method. Since our optical flow estimation formulation 7.1 and our previous method is based on the TV-L^1 formulation, we use the primal-dual method as described in Chapter 3 to retrieve the optical flow from the DAVIS frames. Using this method to retrieve ground truth optical flow is the most reasonable option since an event-based benchmark dataset is not

	Variational (Chapter 4)	Learned Reconstruction
Boxes 6 DoF	0.959	**0.692**
Dynamic 6 DoF	1.01	**0.852**
Office Spiral	0.706	**0.609**
Office Zigzag	0.884	**0.647**
Outdoors Walking	0.616	**0.512**
Poster 6 DoF	1.028	**0.676**

Table 7.1: Mean absolute error of the intensity reconstruction of the contribution of Chapter 4 and the learned contribution using sequences from [61].

available yet. However, by using the same variational formulation for our methods, the ground truth estimates provide a benchmark on the performance of our methods, if the intensity is well estimated. However, since the DAVIS frames are exposure-based, we scale the flow estimate to match the estimation intervals of our methods to the actual frame-rate of the DAVIS, which we describe below.

Intensity Reconstruction

Here we evaluate the performance of our reconstruction methods by comparing them to corresponding DAVIS frames. However, for the comparison, we need to address some caveats of the reconstruction, namely: Brightness and contrast.

Due to its differential nature, event data does not carry any information about absolute brightness, which means that a reconstruction method essentially invents an absolute level of brightness. Another issue is the mismatch between the image contrast and the measured contrast by the event-camera. As mentioned in the previous chapter, the difference between bright and dark regions in the image is related to the event threshold and its relation to the gamma correction function. For our comparison, we address these issues by normalising the brightness and contrast of the camera frames and the image

reconstruction. Formally, the normalisation of a given image I is defined such that

$$\hat{I}(x,y) := \frac{I(x,y) - \mu}{\sigma}, \tag{7.5}$$

where μ is the mean pixel value and σ is the standard deviation of the image. We then define a performance metric for which we use the *mean absolute error* (MAE), which is defined as:

$$\frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} |\hat{I}(x,y) - \hat{I}_{\text{gt}}(x,y)|, \tag{7.6}$$

where $\hat{I}_{\text{gt}}$ is the normalised ground truth frame, which we use as the reference and $\hat{I}$ is the reconstructed frame we are evaluating. As described in the previous sections, we use the corresponding DAVIS frames as the ground truth frames.

With the reconstruction metric described, we provide an overview of the quantitive results in Table 7.1 in which we compare the image reconstruction results of our first contribution from Chapter 4 to our joint learned image reconstruction, which we introduced in this chapter. The Table shows the results of the average MAE over the entire sequence and it shows that the learning based approach performs better in all sequences than the previous handcrafted approach. From the results of the previous chapter, we saw that the learned reconstruction is better in reconstructing details of the scene, while the TV-L^1 smoothness prior tends to suppress these details, which inadvertently increase the reconstruction error.

For a closer look at the "Boxes 6 DoF" sequence, which has a greater difference between the reconstruction results, we visualised the reconstruction error of different parts of the sequence in Figure 7.3. The sequence captures a stack of textured boxes in an indoor environment while the camera is moved freely in a panning motion with increased speed towards the end of the sequence. Figure 7.3 subdivides the sequence into 10 different sequences of equal length, in which we see that the variational method shows comparable results in early parts of the sequence results but in later parts the error increases. One reason for the increased error of the variational estimation is accelerated camera motion in the later parts of the sequence, which would require an adjustment of the frame-rate

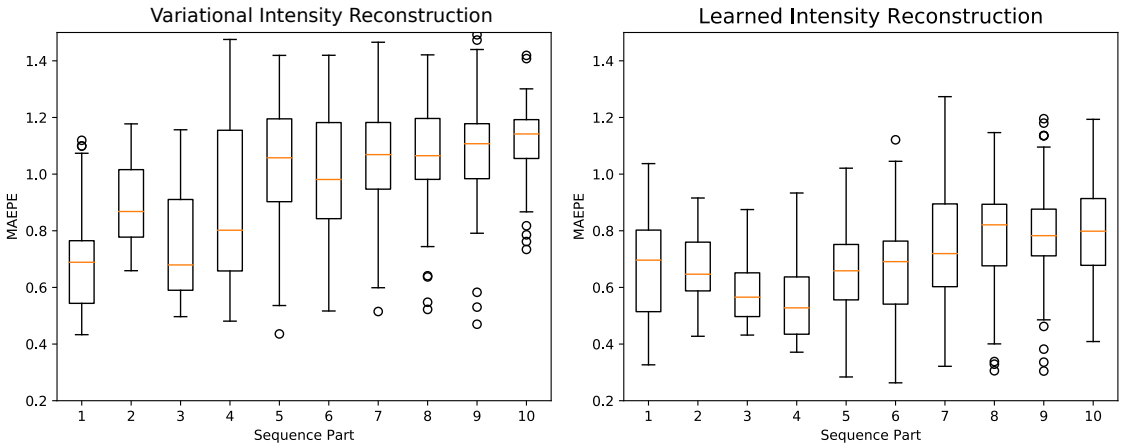


Figure 7.2: Box plot of the MAE of the sequential parts of the "Boxes 6 DoF" sequence from [61]. The sequence is subdivided into 10 part of equal length and ordered chronologically.

of the estimation. The learned methods perform well since its estimation is based on a constant number of events, which independent of the speed of motion. Another reason is the iterative process of the variational estimation, which propagates the previous estimate. If the estimate is suboptimal, then this estimate initialises the estimation of new incoming events into the optimisation window. The learned approach is a batch estimation process, in which each estimation is independent of previous estimations. This means that those sub-optimal estimations are not affecting the results of the overall estimation.

We observe the propagation of sub-optimal results in different sequences and it indicates that iterative event-based processes might be very susceptible to these issues.

Optical Flow Estimation

For the evaluation of our optical flow estimation, we can rely on commonly used metrics for motion estimation. For this work, we use the *mean absolute endpoint error* (MAEPE), which was originally proposed in [71] and is used in other optical flow benchmarks like the Middlebury benchmark dataset [6]. Formally the MAEPE is defined

	Variational (Chapter 4)	Learned Reconstruction+ Handcrafted Flow	Jointly Learned Reconstruction and Flow
Boxes 6 DoF	7.564	6.496	**5.79**
Dynamic 6 DoF	6.768	**4.641**	5.825
Office Spiral	2.424	**1.221**	2.168
Office Zigzag	2.695	**1.708**	2.508
Outdoors Walking	**1.331**	2.215	1.672
Poster 6 DoF	7.077	6.285	**6.128**

Table 7.2: Mean absolute end point error over the entire sequence per entry for the optical flow estimation of the variational estiamtion, optical flow estimation given two recontructed images and learned optical flow estimation. For a fair comparison, we filter out for the learned optical flow estimation would require re-scaling below 0.2 and over 1.8 as described by Equation (7.8).

such that

$$\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\sqrt{(u(x,y)-u_{\text{gt}}(x,y))^2+(v(x,y)-v_{\text{gt}}(x,y))^2}, \tag{7.7}$$

where $\mathbf{u}_{\text{gt}} = (u_{\text{gt}}, v_{\text{gt}})^T$ is the ground-truth optical flow vector. However, since the event-based sensor is frameless, the estimated flow needs to be adjusted to the frame-rate of ground-truth optical flow from the corresponding frame-based camera. We assume that it the mismatch between the frame-rate and the event-based optical flow estimation is relatively small, then the flow estimate $\mathbf{u}$ from a corresponding event integral $E_{i:j}$ can be rescaled such that:

$$\mathbf{u}' = \frac{|t_n - t_p|}{|t_j - t_i|}\mathbf{u}, \tag{7.8}$$

where t_i and t_j are the timestamps of the i-th and j-th event and t_n and t_p with $t_n > t_p$ are the timestamps of the closest camera frames. In other words, we scale the optical flow according to the timestamps of the events and camera frames. If the intervals $|t_n - t_j|$

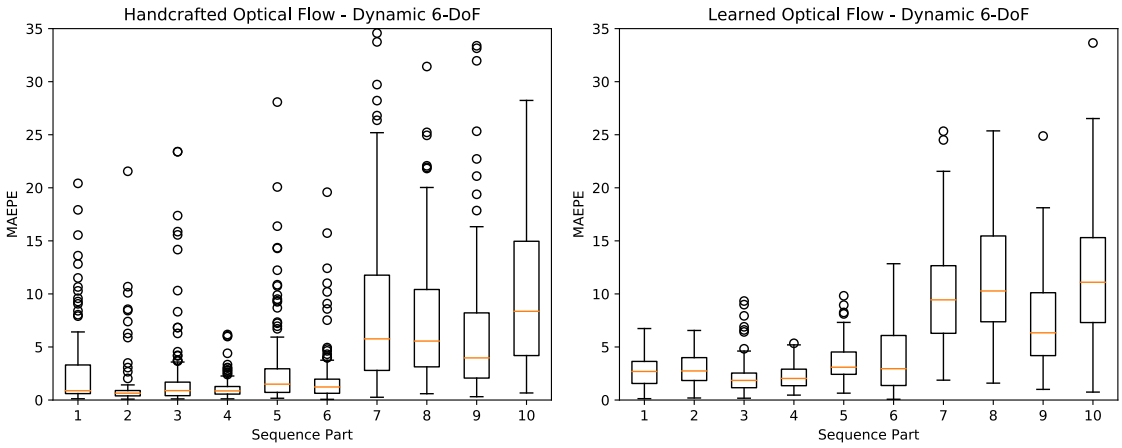


Figure 7.3: Box plot of the MAEPE of the sequential parts of the the "Dynamic 6 DoF" sequence from [61]. The sequence is subdivided into 10 parts of equal length. For a fair comparison, we filter out for the learned optical flow estimation which would require re-scaling below 0.2 and over 1.8 as described by Equation (7.8).

or $|t_p - t_i|$ are too long, we discard the match since the error introduced by rescaling would be greater than the actual endpoint error.

Using the above described metric and normalisation, we provide an overview of the quantitive results in Table 7.2. The results show that the learned approaches perform better in the majority of the sequences, indicating indeed that the learned prior provides better optical flow estimates.

In Figure 7.3, we look more closely into the comparison between the learned reconstruction and the handcrafted optical flow estimation methods based on the image reconstructions. The sequence captures a person sitting at a desk in an office scene, while the camera moves freely. During the sequence, the person stands up from the desk and walks around and the camera motion becomes more rapid. In Figure 7.3 we see that both approaches perform similarly over the duration of the sequence, with both performing worse during the rapid motion of the camera. Overall the handcrafted optical flow method performs better but is also contains more outlier estimates than the learned approach.

Another thing to notice in Table 7.2 is the performance of the learned method in the "Boxes 6 DoF" and the "Poster 6 DoF" sequences, which capture very textured objects.

The learned approach fits very closely to the edges of an object, while not performing well in more flat regions of the image. In both sequences, the textured object creates strong gradients in the scene while having fewer flat image regions.

Lastly, the variational estimation method performs best for the "Outdoors Walking" sequence. This sequence captures an outdoor environment in which the handheld camera is moved from an urban location next to a forest back to the urban location. The sequence contains in part very structured scenes, but largely contains the sky and flat image regions of the buildings. The structure is very well fitted by the TV-L^1 smoothness prior. Another advantage is that prior is very effective in suppressing noise and throughout the sequence between the sky and the ground there are many outlier-events created. The cause for the outliers is not known, but it creates may false measurements, which are detrimental for the learned methods, which do not suppress these measurements.

We continue with a series of qualitative comparisons, in which we evaluate the reconstruction results against the DAVIS frames as well as the optical flow estimates.

7.3.2 Qualitative Comparsion

Face Sequence

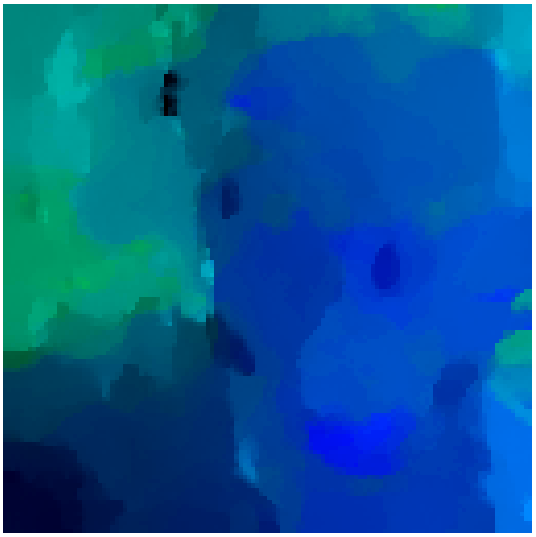
In this experiment, we use a sequence from another dataset which is not part in the event benchmark dataset by Mueggler *et al.* [61]. It shows a close-up of a moving a human head. The sequence is captured by a DAVIS as well, which moves freely with an office environment in the background. The results of the reconstruction and motion estimations are shown in Figure 7.4. We see that the learned image reconstruction is quite capable of reconstructing finer details of the head as well as details of the scenes in the background. However, image artefacts are apparent on the face which do not correspond to the face in the DAVIS frame. Despite these artefacts, the reconstruction shows a clear the separation between the human head and the background and good distinction between the foreground and background.

Handcrafted optical flow estimation based on the reconstruction in Figure 7.4d shows motion directions similar to the ground truth optical flow field. However, the noisy artefacts in the image reconstruction seem also to influence the flow estimation, which creates a rather patchy look than the ground truth flow. Despite the patchiness, the

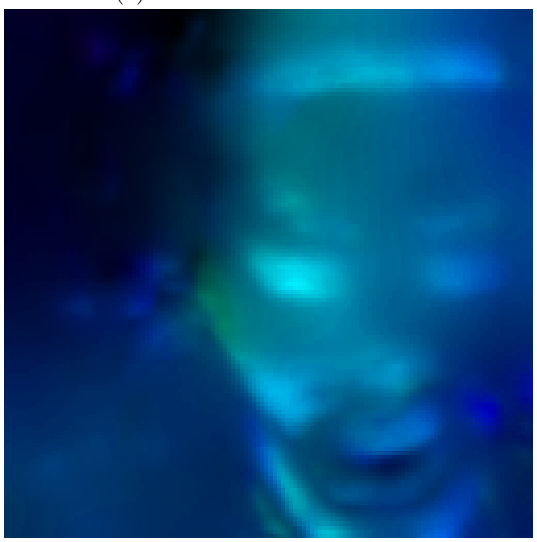
(a) Davis frame

(b) Ground truth flow

(c) Learned reconstruction



(d) Learned reconstruction + handcrafted flow



(e) Jointly learned reconstruction and flow

(f) Variational flow (Chapter 4)

Figure 7.4: Estimation results of the learned image reconstruction, handcrafted optical flow estimation on these reconstructions and the learned optical flow estimate.

flow field is clearly distinguished between the head and the background motion.

In comparison, the learned optical flow estimation is smoother and does not contain the patch artefacts. However, in larger image regions the estimation predicts small or no motion and fits primarily to strong gradient regions, but neglects to propagate the motion estimation to image regions which are further away from those gradients. This behaviour of the estimation causes the flow to create motion direction which only fit the local image regions, which do not necessarily agrees with the global motion like the handcrafted approach.

The variational method in Figure 7.4f, shows a distinct motion boundary between head and background, however, the method fails at estimation the motion direction. The method like the learned flow estimation fits closely to strong gradient regions in the image.

Outdoors Walking

In this experiment, we use the "Outdoors Walking" sequence from the event benchmark dataset by Mueggler *et al.* [61]. We test here the result of our methods in an outdoor environment, which highlights their capacity to handle high contrast. In Figure 7.5, the scene shows a cyclist moving in the foreground with a forest background. The scene otherwise shows parts of the road at the bottom of the image with the sky and trees at the top.

The reconstruction in Figure 7.5c reveals many details of the scene, albeit with image artefacts which makes the appearance noisier than the reconstruction of the previous experiment. At first, the reconstruction seems to fail to reconstruct smaller details like the face of the cyclist. However, comparing it to the DAVIS frame in Figure 7.5a we see that the resolution of the camera is not enough to reveal this in the camera image as well. The reconstruction reveals details of the road, the cyclist and even some finer details of the trees in the background. In the sky, however, "Salt and Pepper" like image artefacts appear which do not correspond to the DAVIS frame. The reason for the artefacts seems to be outliers in the corresponding event signal, which seems to be incorrect measurements of the device which our learned method is not trained to handle.

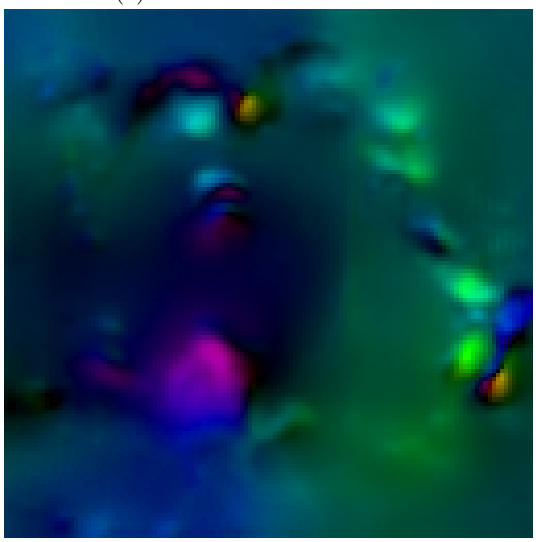
The optical flow estimation by the handcrafted estimation in Figure 7.5d, estimates

(a) Davis frame

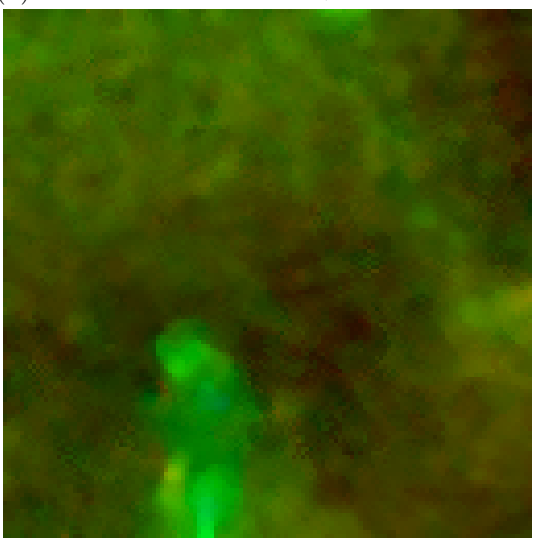
(b) Ground truth flow

(c) Learned reconstruction

(d) Learned reconstruction + handcrafted flow



(e) Jointly learned reconstruction and flow



(f) Variational flow (Chapter 4)

Figure 7.5: Estimation results for the "Outdoors Walking" sequence. The image shows a cyclist in front of a small forest. The results are shown with the corresponding DAVIS frame, event signal and ground truth optical flow.

quite well the optical flow in comparison to our ground truth optical flow field, in which the cyclist is clearly distinct from the motion in the background. The motion direction of the background differs slightly from the ground truth. However, the contour of the forest is aligned with the ground truth. Like in the previous experiment, the handcrafted flow estimation has a block-quality in the motion field.

The learned optical flow estimation in Figure 7.5e perform significantly worse. The motion of the cyclist is visible in the flow field, however, the method fails to estimate the motion direction properly. The flow field seems to strongly fit the boundary between the trees and the sky.

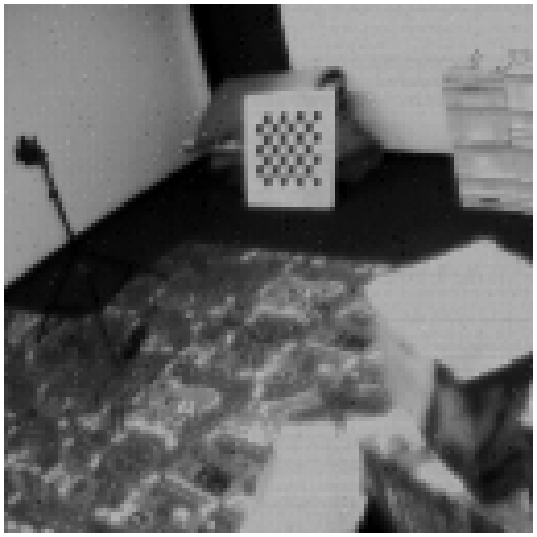
In comparison, the variational estimation method in Figure 7.5f, seems to mostly fit locally well and does not propagate the motion direction well. This can be related to the estimation scheme, which does not include coarse-to-fine estimation. The cyclist's motion is also distinguished from the background motion, but the estimation seems to mostly fit the boundaries of the objects and does not perform as well globally.

Boxes 6 Degrees of Freedom

Here we discuss our results on the "Boxes 6 DoF" sequence, for which we show the results in Figure 7.6. The scene captures a stack of textured boxes in the foreground with a textured carpet on the ground in an indoor environment. The camera is pointed toward the far end corner of the room with a checkerboard standing upright. The reconstruction shows the various objects in the room. For instance, the black and white squares of the checkerboard are visible in the reconstruction results in Figure 7.6c, as well as the tripod standing next to checkerboard. We also note that the textures on the top of the boxes are visible in the image reconstruction while in the DAVIS frame in Figure 7.6a they are over-exposed.

The result of the handcrafted optical flow estimation in Figure 7.6d is lining up closely with the ground truth optical flow estimation but shows greater motion in the background. The motion field of boxes in the foreground is clearly distinguished from the scene motion in the background.

The learned optical flow estimation also gives the same background motion direction as the ground truth flow field, but it seems to overrule the motion of the boxes and the

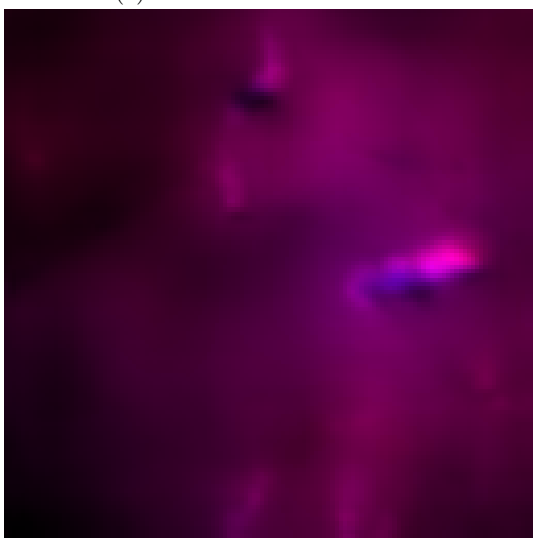


(a) Davis frame

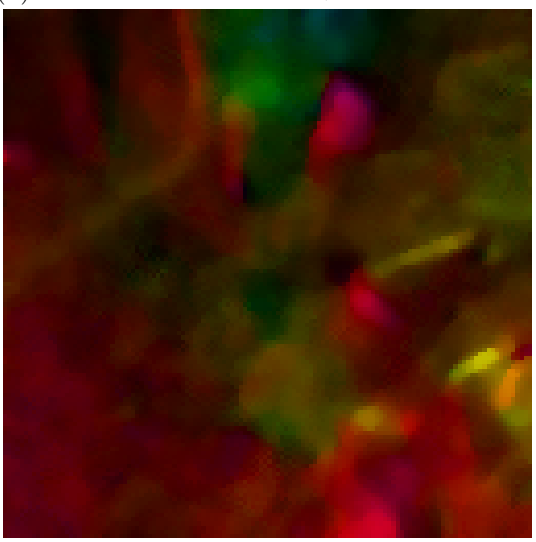
(b) Ground truth flow

(c) Learned reconstruction

(d) Learned reconstruction + handcrafted flow



(e) Jointly learned reconstruction and flow



(f) Variational flow (Chapter 4)

Figure 7.6: Estimation results for the "Boxes 6 DoF" sequence. The image shows an indoor environment with boxes in the foreground and different objects in the background.

learned estimation fails to properly reconstruction the motion. The estimation shows the contour of the boxes, which indicates that the flow estimation fails here to properly propagate the motion from the edges of the boxes.

The variational estimation in Figure 7.6f has similar issues as the previous experiment, in which the flow field fits locally to the edges of the textured objects, but does not propagate to a global consensus. Introducing here a coarse-to-fine approach would also address this issue, and without it seems that the motion field does not reach a global consensus.

Dynamic 6 Degrees of Freedom - Part 1

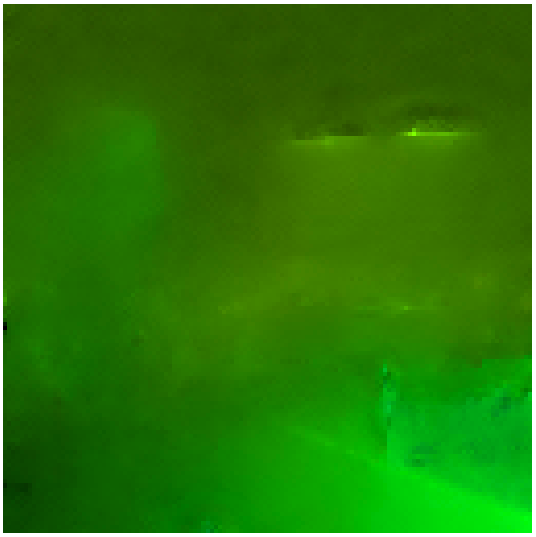
In this experiment, we discuss our results with the "Dynamic 6 DoF" sequence, which are shown in Figure 7.7. The scene shows a person sitting at a desk inside an office environment. On the desk are several objects like a computer monitor, mouse and sheet of paper.

In the reconstruction results in Figure 7.7c, we see that many of the details are reconstructed like the wrinkles on the shirt of the person and several items on the desk. However, there are also noisy artefacts around the objects, like a darker border around the computer monitor. However, the object boundaries are well defined in the image, like in previous experiments.

However, the image artefacts influence the handcrafted optical flow estimation in Figure 7.7d, which creates patches of motion field which do not correspond the ground truth optical flow estimation. Despite these artefacts, the overall estimation lines up with ground truth optical flow field.

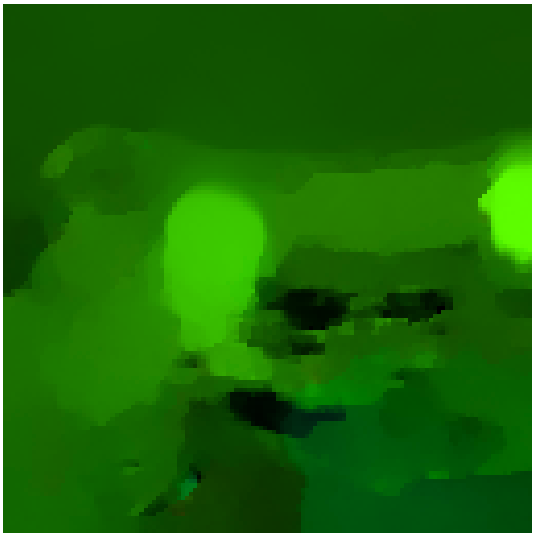
In comparison in Figure 7.7e, the learned optical flow estimation has a smooth motion field, but like in the previous experiments, the results fit closely to edges of the objects while neglecting larger regions. This is especially seen in the textureless background, where the motion is close to zero. Locally this estimation is not unreasonable, since the homogeneous wall does not have an apparent motion in the sequence, but it shows that the learned optical flow method is not capable of properly propagating the motion estimation.

(a) Davis frame

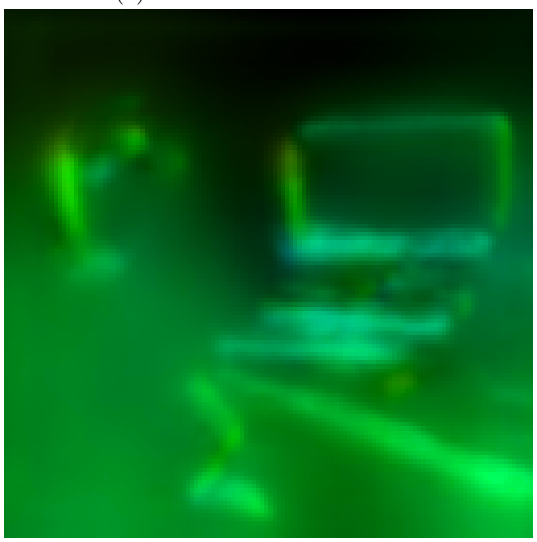


(b) Ground truth flow

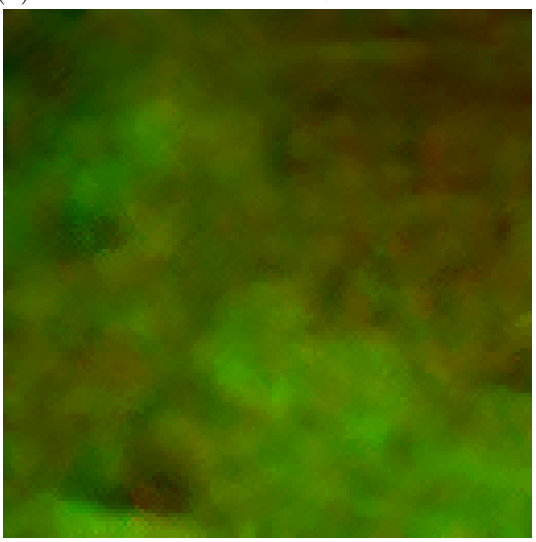
(c) Learned reconstruction



(d) Learned reconstruction + handcrafted flow



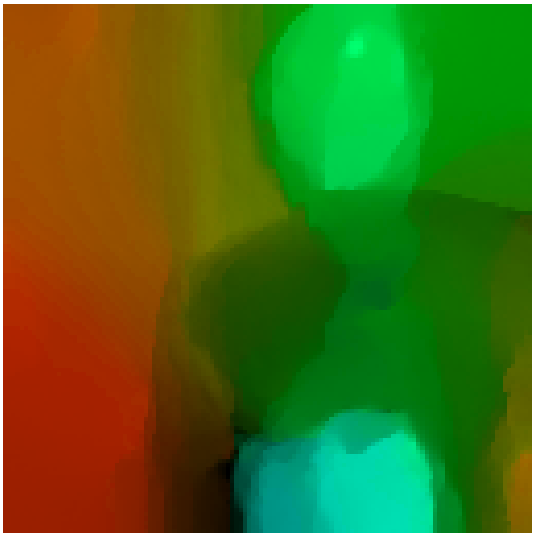
(e) Jointly learned reconstruction and flow



(f) Variational flow (Chapter 4)

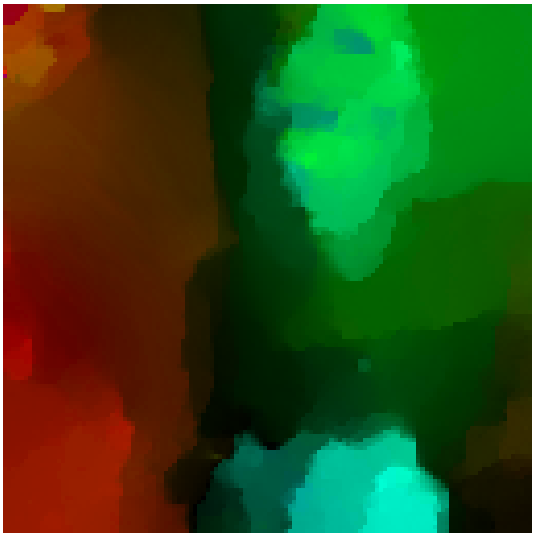
Figure 7.7: Estimation results for the "Dynamic 6 DoF" sequence. The image shows a person sitting at a desk. The results are shown with the corresponding DAVIS frame and ground truth optical flow.

(a) Davis frame

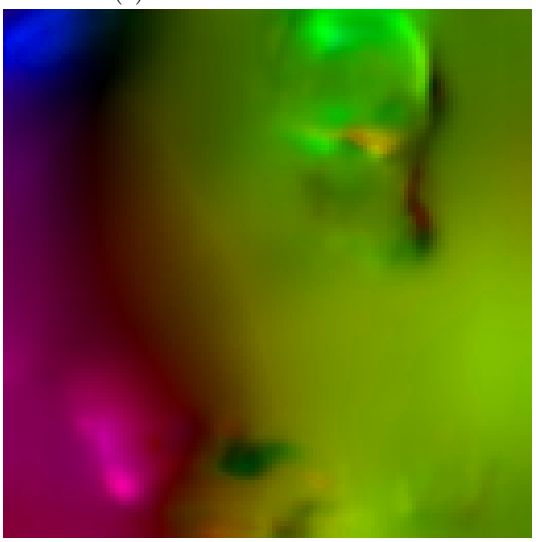


(b) Ground truth flow

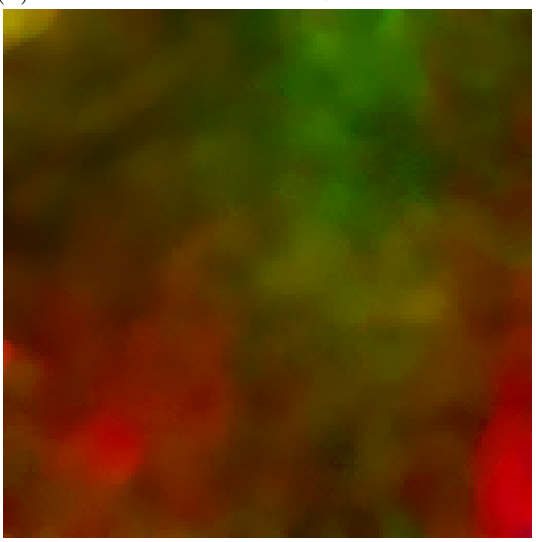
(c) Learned reconstruction



(d) Learned reconstruction + handcrafted flow



(e) Jointly learned reconstruction and flow



(f) Variational flow (Chapter 4)

Figure 7.8: Estimation results for the "Dynamic 6 DoF" sequence. The image shows a person turning to the left and holding a cup in his hand. The results are shown with the corresponding DAVIS frame and ground truth optical flow.

Dynamic 6 Degrees of Freedom - Part 2

In this experiment, we show a later part of the "Dynamic 6 DoF" sequence in which the person is standing and turns with a cup in his hand. The reconstruction shows stronger noise artefact than in other experiments, however many of the details are still visible like the cup and even some writing on the cloth of the person.

The ground truth optical flow field shows a clear difference between the person and the background which is also visible in the handcrafted optical flow results in Figure 7.8d. Overall all the details in the flow field from the ground truth like the cup motion are shown in the estimation.

The learned optical flow results in Figure 7.8e shows the outline of the person, but the boundary between the person and the background is not as clear as in the handcrafted estimation results. In the variational results 7.8f the boundary is not visible. The overall estimation of the variational approach has a patch-like appearance and many of the motion details are not visible.

7.4 Conclusion

In this chapter, we compared our variational method from Chapter 4 against motion estimates using learned reconstruction to gain insights into their individual benefits. During our experiments, we showed that the learned approach outperforms the variational approach in many different scenarios. We presented two motion estimation variants based on the learned approach: A handcrafted optical flow estimation based on the image reconstruction and an unsupervised, learned optical flow method with is trained with the image reconstruction. The quantitative comparisons between these two variants show that the handcrafted one performs better than the learned estimation and the qualitative results that the learned approach tends to give smoother results but it also fitted mostly to the edges of objects and predicts lower or no motion in larger homogenous regions. This observation is supported by the good performance of the learned approach in experiments in which the content of the sequence is dominated by strong textures.

While the experiments suggest that the handcrafted approach provides overall more consistent results, we still think that the investigation of the learned approach has been

worthwhile. At the time of this thesis, deep learning based optical flow estimation is an active field of research and many improvements in this are potentially beneficial to our method as well. For instance, our results suggest that improving the smoothness of the optical flow predictions would help the estimation. Beside this improvement there numerous other approaches for improvements like network architectures, parameterisation, training-data and so on, which have yet to be explored.

Another reason that we believe further development of the learned approach is worthwhile is the potential to avoid image reconstruction entirely for event-based motion estimation. The image reconstruction is helpful to understand the content of the video stream for a human observer and to translate standard computer vision methods to the event camera. However, if our goal is purely motion estimation, this is not required. With techniques from deep-learning like multi-task learning, it is potentially possible to train image reconstruction and optical flow jointly, as proposed in this chapter, and later for the method to discard the parts of the network which are designed for image reconstruction. This approach would additionally reduce the number of parameters used for estimation and increase the efficiency of the method.

Nevertheless, these arguments and the results in this chapter suggests that learning-based methods have desirable qualities for event-based analysis, but require exploration in the future to improve their quality. Deep-learning allows us to incorporate high-quality natural image priors into event-based estimation, which provide the estimation with the information to understand the underlying event signal, which is a powerful tool for working with the sparse signal.

Chapter 8

Conclusions

Contents

In the previous chapters of this thesis, we discussed our contributions to the analysis of the event signal and their theoretical foundations. In the following, we summarise these contributions and discuss their benefits as well as their weaknesses. We end this chapter with a discussion about potential further avenues for event-based signal processing based on our contributions and lay out potential pieces of future work.

8.1 Contributions

We presented in this thesis a series of novel approaches to the analysis of the event camera signal, which we summarise here. To provide some context for these contributions, we would like to note that at the time when we began this body of work, event signal processing methods, in general, posed severe restrictions on their application and relied on strong assumptions. At that stage, the possibility of analysing event data without or with weaker restrictions was an open question which we approached during this thesis by focusing on optical flow analysis using events alone. Throughout this thesis, we demonstrated that this kind of analysis is not only possible but that it is also intrinsically connected to the understanding of the image structure. In Chapter 4, we combined motion estimation and image reconstruction using variational minimisation. We showed

using a sliding window scheme that this estimation can be performed in real-time, but we also discussed the over-smoothing of the intensity reconstruction.

In the subsequent pieces of work in Chapter 6, we built upon these insights and explored more expressive reconstruction approaches using deep-learning. We addressed the problem of creating supervised training sets for the event camera by developing an unsupervised training scheme and demonstrated that is capable of reconstructing finer image details. However, during our experiments, we pointed out that this approach worked well for local image regions, but was globally inconsistent in terms of brightness and contrast. To address this issue, we established a connection between the event threshold and the gamma correction function, which is related to the properties of contrast. We incorporated a gamma correction into the training method and showed that it is indeed able to provide more consistent brightness and contrast in the reconstruction, albeit with artefacts still present.

With improved image reconstruction, we revisited the combined estimation of image and motion in Chapter 7 and compared our first contribution with two optical flow methods using the learned image reconstruction: A handcrafted optical flow method using the image reconstruction and a fully learned optical flow estimation. In a series of experiments, we showed that the learned image reconstruction showed a strong improvement to our first contributions. However, we also see many areas in which this estimation can be improved upon, which we discuss in the next section.

We summarise in the following our contributions. Due to the novel nature of the sensor, the body of work in this thesis has several contributions which are the first methods in this field:

- The first image reconstruction method from an event signal which allows free camera motions as well as dynamic object movements.
- By reconstructing the image presenting the first practical demonstration that the event camera signal contains enough information for monocular computer vision.
- The first dense event-based optical flow estimation method based on the photometric consistency instead of event-to-event matching.

- The first dense optical flow estimation and variational formulation minimisation with an event camera.
- The first unsupervised deep-learning image reconstruction method for event cameras as well as unsupervised optical flow estimation.
- We relate the gamma correction function to the event measurements, which we incorporate during unsupervised network training.

It is our hope that these contributions will provide the foundation to further the use of event cameras for robotic vision and allow researchers to utilise their advantages to improve the stability, robustness and efficiency of these systems. However, there many open problems which need to be addressed for the event camera as well as weaknesses of our approaches, which we address in the next section.

8.2 Discussion and Future Work

In the first chapter of this thesis, we listed the potential benefits of the event camera to the area of robotic vision and in this thesis, we provided some groundwork to make event cameras usable for computer vision applications. However, our body of work is not exhaustive and many questions and problems remain open for further research.

Throughout the body of our work, the common denominator is the understanding of the image structures of the underlying event signal, which enables other estimations like optical flow. The requirement of understanding the image structure is motivated by the discussion in Section 1.2.2 and the body of work of Barranco *et al.* [8], which guided our decision process on our event-based approaches.

Because of this reasoning, we believe that improving image reconstruction is going to be an important avenue for the development of further event-based methods. Deep-learning is a very expressive tool for event-based analysis, however, our methods still present artefacts in their reconstruction. Despite these issues, deep-learning is a fast developing field which offers many approaches to improve upon these results. Another reason for the importance of learning-based methods to event cameras is biological motivation. Both systems, networks and camera, take their inspiration from biological systems and their combination offers the potential to simulate the efficiency of human

vision. However, this research has only begun and there still many open questions about the exact function of human vision, which require answering before we can properly simulate it. However, this prospect gives hope that neural networks will provide the key for efficient event-based algorithms.

Another aspect in which our learned approach can be extended is in the development of an iterative process. Our methods are a one-shot estimation, in which each estimation is built independently. While this is a valid approach, for a sequence it is inefficient. Instead, an iterative approach can take the previous estimates and update them with newer measurements. For neural networks, the most natural choice is the exploration of recurrent neural network architectures. These networks are potentially smaller and therefore more efficient than our feed-forward networks.

However, the above approaches are based on image reconstruction and while we believe that the reasoning of inferring the image structure for the event-based analysis is sound, it is an open question whether or not it is strictly necessary. We discussed in the previous chapter that other pieces of event-based work do not infer the intensity for their estimation and that they show promising results. We are positive that this approach will lead to further promising and interesting results, but it remains to be seen if they completely overcome the necessity of image estimation for more robust estimation. In the long run, we hope that image reconstruction may be avoided for event-based estimation processes like optical flow or visual odometry since the image structure is not necessarily important for potential applications.

Another interesting avenue for future event-based research is the established connection between the gamma correction function and the event threshold which is defined by Equation 2.22. Our training method which incorporates the gamma correction function can be viewed as an attempt at an auto-calibration method, but since we are using neural networks and random standard camera images, we do not view this as a strong claim. However, we hope this will guide future research in developing such a calibration method which focuses on accuracy, which would benefit many other works in the event camera research.

Finally, another area of research is the processing of the event camera signal itself. All our methods process the entire image estimation at once for the incoming event signal

since it is the most efficient processing mode using commercially available hardware. However, it is an open question if for the asynchronous event signal this is the best computing model. An alternative would be an investigation into asynchronous computation models, which only process information as soon as it arrives. Such a system would process the event information only locally in an image region and propagate it if necessary. This area of computer vision only recently received more attention, but it offers a potentially more efficient and faster approach for the event camera.

With this, we conclude our discussion about the event camera. It remains to be seen what impact this sensor has on the computer vision community. However, in conclusion, we think whether or not the event camera will have a lasting impact, it is a worthwhile research field since it allows us to re-evaluate our methods and reflect on whether the devices which we chose are truly the most optimal devices for the problems we chose to solve.

Bibliography

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017. 115, 116

[2] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework: Part 3. Technical report, Carnegie Mellon University, 2003. 25

[3] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework: Part 4. Technical report, Carnegie Mellon University, 2004. 25

[4] S. Baker, R. Gross, I. Matthews, and T. Ishikawa. Lucas-Kanade 20 Years On: A Unifying Framework: Part 2. Technical report, Carnegie Mellon University, 2003. 25

[5] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, 2004. 13, 25

[6] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision (IJCV)*, 2011. 67, 162

[7] P. Bardow, A. J. Davison, and S. Leutenegger. Simultaneous Optical Flow and Intensity Estimation from an Event Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 78

[8] F. Barranco, C. L. Teo, C. Fermüller, and Y. Aloimonos. Contour Detection and Characterization for Asynchronous Event Sensors. 2015. 9, 13, 179

[9] R. Benosman, S. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan. Asynchronous frameless event-based optical flow. *Journal of Neural Networks*, 27:32–37, 2012. 12

[10] D. Berthelot, T. Schumm, and L. Metz. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 118, 119, 132, 135, 136, 138

[11] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988. 109

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 16, 21, 42, 45

[13] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. A 240×180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits (JSSC)*, 49(10):2333–2341, 2014. 8, 49, 52, 58, 86, 125, 132

[14] K. Bredies, K. Kunisch, and T. Pock. Total Generalized Variation. *SIAM Journal of Imaging Sciences*, 3(3):492–526, 2010. 23

[15] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 28

[16] A. Chambolle and T. Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. 64, 73, 88

[17] T. Chan and J. Shen. *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*. 2005. 61

[18] Y. M. Chi, U. Mallik, M. A. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings. CMOS camera with in-pixel temporal change detection and ADC. *IEEE Journal of Solid-State Circuits (JSSC)*, 2007. 48

[19] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. Douglas, and T. Delbruck. A pencil balancing robot using a pair of AER dynamic vision sensors. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2009. 11

[20] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger. Interacting maps for fast visual interpretation. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2011. 12

[21] J. Engel, T. Schoeps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 78

[22] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Image inpainting through neural networks hallucinations. In *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th*, pages 1–5, 2016. 23, 24

[23] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 28, 156

[24] R. Garg, V. K. B. G, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 158

[25] R. Garg, A. Roussos, and L. Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 88

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. 2014. 29, 31, 112, 114, 115, 117

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 116

[28] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Applications of the Legendre-Fenchel transformation to computer vision problems. Technical Report DTR11-7, Imperial College London, 2011. 87

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 134

[30] G. E. Hinton and J. L. McClelland. Learning representations by recirculation. In *Neural Information Processing Systems (NIPS)*, pages 358–366, 1988. 109

[31] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981. 25, 28, 80, 81, 82

[32] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 1133–1141. IEEE, 2017. 30, 31

[33] J. Huang and D. Mumford. Statistics of natural images and models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999. 20

[34] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 28

[35] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. 113, 117

[36] J. Y. Jason, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–10, 2016. 157

[37] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2018. 32, 148

[38] D. Kim and E. Culurciello. A Compact-pixel Tri-mode vision sensor. *IEEE International Symposium on Circuits and Systems*, 2010. 48

[39] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous Mosaicing and Tracking with an Event Camera. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014. 12, 55

[40] H. Kim, S. Leutenegger, and A. J. Davison. Real-time 3D reconstruction, 6-DoF tracking and intensity reconstruction with an event camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 13, 49, 138

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 138

[42] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 29, 30, 31

[43] N. Kodali, J. Abernethy, J. Hays, and Z. Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017. 115, 132, 148

[44] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2012. 107, 150

[45] S. W. Kuffler. Discharge patterns and functional organization of mammalian retina. *Journal of neurophysiology*, 16(1):37–68, 1953. 45

[46] H. Kurino, M. Nakagawa, K. W. Lee, T. Nakamura, Y. Yamada, K. T. Park, and M. Koyanagi. Smart vision chip fabricated using three dimensional integration technology. *Advances in Neural Information Processing Systems*, 13:720–726, 2001. 53

[47] W.-S. Lai, J. Huang, N. Ahuja, and M.-H. Yang. Deep Laplacian pyramid networks for fast and accurate super-resolution. *arXiv preprint arXiv:1704.03915*, 2017. 137

[48] J. A. Leñero Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco. A signed spatial contrast event spike retina chip. *IEEE International Symposium on Circuits and Systems*, 2010. 48, 56

[49] Y. LeCun. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989. 107

[50] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015. 101, 102, 103, 104, 105, 110, 111

[51] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 15, 107

[52] C. Li, C. Brandli, R. Berner, H. Liu, M. Yang, S.-C. Liu, and T. Delbruck. Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor. *International Image Sensor Workshop (IISW)*, 2015. 57

[53] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128×128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits (JSSC)*, 43(2):566–576, 2008. 4, 5, 7, 8, 48, 49, 58, 84, 89, 129, 137

[54] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 30, 32

[55] M. Mahowald and C. Mead. The Silicon Retina. *Scientific American*, 264(5):76–82, 1991. 46

[56] R. H. Masland. The fundamental plan of the retina. *Nature neuroscience*, 4(9):877, 2001. 45, 47

[57] S. Meister, J. Hur, and S. Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. *arXiv preprint arXiv:1711.07837*, 2017. 157

[58] Microsoft Corp. Microsoft Kinect. https://www.xbox.com/en-US/xbox-one/accessories/kinect, 2010. 14

[59] Microsoft Corp. Microsoft HoloLens. https://www.microsoft.com/en-us/hololens, 2016. 5

[60] E. Mueggler, B. Huber, and D. Scaramuzza. Event-based , 6-DOF Pose Tracking for High-Speed Maneuvers. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2014. 11

[61] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *arXiv preprint arXiv:1610.08336*, 2016. 15, 143, 148, 159, 160, 162, 164, 165, 167

[62] R. A. Newcombe and A. J. Davison. Live Dense Reconstruction with a Single Moving Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 62

[63] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 33, 75, 82

[64] A. Nguyen, T. Do, D. G. Caldwell, and N. G. Tsagarakis. Real-Time Pose Estimation for Event Cameras with Stacked Spatial LSTM Networks. 2017. 15, 125

[65] M. Nikolova. A variational approach to remove outliers and impulse noise. *Journal of Mathematical Imaging and Vision*, 20(1-2):99–120, 2004. 22

[66] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *arXiv preprint arXiv:1505.04366*, 2015. 111

[67] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7, 2013. 15

[68] Oculus VR, LLC. Oculus Rift. https://www.oculus.com/rift, 2016. 5

[69] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997. 23

[70] M. Osswald, S.-H. Ieng, R. Benosman, and G. Indiveri. A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems. *Scientific reports*, 7:40703, 2017. 15

[71] M. Otte and H.-H. Nagel. Optical flow estimation: advances and comparisons. 1994. 162

[72] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2013. 65

[73] T. Pock and A. Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 73, 87, 88

[74] T. Pock, A. Chambolle, D. Cremers, and H. Bischof. A convex relaxation approach for computing minimal partitions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 27, 28, 60, 61, 82

[75] 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 (JSSC)*, 2011. 8, 58

[76] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck. Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014. 8, 45, 47, 50, 54

[77] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016. 32, 113

[78] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2016. 13

[79] C. Reinbacher, G. Graber, and T. Pock. Real-Time Intensity-Image Reconstruction for Event Cameras using Manifold Regularisation. *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 124, 129, 137, 138

[80] E. Reinhard, P. Shirley, and T. Troscianko. Natural image statistics for computer graphics. *Univ. Utah Tech Report UUCS-01-002 (Mar. 2001)*, 2001. 82

[81] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 27, 28

[82] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. 42

[83] R. W. Rodieck. The primate retina. *Comput. Primate Biol.*, 4:203–278, 1998. 45

[84] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck. Asynchronous event-based binocular stereo matching. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):347–353, 2012. 14

[85] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015. 136

[86] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 23

[87] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992. 27, 82

[88] T. Serrano-Gotarredona and B. Linares-Barranco. A 128×128 1.5% Contrast Sensitivity 0.9% FPN 3 μs Latency 4mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers. *IEEE Journal of Solid-State Circuits (JSSC)*, 48(3):827–838, 2013. 57

[89] P. A. Shedligeri, K. Shah, D. Kumar, and K. Mitra. Photorealistic image reconstruction from hybrid intensity and event based sensor. *arXiv preprint arXiv:1805.06140*, 2018. 154

[90] A. A. Stocker. An improved 2D optical flow sensor flor motion segmentation. *IEEE International Symposium on Circuits and Systems*, 2002. 48

[91] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, New York, NY, USA, 2010. 145

[92] A. N. Tikhonov. On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39(5):195–198, 1943. 17

[93] J. Tumblin, A. Agrawal, and R. Raskar. Why I want a Gradient Camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 48

[94] H.-Y. F. Tung, A. W. Harley, W. Seto, and K. Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. 120, 121

[95] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *arXiv preprint arXiv:1711.10925*, 2017. 107

[96] M. Unger, T. Pock, M. Werlberger, and H. Bischof. A Convex Approach for Variational Super-Resolution. In *Proceedings of the DAGM Symposium on Pattern Recognition*, 2010. 89

[97] Valve Corp. HTC Vive. https://www.vive.com, 2016. 5

[98] H. Wässle. Parallel processing in the mammalian retina. *Nature Reviews Neuroscience*, 5(10):747, 2004. 47

[99] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An Improved Algorithm for TV-L1 Optical Flow. In *Proceedings of the Dagstuhl Seminar on Statistical and Geometrical Approaches to Visual Motion Analysis*, 2009. 82

[100] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt. Event-based 3D SLAM with a depth-augmented dynamic vision sensor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 14

[101] Z. Yin and J. Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 156

[102] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Proceedings of the DAGM Symposium on Pattern Recognition*, 2007. 80, 81, 82

[103] K. A. Zaghloul and K. Boahen. A silicon retina that reproduces signals in the optic nerve. *Journal of Neural Engineering*, 2006. 46

[104] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. 2014. 109

[105] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. 117, 132

[106] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 158

[107] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv 1802.06898v2*, 2018. 125, 154