Imperial College London
Department of Computing

Real-Time Visual SLAM with an Event Camera

Hanme Kim

September 2017

Supervised by Prof. Andrew J. Davison

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

Abstract

Simultaneous localisation and mapping (SLAM) based on computer vision has remarkably matured over the past few years, and is now rapidly transitioning into practical applications such as autonomous vehicles, drones, *augmented reality* (AR) / *virtual reality* (VR) devices, and service robots to name a few. These real-time, real-world SLAM applications require instantaneous reaction to dynamic, real-world environments, the ability to operate in scenes which contain extreme lighting variation, and high power efficiency. The standard video cameras on which they rely, however, run into problems when trying to supply these, due to either huge bandwidth requirements and power consumption at high frame-rates, or diminishing image quality with blur, noise or saturation.

The core work of this research concerns these constraints imposed by standard cameras, and was motivated by silicon retinas in neuromorphic engineering mimicking some of the superior properties of human vision. One such bio-inspired imaging sensor called an *event camera* offers a breakthrough new paradigm for real-time vision, with its high measurement rate, low latency, high dynamic range, and low data rate properties. The event camera outputs not a sequence of video frames like a standard camera, but a stream of asynchronous events at microsecond resolution, indicating when individual pixels record log intensity changes of a pre-set threshold size. But it has proven very challenging to use this novel sensor in most computer vision problems, because it is not possible to apply standard computer vision techniques, which require synchronous intensity information, to its fundamentally different visual measurements.

In this thesis, we show for the first time that an event stream, with no additional sensing, can be used to track accurate camera rotation while building a persistent and high quality mosaic of an unstructured scene which is super-resolution accurate and has high dynamic range. We also present the first algorithm provably able to track a general 6D motion along with reconstruction of arbitrary structure including its intensity and the reconstruction of grayscale video that exclusively relies on event camera data. All of the methods operate on an event-by-event basis in real-time and are based on probabilistic filtering to maximise update rates with low latency. Through experimental results, we show that extremely rapid motion tracking and high dynamic range scene reconstruction without motion blur can be achievable by harnessing the superior properties of the event camera, and the potential for consuming far less power based on the low data rate property. We hope that this work opens up the door to practical solutions to the current limitations of real-world visual SLAM applications.

I dedicate this thesis to my parents, my wife and my sons.

Acknowledgements

About 10 years ago in October 2007 in South Korea, I made a plan to move to the United Kingdom to work at first and then do my PhD once I secure a scholarship somehow. It was nothing but a dream at that time, but while I was writing this thesis, I realised how lucky I was to make the dream come true. Surely, it would not have been possible without the help and encouragement of a number of people around me during my time at Imperial College London and in the UK in general.

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisor Prof. Andrew Davison for his enduring support, guidance and inspiration. He is truly the best supervisor any student could ever ask for, and I know I will miss working with him terribly. I am also deeply grateful to Dr. Stefan Leutenegger for his generous offering of ideas and expertise, and to my examiners, Dr. Aldo Faisal and Dr. Margarita Chli, for their valuable comments during the viva voce examination.

I would also like to thank Dr. Ryad Benosman who introduced us to event cameras, and all the past and current members and visitors of the Robot Vision Group and the Dyson Robotics Laboratory for their support and friendship throughout all ups and downs of my PhD. My thanks also go to all my friends and colleagues at Imperial College London who made my study more enjoyable. I also truly appreciate that I am very fortunate for the EPSRC DTA scholarship, the Qualcomm Innovation Fellowship 2014, the Best Industry Paper award at the British Machine Vision Conference (BMVC) 2014 and the Best Paper Award at the European Conference on Computer Vision (ECCV) 2016 I received for my PhD study.

Last but not least, I would like to thank my dad Kwanhong Kim, my mum Meja Chuen, my wife Soyeon Kim, my sons Eeru and Haru, and all my family in South Korea and USA whose love, support and patience carried me through this journey. I will always keep going for them, they are the reason I take each breath. Love you all.

Contents

Chapter 1

Introduction

Contents

Simultaneous localisation and mapping (SLAM) based on computer vision has remarkably matured over the past few years, and this technology is now rapidly transitioning into a range of real products in robotics such as autonomous vehicles, drones and service robots, wearable devices for *augmented reality* (AR)/*virtual reality* (VR) and smartphones to name a few. These real-time, real-world applications require instantaneous reaction to dynamic in real-world environments, the ability to operate in scenes which contain extreme lighting variation, and high power efficiency especially when operating on battery-powered platforms.

However, the standard vision cameras on which they heavily rely run into problems when trying to supply these, either of huge bandwidth requirements and power consumption at high frame-rates, or diminishing image quality with blur, noise or saturation. Although a number of sophisticated visual SLAM methods have been recently developed which are more efficient in terms of computation and resource usage, and more resilient to difficult scene conditions, their heavy reliance on conventional imaging sensors still prevents them from becoming mass market products.

The core work of this research concerns these constraints imposed by standard cameras, and was motivated by silicon retinas in neuromorphic engineering mimicking some of the superior properties of human vision. One such bio-inspired imaging sensor called an *event camera* offers a breakthrough new paradigm for real-time vision, with its high measurement rate, low latency, high dynamic range, and low data rate properties. The event camera outputs not a sequence of video frames like a standard camera, but a stream of asynchronous events at microsecond resolution, indicating when individual pixels record log intensity changes of a pre-set threshold size. But it has proven very challenging to use this novel sensor in most computer vision problems, because it is not possible to apply standard computer vision techniques, which require synchronous intensity information, to its fundamentally different visual measurements.

In this thesis, we show for the first time that an event stream, with no additional sensing, can be used to track accurate camera rotation while building a persistent and high quality mosaic of an unstructured scene which is super-resolution accurate and has high dynamic range. We also present the first algorithm provably able to track a general 6D motion along with reconstruction of arbitrary structure including its intensity and the reconstruction of grayscale video that exclusively relies on event camera data. All of the methods operate on an event-by-event basis in real-time and are based on probabilistic filtering to maximise update rates with low latency. Through experimental results, we show that extremely rapid motion tracking and high dynamic range scene reconstruction without motion blur can be achievable by harnessing the superior properties of the event camera, and the potential for consuming far less power based on the low data rate property. We hope that this work opens up the door to practical solutions to the current limitations of real-world visual SLAM applications.

1.1 Visual SLAM

In this section, we will give a brief introduction to the history of visual SLAM and the current state-of-the-art methods before we discuss the remaining challenges which still restrict their applicability in many real-world situations in Section 1.2.

We face the SLAM problem whenever we have a moving robot or any other device with various on-board sensors which is dropped into an unknown space where it must localise, navigate and understand in order to achieve a given task. Such a SLAM system takes input in the form of a stream of data from one or more sensors, and estimates its continuously changing position and orientation (*pose*) while building and expanding a model of the environment surrounding the device. SLAM research originated in the robotics community where achieving real-time processing is essential to be useful, and was mainly about 2D mapping. Most early SLAM systems therefore relied on sensors which could directly measure both bearing angle and depth such as sonar sensors or laser range finders as well as wheel odometry information based on strong assumptions about planar robot motion [91, 129].

In visual SLAM, the main sensors are some type of vision sensors such as *charge-coupled device* (CCD) or *complementary metal-oxide-semiconductor* (CMOS) cameras, depth cameras, or event cameras in our case. Visual SLAM has received a great deal of attention from the computer vision and robotics communities over the last 20 years, mainly because these vision sensors have become relatively cheap and small size, and ubiquitously found in portable devices such as mobile phones and tablets. Cameras are also the natural sensor of choice for high level scene understanding capability which can infer meaningful semantic information such as the identities of objects.

While *structure from motion* (SfM) estimates the positions of a set of cameras and the structure of the scene given a collection of images focussing on off-line applications such as movie post production [60, 133], visual SLAM *incrementally* estimates cameras' motion and the scene structure as images are received in real-time.

Initially, localisation and mapping were tackled separately due to the closely correlated nature of both problems — an accurate map representation is required to perform localisation, and precise localisation information is needed to create a consistent map. The theoretical foundation of SLAM was established by the pioneering work of Smith *et al.* [153] and Moutarlier and Chatila [121] using a sequential probabilistic filter whose job is to maintain joint probabilistic estimates for both the shape of a space around a device and the device's own motion through that space which are refined over time as new sensor measurements arrive. They proposed the use of the *extended Kalman filter* (EKF) [79], and this became the mainstream framework of many early successful SLAM systems [92, 14, 25, 129].

One of the first full joint EKF-based real-time SLAM system using computer vision was developed by Davison *et al.* [40] which utilised an active stereo head on a wheeled robot to fixate and track selective sparse landmark features, and navigated in previously unknown environments over extended periods of time as shown in Figure 1.1. Its hardware complexity (i.e. the active stereo camera and wheel odometry) and hard constraint (i.e. the planar robot motion assumption) however limited its applicability in many real-world scenarios similar to

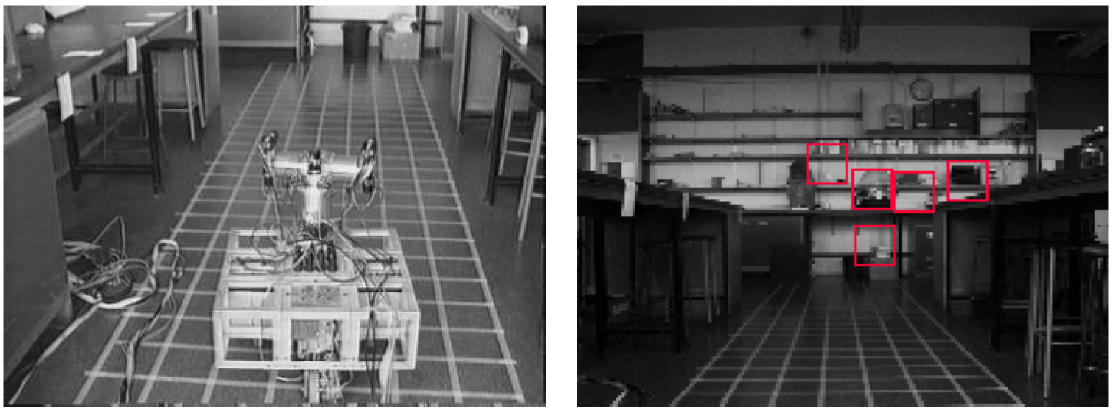


Figure 1.1: One of the first full joint EKF-based real-time visual SLAM system which navigates in unknown environments by tracking selective sparse landmark features with an active stereo camera on a wheeled robot, showing a highly constrained and hardware-dependent setup in the early period of the visual SLAM research. Figures courtesy of Davison *et al.* [40].

most early SLAM systems, and therefore more general visual SLAM algorithms which could operate in a much less constrained and hardware-dependent setup were highly demanded.

The most general and simplest setup in visual SLAM is using a single camera with no additional sensing to track 6 *degree-of-freedom* (DoF) motion and build a map of unknown and unstructured scenes. This configuration is known as *monocular SLAM*, which is much more challenging. Chiuso *et al.* [29, 30] tackled this monocular camera motion estimation and mapping problem based on a sequential probabilistic filtering approach, but the presented results were limited to tracking small camera motions with small groups of objects.

A breakthrough real-time monocular SLAM system called MonoSLAM was developed by Davison *et al.* [41, 42] — its implementation is available open-source[1,2]. As shown in Figure 1.2, MonoSLAM was able to track the position and orientation of a hand-held camera and estimate the 3D locations of a sparse set of feature points represented by small image patches using an EKF maintaining full covariance over the stacked state vector composed of the camera and feature position estimates. It was however able to handle only around 100 features in the state vector in order to maintain real-time operation because of the computational complexity of the EKF which is, in the worst case, proportional to the cube of the size of the state vector, severely limiting its scalability. In addition, repeatedly linearising non-

[1]SceneLib 1.0: `https://www.doc.ic.ac.uk/~ajd/Scene/index.html` (accessed September 2017)
[2]SceneLib 2.0: `https://github.com/hanmekim/SceneLib2` (accessed September 2017)

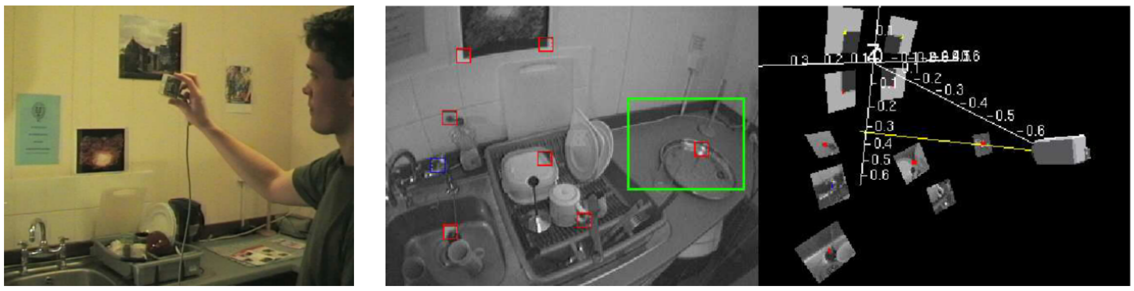


Figure 1.2: MonoSLAM was one of the first real-time monocular SLAM methods which tracked 6-DoF hand-held camera motion while building a sparse set of features based on a sequential probabilistic filter showing the most general and simplest setup in visual SLAM. Figures courtesy of Davison *et al.* [42].

linear sensor and motion models in the sequential probabilistic filtering framework propagates linearisation errors and eventually tends to cause inconsistencies over longer trajectories. To mitigate both problems and improve the scalability of EKF-based SLAM methods, *submapping* strategies were introduced which generally decompose a large map into small local maps of bounded complexity such as Bosse *et al.* [18] and Clemente *et al.* [35]. Also, Eade and Drummond [54] took a different approach to submapping to improve scalability by adapting the FastSLAM algorithm [118] for a monocular SLAM system which was able to handle hundreds of features in real-time.

A more general way to address the SLAM problem over probabilistic filtering formulations, especially for large scale maps, is the *graph-based* method which constructs a graph whose nodes represent poses or landmarks and edges correspond to constrains between connected nodes based on sensor measurements, and then finds the best configuration of the graph consistent with the constraints via graph optimisation such as the TORO [67] or g^2o [86][3] libraries. This graph-based formulation was first proposed by Lu and Milios [102] in 1997, but it took several years to gain popularity over EKF-based methods due to the high computational complexity of the error minimisation. The first monocular SLAM system based on such a graph-based formulation was proposed by Eade and Drummond [55]. The system operates in real-time with several hundred landmark features producing accurate estimation considerably better than probabilistic filtering-based methods and comparable to off-line bundle adjustment in SfM by performing global graph optimisation. Another interesting and closely related method to tackle large scale SLAM was RatSLAM developed by

[3] g^2o: https://github.com/RainerKuemmerle/g2o (accessed September 2017)

Milford and Wyeth [114] which is a lightweight vision system that provides rough trajectory of a ground-based platform (e.g. car) based on a very simple visual odometry algorithm and then produces a very large coherent map by closing many loops based on a simple visual place recognition method.

In 2007, a groundbreaking real-time visual SLAM system called *parallel tracking and mapping* (PTAM) was presented by Klein and Murray [84] which unarguably outperformed MonoSLAM and other existing methods in terms of accuracy and robustness — its implementation is also available open-source[4]. They proposed to split the visual SLAM problem into two separate tracking and mapping components processed in parallel on different threads of a multi-core computer, based on the assumption that the current estimate from one component is accurate enough to lock for the purposes of estimating the other. While its tracking operates in real-time on every single frame against a 3D scene model from the mapping part consisting of thousands of feature points, its mapping component produces the 3D map from carefully selected previous frames (*keyframes*) based on a computationally expensive batch optimisation technique but running at lower frequency than frame rate. As shown in Figure 1.3, it was able to enable more realistic AR experiences by recovering more detailed 3D geometric information of a scene, but limited to small AR workspaces bounded by the maximum number of keyframes the bundle adjustment back-end can handle in real-time.

Since the impact PTAM has made on the visual SLAM research was significant, a thorough investigation was conducted into the question of which one is more beneficial to maximise accuracy and robustness of a real-time visual SLAM system between increasing the number of frames and increasing the number of features given available processing resources. And, it has proven that keyframe-based batch optimisation is the method of choice for real-time visual SLAM over iterative probabilistic filtering [157, 160]. Later, Strasdat *et al.* [159] proposed a two-level optimisation approach to improve scalability to much larger workspaces which operates like PTAM in the primary region of interest but has a second coarse level which is similar to pose-graph optimisation. Most recently, Mur-Artal *et al.* [125][5,6] extended this type of approach much further by redesigning a new monocular SLAM system called ORB-SLAM incorporating other well-developed components for feature detection [142], loop detection [65], a loop closing and optimisation strategy [158, 159], and optimisation framework [86], and demonstrated its high accuracy and reliability in many

[4]PTAM: `http://www.robots.ox.ac.uk/~gk/PTAM/` (accessed September 2017)
[5]ORB-SLAM: `https://github.com/raulmur/ORB_SLAM` (accessed September 2017)
[6]ORB-SLAM2: `https://github.com/raulmur/ORB_SLAM2` (accessed September 2017)

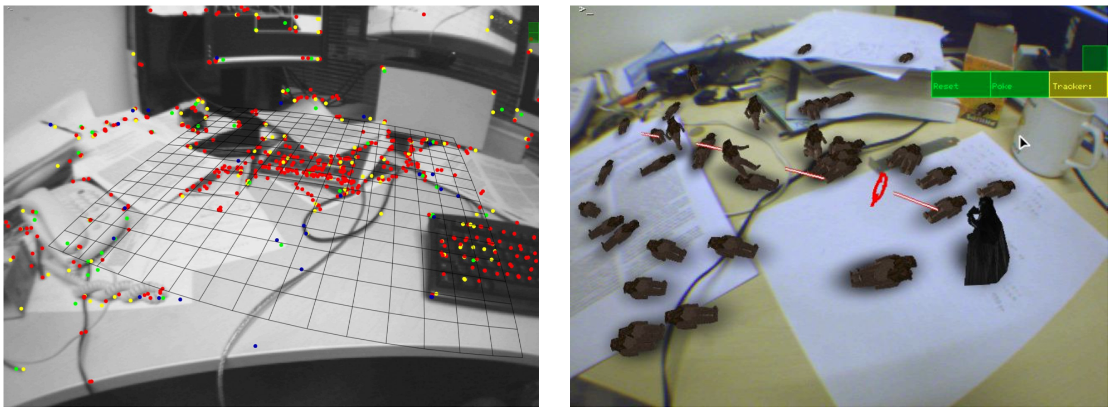
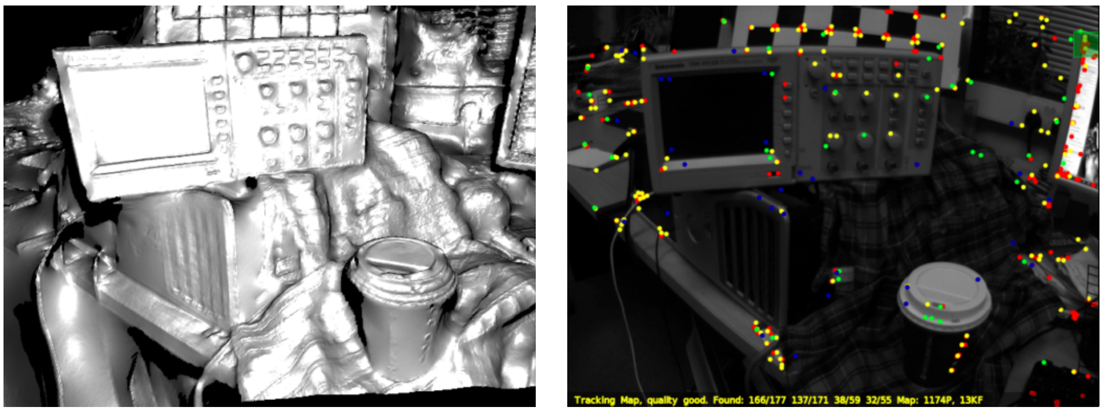


Figure 1.3: PTAM separates tracking and mapping into two parallel tasks running on different threads of a multi-core computer. The tracking component operates on every single frame while the mapping part performs at lower frequency to use a computationally expensive batch optimisation method. The system runs in real-time with thousands of feature points enabling realistic AR experiences. Figures courtesy of Klein and Murray [84].

different scenarios.

Although the increased number of feature points that visual SLAM systems can handle in real-time enabled more robust camera pose tracking and more realistic AR experiences, they are still not sufficient for many robotic and AR applications. For instance, a robot needs to know the geometry of blank walls to avoid collisions or the complete surface model of objects in the world is required in order for AR virtual characters to interact with them. Therefore, great research attention turned from feature-based to *direct* or *dense* visual SLAM which makes use of all of the pixels in an image, not only extracted features such as the work of Comport *et al.* [37].

Initially, researchers tried to tackle reconstructing dense surface models in real-time while tracking still relied on a feature-based method such as Newcombe and Davison [126] and Stuehmer *et al.* [161]. The first fully dense tracking and mapping method working with a hand-held camera, called *dense tracking and mapping* (DTAM), was developed by Newcombe *et al.* [128]. The system was able to produce a dense surface model with millions of vertices by fusing detailed textured depth maps estimated at selected keyframe positions while robustly tracking the 6-DoF camera motion by whole image alignment against the dense model at frame rate. They achieved real-time performance of their computationally intensive but highly parallelisable algorithms using a *graphics processing unit* (GPU) enabled computer, and demonstrated the usefulness of the dense representation for robust tracking



```
Tracking Map, quality good. Found: 166/177 137/171 38/59 32/55 Map: 1174P, 13KF
```

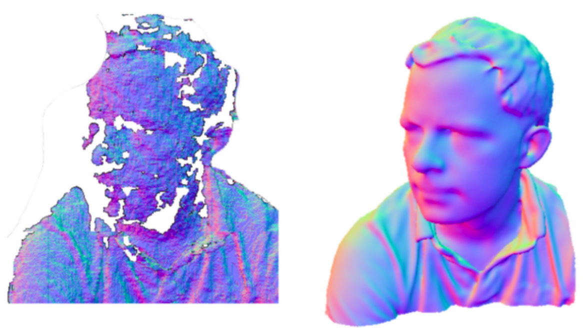
Figure 1.4: DTAM demonstrated the usefulness of its dense surface model (left) for robust tracking and real-time scene interaction in AR applications over the sparse feature-based representation (right). Figures courtesy of Newcombe *et al.* [128].

even under rapid motion and physics-enhanced AR applications over the sparse feature-based representation as shown in Figure 1.4.

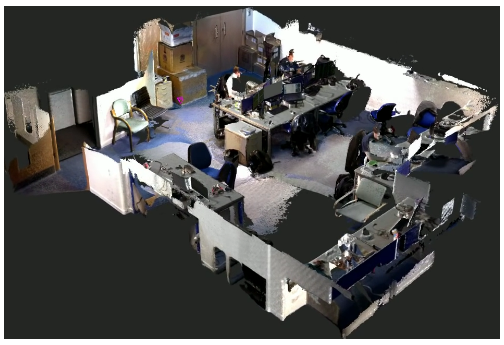
In addition to the parallel processing power of GPU processors which massively contributed to realising real-time dense monocular SLAM methods, the arrival of commodity depth sensors like Microsoft Kinect in 2010 made possible high quality dense surface reconstruction and beyond using depth sensor readings measured in hardware, at no computational cost. One of the most influential pieces of work in RGB-D camera-based dense visual SLAM, called KinectFusion, was introduced by Newcombe *et al.* [127]. While a low-cost depth sensor browses a complex and arbitrary indoor scene, tracked using *iterative closest point* (ICP) alignment between the predicted and actual depth measurement, the system was able to produce a highly accurate dense map of the scene in real-time by fusing all of the incomplete, noisy depth data from the sensor into a volumetric, *truncated signed distance function* (TSDF) representation as shown in Figure 1.5 (a).

However, KinectFusion came with a few limitations such as its restricted scalability to a fixed small workspace, mainly caused by the predefined TSDF voxel model. Also there were potential tracking failures in scenes with poor 3D structure because of its sole reliance on geometric information. Newer related developments such as Kintinuous [171, 174, 172][7] and ElasticFusion [173][8] tackled these limitations by introducing more scalable volumetric

[7]Kintinuous: https://github.com/mp3guy/Kintinuous (accessed September 2017)
[8]ElasticFusion: https://github.com/mp3guy/ElasticFusion (accessed September 2017)



(a)



(b)

Figure 1.5: (a) KinectFusion produces high quality dense surface reconstruction (right) of complex and arbitrary objects or indoor scenes in real-time by fusing all of the incomplete, noisy depth data (left) from a Kinect sensor. Figure courtesy of Newcombe *et al.* [127]; (b) high quality globally consistent large scale reconstruction of ElasticFusion. Figure courtesy of Whelan *et al.* [173].

representations (e.g. sliding volume), robust tracking (e.g. based on both geometric and photometric constraints), and loop closure optimisations, and they managed to produce high quality globally consistent large scale reconstructions such as the result from ElasticFusion shown in Figure 1.5 (b).

The real-time dense monocular SLAM methods either using a standard camera or RGB-D sensor described so far have become very practical technologies and have great potential for serious commercial products. However, their high computational requirements which can be processed in real-time only with powerful hardware (e.g. GPUs) mean that use on mass market, low-cost, low-power platforms is difficult. Therefore recently, a number of sophisticated visual SLAM methods which try to benefit from both the efficiency of feature-based approaches and the accuracy and robustness of dense methods have been introduced.

Engel *et al.* developed a *large-scale direct monocular SLAM* (LSD-SLAM) system [57][9] built on top of their previous *semi-dense* visual odometry method [56] which can build large-scale, globally consistent maps as shown in Figure 1.6 along with highly accurate camera pose estimation based on direct (i.e. featureless) image alignment. The key idea is to continuously estimate pixel-wise depth values in a probabilistic filtering framework but only for the pixels with non-negligible intensity gradients (i.e. semi-dense). They managed to run this method in real-time on a CPU. A similar idea but using more abstract information (*edges*) was pro-

[9]LSD-SLAM: `https://github.com/tum-vision/lsd_slam` (accessed September 2017)

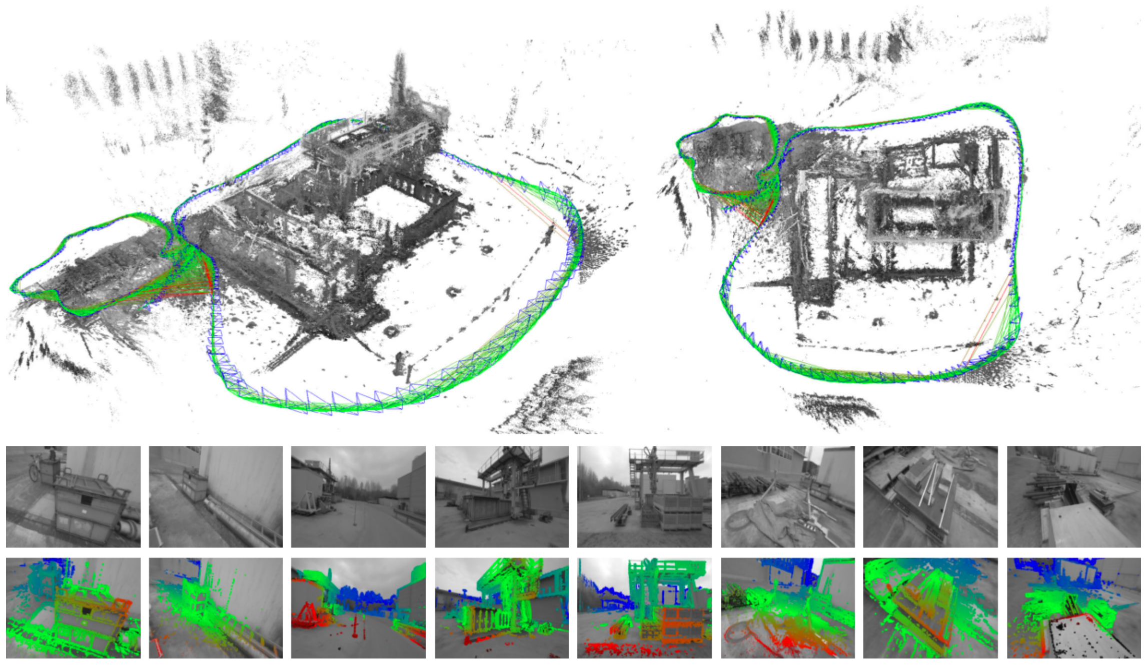


Figure 1.6: LSD-SLAM can build a large-scale, globally consistent map (shown as 3D point clouds on the top) consisting of keyframes with associated semi-dense depth maps (shown in the bottom two rows). This method can run in real-time on a CPU. Figure courtesy of Engel *et al.* [57].

posed by Tarrio and Pedre called *real-time edge based visual odometry* (REBVO) [164][10] which can run in real-time on embedded platforms with limited processing resources. Once edge information is extracted in a highly parallelisable manner, the system tracks 6-DoF camera pose by fitting the current edges into the previous ones in an energy minimisation form, and estimates the depth value per edge pixel using a pixel-wise EKF based on edge point correspondences between the latest two frames. Lastly, Forster *et al.* proposed a hybrid approach called *semi-direct visual odometry* (SVO) [61][11] which explicitly combines the feature-based approaches and direct methods. SVO produces a sparse map consists of feature points, but performs feature extraction and matching only when a keyframe needs to be created while estimating the camera pose directly based on pixel values at every frame. By reducing the number of feature correspondence steps required, they managed to increase processing speed, and at the same time increase the accuracy and robustness of the method through sub-pixel alignments.

[10]REBVO: `https://github.com/JuanTarrio/rebvo` (accessed September 2017)
[11]SVO: `https://github.com/uzh-rpg/rpg_svo` (accessed September 2017)

One very interesting approach enabled by high quality dense surface reconstruction is to use high-level scene understanding *in the loop* to improve real-time visual SLAM systems with robust camera localisation, more efficient and compressed scene representation, and perception capabilities. For instance, SLAM++ [145] used online 3D object recognition of domain-specific objects such as chairs or tables, and Salas-Moreno *et al.* [146] took advantage of prior knowledge that many man-made environments consist of repeated planar structures in their RGB-D camera-based SLAM system. We believe that visual SLAM will continue to evolve towards *semantic SLAM* integrating real-time geometry and semantic understanding of scenes, and this is a current research frontier such as the work of McCormac *et al.* [110].

In this section, we gave a brief introduction to the history of visual SLAM and current state-of-the-art methods, but for a deeper understanding of the SLAM research, we refer to the relevant literature such as textbooks [74, 165, 33], through historical reviews [53, 6, 5, 50, 147, 62], more recent survey papers [176, 77, 21], graph-based SLAM [68], visual place recognition [101], and multiple robot SLAM [144].

1.2 Limitations

As introduced in the previous section, real-time visual SLAM has progressed rapidly and continuously in the past 20 years. While early visual SLAM methods made rather sparse maps of the world in the form of point clouds, recent approaches show that it is now evolving into a general scene understanding capability which can reconstruct detailed surface geometry and meaningful semantic information such as the identities of objects. And we are now witnessing a rapid transition of this technology into a range of real products such as autonomous vehicles, drones, smartphones, AR/VR and services robots (see Figure 1.7). Most of these use additional complementary sensors like an IMU. However, despite a number of sophisticated visual SLAM methods recently developed which are more efficient and robust, we believe that such mass market products will not be possible until further breakthrough improvements in several challenges to current visual SLAM technology are achieved: especially the response to rapid motion with low latency, handling of extreme lighting variation, and high power consumption.

Standard frame-rates cannot cope with rapid motion.

Many real-world, real-time vision applications require much faster control feedback than standard frame rates (i.e. 25-60Hz) to cope with dynamics in the world. Normal frame rates run into problems when trying to supply this, either of vast motion displacement between

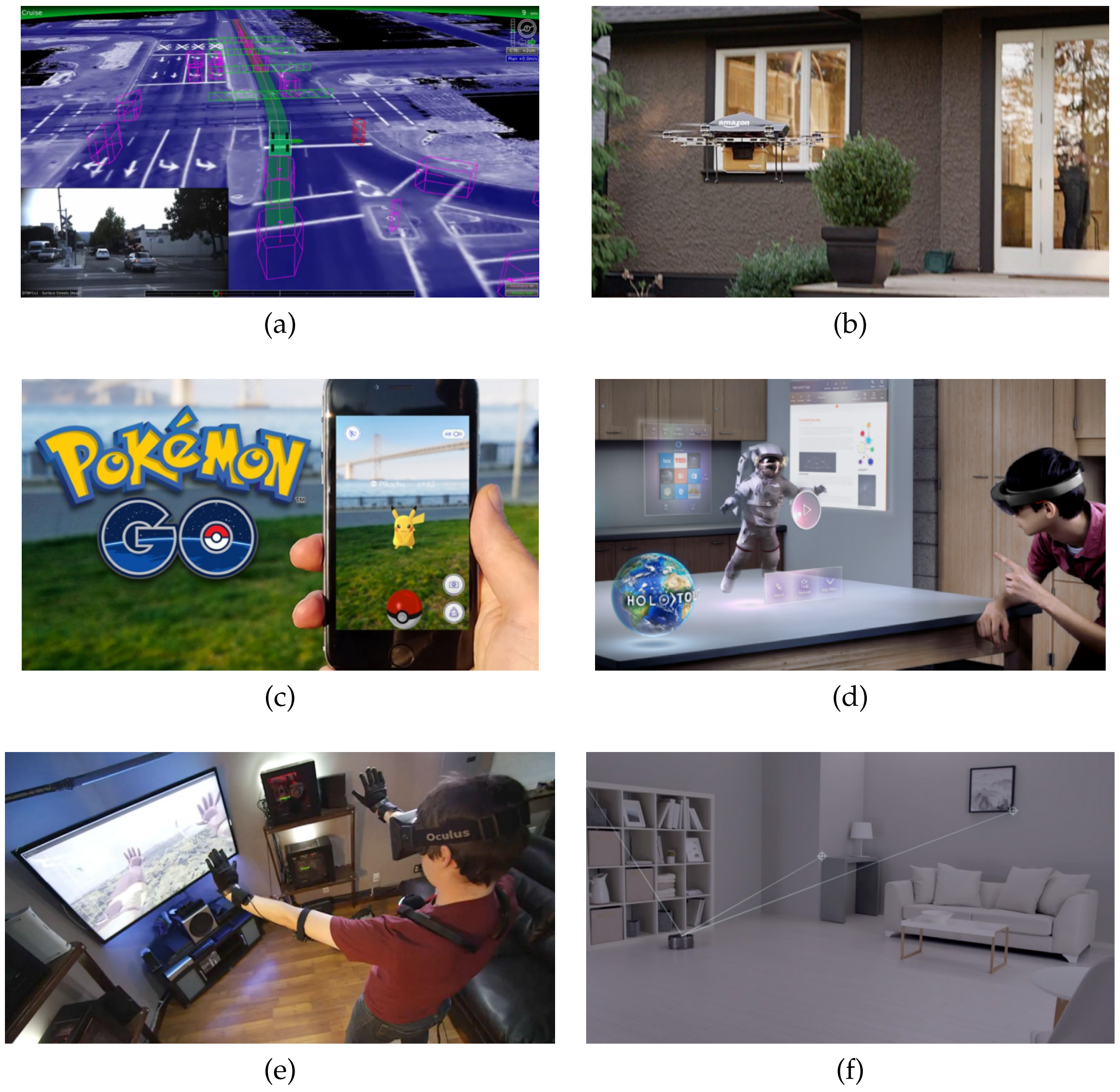


Figure 1.7: Visual SLAM is a key enabler for a number of applications such as (a) autonomous vehicles, (b) drones, (c) smartphones, (d) augmented reality, (e) virtual reality, and (f) service robots. Figures courtesy of Google, Amazon, Niantic, Microsoft, Facebook and Dyson.

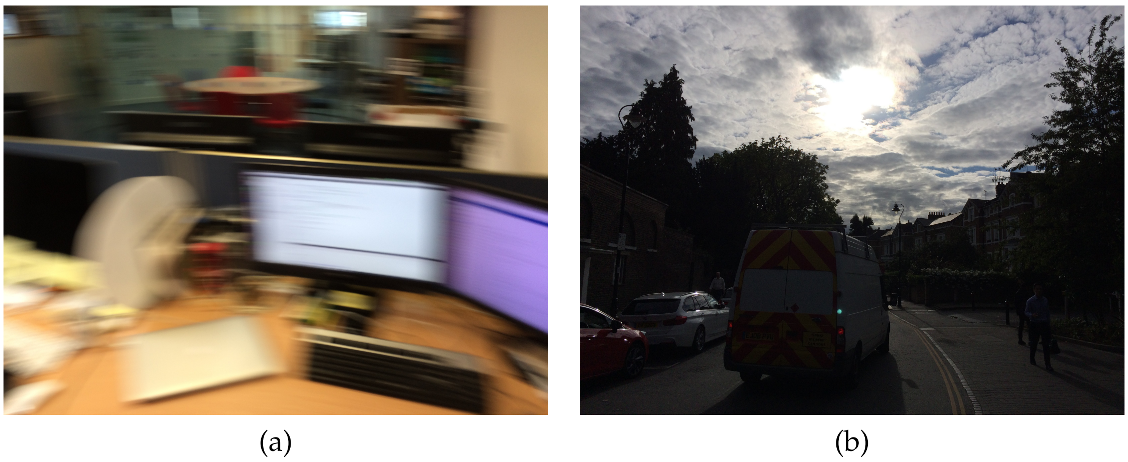


Figure 1.8: Computer vision methods relying on standard cameras suffer from: (a) severe motion blur with rapid camera motion washing out all the detailed texture in the scene; (b) low dynamic range under extreme lighting variation, causing low contrast in the areas around bright or dark regions.

frames or diminishing image quality with motion blur (e.g. Figure 1.8 (a)), which degrade the performance of any kind of computer vision algorithm significantly.

Alternatively, higher frame-rates give us the opportunity to track rapid motion by providing smaller motion displacement between frames. However, without a dedicated special hardware architecture such as that of [78], the amount of data to be transmitted, processed and saved would be too great, and the nature of high shutter timing increases noise levels and makes output too dark. Handa *et al.* [70] showed that the overall computational cost of high frame-rate vision-based tracking applications is mitigated by reduced inter-frame processing complexity which takes the advantage of motion prediction, but frame-rate cannot be arbitrarily increased because of the limitation of available computational budget and image degradation with noise.

Standard cameras suffer from low dynamic range.

A standard CCD/CMOS camera generates video by regularly and synchronously opening its shutter to expose all pixels (or a line of pixels in the rolling shutter case) and capture frames, and has relatively low dynamic range around 60dB. It is therefore unable to sense over a high dynamic range in scenes which contain large intensity differences as shown in Figure 1.8 (b) — we lose visibility in an area around the bright sun and in the dark regions. Of course, this can be improved by sophisticated post-processing algorithms based on multiple images with different shutter speeds in hardware or software, but there is still a significant gap between the enhanced results and natural high dynamic scenes. Like motion blurred images, degraded low dynamic range frames make most computer vision algorithms difficult to perform correctly.

Powerful hardware requirements and high power consumption.

The accurate, robust tracking and high quality reconstruction capabilities of current visual SLAM systems in general require dedicated powerful hardware such as multi-core CPUs and GPUs, and their high computational requirements consume a lot of power which also leads to significant heat. LiKamWa *et al.* [97] in their case study of the power and thermal characteristics of Google Glass[12] showed that even a simple OpenCV face detection application brings the device to a full 100% CPU utilisation and shortens battery lifetime to only 38 minutes. These are serious limiting factors for such systems to be deployed in mass market, low-cost, always-on, low-power platforms such as the ones shown in Figure 1.7 to reach their full potential.

1.3 Contributions

We strongly believe that next generation image sensor and processor technologies have great potential to solve all of the limitations of real-time visual SLAM imposed by the conventional hardware architecture and design principles of the standard cameras and processors when integrated with a new class of computer vision algorithms, and this strong belief has inspired the direction of this work.

In this thesis, we present the first high performance real-time visual SLAM algorithms based on a single event camera, a paradigm shift in visual sensing. Event sensors generate low bit-rate, information-rich data streams which are free of the redundancy of video while providing high dynamic range and high time resolution by reporting asynchronous intensity changes rather than full frames. Event sensors have no shutter or global exposure settings, which leads to a very high dynamic range (e.g. 130dB), and report events with almost continuous micro second timestamps, allowing detailed observation of even spinning fan blades.

In Chapter 4, we show for the first time that an event stream, with no additional sensing, can be used to track accurate camera rotation while building a persistent and high qual-

[12]Google Glass: `https://en.wikipedia.org/wiki/Google_Glass` (accessed September 2017)

ity mosaic of a scene which is super-resolution accurate and has high dynamic range. The method involves parallel camera rotation tracking and template reconstruction from estimated gradients, both operating on an event-by-event basis and based on probabilistic filtering to maximise update rates with low latency. We then present its real-time implementation whose overall structure and functionalities are the same, but a substantial speed up has been achieved by adapting a computationally efficient estimation method for tracking as well as a parallelisable log intensity reconstruction running on a GPU in Chapter 5. The experimental results also show that this speed-up increases the quality of estimation and reconstruction as it guarantees a higher fidelity of the independence assumption.

Then, to move towards 3D visual SLAM, we propose a method which can perform real-time 3D reconstruction from a single hand-held event camera with no additional sensing, and works in unstructured scenes of which it has no prior knowledge in Chapter 6. It is based on three decoupled probabilistic filters, each estimating 6-DoF camera motion, scene logarithmic (log) intensity gradient and scene inverse depth relative to a virtual keyframe, and we build a real-time graph of these to track and model over an extended local work-space. We also upgrade the gradient estimate for each keyframe into an intensity image, allowing us to recover a real-time video-like intensity sequence with spatial and temporal super-resolution from the low bit-rate input event stream. To the best of our knowledge, this is the first algorithm provably able to track a general 6D motion along with reconstruction of arbitrary structure including its intensity and the reconstruction of grayscale video that exclusively relies on event camera data.

All of the methods described in this thesis are able to harness the superior properties of the event camera such as high speed measurement, low latency, high dynamic range, and low data rate, and we hope our work opens up the door to practical solutions to the current limitations of real-world SLAM applications.

1.4 Publications

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

Simultaneous Mosaicing and Tracking with an Event Camera [82]
Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoï Ieng and Andrew J. Davison
Proceedings of the British Machine Vision Conference (BMVC), 2014
(**Best Industry Paper**)

Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera [83]
Hanme Kim, Stefan Leutenegger and Andrew J. Davison
Proceedings of the European Conference on Computer Vision (ECCV), 2016
(**Best Paper Award**)

and other papers produced during the course of my PhD are as follows:

Place Recognition with Event-based Cameras and a Neural Implementation of SeqSLAM [113]
Michael Milford, Hanme Kim, Michael Mangan, Tom Stone, Stefan Leutenegger, Barbara Webb and Andrew Davison
The Innovative Sensing for Robotics: Focus on Neuromorphic Sensors workshop at the IEEE International Conference on Robotics and Automation (ICRA), 2015

Towards Visual SLAM with Event-based Cameras [112]
Michael Milford, Hanme Kim, Stefan Leutenegger and Andrew Davison
The Problem of Mobile Sensors: Setting future goals and indicators of progress for SLAM Workshop in conjunction with Robotics: Science and Systems (RSS), 2015

1.5 Thesis Structure

The structure of the rest of this thesis is as follows. In Chapter 2, we introduce the event camera which senses scene changes in the form of asynchronous events rather than synchronous frames, and some related work. Chapter 3 first presents the notational convention and some important mathematical foundations, and then gives a brief overview of some of the prerequisites required to work with an event camera and a number software libraries and APIs used in our implementations. After these introductory chapters, we move on to the core work of this thesis. In Chapter 4, we present the first method demonstrating high quality HDR and super-resolution intensity panorama reconstruction and motion estimation from a purely rotating event-based camera followed by its real-time version whose overall structure and functionalities are the same, but a substantial speed up is achieved by adapting a computationally efficient tracking method as well as a parallelisable reconstruction running on a GPU in Chapter 5. In Chapter 6, we present the first method demonstrating joint estimation of general 6-DoF camera motion, scene intensity and scene 3D depth from pure event data running on a standard PC in real-time. Finally, we conclude in Chapter 7 with a summary of the work and the contributions made throughout this thesis along with some thoughts on potential future work.

Chapter 2

Event Cameras

Contents

In this chapter, we introduce novel image sensors called event cameras which have strong potential, when integrated with new event-driven computer vision algorithms as proposed in this thesis, to solve most of the problems caused by the design principles of conventional cameras as described in Section 1.2. In contrast to standard frame-based cameras, event vision sensors capture scenes in a totally different way. They generate low bit-rate, information-rich data streams which are free of the redundancy of video while providing additional advantages such as high dynamic range, high temporal resolution and low latency by reporting asynchronous intensity changes over time or space rather than synchronous full image frames. They originate from the bio-inspired silicon retina research in neuromorphic engineering whose goal is to emulate some superior properties of biological vision [105, 104, 17].

2.1 Neuromorphic Silicon Retina

Despite AlphaGo's historic victory in the Go match between Sedol Lee, 18-time world champion, and AlphaGo, a computer program developed by Google DeepMind [150], in March 2016, biological creatures outperform their artificial counterparts still in many areas. Humans, for instance, are able to recognise heavily occluded objects with which state-of-the-art machine learning methods are still struggling. More importantly, if we take into account operational efficiency, there is no competition at all. In the historic Go match, AlphaGo consumed approximately 1MW of power to operate its cloud computer consisting of 1,920 CPUs and 280 GPUs[1] while its opponent used only about 20W [73, 20] to make his moves. Therefore, understanding how biology functions in such an efficient way will undoubtedly provide profound insights into new paradigms of sensing and processing. With this aim, researchers in neuromorphic engineering have been trying to copy neural architecture and function to create electronics systems with the same efficient style of sensing and computation.

Within this research branch, neuromorphic vision systems specifically aim to mimic the biological retina and subsequent vision processing. In biology, the vertebrate retina, which is a thin sheet of tissue lining the inner surface of the eye, converts raw light into electrical pulses (known as spikes) in proportion to the relative change in light intensity over time or space. The spike signals are transmitted to the brain along the optic nerve to be interpreted as visual images and to stimulate high level perception and reaction. Even though individual neurons in the retina react much slowly and less precisely than electronic devices (e.g. transistors), biological vision systems outperform any artificial counterparts, extracting all the essential features of visual scenes rapidly and reliably even under very dynamic lighting conditions, and are extremely efficient, consuming far less power.

Motivated by this deep understanding of how visual information is encoded by the bio-

[1]AlphaGo versus Lee Sedol: https://en.wikipedia.org/wiki/AlphaGo_versus_Lee_Sedol (accessed September 2017)

logical input sensor, transmitted to the brain, and processed to perform high level tasks rapidly, reliably, and much more efficiently, researchers in neuromorphic engineering have developed a new type of visual sensors. They are generally called as *neuromorphic silicon retinas*, and aiming at emulating some of the properties of the biological counterpart. In 1991, Mahowald and Mead [105] successfully reproduced the first three of the biological retina's five layers in silicon, and demonstrated the same output signals observed in real retinas in real-time. Since the pioneering work, a variety of neuromorphic vision devices have been developed such as Visio1 developed by Zaghloul and Boahen which reproduced all five layers of the retina in 2001 [178].

2.2 Event-Based Vision Sensors

Up to the early 2000s, neuromorphic vision research was mostly aimed at creating complete neuromorphic systems mimicking their biological counterparts as precisely as possible. Since then, many different types of silicon retina have been developed which can cooperate with conventional processors and be used in practical applications as alternative vision devices. Inspired by biological vision, they abstract information from scenes in various forms to reduce redundancy and latency and increase dynamic range. Through historical reviews of the event-based vision sensors research in general can be found in [48, 45, 134, 135, 43].

Broadly speaking, any imaging sensor could be classified as a bio-inspired event camera as long as the core aspect of its architecture is to emulate the use of biology's sparse and data-driven signalling. A number of different sensing modalities have been proposed [116], such as spatial difference or contrast sensors which reduce spatial redundancy based on intensity differences [81] or ratios [89] over space, and temporal difference or contrast sensors which reduce temporal redundancy based on absolute [28] or relative [96] intensity changes over time. More high-level abstraction sensors include gradient [166] and optical flow [156] cameras. In this thesis, however, we only focus on temporal contrast type event cameras as they are easily accessible via commercial products or research collaboration, and therefore when we say an event camera we mean a temporal contrast type sensor which responds to relative intensity changes over time.

2.2.1 Event Camera vs Standard Camera

In order to understand how event cameras work and appreciate how they could be beneficial for real-time computer vision applications, it is inspiring to look at the differences between event cameras and standard cameras as illustrated in Figure 2.1. We recommend readers to view our animation video[2] which illustrates the comparison between event cameras and standard cameras in a form better than a still figure (also see Appendix A).

Standard cameras record scenes at fixed time intervals (i.e. global or rolling shutter) and output a sequence of image frames. For instance, as shown in Figure 2.1, if a fixed standard camera looks at the spinning disc with a black dot shown on the left, we get a sequence of snapshots as illustrated in the upper spatial-temporal graph on the right. The graph visualises some of the main properties of the standard video frames: there are blind time intervals between frames, the sensor keeps sending redundant data even when the disc stays still (no new information produced), and we suffer from motion blur (illustrated by the grey tails along the trajectory of the block dot) if the disc spins too fast.

In contrast, event cameras output not a sequence of video frames like a standard camera, but a stream of asynchronous *events* (also called *spikes*), each with a pixel location, polarity and microsecond-precise timestamp, indicating when individual pixels record log intensity changes of a pre-set threshold size. Positive and negative changes produce positive and negative events respectively. By encoding only image changes, the bandwidth needed to transmit, process and store a stream of events is much lower than that for standard video, removing the redundancy in continually repeated image values; but this stream should in principle contain all of the information of standard video, at least up to scale, and without the usual bounds on frame-rate and dynamic range. For instance, if we observe the same spinning disc with a fixed event camera, we get the stream of events illustrated in the lower spatial-temporal graph on the right of Figure 2.1 — red and blue dots represent positive and negative events respectively. This graph also visualises some of the main properties of the event stream; in particular the almost continuous response to very rapid motion and the way that the output data-rate depends on scene motion, though in practice is almost always dramatically lower than that of standard video. These properties offer the potential to overcome the limitations of real-world computer vision applications, relying on conventional imaging sensors, such as low frame rate, high latency, low dynamic range, and high power consumption.

We now introduce some of the event cameras available, including the one we used in this thesis. They are still research prototypes, but most of them are accessible via commercial products or research collaboration.

[2]Event Camera vs Standard Camera: https://youtu.be/kPCZESVfHoQ (accessed September 2017)

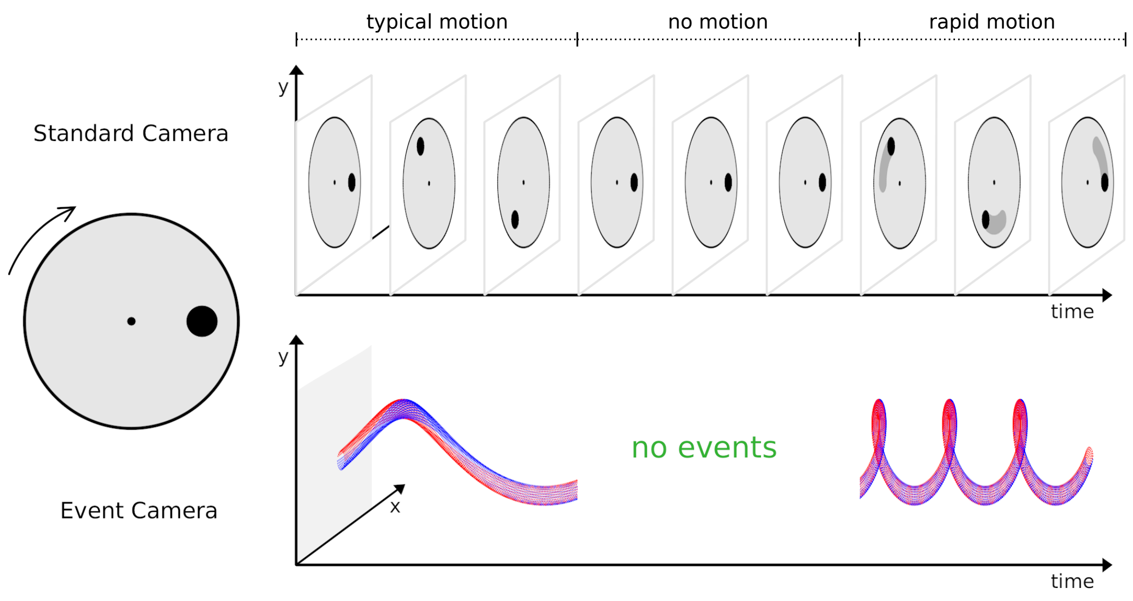


Figure 2.1: Event camera vs standard camera: in contrast to a sequence of video frames from a standard camera shown in the upper graph, a stream of events from an event camera, plotted in the lower graph, offers no redundant data output (only informative pixels or no events at all), no motion blur and high dynamic range. Red and blue dots represent positive and negative events respectively, and this figure was recreated inspired by the associated animation of [122]: `https://youtu.be/LauQ6LWTkxM?t=35s`.

2.2.2 The DVS Sensor

In this thesis, we use the first commercialised event camera from iniLabs[3], the *Dynamic Vision Sensor* (DVS), based on the research paper of Lichtsteiner *et al.* [96] as shown in Figure 2.2 (a). It has 128×128 resolution, 120 dB dynamic range and 15 microsecond latency, and communicates with a host computer using USB 2.0. It outputs a stream of events, each consisting of a pixel location u and v, a polarity bit p indicating either a positive or negative change in log intensity, and a timestamp t in microseconds as illustrated in Figure 2.2 (b). We can visualise its output as shown in Figure 2.2 (c) by accumulating events within a time interval; in this figure, white and black pixels represent positive and negative events respectively.

Specifically, each pixel of the event camera consists of three hardware components as shown as an abstracted pixel schematic in Figure 2.3 (a): a logarithmic photoreceptor, a differencing circuit, and two comparators. The photoreceptor continuously outputs a voltage

[3]iniLabs Ltd: `www.inilabs.com` (accessed September 2017)

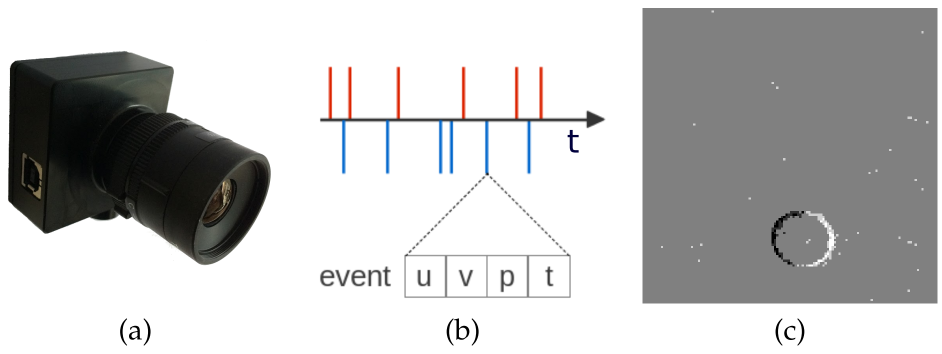


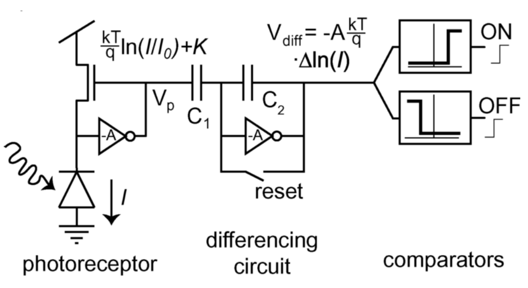
Figure 2.2: The first commercial event camera, which we have used in this thesis: (a) DVS128 from iniLabs; (b) a stream of events illustrated as upward and downward spikes for positive and negative events respectively. Each event is a tuple $\langle u, v, p, t \rangle$ where u and v are the pixel coordinates of the event, p is the polarity of the event, and t is the timestamp of the event in microseconds; (c) an image-like visualisation of accumulated events within a time interval — here white and black pixels represent positive and negative events respectively.

signal which encodes the incoming intensity logarithmically as plotted in the upper graph in Figure 2.3 (b), and the signal is then monitored by the differencing circuit and the two comparators for changes compared to a log intensity value recorded when the last event was emitted by the pixel. Once a change in log intensity which exceeds either a pre-set ON or OFF event threshold is detected, one of the comparators generates an ON or OFF event accordingly, and the newly generated event then resets and causes the pixel to memorise a new log intensity value as plotted in the lower graph in Figure 2.3 (b). The event threshold is typically set to 10% contrast, and can be adjusted by the sensor's bias settings — for a more thorough description of the bias settings, refer to the technical reference manual from the manufacturer[4]. Each DVS pixel repeats this process continuously and independently, yielding a stream of asynchronous events which encode relative changes in pixel illumination.

2.2.3 The ATIS Sensor

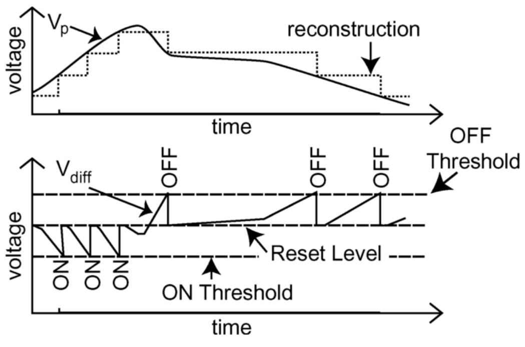
Posch *et al.* [136] developed the *Asynchronous Time-based Image Sensor* (ATIS) camera which provides absolute intensity values along with events. This was realised by combining the DVS temporal contrast pixel with a new time-based intensity measurement pixel. Each pre-set sized scene intensity change now causes three consecutive events: the first event

[4]User Guide: Biasing Dynamic Sensors: www.inilabs.com/support/hardware/biasing (accessed September 2017)



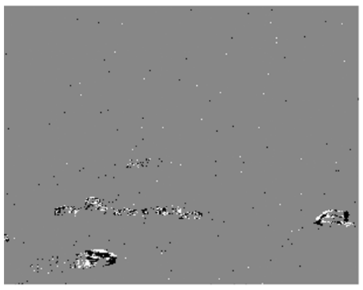
$\frac{kT}{q}\ln(I/I_0)+K$
$V_{diff} = -A\frac{kT}{q} \cdot \Delta\ln(I)$
V_p
C_1
C_2
ON
OFF
I
reset
photoreceptor
differencing circuit
comparators

(a) abstracted DVS pixel schematic

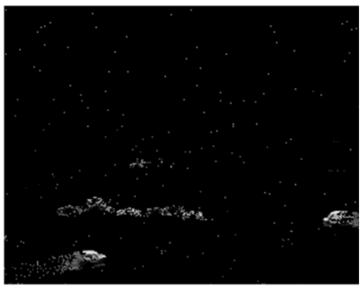


(b) principle of operation

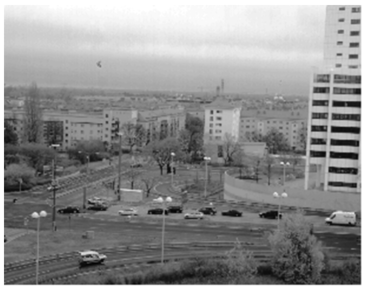
Figure 2.3: DVS pixel architecture and the principle of operation: (a) each pixel of the DVS sensor consists of three parts: a logarithmic photoreceptor, a differencing circuit, and two comparators; (b) the output signal from the photoreceptor and the principle of operation governed by the differencing circuit and comparators are plotted in the upper and lower graphs respectively. Figures courtesy of Lichtsteiner *et al.* [96].



(a)



(b)



(c)

Figure 2.4: ATIS output: (a) an image-like visualisation of collected events within a time interval; (b) an image-like visualisation of decoded intensity measurements within a time interval; (c) a reference video frame of a typical surveillance scene. Figures courtesy of Posch *et al.* [136].

is the same as the one from a DVS pixel, and the other two encode an absolute greyscale value in the inter-event time interval. Figure 2.4 (a) and (b) show an image-like visualisation of accumulated events and decoded intensity measurements within a time interval while looking at a typical surveillance scene like (c). The main advantages compared to the DVS sensors are its higher resolution (304$\times$240), higher dynamic range (143 dB) and lower latency (3 μs); and more importantly, the event-triggered wide dynamic range intensity readout. The superior properties however come at the cost of a larger pixel size and smaller fill factor as well as a higher power consumption.

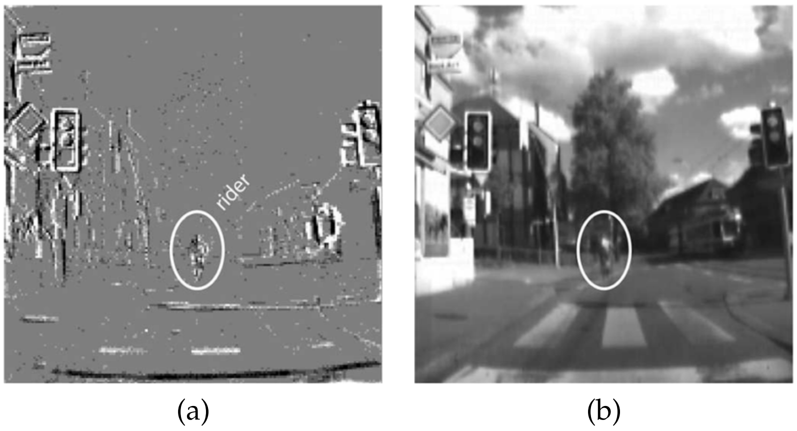


Figure 2.5: DAVIS output: (a) an image-like visualisation of collected events within a time interval; (b) an intensity frame from the DAVIS camera. Figures courtesy of Brandli *et al.* [19].

2.2.4 The DAVIS Sensor

To address some of the drawbacks of the ATIS camera, Brandli *et al.* [19] designed the *dynamic and active pixel vision sensor* (DAVIS) which interleaves event data with conventional intensity frames rather than per-event intensity measurements. The main advantage of the DAVIS pixel design is sharing the same photocurrent between the asynchronous detection of brightness changes and the synchronous readout of intensities, and as a consequence it requires only five additional transistors per pixel to add a global and rolling shutter readout yielding a smaller pixel size. The DAVIS240C camera from iniLabs has 240$\times$180 resolution and 130 dB dynamic range (55dB for greyscale frames), and communicates with a host computer using USB 2.0. It also has a time-synchronised *inertial measurement unit* (IMU) on board that provides gyro, accelerometer and magnetometer data enabling to design event-based visual-inertial methods. Figure 2.5 shows output data from the DAVIS camera — an image-like visualisation of accumulated events within a time interval (a) and a greyscale frame (b).

2.2.5 Other Advanced Event Cameras

Leñero-Bardallo *et al.* presented a 128$\times$128 dynamic vision sensor which managed to improve the minimum detectable contrast to 10% and the latency to 3.6μs while reducing the pixel size compared to previous temporal contrast event cameras at the time of the publication [90]. Two years later, Serrano-Gotarredona and Linares-Barranco introduced an improved version of their previous design which has better contrast sensitivity (1.5%), lower power consumption (4mW), and lower fixed pattern noise level while maintaining the shortest reported latency (3μs), good dynamic range (120dB), and small pixel size (30$\times$31μm^2) [149]. Recently, the minimum temporal contrast sensitivity has been further improved to 1% by Yang *et al.* [175]. More remarkably, we have now witnessed a colour dynamic and active-pixel vision sensor which outputs rolling and global shutter RGBW coded VGA resolution frames along with asynchronous monochrome QVGA resolution temporal contrast events (C-DAVIS by Li *et al.* [95]), and a 640$\times$480 VGA resolution DVS camera with the smallest reported pixel size (9$\times$9μm^2) by Son *et al.* [154].

2.2.6 Limitations

Despite their different design principles compared to standard vision sensors, event cameras are also in practice subject to noise and limited in what they can perceive. Noises arise from two primary factors. First, all the electronic components such as photodiodes and transistors contribute some electronic noise. For instance, even in complete darkness, there is still a small electric current across photodiodes which could produce noise events especially noticeable in low-light conditions or in darker areas of scenes. Second, even in well-lit conditions with little electronic noise, all existing event cameras have undesired background events which are not correlated to scene changes [175]. They are all positive events regularly produced at a certain rate which could be for instance once every 10 seconds depending on the positive event threshold.

Also, what event cameras can perceive is limited by the minimum timestamp resolution which is mostly 1μs, and two major bandwidth limitations. First, the chip bandwidth limits the maximum number of events that can be transmitted from event cameras which is about 1M *events-per-second* (eps) for the DVS128 sensor [96] and about 10Meps for the DAVIS sensor [19]. Second, the pixel bandwidth defines the maximum frequency at which an event pixel can reliably produce events in response to rapid illumination changes, and that is in the order of kHz in most event cameras. Therefore, it is very important to accordingly configure event cameras, for instance via the DVS biases described in Section 3.5.3, to achieve desirable characteristics (i.e. the noise level, sensitivity and sparsity of events) considering these limitations.

2.2.7 Summary

We summarise and compare the characteristics of the notable event cameras described in this chapter in Table 2.1. As shown, over a decade, the event camera technology has been improved significantly especially in terms of spatial resolution, pixel size, sensitivity, latency and power consumption, and we can expect much more innovation in this area in the near future. Each pixel of these event cameras is more complex at the hardware level than that of standard cameras, paying the price in terms of fill factor and pixel size. Therefore, it requires huge investment by industry to bring event cameras to mass production, and it may take longer to reach the point where current CCD or CMOS sensors are. But, we believe that this technology has a bright future by looking at the development history of standard imaging sensors, and the latest event camera from a giant company like Samsung [154].

Event cameras can be seen as the logical conclusion of devices such as rolling shutter cameras which have some degree of non-global capture; but are a much more powerful proposition since their output is purely data-driven and they do a lot of the hard things we are used to doing in computer vision to determine which pixels are useful *in hardware, at no computational cost*. It is also worth restating the comparison between the data rate of an event camera — typically on the order of 100-400kB/s — and for instance a standard monochrome VGA video feed at 30Hz: 10MB/s. The event camera generates low bit-rate, information-rich data streams which are free of the redundancy of video while removing the problems of motion blur, low dynamic range and high latency which standard cameras have.

2.3 Related Work

As introduced in the previous section, event cameras offer a breakthrough new paradigm of sensing and processing for real-time computer vision applications, with high potential in robotics, wearable devices and autonomous vehicles, but it has proven very challenging to use them in most standard computer vision problems. The clear difficulty is that most well defined computer vision methods over decades of research require synchronous full image frames to work on, and therefore they cannot be directly applied to the fundamentally different asynchronous stream of events from such an event camera. A paradigm shift from traditional frame-based methods to novel event-based approaches is required. In this section, we introduce some of the notable related work in different computer vision application areas, emphasising SLAM-like work especially, since the DVS camera became available in

Table 2.1: Summary and comparison of the event cameras described in Section 2.2.

	DVS [96]	ATIS [136]	Leñero-Bardallo *et al.* [90]	Serrano-Gotarredona and Linares-Barranco [149]	DAVIS [19]	Yang *et al.* [175]	Son *et al.* [154]
Year	2008	2011	2011	2013	2014	2015	2017
Resolution	128×128	304×240	128×128	128×128	240×180	60×30	640×480
Latency (μs)	15	3	3.6	3	3	N/A	N/A
Dynamic range (dB)	120	143	100	120	130	130	N/A
Min. sensitivity (%)	17	13	10	1.5	11	1	9
Power (mW)	24	175	145	4	14	0.72	27-50
Greyscale output	×	○	×	×	○	×	×
Chip size (mm^2)	6.3×6	9.9×8.2	5.6×5.5	4.9×4.9	5×5	3.2×1.6	8×5.8
Pixel size (μm^2)	40×40	30×30	35×35	31×30	18.5×18.5	31.2×31.2	9×9
Fill factor (%)	8.1	10	8.7	10.5	22	10.3	N/A
Supply voltage (V)	3.3	1.8/3.3	3.3	3.3	1.8/3.3	1.8	1.2/2.8
Technology	0.35μm MM/RF 2P4M	0.18μm MM/RF 1P6M	0.35μm MM/RF 2P4M	0.35μm IS 2P4M	0.18μm IS 1P6M	0.18μm MM/RF 1P6M	90nm 1P5M BSI

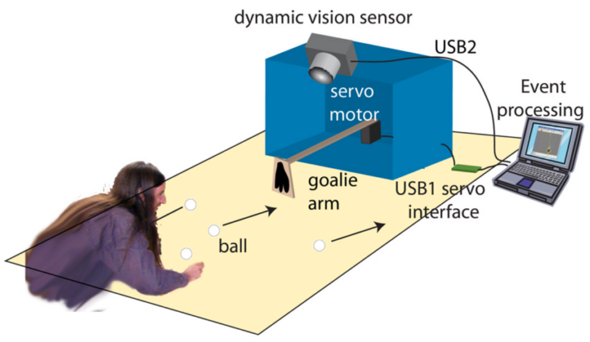
2008.

Since the emergence of event-based cameras, most computer vision work using them has focused on tracking and recognising moving targets from a fixed point of view, where almost all events are generated by the dynamic object motion. In this scenario, tasks become much easier by having additional information to spot target objects and eliminating the need to extract them from backgrounds even under difficult lighting conditions. Also, the high measurement rate and low latency properties of the event camera enable rapid motion tracking and very fast feedback control in a very efficient way, compared to the use of a high frame rate camera with its associated high computational cost and diminishing image quality with blur, noise or saturation [70].

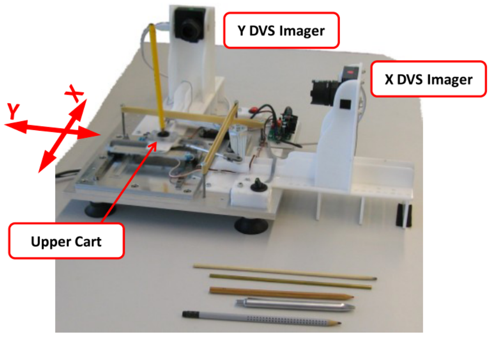
For instance, a robotic goalkeeper system which controls its robot arm to block rapidly moving balls using a DVS camera was successfully demonstrated by [47, 46] as shown in Figure 2.6 (a) — we recommend readers to view their demo video[5]. By harnessing a simple event-driven cluster tracking algorithm, the robot system achieves high update rates around 550Hz with 3ms reaction time at a peak CPU load of less than 4%. A similar application which requires very fast feedback control with latencies on the order of milliseconds was also presented by Conradt *et al.* [38] as shown in Figure 2.6 (b). This pencil balancing robot utilises a pair of DVS cameras to estimate the 3D position and angle of a pencil (or any small and thin object) by fitting incoming events to a vertical shape model, and maintains the pencil balanced on its tip by controlling an actuated table underneath accordingly. We also recommend viewing their demo video which shows its real-time performance[6].

Artificial silicon retinas have also been used in conjunction with biologically inspired learning approaches to solve high level scene understanding and perception problems. Bichler *et al.* [15] presented a feed-forward spiking neural network which can learn temporally correlated patterns from an event camera in a fully unsupervised scheme, and they showed that the proposed neural network is able to extract car trajectories on a motorway after only 10 minutes of traffic learning, and is extremely robust to noise. An event-driven *convolutional neural network* (ConvNet) was also developed by Pérez-Carrasco *et al.* [132] to recognise the orientation of human walking silhouettes and more remarkably high speed poker card symbols when flickering through a whole card deck in about 1 second in front of a DVS camera. The weights of the convolutional kernel are however trained based on a frame-based ConvNet, and then mapped to an event-driven one at a later stage. More

[5]DVS robo goalie: `https://youtu.be/IC5x7ftJ96w` (accessed September 2017)
[6]Pencil Balancer using Vision Input (DVS) only: `https://youtu.be/XVR5wEYkEGk` (accessed September 2017)



(a)



(b)

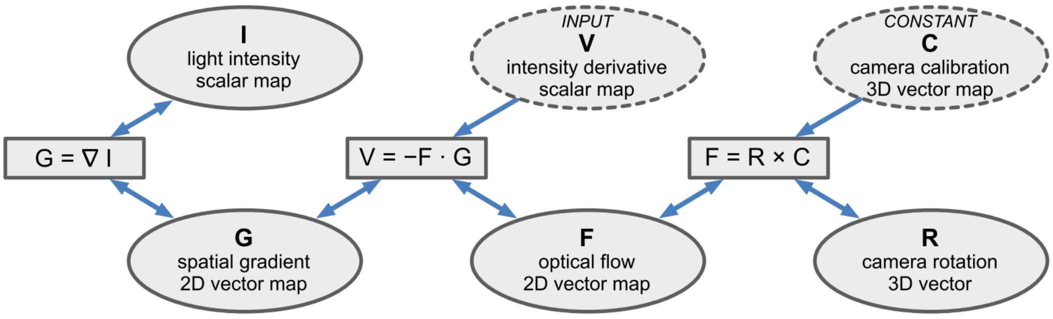
Figure 2.6: (a) A goalie robot controls its robot arm to block rapidly moving balls using a DVS camera. Figure courtesy of Delbruck and Lang [46]; (b) A pencil balancing robot maintains a pencil balanced on its tip by controlling an actuated table underneath using two DVS cameras. Figure courtesy of Conradt *et al.* [38].

recently, Lee *et al.* [88] proposed a real-time hand gesture recognition system based on a spiking neural network which processes events from a pair of DVS cameras as they arrive, and showed that their architecture has the potential to be implemented in a dedicated neuromorphic processor such as IBM's TrueNorth [111] or SpiNNaker [63] from the University of Manchester which will make the system more energy efficient and faster.

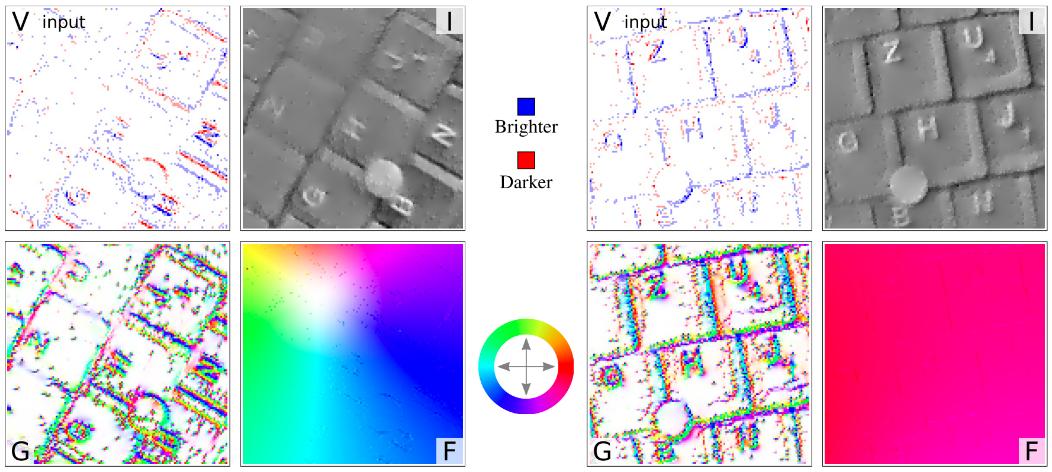
Work on tracking, reconstruction and understanding of more general, previously unknown scenes with a freely moving event camera, which we believe is the best place to take full advantage of the remarkable properties of event cameras, is still at an early stage. In recent years, however, the computer vision and robotics communities have made significant progress, and in the following sub-sections, we introduce some of the notable SLAM related work.

2.3.1 Estimating Visual Motion and Structure

Cook *et al.* [39] proposed a loosely coupled interacting network which interprets a stream of events to recover different visual estimate maps of scenes such as light intensity, spatial gradient and optical flow while estimating global rotating camera motion as shown in Figure 2.7. As their network involves only local computations, it is highly parallelisable and can be performed asynchronously, and has the high potential to be implemented in dedicated neuromorphic or graph processors placed directly behind and connected in parallel to the pixels of event cameras. In this concept, incoming events could wake up and activate local



(a)



(b)

Figure 2.7: Interacting visual maps: (a) the proposed interacting network architecture which estimates different visual quantities (shown as solid ellipses) from a stream of events based on their loosely coupled relationships (shown as solid rectangles); (b) the network input and output — accumulated input events (V), light intensity output (I), spatial gradient output (G) and optical flow output (F). Figures courtesy of Cook *et al.* [39].

computation and message passing, while the majority of the processor remains in a sleep state to conserve power. As a result of their ongoing research, they have recently implemented an extended version in the *Cellular Processor Array* (CPA) [9, 23, 98] simulator and hardware as well as general purpose GPUs demonstrating its potential to be a high speed and low power computer vision chip [108, 107].

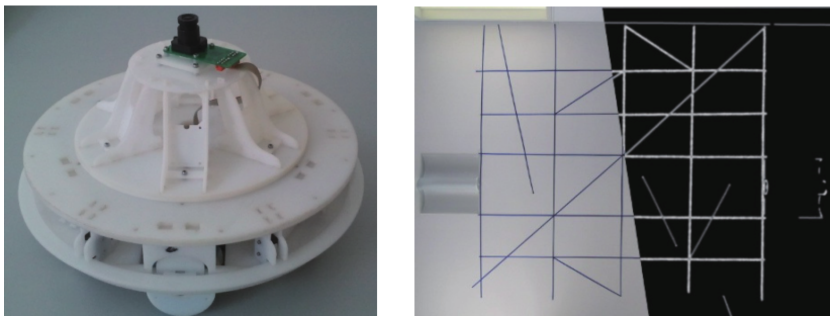
Benosman *et al.* [13] proposed a dense optical flow estimation algorithm using an event camera which precisely estimates visual flow orientation and amplitude based on a local differential approach. Events are clustered in the spatio-temporal domain with microsecond accuracy and at very low computational cost. It however only works well with a fixed event camera setup and for low textured scenes with sharp edges. Recently, Bardow *et al.* [8] proposed a new event-based optical flow estimation method which supports generic camera motion as well as dynamic scenes. The method estimates optical flow and intensity simultaneously purely based on a stream of events from a single DVS camera by minimising a cost function consisting of the asynchronous event data term and spatial and temporal regularisation terms within a sliding time window.

Miyatani *et al.* [115] adapted a patch-based dictionary learning approach to infer image spatial gradients from sparse, accumulated events. These are then upgraded to log intensities via Poisson reconstruction. They also demonstrated direct face detection from event streams without needing reconstructed intensity frames. Their framework does not require recovery of camera movement to estimate intensity information, and can produce high-speed video reconstruction at 2,000 fps even with significant noise within the captured event streams. Reinbacher *et al.* [141] also showed that intensities can be recovered from events in real-time without explicitly estimating optical flow or relying on a learning approach which reduces its complexity substantially. They formulated the intensity reconstruction problem in a variational denoising framework which accurately models the characteristics of event cameras, and defined the optimisation problem on the event manifold induced by the relative timestamps of events. Real-time performance is achieved by implementing the energy minimisation on a GPU, and their current per-event-basis implementation[7] is able to reconstruct at about 500 fps.

2.3.2 Tracking

Initially, event-based SLAM research focused on event camera motion tracking with artificially created or previously known scenes since this simplifies the problem dramatically. For instance, Weikersdorfer and Conradt [168] presented a particle filter based tracking algorithm which tracks relatively slow 3-DoF planar robot motion while an embedded upward looking event-based sensor observes planar scenes parallel to the plane of motion such as a ceiling covered with artificial line patterns as shown in Figure 2.8 (a). Later, Mueggler *et al.* [122] proposed an onboard 6-DoF localisation flying robot system equipped with a DVS camera which is able to track very rapid motion such as flips of a quadrotor. The system requires a black and white line-based known target such as the black square shown in Figure 2.8 (b), and updates the current 3D position and orientation estimates by minimising the

[7] VLOGroup/dvs-reconstruction: `https://github.com/VLOGroup/dvs-reconstruction` (accessed September 2017)


(a)

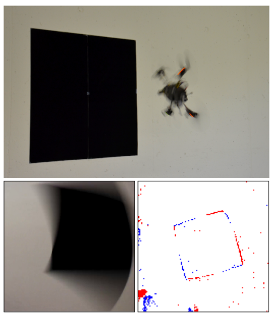

(b)

Figure 2.8: Early event-based tracking or visual odometry methods tried to simplify the problem for instance by (a) restricting to planar motion with artificially created scenes or (b) previously known scenes. Figure (a) courtesy of Weikersdorfer and Conradt [168] and Figure (b) courtesy of Mueggler *et al.* [122].

point-to-line reprojection error. Recently, a more general 6-DoF motion tracking algorithm was proposed by Gallego *et al.* [64] which can handle arbitrarily fast motions in realistic and natural scenes with a single event camera. It is however still quite limited by the nearly planar scene assumption and the reference photometric map requirement, for which they currently use image frames from a standard camera.

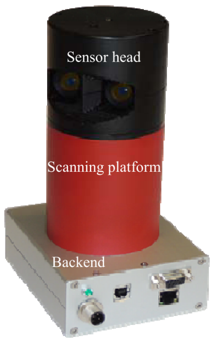
Because of the absence of synchronous reference information (i.e. reference image frames or 2D/3D maps) in this event-based tracking problem with a single event camera, Censi and Scaramuzza [26] tried to combine a DVS camera with a standard greyscale camera to estimate the small relative motion from the previous frame of a standard camera for every incoming event. The method still requires planar motion and an artificial black and white striped background to work, and, more importantly, it is subject to high latency, motion blur and low dynamic range because of their reliance on a standard camera. This is, of course, a possible practical way of using an event camera for solving SLAM problems. However, we believe that resorting to standard sensors discards many of the advantages of processing the efficient and data-rich pure event stream, as well as introducing extra complications including synchronisation and calibration problems to be solved.

One very interesting approach if the application permits is to combine an event camera with an IMU which has a high potential for the complementary nature of event and inertial data. Yuan and Ramalingam [177] used the embedded IMU of the DAVIS camera to keep the camera upright in order to support their vertical line-based tracking. More recently, Zhu *et al.* [180] presented the first algorithm to fuse a purely event-based tracking algorithm with an IMU, to provide accurate metric tracking of a camera's full 6-DoF pose. Rebecq *et al.* [140] also proposed an accurate keyframe-based, tightly-coupled visual-inertial odometry algorithm based on nonlinear optimisation, and showed their system works well even in challenging conditions by utilising the event camera's outstanding properties.

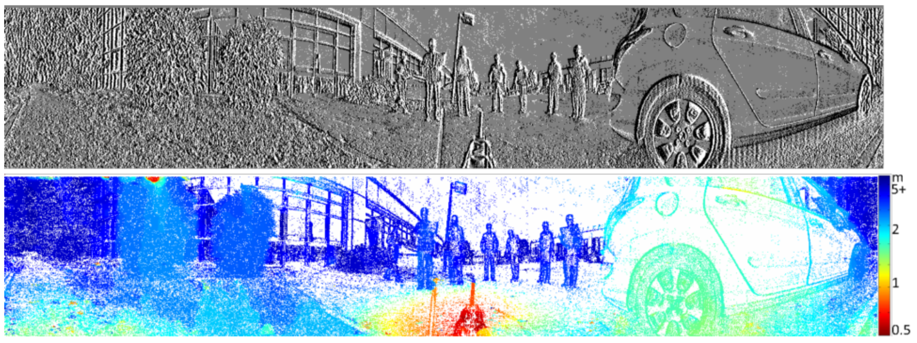
2.3.3 Reconstruction

3D reconstruction purely based on a sequence of events is a hard problem, and initially, researchers used multiple event cameras or special hardware to tackle the problem. For instance, Carneiro *et al.* [24] presented a N-camera 3D reconstruction algorithm applied to multi-cameras systems of event-based sensors. The method achieved highly accurate reconstructions despite the low spatial resolution of the DVS sensor by processing incoming events as they arrive and using only geometrical and temporal constraints. If an application permits two or more event cameras to be combined in a stereo setup (which introduces extra complication including synchronisation and calibration), the nicest part of this method is the way that stereo matching of events can be achieved based on coherent timestamps. In a much more constrained and hardware-dependent setup, a special event-based HDR depth camera was developed by [12, 148] that consists of a pair of bio-inspired dynamic vision line sensors (each with $1\times1{,}024$ pixels) which creates real-time 3D 360° panoramas aided by its high-speed rotating mechanical device and stereo event streams as shown in Figure 2.9. The key advantage compared to conventional active 3D sensors is its real-time 3D panoramic reconstruction of natural scenes with dense vertical resolution and high dynamic range properties. The system also potentially has a low processing cost and low power consumption. Matsuda *et al.* [109] also developed a new structured light scanning system called the *Motion Contrast 3D Laser Scanner* (MC3D) which combines a DVS camera with an active projector, and showed that high quality 3D object reconstruction can be achievable which is better than that of laser scanners or RGB-D cameras in some specific situations as shown in Figure 2.10. By replacing a standard camera with a DVS camera, they managed to avoid performance trade-offs in acquisition speed, resolution, and light efficiency with which traditional structured light sensors struggle.

Recently, a much more lightweight event-based 3D reconstruction method was proposed by Rebecq *et al.* [138], which only requires a single moving event camera if its trajectory is provided by an external pose estimator. In contrast to traditional *multi-view stereo* (MVS) which estimates dense 3D structure from a set of known viewpoints by solving the data association problem, they introduced the concept of *event-based multi-view stereo* (EMVS)

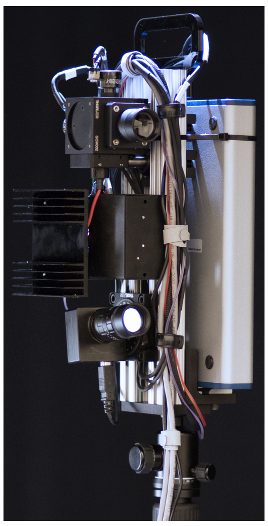


(a)

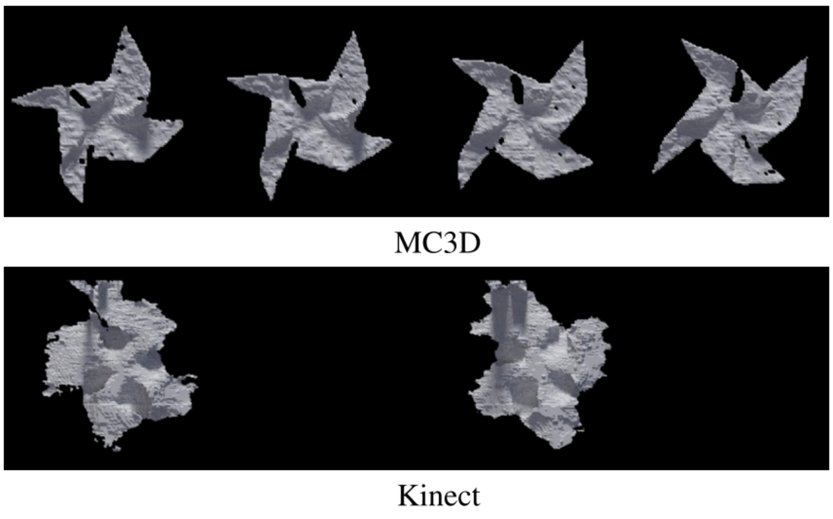


(b)

Figure 2.9: A special event-based HDR depth camera consists of a pair of dynamic vision line sensors which creates real-time 3D 360° panoramic scenes aided by its high-speed rotating mechanical device and stereo event streams. Figure (a) courtesy of Belbachir *et al.* [12] and Figure (b) courtesy of Schraml *et al.* [148].



(a)



(b)

Figure 2.10: MC3D: Motion Contrast 3D Laser Scanner which consists of an event camera with an active projector. It is able to produce high quality 3D object reconstruction which is better than laser scanners or RGB-D cameras in some specific situations. Figure (a) courtesy of the Computational Photography Lab at Northwestern University and Figure (b) courtesy of Matsuda *et al.* [109].

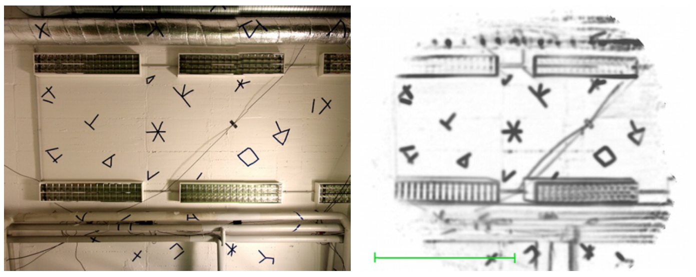
which estimates semi-dense 3D structure without the need for explicit data association or pixel intensity values. Similar to the space-sweep approach [36] in traditional MVS, the proposed EMVS method accumulates the number of rays from an event pixel through a voxel in a discretised 3D volume for every incoming event, and then finds 3D structure information from it.

2.3.4 Localisation and Mapping

When it comes to the SLAM problem (i.e. performing localisation and mapping simultaneously), most researchers have assumed that this problem is too difficult, especially estimating 3D depth purely based on a stream of events, and attempted to simplify the problem by imposing restrictions on motion and scenes, or by combining an event camera with other sensors.

For instance, an early 2D SLAM method was proposed by Weikersdorfer *et al.* [169] which tracks a ground robot's pose while reconstructing a planar ceiling map with an upward looking DVS camera as shown in Figure 2.11 (a). This is an extended version of their previous work [168] to generate a map automatically during self-localisation, but it is still quite limited to planar scenes parallel to the plane of relatively slow motion. To overcome the limitations of their previous methods, they also proposed an event-based 3D SLAM method with a DVS camera combined with a RGB-D sensor [170] as shown in Figure 2.11 (b). By finding pixel correspondences between two sensors through a special off-line calibration procedure, the multi-camera system produces a stream of sparse events with depth information which enables the estimation of semi-dense 3D structure of the scene. Similarly, Kueng *et al.* [85] developed an event-based low latency visual odometry algorithm which combines the benefits of event cameras and that of conventional cameras. They utilised the DAVIS camera which has an event-based sensor and a conventional frame-based sensor built into the same pixel array. They achieved low latency pose updates by tracking visual features extracted from image frames whenever a new event arrives, and tracked features were used to infer 3D quantities using probabilistic depth filters for mapping. These are again, of course, possible practical ways of using an event camera for solving SLAM problems. However, we would like to again emphasise that resorting to standard sensors discards many of the advantages of processing a pure event stream.

The first event-based 2D SLAM method demonstrating high quality HDR and super-resolution intensity panorama reconstruction and motion estimation from a purely rotating event-based camera [82], and the first event-based real-time 3D SLAM method demonstrat-


(a)

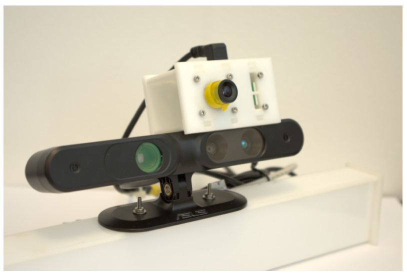

(b)

Figure 2.11: Early event-based SLAM methods have tried to simplify the problem for instance by (a) restricting to planar motions and scenes, or (b) combining an event camera with an additional sensor. Figure (a) courtesy of Weikersdorfer *et al.* [169] and Figure (b) courtesy of Weikersdorfer *et al.* [170].

ing joint estimation of general 6-DoF camera motion, scene intensity and scene 3D depth from pure event data [83] were proposed as part of the work in this thesis, and will be described in more detail in the following chapters. Also, to investigate whether current techniques can be applied to a large scale visual SLAM problem, Milford *et al.* [112] presented a simple visual odometry system using a DVS camera with loop closure built on top of the SeqSLAM algorithm using events accumulated into frames [113].

Most recently, Rebecq *et al.* [139] presented a real-time event-based 6-DoF tracking and mapping method with an event camera by combining a new event-based tracking approach based on event frame to model alignment with their recent EMVS method [138] for semi-dense 3D reconstruction. In contrast to our method [83] which requires a GPU to operate in real-time, their tracking component does not require intensity information which means the whole system can run in real-time only on a CPU up to moderate motions.

We however believe that relying on artificial event frames created by accumulating events within a time interval introduces unnecessary bounds on the update rate and latency. Therefore, all of the methods described in this thesis operate on an event-by-event basis in real-time and based on probabilistic filtering to maximise update rates with low latency. Moreover, we have taken this event-based approach with our long-term research in mind that moving towards a future where event-based SLAM algorithms are implemented on appropriate neuromorphic or graph-based processing architectures sitting directly behind and connected in parallel to the pixels of future event sensors. Incoming events will wake up local computation and message passing, while the majority of the processor sleeps to conserve power taking full advantage of the low data rate. We also believe that reconstructed intensity information will undoubtedly be useful for loop closure or high level perception in such an event-based SLAM system.

2.3.5 Dataset

Since established benchmarks in the computer vision field have greatly contributed to the advance of algorithms in many areas, an important piece of future work is to design and release suitable comparative benchmarks for event camera-based SLAM research. Recently, a few useful datasets have become available for optical flow estimation [143][8], for visual navigation [10][9], and for pose estimation, visual odometry, and SLAM in general [124][10], and we expect many more event camera-based benchmarks will be designed and released in the near future.

2.4 Discussion and Summary

As shown in this chapter, the properties of event cameras offer the potential to overcome the limitations of real-time, real-world computer vision applications. The event camera is gradually becoming more widely known by researchers in computer vision, robotics and related fields, in particular since the release as commercial products of the DVS and DAVIS cameras. Over a decade, event camera technology has been improved significantly in terms of spatial resolution, pixel size, sensitivity, latency and power consumption, and we can expect many more innovations in this area in the near future.

The uses demonstrated of using event cameras to date however have been limited. We are interested in tracking and reconstruction of general, previously unknown scenes with a freely moving event camera, which we believe is the best place to take full advantage, and there have been little work on building coherent scene models from event data in this case. The clear difficulty is that most of the methods and abstractions normally used in reconstruction and tracking, such as feature detection and matching (e.g. [100, 11, 93]) or iterative image alignment (e.g. [103, 106, 7]) cannot be directly applied to its fundamentally different visual measurement stream. To keep up to date with research and development in this field, we refer to the *Event-based Vision Resources*[11].

[8]DAVIS Optical Flow Dataset: `http://sensors.ini.uzh.ch/databases.html` (accessed September 2017)

[9]Dataset for Visual Navigation with Neuromorphic Methods: `https://github.com/fbarranco/eventVision-evbench` (accessed September 2017)

[10]The Event-Camera Dataset and Simulator: `http://rpg.ifi.uzh.ch/davis_data.html` (accessed September 2017)

[11]Event-based Vision Resources: `https://github.com/uzh-rpg/event-based_vision_resources` (accessed September 2017)

In the following chapters, we will introduce the first event-based 2D SLAM method demonstrating high quality HDR and super-resolution intensity panorama reconstruction and motion estimation from a purely rotating event-based camera [82], its real-time implementation, and the first event-based real-time 3D SLAM method demonstrating joint estimation of general 6-DoF camera motion, scene intensity and scene 3D depth from pure event data [83] in more detail.

CHAPTER 3

Preliminaries

Contents

In this chapter, we first clarify the notational conventions used in equations throughout this thesis and review some important mathematical foundations of geometry-based 3D visual SLAM methods. And, as event cameras have not been well established as common computer vision devices just yet, we will give a brief overview of some of the prerequisites required to work with an event camera; in particular how to communicate with the DVS128 camera through its USB interface. In addition, we will list a number of software libraries and APIs which have helped us to implement our methods more easily and rapidly.

3.1 Notation

We denote m-dimensional vectors by boldfaced lowercase letters such as:

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} , \tag{3.1}$$

or $\mathbf{v} = (v_1, v_2, \cdots, v_m)^\top$, and m×n matrices by uppercase letters such as:

$$\mathtt{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} . \tag{3.2}$$

Also, to facilitate a great range of transformations using linear algebra, we use *homogeneous coordinates* which increase the size of a vector by one, and are represented using the dot notation such as:

$$\dot{\mathbf{v}} = \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} . \tag{3.3}$$

The reverse operation is called *homogeneous projection* denoted by π, and it simply divides all the elements by the last one which is then truncated such as:

$$\pi \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} , \tag{3.4}$$

for a 3-dimensional vector for instance.

3.2 Frames of Reference

We call the global coordinate system the world frame of reference w, and even though the event camera does not have the frame concept of standard cameras, we still use frames of

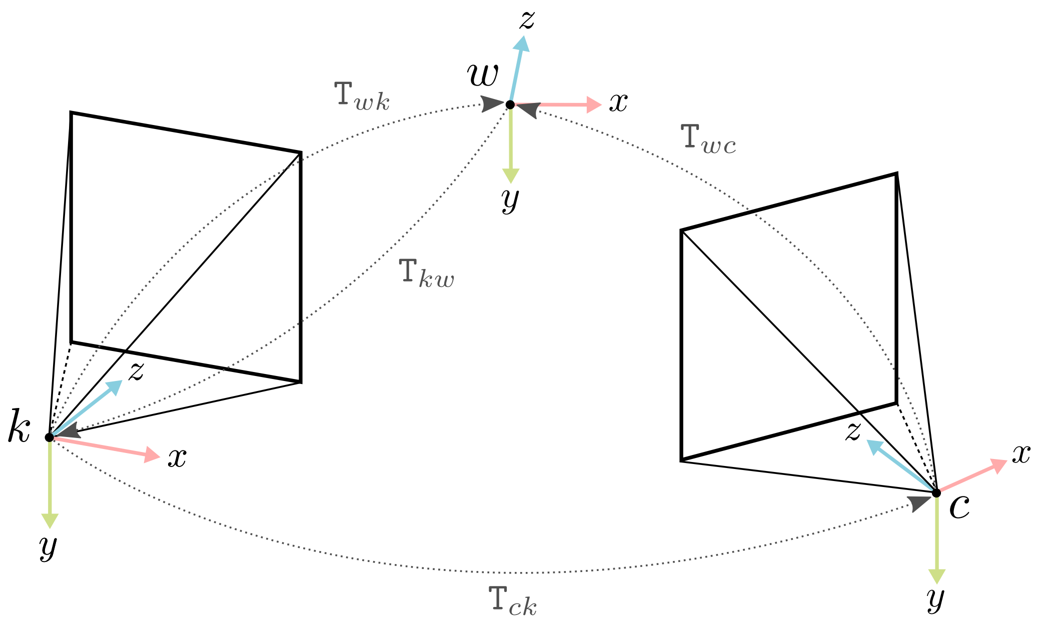


Figure 3.1: Frames of reference: the world frame of reference w represents the global coordinate system, and the camera frames of reference k and c represent local coordinate systems. All of them follow the right-hand rule, and the camera's optic centre is located at $(0, 0, 0)^\top$, its up vector is aligned with the negative y-axis, and its principle point is aligned with the positive z-axis in its local coordinate system. Interacting these coordinates is represented by a rigid body transformation matrix such as $\mathtt{T}_{wk}$, $\mathtt{T}_{kw}$, $\mathtt{T}_{wc}$, and $\mathtt{T}_{ck}$.

reference when we refer to local coordinate systems such as k or c in Figure 3.1. By default, we use a right-handed coordinate system, and for a camera frame of reference, its optic centre is located at $(0, 0, 0)^\top$ while its up vector is aligned with the negative y-axis and its principal point is aligned with the positive z-axis. A position vector $\mathbf{p}$ in the world frame of reference w and in a local coordinate system c are denoted as $\mathbf{p}_w$ and $\mathbf{p}_c$ respectively with appropriate subscripts.

3.3 Rigid Body Transformations

Transforming between coordinate systems is represented by a 4×4 rigid body transformation matrix such as $\mathtt{T}_{wk}$, $\mathtt{T}_{kw}$, $\mathtt{T}_{wc}$, and $\mathtt{T}_{ck}$ in Figure 3.1 — the first and second subscripts of these matrices represent the target and source frames of reference respectively. For example, $\mathtt{T}_{wk}$ transfers a homogeneous point $\dot{\mathbf{p}}_k$ in the frame of reference k to an equivalent homogeneous point $\dot{\mathbf{p}}_w$ in the world frame of reference w via matrix-vector multiplication such that:

$$\dot{\mathbf{p}}_w = \mathtt{T}_{wk}\dot{\mathbf{p}}_k \ . \tag{3.5}$$

The transformation matrix $\mathtt{T}_{wk}$ is decomposed as:

$$\mathtt{T}_{wk} = \begin{pmatrix} \mathtt{R}_{wk} & \mathbf{k}_w \\ \mathbf{0}^\top & 1 \end{pmatrix} = \begin{pmatrix} \mathtt{R}_{wk} & -\mathtt{R}_{wk}\mathbf{w}_k \\ \mathbf{0}^\top & 1 \end{pmatrix} \ , \tag{3.6}$$

where $\mathtt{R}_{wk}$ is a 3×3 orthonormal rotation matrix satisfying ${\mathtt{R}_{wk}}^{-1} = {\mathtt{R}_{wk}}^\top$, and $\mathbf{k}_w$ represents the position of the optic centre of the frame of reference k in the world frame of reference. These rigid body transformation matrices also satisfy:

$$\mathtt{T}_{ck} = \mathtt{T}_{cw}\mathtt{T}_{wk} \ , \tag{3.7}$$

$$\mathtt{T}_{kw} = {\mathtt{T}_{wk}}^{-1} = \begin{pmatrix} {\mathtt{R}_{wk}}^\top & -{\mathtt{R}_{wk}}^\top\mathbf{k}_w \\ \mathbf{0}^\top & 1 \end{pmatrix} \ . \tag{3.8}$$

3.4 Projection

In the field of geometry-based SLAM and multi-view stereo, we are interested in projecting 3D points from the world into a camera frame of reference to find their corresponding 2D image coordinates. If we know the position and orientation of the camera as well as its intrinsic parameters from a calibration procedure (as described in Section 3.6 for instance), they can be calculated based on the pinhole camera model [74].

3.4.1 Pinhole Camera Model

As illustrated in Figure 3.2, a 3D point $\mathbf{p}_c$ in the camera frame of reference c can be projected to a 2D image position $\mathbf{u} = (u, v)^\top$ as:

$$\mathbf{u} = \pi\left(\mathtt{K}\mathbf{p}_c\right) \ , \tag{3.9}$$

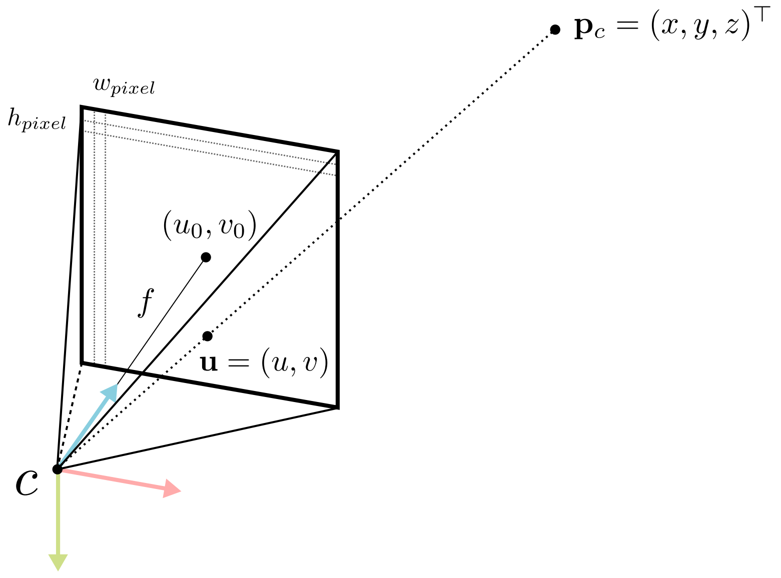


Figure 3.2: A 3D point $\mathbf{p}_c$ in the local coordinate system c can be projected to a 2D point $\mathbf{u} = (u, v)^\top$ in image space using the pinhole camera model. Here f is the focal length and (u_0, v_0) is the principal point of the camera.

where K is the intrinsic or calibration matrix:

$$\mathtt{K} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} . \tag{3.10}$$

Here (u_0, v_0) is the principal point which is the intersection between the optical axis (i.e. the positive z-axis) and the image plane, and f_u and f_v are:

$$f_u = \frac{f}{w_{pixel}} , \qquad f_v = \frac{f}{h_{pixel}} , \tag{3.11}$$

where f is the focal length which is the distance between the optic centre and the principal point, and w_{pixel} and h_{pixel} are the width and height of the camera's pixels and are often treated as the same (i.e. square pixels). The inverse intrinsic matrix which transfers a projected point in image space into the camera frame of reference is defined as:

$$\mathtt{K}^{-1} = \begin{pmatrix} \frac{1}{f_u} & 0 & -\frac{u_0}{f_u} \\ 0 & \frac{1}{f_v} & -\frac{v_0}{f_v} \\ 0 & 0 & 1 \end{pmatrix} . \tag{3.12}$$

3.5 Event Camera Interface

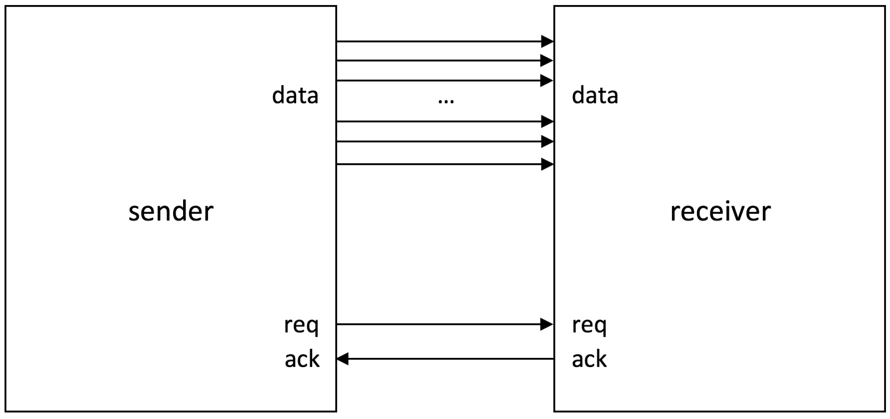
Although event cameras are gradually becoming more widely known by researchers in the computer vision, robotics and related fields, they have not been well established as common computer vision devices just yet. Therefore, in this section, we introduce some of the prerequisites required to work with an event camera, and especially how to communicate with the DVS128 camera used in this thesis through its USB interface.

3.5.1 Address Event Representation

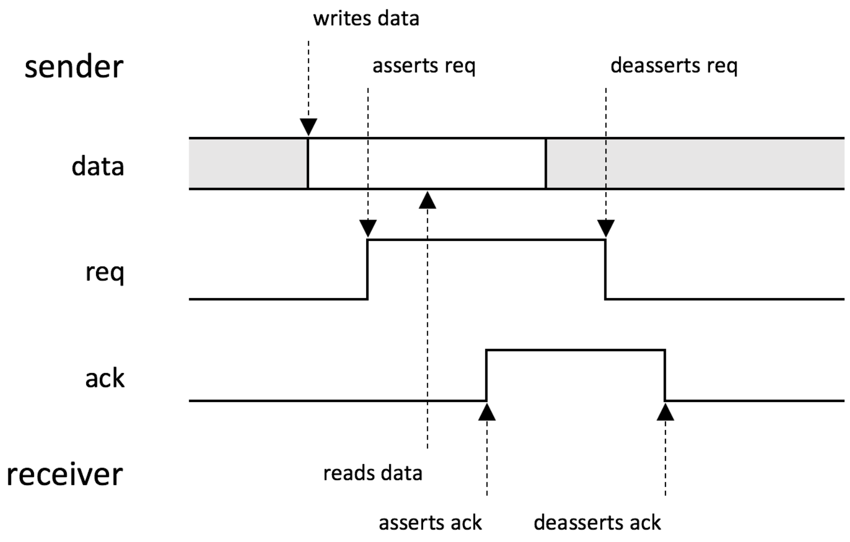
The sparse and asynchronous nature of spikes (or events) in the neuromorphic design principle has called for new communication protocols which can easily distinguish which neuron (or event camera pixel) has fired a specific spike or event. One such new communication strategy is the *Address Event Representation* (AER) [152, 104, 87], which is now the de facto standard protocol used in much neuromorphic hardware including most current event cameras.

As shown in Figure 3.3 (a), simple AER protocol-based neuromorphic systems use two control signals *req* and *ack* to synchronise the data transfer through the data bus between the sender and the receiver based on a four-phase handshake. Specifically, as illustrated in Figure 3.3 (b), the sender asserts a req signal once it has written data onto the multiline data bus to notify the receiver, and at this point, the data on the bus can be considered valid. The receiver then reads the data and confirms that by asserting an ack signal which yields the subsequent sequential deassertions of the req and ack signals by the sender and receiver respectively, and goes back to wait for the next transaction. The actual AER data bus width and protocol vary depending on specific hardware — for instance, the DVS128 camera used in our work has the 15-bit width AER bus and its data is encoded in a packet as shown in Figure 3.3 (c), where p is the polarity, and u and v are the pixel address of an event.

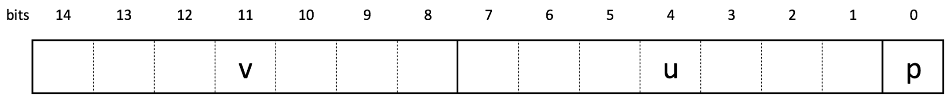
For a deeper understanding of the AER protocol, which is only required for the interface between an event camera and a microcontroller or another neuromorphic hardware, we refer



(a)



(b)



(c)

Figure 3.3: The AER protocol: (a) the sender and the receiver of a simple AER system communicate each other through two control signals (req and ack) and a data bus; (b) a simple AER communication sequence logic diagram based on a four-phase handshake; (c) the AER data packet of the DVS128 camera where p is the polarity, and u and v are the pixel address of an event.

the reader to the relevant literature [152, 104, 87] or the technical reference manual from the manufacturer[1]. All of our implementations described in this thesis run on a standard PC and communicate with an event camera (i.e. the DVS128 camera) via its USB interface which is described in the next subsection.

3.5.2 DVS USB Interface

To communicate with the DVS camera connected to a standard PC through its USB cable, we initially used the open source jAER software project [44][2] written mostly in Java, but quickly realised that we needed a more lightweight software interface which can be easily integrated with our implementations of the methods described in this thesis written in C/C++. We therefore developed our own USB driver using the libusb library[3] which provides generic access to USB devices. Recently, the manufacturer also released its minimal C library called *libcaer* to access, configure and communicate with their event cameras[4].

The DVS event camera, like a standard USB device, can be identified by its *vendor ID* (VID) and *product ID* (PID), and communicated with using an application running on a standard PC using its predefined commands — we list all the identifiers and commands of the DVS camera in Table 3.1. 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 3.4, where p is the polarity, u and v are the pixel address, and t is the timestamp of an event. The maximum bandwidth of the DVS128 camera is 1Meps, and for a more thorough description of the libusb API we refer the reader to the technical reference manual on the library website.

3.5.3 DVS Biases

The biases are the on-chip parameters which define the operating characteristics of the event camera — they represent voltages and currents which stay stable during operation and are applied to the analog electronic circuits of the event camera's pixels. For instance, the DVS128 camera used in our work has 12 biases as listed in Table 3.2 which can be dynamically adjusted via its USB interface — we need to send the `SEND_BIAS_BYTES` command followed by all 12 3-byte long biases.

[1]Address-Event Representation (AER) protocol: `https://inilabs.com/support/hardware/aer/` (accessed September 2017)
[2]jAER: `https://sourceforge.net/projects/jaer/` (accessed September 2017)
[3]libusb: `https://github.com/libusb/libusb` (accessed September 2017)
[4]libcaer: `https://github.com/inilabs/libcaer` (accessed September 2017)

Table 3.1: DVS128 USB Identifiers and Commands

Identifiers / Commands	Value	Description
VID	0x152A	Vendor ID
PID	0x8400	Product ID
START_TRANSFER	0xB3	start sending events from FIFO endpoint
STOP_TRANSFER	0xB4	stop sending events from FIFO endpoint
EARLY_TRANSFER	0xB7	transfer whatever you have now
SEND_BIAS_BYTES	0xB8	send bias bytes out on SPI interface
POWERDOWN	0xB9	stop working
FLASH_BIASES	0xBA	flash the bias values to EEPROM
RESET_TIMESTAMP	0xBB	reset timestamp
SET_ARRAY_RESET	0xBC	set array reset of retina
DO_ARRAY_RESET	0xBD	do an array reset (toggle arrayReset for a fixed time)
SET_LED	0xBF	set the board's LED
FIRMWARE	0xA0	download/upload firmware
VR_EEPROM	0xA2	load (upload) EEPROM
VR_RAM	0xA3	load (upload) external RAM
VR_RESET_FIFO	0xC4	reset FIFO

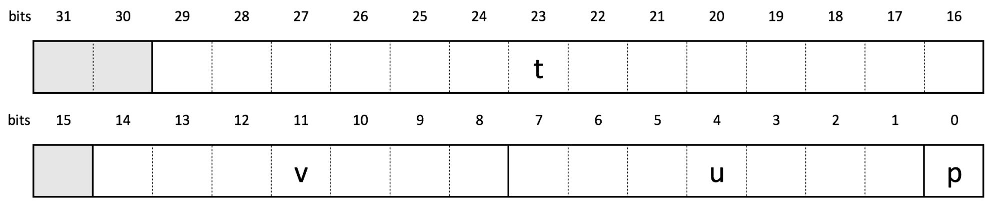


Figure 3.4: The 4-byte long USB event packet of the DVS128 camera, where p is the polarity, u and v are the pixel address, and t is the timestamp of an event.

Table 3.2: DVS128 Biases

Bias	Value		Description
	Dec	Hex	
pr	3	0x000003	photoreceptor: controls the reponse speed to changes in light
cas	54	0x000034	first stage (photoreceptor) cascode: controls the speed and stability of the circuit — **need to be higher than the pr bias**
foll	51	0x000033	source follower: separates the first and second (differential) stages
diff	30153	0x0075C9	differential: controls the response speed to a change in the light-related signal
diffOn	482443	0x075C8B	On event threshold: — **need to be higher than the diff bias**
diffOff	132	0x000084	Off event threshold: — **need to be lower than the diff bias**
injGnd	1108364	0x10E98C	injected ground: controls the response speed to the acknowledge in order to reset
req	159147	0x026DAB	pull down for passive load inverters in digital AER pixel circuitry
refr	6	0x000006	refractory period: defines the time period during which the pixel will be reset
PuY	16777215	0xFFFFFF	pull up on request from Y arbiter
PuX	8159221	0x7C7FF5	pull up on request from X arbiter
reqPd	16777215	0xFFFFFF	pull down on chip request

We present all the bias setting values used in our work and a brief description of each bias in the table. Again, for more details, please see technical reference manual from the manufacturer[5].

3.6 Event Camera Calibration

Although the sensors of event cameras are fundamentally different from conventional imaging sensors, they use the same optics to which traditional perspective projection applies as described in Section 3.4, and therefore a calibration procedure is required for intrinsic

[5]User Guide: Biasing Dynamic Sensors: `www.inilabs.com/support/hardware/biasing` (accessed September 2017)

Table 3.3: DVS Intrinsic Parameters

Intrinsic Parameter	Value	Description
f_u	115.534	camera focal length
f_v	115.565	camera focal length
u_0	79.262	optical center
v_0	65.531	optical center
k_1	-0.308	distortion parameter
k_2	0.110	distortion parameter
p_1	0.001	distortion parameter
p_2	0.002	distortion parameter

calibration and to compensate radial distortion to make use of the projective geometry of events.

While such a calibration method is now a standard process for conventional cameras, the obvious difficulty for event cameras is that the standard *static* calibration pattern cannot be used as the event camera only responds to scene changes — we need to either move the camera or the calibration chart, or have an active pattern such as blinking LEDs as in [123]. We therefore first create a *special* calibration pattern consisting of a grid of dots using the Animated *Graphics Interchange Format* (GIF)[6], and display it on a computer screen as shown on the left monitor in Figure 3.5. As the pattern is blinking frequently enough to stimulate the static event camera's pixels, we generate a sequence of image-like event frames by accumulating events within a time interval as shown on the right monitor in the same figure, and apply a standard intrinsic calibration method such as the polynomial power series model [179] to the event video stream. Table 3.3 presents the intrinsic parameters from the calibration method as an example, and we recommend readers to view our video[7] which illustrates the calibration procedure in a better form than a still figure (also see Appendix A).

Within all the methods described in this thesis, we use the same lens with 4.5*mm* focal length and 60° horizontal/vertical field of view, and always remove lens distortion based on the distortion parameters estimated from event camera calibration to make use of the projective geometry of events at the start of the processing pipeline. Figure 3.6 shows image-like visualisations of events with and without lens distortion.

[6]Animated Graphics Interchange Format (GIF): `https://en.wikipedia.org/wiki/GIF` (accessed September 2017)

[7]Event Camera Calibration: `https://youtu.be/OK_m6OobntE` (accessed September 2017)

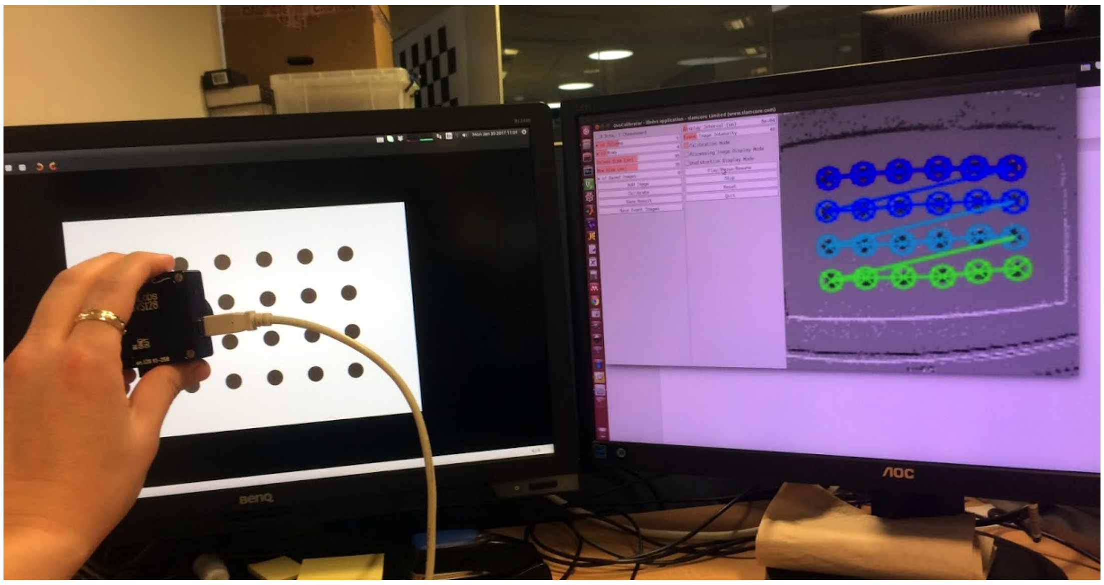


Figure 3.5: Event camera calibration. We apply a standard intrinsic calibration method to a sequence of image-like event frames (shown on the right monitor) created by accumulating events within a time interval while the event camera is looking at the blinking calibration pattern consists of a grid of dots displayed on the left monitor.

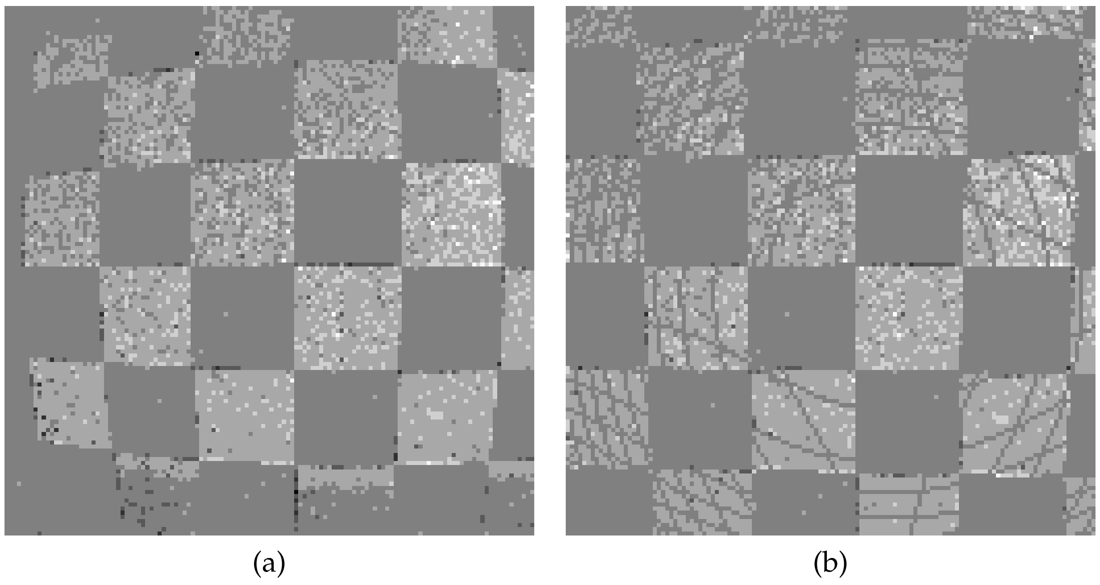


Figure 3.6: Event camera lens distortion compensation: (a) an image-like visualisation of events with lens distortion — straight lines appear curved; (b) an image-like visualisation of events with lens distortion removed by the compensation process — straight lines remain straight.

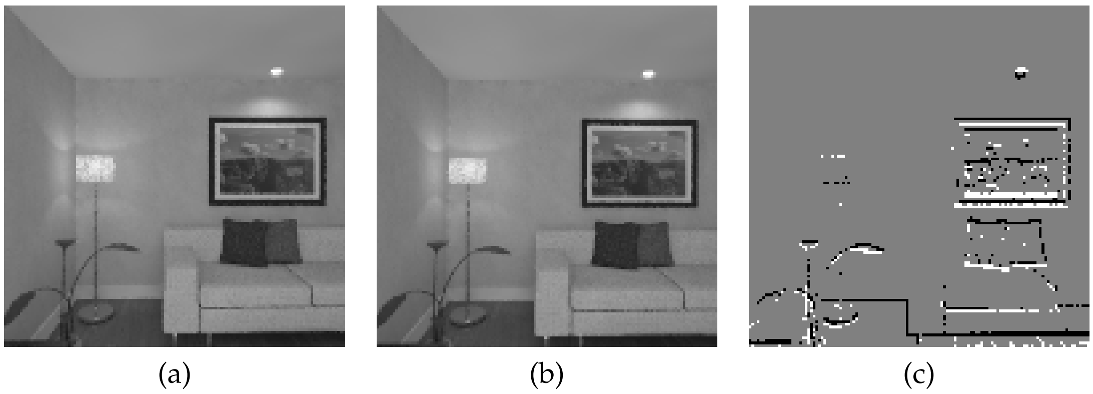


Figure 3.7: In our simulator, synthetic events are generated as shown in (c) from any pixel where the log intensity difference between corresponding pixels in the rendered frame at $t-1$ and t (shown in (a) and (b) respectively) satisfies the positive or negative event conditions.

3.7 Event Camera Simulator

As well designed synthetic benchmark methods often greatly contribute to the development and evaluation of new algorithms, especially with new sensors, we have developed an event camera simulator which mimics the characteristics of the event camera to provide a repeatable experimental environment.

Our implementation is largely inspired by Handa *et al.* [70] who used POV-Ray[8] and synthetic trajectories in their framework to obtain photo-realistic rendered frames and ground truth depth maps. However, in our case, simulating a realistic sensor noise model for the event camera has not been well studied yet and this remains as an important issue for future research. We therefore assume a perfect noise-free event camera, and render greyscale frames at the same 128×128 resolution and at very fine time resolution based on various synthetic scenes and trajectories. We then generate synthetic events from any pixel whose log intensity difference between corresponding pixels in two consecutive rendered frames satisfies the positive or negative event conditions. Figure 3.7 shows the synthetic living room scene of the ICL-NUIM dataset [71][9] as an example.

Our event camera simulator generates a sequence of synthetic events along with ground truth data such as camera trajectory, IMU data, scene intensity, log intensity gradient, scene

[8]POV-Ray: `http://www.povray.org` (accessed September 2017)
[9]ICL-NUIM dataset: `https://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html` (accessed September 2017)

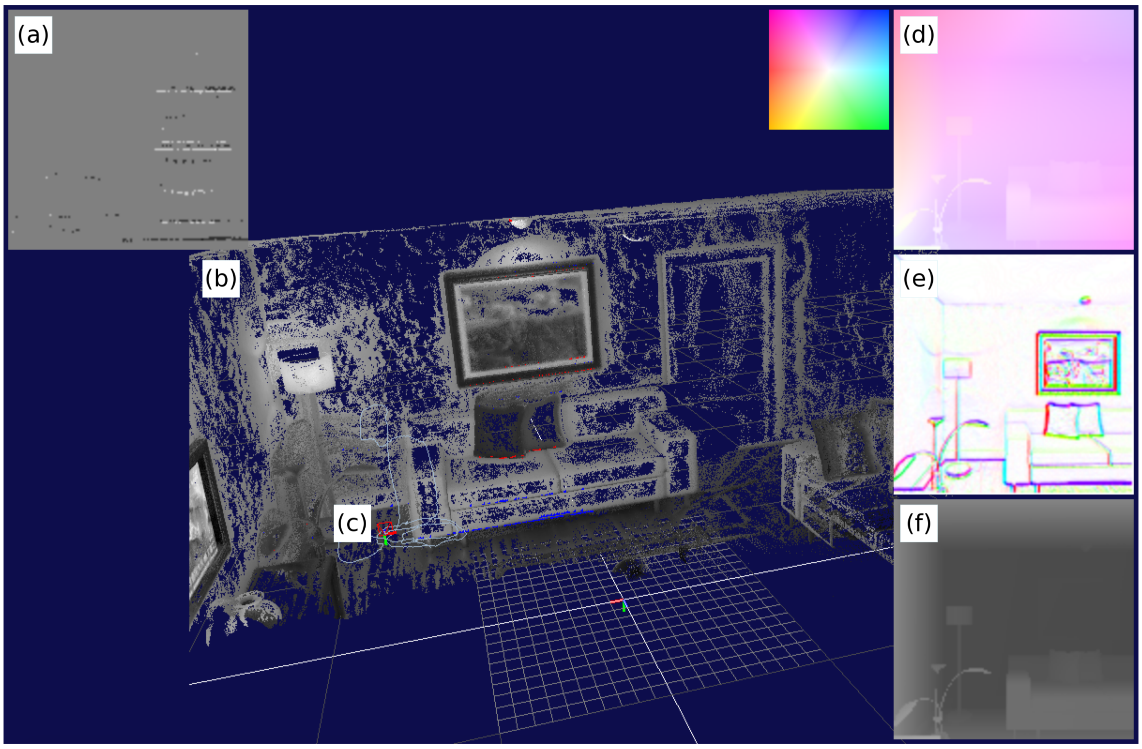


Figure 3.8: Our event camera simulator generates synthetic events as shown in (a) along with various ground truth data such as: (b) 3D point cloud; (c) camera trajectory; (d) motion flow; (e) log intensity gradient; (f) scene depth.

depth, 3D point cloud, and motion flow. Figure 3.8 shows a graphical representation of the various outputs of our event camera simulator.

3.8 Software

When developing the software systems to implement the methods described throughout this thesis, we have greatly benefited from many publicly available software libraries and APIs including CUDA, OpenGL, OpenCV and Eigen, which have already become very useful common tools for computer vision or robotics researchers.

In addition, we largely relied on the open source Pangolin library[10] for OpenGL display, user interface, managing config files, and real-time data plotting, and also the open source Sophus library[11] to utilise Lie groups more easily for 2D and 3D geometric problems (e.g. the

[10]Pangolin: https://github.com/stevenlovegrove/Pangolin (accessed September 2017)
[11]Sophus: https://github.com/strasdat/Sophus (accessed September 2017)

special orthogonal group **SO**(3) or the special Euclidean group **SE**(3)). For a more thorough description of these libraries and APIs, we refer the reader to the technical reference manuals on their websites.

3.9 Summary

In this chapter, we have presented the mathematical notation conventions and geometrical foundations used throughout this thesis. We have also given introduction to some important prerequisites required to work with an event camera, especially how to communicate with the DVS128 camera used in our research, and a number of software libraries and APIs which were highly useful to implement our methods. In the following chapters, we will now present our event-based visual SLAM methods in detail.

Chapter 4

Simultaneous Mosaicing and 3-DoF Tracking with an Event Camera

Contents

4.1 Introduction

In this chapter, we show that an event stream from a single hand-held event camera, with no additional sensing, can be used to track accurate camera rotation while building a persistent and high quality mosaic of a scene which is super-resolution accurate and has high dynamic range. A set of video frames from a standard camera with no parallax observed can be fused into a panoramic image, and this technique has been considered well established in computer vision for a long time [162]. It is however not possible to apply the same methodologies such as a feature based filtering method [32] or a whole image alignment with global optimisation method [99] as event cameras provide not image frames but an asynchronous sequence of per-pixel log intensity changes. Therefore, approaches aiming at simultaneous camera motion estimation and building a scene mosaic with an event camera need also to jointly estimate the intensity appearance of the scene, or at least a highly descriptive synchronous function of this such as a gradient map, and this is the essential mechanism built into our event-based mosaicing method.

As shown in Figure 4.1, the method takes input in the form of a stream of events from a single pure rotation event camera looking at an unknown and unstructured natural scene, and operates on an event-by-event basis to maximise the update rate with low latency. Our approach relies on two parallel probabilistic filters as shown in the blue rounded boxes to jointly track the global rotational motion of a camera and estimate the log intensity gradients of the scene around it. The gradient map is then upgraded to a full image-like mosaic, from which we calculate the value of a measurement for the camera pose estimation, with super-resolution and high dynamic range properties as shown in the grey rounded box. Each of these components essentially assumes that the current estimate from one component is accurate enough to lock for the purpose of estimating the other, following the alteration approach of most recent successful data-rich SLAM systems such as PTAM [84], DTAM [128], LSD-SLAM [57], or ORB-SLAM [125].

Note that we do not currently explicitly address bootstrapping in our method. We have found that simple alternation of the tracking and mapping components, starting from scratch, will very often lead to rapid convergence of joint tracking and mapping, though there are sometimes currently gross failures and this is an important issue for future research.

4.2 Preliminaries

We use the notation $\mathbf{e}(u, v) = (u, v, p, t)^\top$ to denote an event with pixel location u and v, polarity p and timestamp t. Our event camera has the fixed pre-calibrated intrinsic matrix $\mathtt{K}$ and all event pixel locations are pre-warped to remove radial distortion via the event camera calibration method described in Section 3.6. We define two important time intervals τ and τ_c which are used in our algorithm. For a simple example to illustrate them, let us assume

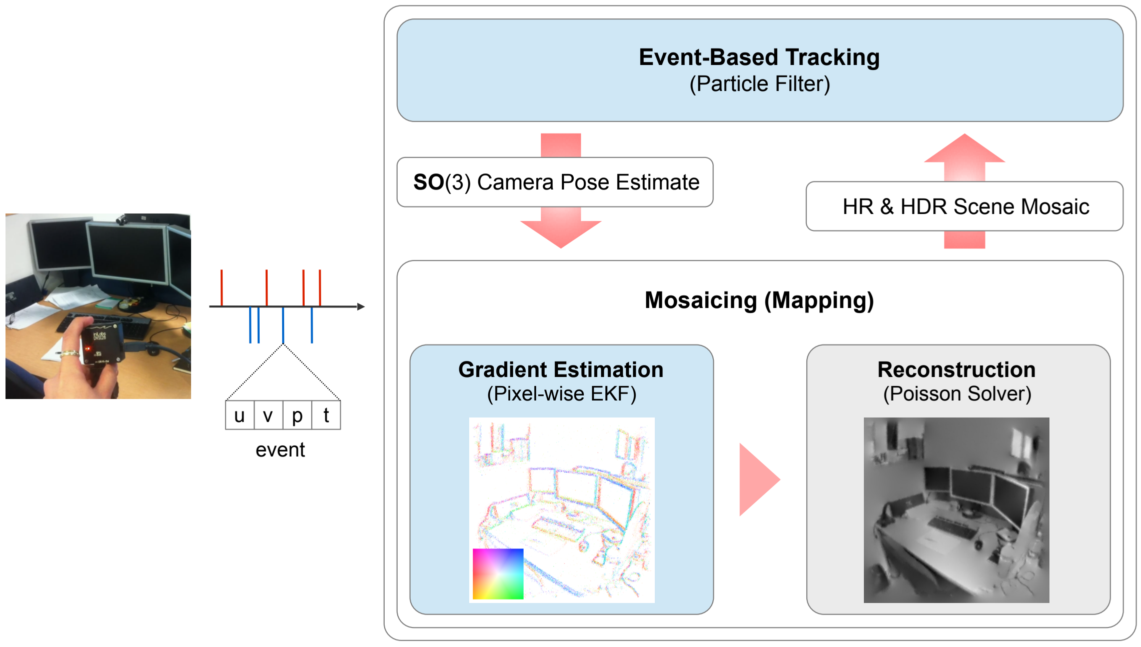


Figure 4.1: Method overview showing separation of tracking and mapping: the method takes input in the form of a stream of events from a single purely rotating event camera looking at an unknown and unstructured scene, and operates on an event-by-event basis. The core of the method is two decoupled probabilistic filters as shown in the blue rounded boxes, each estimating camera rotation motion and scene log intensity gradient. The gradient map is also in parallel upgraded into a full image-like log intensity map as shown in the grey rounded box (converted to intensity values for visualisation purposes).

we have a simple 2×2 event camera moving over a simple scene following the trajectory illustrated as the grey arrow shown in Figure 4.2 (a). We then receive a stream of positive and negative spikes from the camera; those spike events are depicted along the time axis and can be associated with a specific pixel by its colour in Figure 4.2 (b). When a new event arrives from a certain pixel, we define τ as the time elapsed since the most recent previous event from *any pixel*; and τ_c as the time since the most previous event at *the same pixel*. Here, τ is significant as the *blind time* since any previous visual information was received and is used in the motion prediction component of our tracker; while τ_c is important since its inverse serves as a local measurement of the rate of events at a particular location in image space and is used in our gradient estimation filter. In Figure 4.3 we show event time interval distributions for a typical scene and motion (the same pre-recoded event camera dataset as in Figure 4.10 (a)) represented in the form of a normalised histogram for τ and τ_c.

As illustrated in Figure 4.4 (a), the current estimate of the event camera's pose is represen-

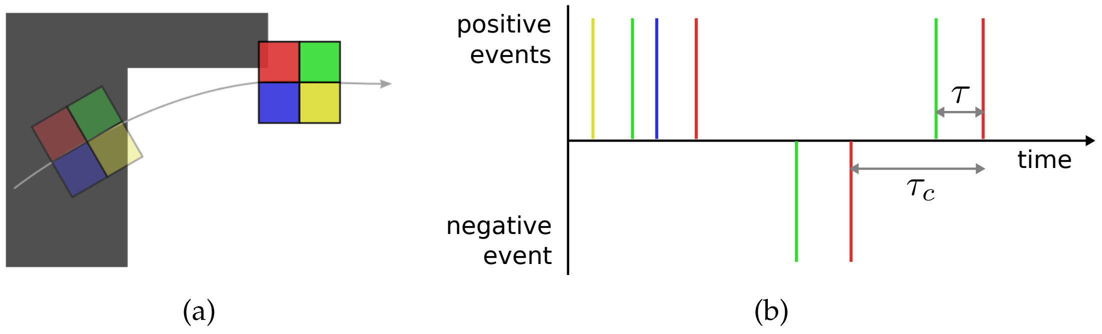


Figure 4.2: Event time intervals τ and τ_c: (a) a simplified 2×2 event camera moving over a scene following the simple trajectory illustrated as the grey arrow, with different colours to identify each pixel. (b) a stream of events generated by the camera. Upward and downward spikes represent positive and negative events respectively, and their colours visualise the pixel from which each event came. As shown, τ is the time elapsed since the previous event at *any pixel*, and τ_c is the time since the previous event at *the same pixel* — both play important roles in our camera pose and gradient estimation filters.

ted by a 3×3 orthonormal rotation matrix $\mathtt{R}_{wc}$ with respect to the world frame of reference w. When an event is received at a pixel location $(u, v)^\top$ in the event camera frame of reference c, we obtain a corresponding 3D point $\mathbf{p}_w$ on the view-sphere as:

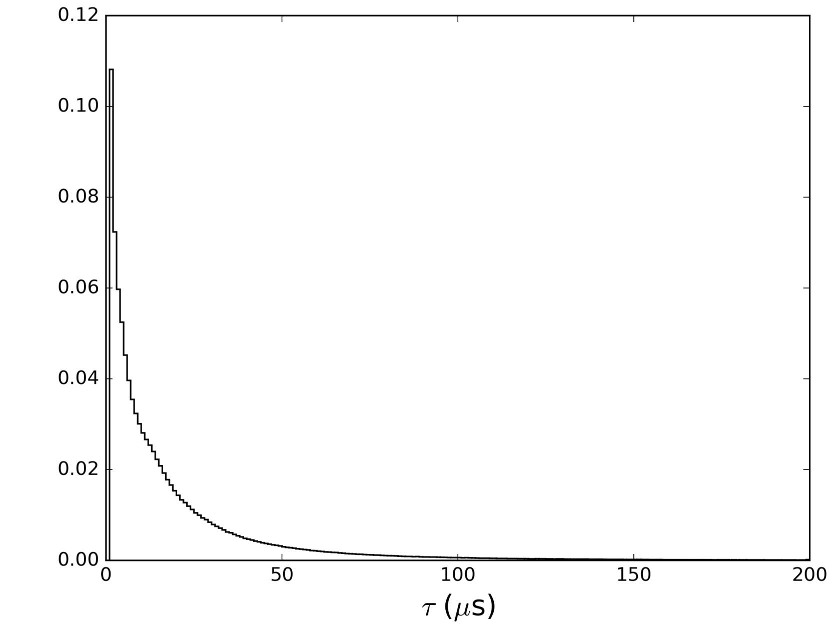
$$\mathbf{p}_w = \begin{pmatrix} p_{w1} \\ p_{w2} \\ p_{w3} \end{pmatrix} = \mathtt{R}_{wc}\mathtt{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} , \tag{4.1}$$

where $(u, v, 1)^\top$ is the camera pixel position in homogeneous coordinates and $\mathtt{K}$ is the camera intrinsics, which can be also interpreted as yaw ($\theta \in [-\pi, +\pi]$) and pitch ($\phi \in [-\frac{\pi}{2}, +\frac{\pi}{2}]$) angles in spherical coordinates [162].

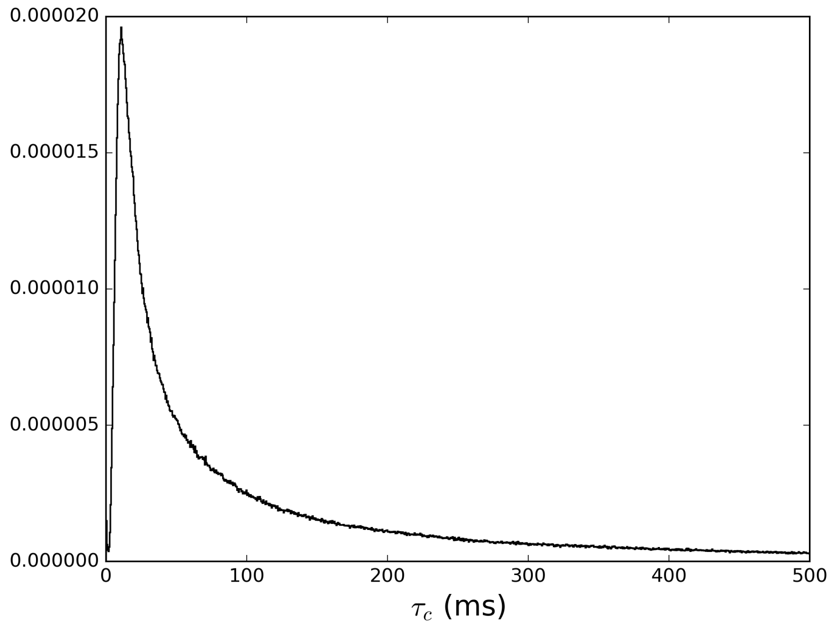
The log intensity rotational mosaic or template we aim to reconstruct is denoted $\mathtt{M}_l(\mathbf{p}_m)$ as shown in Figure 4.4 (b), and has its own fixed 2D coordinate frame with pixel position vector $\mathbf{p}_m$ as in [163]:

$$\mathbf{p}_m = \begin{pmatrix} \theta \\ \phi \end{pmatrix} = \begin{pmatrix} \tan^{-1}\left(p_{w1}/p_{w3}\right) \\ \tan^{-1}\left(p_{w2}/\sqrt{p_{w1}^2 + p_{w3}^2}\right) \end{pmatrix} . \tag{4.2}$$

To reconstruct super resolution scenes by harnessing the very high speed measurement

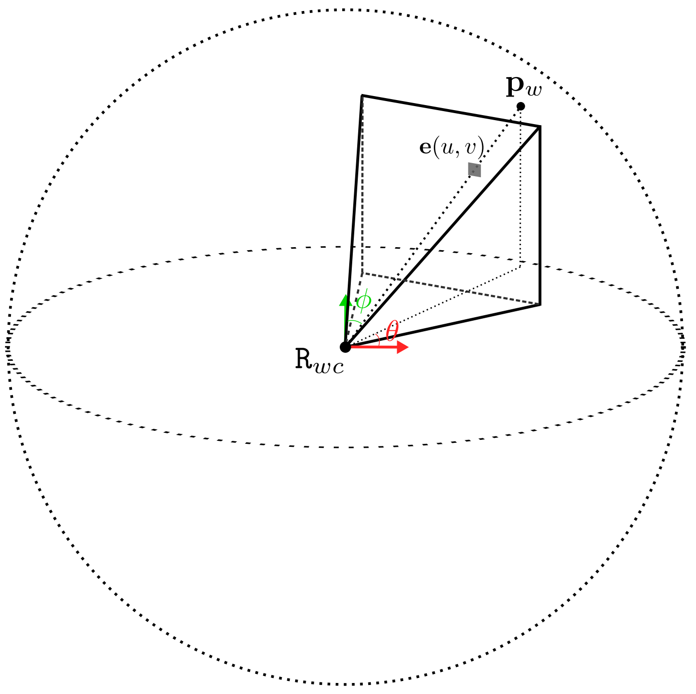


(a)

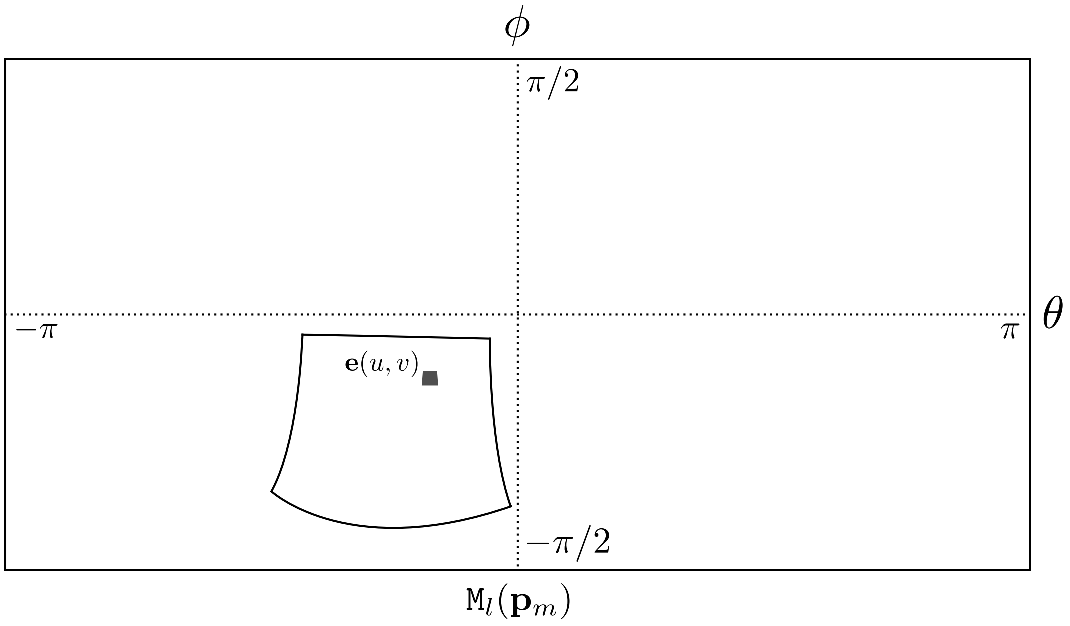


(b)

Figure 4.3: Typical event time interval distributions represented in the form of a normalised histogram for τ and τ_c as shown in (a) and (b) respectively.



(a)



(b)

Figure 4.4: Basic geometry: (a) the current estimate of the event camera's pose is represented by a 3×3 orthonormal rotation matrix $\mathtt{R}_{wc}$ with respect to the world frame of reference w, and when an event $\mathbf{e}(u,v)$ is received, we obtain a corresponding 3D point $\mathbf{p}_w$ on a view-sphere which can be also interpreted as yaw θ and pitch ϕ angles in spherical coordinates. (b) the log intensity rotational mosaic we aim to reconstruct is denoted $\mathtt{M}_l(\mathbf{p}_m)$, and has its own fixed 2D coordinate frame with pixel position vector $\mathbf{p}_m = (\theta,\phi)^\top$.

property of the event camera which enables sub-pixel accurate camera tracking, we use a higher resolution for the panoramic mosaic than for the low resolution sensor used in the experiments (shown in Table 4.1).

4.3 Event Camera Pure Rotation Tracking

We first explain the tracking component of our algorithm, whose job is to provide an event-by-event updated estimate of the rotational location of the camera with respect to the scene mosaic (assumed correct). The basic idea is, as soon as a new event arrives, to find current camera pose $\mathtt{R}_{wc}$ most consistent with the predicted log intensity change since the previous event at the same pixel.

We have chosen a particle filter as a straightforward sequential Bayesian way to estimate the rotation motion of our camera over time with the multi-hypothesis capability to cope with the sometimes noisy event stream. In our event-based particle filter, the posterior density function at time t is represented by N particles, $\{\mathbf{p}_1^{(t)}, \mathbf{p}_2^{(t)}, ..., \mathbf{p}_N^{(t)}\}$. Each particle $\mathbf{p}_i^{(t)}$ is a set consisting of a hypothesis of the current state $\mathtt{R}_i^{(t)} \in \mathbf{SO}(3)$ and a normalised weight $w_i^{(t)}$. Initially, all particles are set to the same nominal value with the same weight i.e. $\frac{1}{N}$.

At every new event arrives, the particles evolve in the standard particle filter prediction and measurement update steps as follows.

4.3.1 Motion Prediction

For motion prediction in our particle filter, we use a constant position (random walk) motion model where the predicted mean rotation of a particle at any given time remains constant while the variance of the prediction is proportional to the time interval as shown in Figure 4.5.

Specifically, we perturb the current $\mathfrak{so}(3)$ vector on the tangent plane with Gaussian noise independently in all three axes and reproject it onto the $\mathbf{SO}(3)$ unit sphere to obtain the corresponding predicted mean rotation. The noise is the predicted change the current rotation might have undergone since the previous event was generated. This is further simplified by the composition property of rotation matrices and yields the final update at the current time

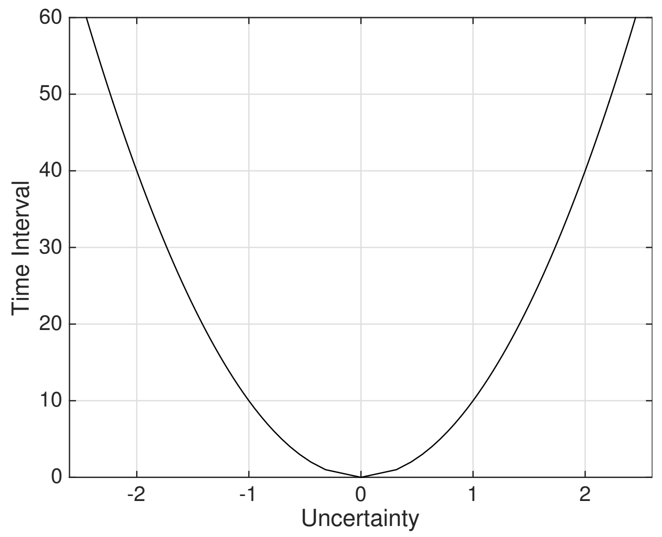


Figure 4.5: By modelling random walk in our motion model, the variance of the motion prediction is proportional to the time interval i.e. $\sigma_i^2\tau$.

t via the matrix exponential map [155]:

$$\mathtt{R}_i^{(t)} = \mathtt{R}_i^{(t-\tau)} \exp\left(\sum_{k=1}^{3} \mathbf{n}_k \mathtt{G}_k\right) , \tag{4.3}$$

where $\mathtt{R}_i$ is a 3×3 orthonormal rotation matrix for the i^{th} particle, the noise vector $\mathbf{n} = (n_1, n_2, n_3)^\top$ is obtained by generating random numbers sampled from Gaussian distributions independently in all three directions i.e. $n_i \sim \mathcal{N}(0, \sigma_i^2\tau)$, and $\mathtt{G}_k$ are the Lie group generators for $\mathbf{SO}(3)$ as follows:

$$\mathtt{G}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} , \quad \mathtt{G}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} , \quad \mathtt{G}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} . \tag{4.4}$$

4.3.2 Measurement Update

The weights of these perturbed particles are now updated through the measurement update step which applies Bayes rule to each particle:

$$w_i^{(t)} = P(z|\mathtt{R}_i^{(t)}) w_i^{(t-\tau)} . \tag{4.5}$$

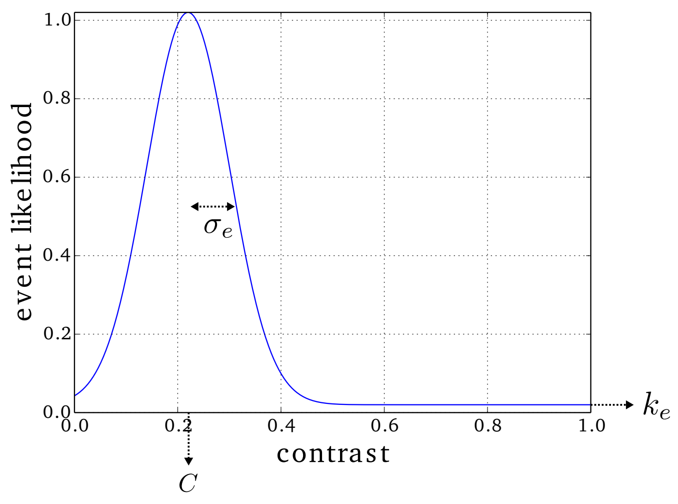


Figure 4.6: Mexican hat shaped event likelihood function of absolute log intensity difference (contrast) with its mean aligned to a known event contrast C, which highly likely generates an event [96], an appropriate standard deviation σ_e, and the minimum constant k_e. All the parameters used in the experiments are given in Table 4.1.

We calculate the value of a 'measurement' z given an event $\mathbf{e}(u, v)$, the current state $\mathtt{R}_i^{(t)}$ and the previous state $\mathtt{R}_i^{(t-\tau_c)}$ by taking the log intensity difference between the corresponding log intensity map positions:

$$z = \mathtt{M}_l(\mathbf{p}_m^{(t)}) - \mathtt{M}_l(\mathbf{p}_m^{(t-\tau_c)}) \ , \tag{4.6}$$

where $\mathbf{p}_m^{(t)}$ and $\mathbf{p}_m^{(t-\tau_c)}$ can be obtained as in Equation (4.1) and (4.2). The measurement z is now used to calculate the likelihood $P(z|\mathtt{R}_i^{(t)})$ for each particle, essentially asking 'how likely was this event relative to our mosaic given a particular hypothesis of camera pose?'. We first compare the sign of the log intensity difference with the polarity of the event, and we give a particle a fixed low likelihood P_{min} if the signs do not agree. Otherwise, we look up the likelihood of this absolute log intensity difference (contrast) in the Mexican hat shaped curve shown in Figure 4.6, with its mean aligned to a known contrast C, which highly likely generates an event [96], an appropriate standard deviation σ_e, and the minimum constant k_e. All the parameters used in the experiments are given in Table 4.1.

After updating all the weights of the particles and normalising them i.e. $w_i = \frac{w_i}{\sum_1^N w_i}$, we resample the particle distribution in the standard way — making a new particle set which copies old particles with probability according to their new weights [51, 34]. However, due to the very high frequency of events, we do not resample at every iteration to avoid unnecessarily deleting good particles in cases where all weights are similar. Instead, we follow a selective resampling scheme [52] to determine whether the resampling should be carried out depending on the so-called effective number of particles N_{eff}:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_i^{(t)})^2}\,, \tag{4.7}$$

and we only resample the set of particles whenever N_{eff} is less than $N/2$.

Lastly, for the following steps and iterations, a mean pose is saved for the event pixel. To calculate the mean of particles, we apply the matrix logarithm [155] to all particles' $\mathbf{SO}(3)$ components to map them to the tangent space, e.g. $\mathbf{r}_i = \log(\mathtt{R}_i)$ and $\mathbf{r}_i \in \mathbb{R}^3$, calculate the arithmetic mean, and re-map to the $\mathbf{SO}(3)$ group's manifold by applying the matrix exponential. However, because of the random walk nature of our motion model which generates a noisy particle mean pose $\tilde{\mathbf{r}}$, a new mean pose $\bar{\mathbf{r}}^{(t)}$ is calculated in a form of a weighted average with the previous mean pose $\bar{\mathbf{r}}^{(t-\tau_c)}$ as:

$$\bar{\mathbf{r}}^{(t)} = (1-\gamma)\bar{\mathbf{r}}^{(t-\tau_c)} + \gamma\tilde{\mathbf{r}}, \tag{4.8}$$

where γ is a weight variable which controls the smoothness of estimated camera motion. In fact, a small γ value implicitly implements a strong accumulation of events, but our method still provides an event-by-event updated estimate of the rotational location of the event camera as they arrive.

4.4 Spherical Mosaic Reconstruction

We now turn to the other main part of our algorithm, which having received an updated camera pose estimate from tracking must incrementally improving our estimate of the log intensity mosaic. This takes two steps; pixel-wise incremental EKF estimation of the log intensity gradient at each template pixel, and interleaved Poisson reconstruction to recover absolute log intensity.

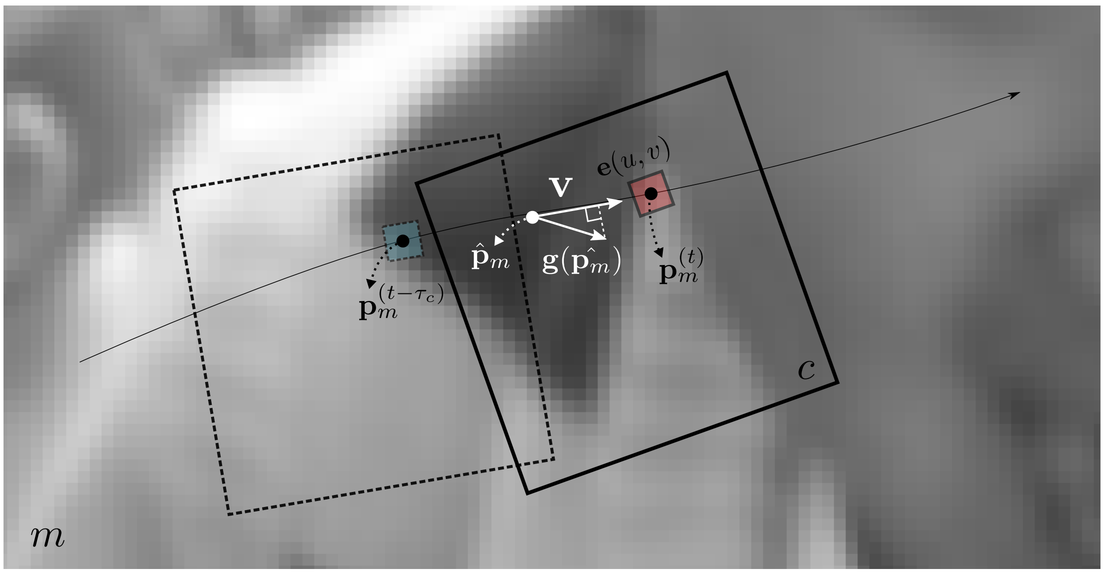


Figure 4.7: Pixel-wise EKF-based gradient estimation overview: for each incoming new event $\mathbf{e}(u, v)$ in the current camera frame of reference c, we obtain its corresponding map location $\mathbf{p}_m^{(t)}$ and know the previous one $\mathbf{p}_m^{(t-\tau_c)}$ in the map frame of reference m based on the assumption that the camera pose estimates are correct. We then find a displacement vector between them to calculate a motion vector $\mathbf{v}$ which is used to compute the value of a measurement $(\mathbf{g}(\hat{\mathbf{p}}_m) \cdot \mathbf{v})/C$ at the midpoint $\hat{\mathbf{p}}_m$ based on the brightness constancy and the linear gradient assumptions. We then update the gradient estimate using a pixel-wise EKF framework. Essentially, each new event which lines up with a particular map pixel improves our estimate of its gradient in the direction parallel to the motion of the camera over the scene at that pixel while we learn nothing about the gradient in the direction perpendicular to camera motion.

4.4.1 Pixel-Wise EKF-Based Gradient Estimation

We receive an event $\mathbf{e}(u, v)$ at a pixel location $(u, v)^\top$ in the event camera frame of reference c and, using our tracking algorithm as described in Section 4.3, we find the corresponding map location $\mathbf{p}_m^{(t)}$ and know the previous one $\mathbf{p}_m^{(t-\tau_c)}$ in the mosaic frame of reference m as shown in Figure 4.7. Each pixel of the gradient map has an independent gradient estimate $\mathbf{g}(\mathbf{p}_m) = (g_x, g_y)^\top$, consisting of log intensity gradients g_x and g_y along the horizontal and vertical axes in image space respectively, and a 2×2 uncertainty covariance matrix $\mathtt{P}_{\mathbf{g}}(\mathbf{p}_m)$. At initialisation, all estimated gradients are initialised to zero vectors with large variances such as $\mathbf{g}_0$ and $\mathtt{P}_{\mathbf{g}_0}$, used in the experiments shown in Table 4.1.

Now, we want to improve the gradient estimate at a point in the mosaic based on a new incoming event and a tracking result using the pixel-wise EKF. We first find motion vector

$\mathbf{v}$ which is the displacement vector between $\mathbf{p}_m^{(t)}$ and $\mathbf{p}_m^{(t-\tau_c)}$ divided by the elapsed time τ_c:

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \frac{\mathbf{p}_m^{(t)} - \mathbf{p}_m^{(t-\tau_c)}}{\tau_c} \; . \tag{4.9}$$

Assuming, based on the rapidity of events, that the gradient in the template and the camera velocity can be considered locally constant, we update the midpoint $\hat{\mathbf{p}}_m$ of the two points $\mathbf{p}_m^{(t)}$ and $\mathbf{p}_m^{(t-\tau_c)}$. We now say $(\mathbf{g}(\hat{\mathbf{p}}_m) \cdot \mathbf{v})\tau_c$ is the amount of log grey level change that has happened since the last event. Therefore, if we have an event camera where log intensity change C should trigger an event, the brightness constancy constraint [76] tells us that:

$$\left(\mathbf{g}(\hat{\mathbf{p}}_m) \cdot \mathbf{v}\right) \tau_c = \pm C \; , \tag{4.10}$$

where the sign of C depends on the polarity of an event (i.e. $+C$ for a positive event, and $-C$ for a negative one). We now define z, a measurement of the instantaneous *event rate* at this pixel, and its measurement model h, as follows:

$$z = \frac{1}{\tau_c} \; , \tag{4.11}$$

$$h = \frac{\mathbf{g}(\hat{\mathbf{p}}_m) \cdot \mathbf{v}}{C} \; . \tag{4.12}$$

In the EKF framework, the gradient estimate and the uncertainty covariance matrix are updated using the standard equations at every event:

$$\mathbf{g}(\hat{\mathbf{p}}_m)^{(t)} = \mathbf{g}(\hat{\mathbf{p}}_m)^{(t-\tau_c)} + \mathbf{W}\nu \; , \tag{4.13}$$

$$\mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_m)^{(t)} = \mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_m)^{(t-\tau_c)} - \mathbf{W}\mathtt{S}\mathbf{W}^\top \; , \tag{4.14}$$

where the Kalman gain $\mathbf{W}$ is:

$$\mathbf{W} = \mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_m)^{(t-\tau_c)} \frac{\partial h}{\partial \mathbf{g}(\hat{\mathbf{p}}_m)^{(t-\tau_c)}}^\top \mathtt{S}^{-1} \; , \tag{4.15}$$

the innovation ν is:

$$\nu = z - h \,, \tag{4.16}$$

and the innovation covariance $\mathtt{S}$ is:

$$\mathtt{S} = \frac{\partial h}{\partial \mathbf{g}(\hat{\mathbf{p}}_m)^{(t-\tau_c)}} \mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_m)^{(t-\tau_c)} \frac{\partial h}{\partial \mathbf{g}(\hat{\mathbf{p}}_m)^{(t-\tau_c)}}^{\top} + \mathtt{R} \,, \tag{4.17}$$

where $\mathtt{R}$ is the measurement noise, in our case scalar σ_m^2. Finally, Jacobian $\frac{\partial h}{\partial \mathbf{g}}$ (simplified notation for clarity) is derived as:

$$\frac{\partial h}{\partial \mathbf{g}} = \frac{\partial}{\partial \mathbf{g}} \left(\frac{\mathbf{g} \cdot \mathbf{v}}{C} \right) \tag{4.18}$$

$$\tag{4.19}$$

$$= \left(\frac{\partial}{\partial g_x} \left(\frac{g_x v_x + g_y v_y}{C} \right) \quad \frac{\partial}{\partial g_y} \left(\frac{g_x v_x + g_y v_y}{C} \right) \right) \tag{4.20}$$

$$\tag{4.21}$$

$$= \left(\frac{v_x}{C} \quad \frac{v_y}{C} \right) \,. \tag{4.22}$$

Essentially, each new event which lines up with a particular template pixel improves our estimate of its gradient in the direction parallel to the motion of the camera over the scene at that pixel while we learn nothing about the gradient in the direction perpendicular to camera motion. We visualise an estimated gradient map in Figure 4.8; the colours and intensities of the figure represent the orientations and strengths of the gradients of the scene respectively — refer to the overlaid colour chart.

4.4.2 Reconstruction from Gradients

The estimated log intensity gradient map as shown in Figure 4.8 is now upgraded into a full image-like log intensity map as shown in Figure 4.9, from which we calculate the value of a measurement for the camera pose estimation, with super-resolution and high dynamic range properties. This reconstruction module does not run at the event rate but instead runs in a separate thread as often as computing resources allow.

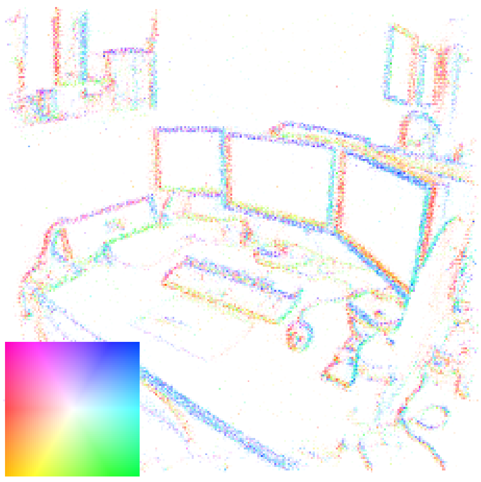


Figure 4.8: Estimated gradient map — the colours and intensities of this figure represent the orientations and strengths of the gradients of the scene respectively (refer to the overlaid colour chart).

Reconstructing a 1D scalar function from its derivatives is a matter of simple integration up to an additive constant, but in the 2D case we cannot simply obtain an image by integrating its gradients since it is not necessarily integrable. In fact, the gradient vector field should have zero curl to be integrable — i.e. it must be *a conservative vector field* meaning that the integral along any closed path should be equal to zero [72]. In practice, the estimated gradient map is rarely integrable due to the various sources of noise in the sensing and estimation processes — i.e. the integral of such a gradient vector field depends on the chosen path.

As one possible solution to this problem, we perform a reconstruction from gradients by solving a Poisson equation method [58] which has proven very useful for many applications such as high dynamic range tone mapping [58], removing shadows from images [59], novel image editing tools [131], surface reconstruction [2, 3], and even the novel concept of a gradient camera [166]. Specifically, we wish to reconstruct a log intensity mosaic $\mathtt{M}_l(\mathbf{p}_m)$ whose gradients across the whole image domain are close to the estimated gradients $\mathbf{g}(\mathbf{p}_m) = (g_x, g_y)^\top$ in a least squares sense; in other words, $\mathtt{M}_l$ should minimise:

$$\int\int J(\nabla \mathtt{M}_l, \mathbf{g}) dxdy \; , \tag{4.23}$$

where:

$$J(\nabla \mathtt{M}_l, \mathbf{g}) = \|\nabla M_l - \mathbf{g}\|^2 = \left(\frac{\partial M_l}{\partial x} - g_x\right)^2 + \left(\frac{\partial M_l}{\partial y} - g_y\right)^2 . \tag{4.24}$$

Based on the Variational Principle, $\mathtt{M}_l$ that minimises Equation (4.23) must satisfy the Euler-Lagrange equation:

$$\frac{\partial J}{\partial \mathtt{M}_l} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial J}{\partial M_{lx}} - \frac{\mathrm{d}}{\mathrm{d}y}\frac{\partial J}{\partial M_{ly}} = 0 , \tag{4.25}$$

which is a partial differential equation in $\mathtt{M}_l$. Substituting J with Equation (4.24) we obtain:

$$2\left(\frac{\partial^2 \mathtt{M}_l}{\partial x^2} - \frac{\partial g_x}{\partial x}\right) + 2\left(\frac{\partial^2 \mathtt{M}_l}{\partial y^2} - \frac{\partial g_y}{\partial y}\right) = 0 . \tag{4.26}$$

Dividing by 2 and rearranging the equation, we obtain:

$$\frac{\partial^2 \mathtt{M}_l}{\partial x^2} + \frac{\partial^2 \mathtt{M}_l}{\partial y^2} = \frac{\partial g_x}{\partial x} + \frac{\partial g_y}{\partial y} , \tag{4.27}$$

which is the well known Poisson equation:

$$\nabla^2 \mathtt{M}_l = \mathrm{div}\mathbf{g} , \tag{4.28}$$

where ∇^2 is the Laplacian operator and $\mathrm{div}\mathbf{g}$ is the divergence of the vector field $\mathbf{g}$.

To solve Equation (4.28), we use a sine transform based method with Dirichlet boundary conditions assuming the log intensity values of the boundary are known (initially, they are set to nominal values) [137]. In bootstrapping, we initialise all log intensity mosaic pixels to log(128), and Figure 4.9 shows a reconstructed intensity map using the method described in this section.

4.5 Evaluation and Results

We recommend readers to view our video[1] which illustrates all of the key results below in a form better than still pictures (also see Appendix A). We have conducted spherical mosa-

[1]Simultaneous Mosaicing and Tracking with an Event Camera Video: `https://youtu.be/l6qxeM1DbXU` (accessed September 2017)

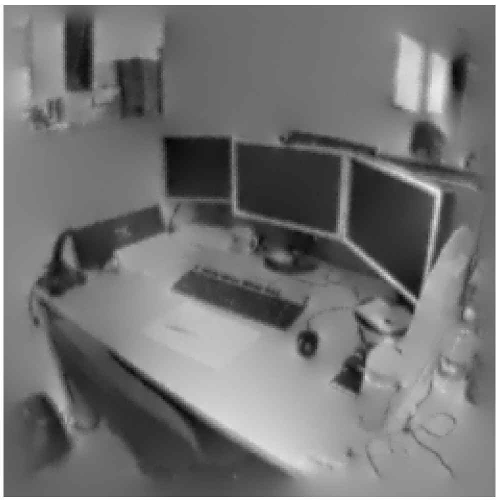


Figure 4.9: Reconstructed log intensity map from the gradient map shown in Figure 4.8 — converted to intensity values for visualisation purposes.

icing in both indoor and outdoor scenes. Also, we show the potential for reconstructing high resolution and high dynamic range scenes from very small camera motion. Our algorithm runs in real-time on a standard PC for a low number of particles and low template resolutions; the results we show here were generated using high number of particles at higher resolution and are currently off-line. We present a real-time version of this by adopting faster algorithms in Chapter 5.

In all experiments, we have used the DVS camera from iniLabs[2] with 128×128 resolution, 120 dB dynamic range, and 15 microsecond latency, and communicated with a host computer using our own USB 2.0 driver. The camera has pre-calibrated intrinsics and all event pixel locations are pre-warped to remove radial distortion via the event camera calibration method described in Section 3.6. We run both on a standard desktop PC with an Intel Xeon W5590 3.33GHz quad-core CPU, and an Apple MacBook Pro with an Intel i7 2.6Ghz dual-core CPU.

We present all the parameters used in the experiments in Table 4.1. As shown, we have used the same values for most of the parameters across a wide range of experimental environments, except the ones related to the motion estimation which are sensitive to different motion speeds.

[2]iniLabs Ltd: `www.inilabs.com` (accessed September 2017)

Table 4.1: Parameters used in the experiments

Parameter	Value	Reference
map size	2304×1152	Section 4.2
# of particles N	100	Section 4.3
$\sigma_1 \sim \sigma_3$	$2.3 \times 10^{-8} \sim 7.0 \times 10^{-5}$	Section 4.3.1
P_{min}	1.0×10^{-6}	Section 4.3.2
C	0.22	Section 4.3.2
σ_e	8.0×10^{-2}	Section 4.3.2
k_e	1.0×10^{-3}	Section 4.3.2
γ	2.0×10^{-3}	Equation (4.8)
$\mathbf{g}_0$	$\mathbf{0}_{2\times 1}$	Section 4.4.1
$\mathrm{P}_{\mathbf{g}_0}$	$\begin{pmatrix} 3.0 \times 10^1 & 0 \\ 0 & 3.0 \times 10^1 \end{pmatrix}$	Section 4.4.1
σ_m^2	2.5×10^3	Section 4.4.1
$M_{l0}(\cdot)$	$\log(128)$	Section 4.4.2

4.5.1 Spherical Mosaicing

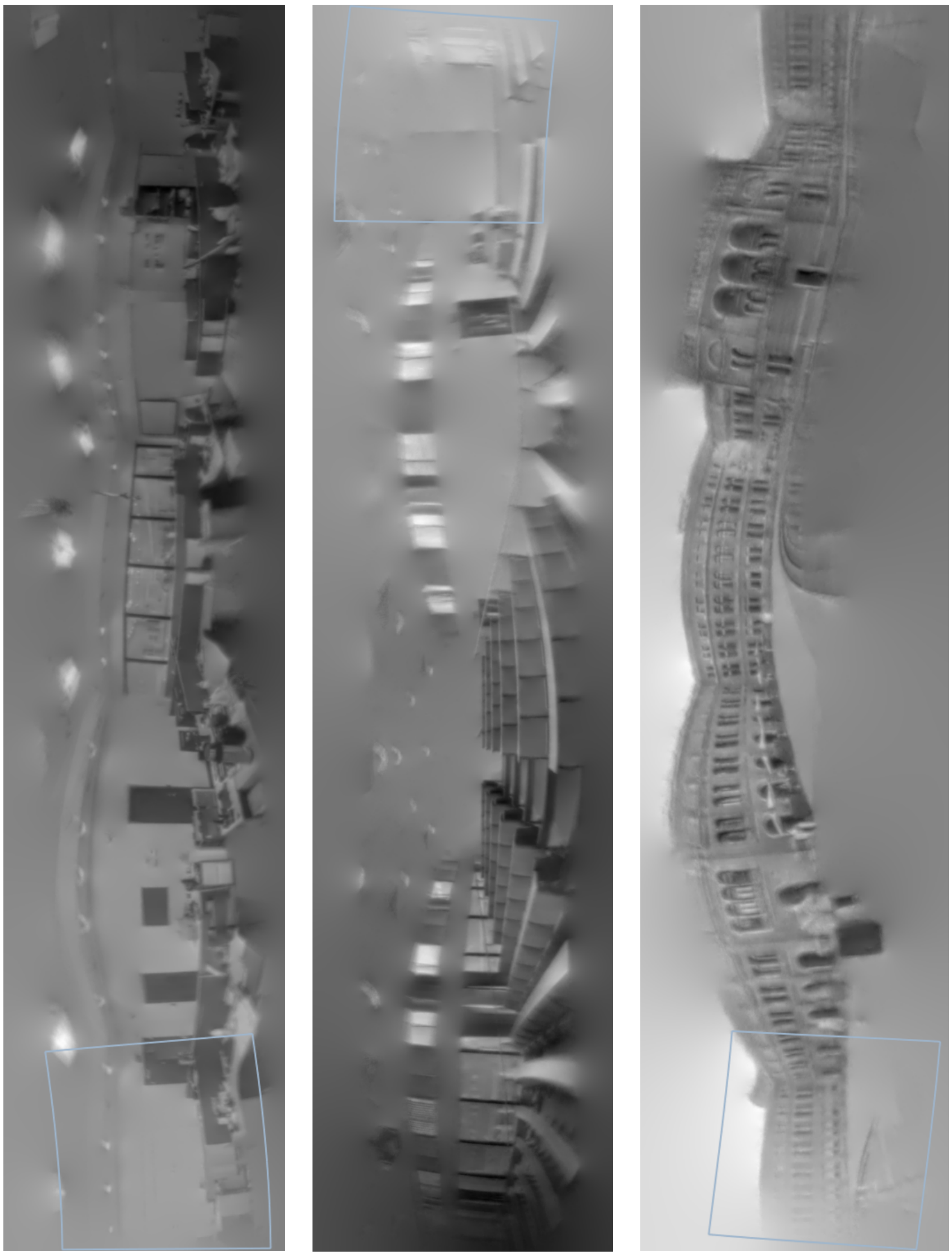
As shown in Figure 4.10, our algorithm is able to reconstruct indoor and outdoor scenes with super resolution and high dynamic range properties. In these mosaics, the overlaid box represents the current tracked field of view of the event camera.

4.5.2 High Resolution Reconstruction

Even though current event cameras have very low resolution (the DVS has a 128×128 pixel array), as they provide very fast visual measurements we can reconstruct high resolution scenes since our algorithm tracks rotation at sub-pixel accuracy. In Figure 4.11 we compare (a) an image from a standard camera down-sampled to 128×128 resolution with (b) our reconstructed super resolution result, showing sharper details.

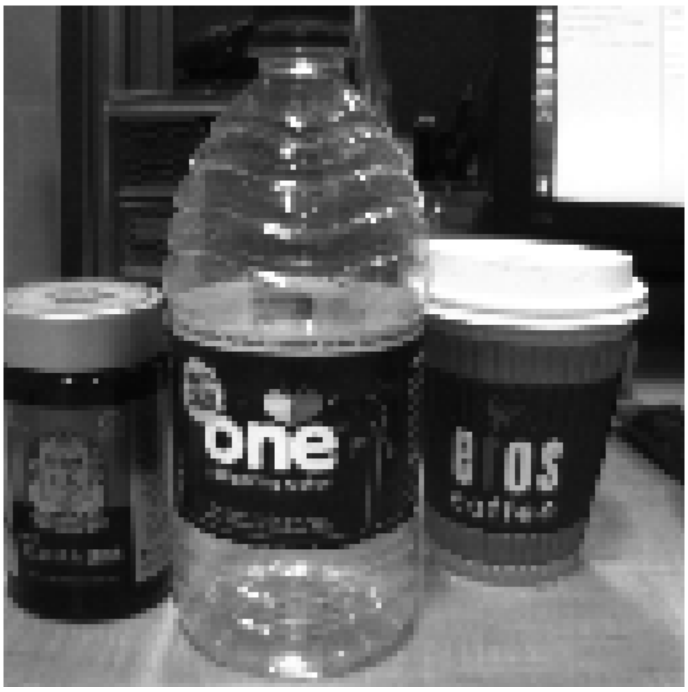
4.5.3 High Dynamic Range Reconstruction

Another key characteristic of the event camera is its sensitivity over a very high dynamic range (e.g. 120dB for DVS). Our algorithm can build mosaics which make use of this range, to deal with scenes where there are large intensity difference between the brightest and darkest parts. We created a scene with a very high range of light intensity by placing a row of bright LED lights on top of a poorly lit sketch pad. A standard global shutter camera

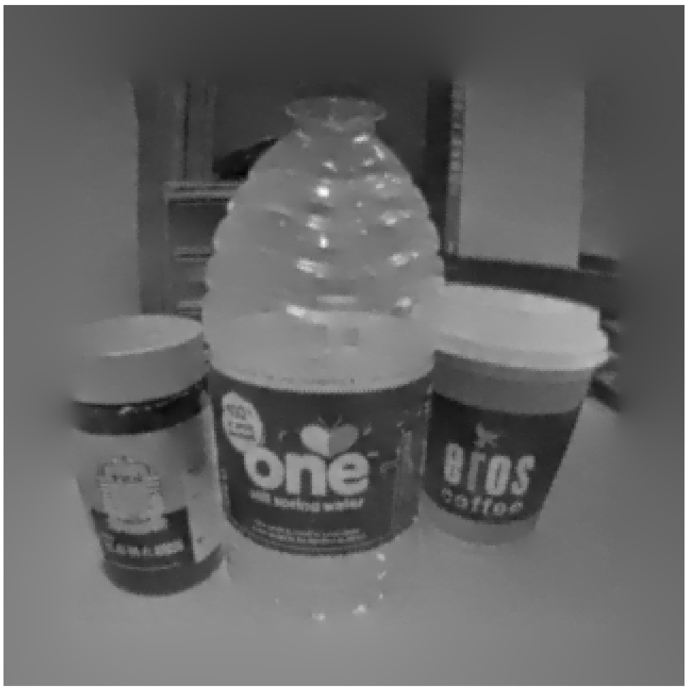


(a) (b) (c)

Figure 4.10: Reconstructed spherical mosaics for indoor (shown in (a) and (b)) and outdoor (shown in (c)) scenes with super resolution and high dynamic range properties — converted to intensity values for visualisation purposes. The overlaid boxes represent the current field of view of the event camera being tracked.



(a)



(b)

Figure 4.11: High resolution reconstruction: (a) a 128×128 down sampled normal camera image for a comparison; (b) a reconstructed high resolution scene by harnessing the very high speed measurement property of the event camera which enables sub-pixel accurate camera tracking — converted to intensity values for visualisation purposes.

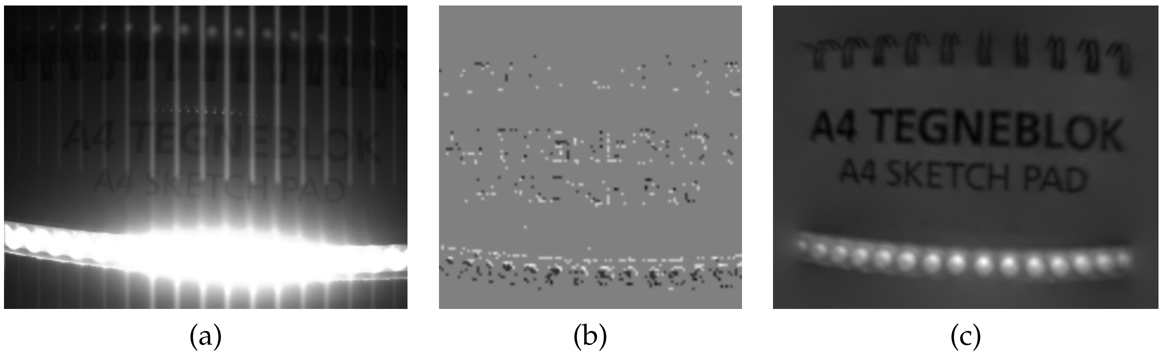


Figure 4.12: High dynamic range reconstruction: (a) a saturated normal CCD camera image with the smear effect for a comparison; (b) a visualisation of a stream of events from the DVS camera showing the decent number of events being generated from both the bright and dark areas; (c) a reconstructed high dynamic range scene — converted to intensity values for visualisation purposes.

generates an image which is partly saturated, partly very dark and also has smearing effects (Figure 4.12 (a)). However, the event camera and our algorithm are able to reconstruct the high dynamic range log intensity image in Figure 4.12 (c) where all elements are clear.

4.6 Discussion and Summary

In this chapter, we have presented breakthrough results, showing how joint sequential and global estimation permits the great benefits of an event camera to be applied to the real problem of mosaicing. The proposed method takes input in the form of a stream of events from a single pure rotation event camera looking at an unknown and unstructured natural scene, and operates on an event-by-event basis to maximise the update rate with low latency – Equation (4.8) with a small γ value implicitly implements a strong accumulation of events, but still provides an event-by-event updates. Our approach relies on two parallel probabilistic filters to jointly track the global rotational motion of a camera and estimate the log intensity gradients of the scene around it. The gradient map is then upgraded to a full image-like mosaic with super-resolution and high dynamic range properties.

Chapter 5

Real-Time Mosaicing and 3-DoF Tracking with an Event Camera

Contents

5.1 Introduction

The method described in Chapter 4 presents breakthrough results showing how joint sequential and global estimation permits the great benefits of an event camera to be applied to a real problem of mosaicing and camera pose tracking. However, its practicality is somewhat limited because of the high computational complexity of the particle filter used in tracking and the Poisson solver-based log intensity reconstruction running on a CPU. As addressed in the previous chapter, it runs in real-time on a standard PC for a low number of particles and low template resolutions, but to obtain the high quality of the results presented in Section 4.5, a high number of particles and a high resolution template are required and it is, therefore, currently off-line. Moreover, the relatively slow update rate of log intensity reconstruction tends to cause poor tracking, then data association errors, and finally corruption of the other parts of the estimation process. This is because the interaction of these estimation processes and reconstruction is key as the approach of interleaved probabilistic filters and separated log intensity reconstruction makes many assumptions about independence which are certainly an approximation.

Bearing the above in mind, in this chapter, we first discuss the real-time processing requirements for event cameras by analysing measured event rates in different experimental conditions and the current processing time of the method described in Chapter 4. We then present a new real-time implementation of simultaneous mosaicing and 3-DoF camera pose estimation whose overall structure and functionality is the same as the previous method, but where a substantial speed up has been achieved via adapting a computationally efficient estimation method for tracking as well as parallelisable log intensity reconstruction running on a GPU. The experimental results in Section 5.4 show that this speed-up also increases the quality of estimation and reconstruction as it guarantees higher fidelity of the independence assumption.

5.2 Real-Time Processing Requirements for Event Cameras

When we use conventional cameras operating at the standard frame rate (e.g. 25-60Hz), we normally have a time budget within the range of 16 to 40ms to process each incoming frame to run any computer vision method in real-time. However, one of the superior characteristics of event cameras, the high measurement rate, comes at a price — a highly limited time budget for real-time processing. In the standard DVS128 device used in our work case, for instance, we need to process each event within 1μs in the worst scenario as its maximum event rate is one million eps. Newer event cameras provide even higher maximum bandwidth such as 12Meps from the DAVIS sensor [19]. Of course, we can process a bundle of events accumulated within a small time window at once or only selected events based on a certain filtering criterion to mitigate the greatly restricted real-time processing time budget per event. In our work, however, we aim to process all of the incoming events on an event-by-event basis to minimise latency and to maximise the generality of our methods.

5.2.1 Current and Future Processors

Despite the high speed processing capabilities of today's hardware and the small information content of each event, performing meaningful tasks per event within such a limited time initially appears infeasible. This is partly due to the limitation of the traditional von Neumann computing architecture[1], which separates its processing and memory components, and we expect special hardware become available in the near future. For instance, neuromorphic processors or graph processors placed directly behind and connected in parallel to the pixels of event sensors, and incoming events wake up and activate local computation and message passing, while the majority of the processor remains in a sleep state to conserve power. However, although significant progress have been achieved recently in the related hardware research and development such as IBM's TrueNorth [111], Qualcomm's Zeroth[2], Graphcore's IPU[3], and CPA [9, 23, 98], SCAMP [22] and SpiNNaker [63] from the University of Manchester, the challenges are to map estimation methods into message passing (belief propagation) algorithms and to obtain globally consistent estimates of non-local parameters such as ego-motion, and they still remain as subjects for future research. Therefore, for now, we only consider serial processing of events on a standard PC.

In practice, however, saturating an event camera to the maximum event rate is relatively rare as it requires extremely rapid motion while looking at densely textured scenes. In the next section, we examine the typical event rates measured under different experimental conditions to determine the *practical* maximum event rate and real-time processing time budget for the event camera used in our work.

5.2.2 Typical Event Rates

In Table 5.1, we present the measured maximum and average event rates (i.e. eps: events-per-second) from twenty different datasets which were captured using a standard DVS128 device with fixed bias settings, and we visualise them in the form of accumulated events as shown in Figure 5.1. The presented datasets include different motion types: pure rotation,

[1]Von Neumann architecture: https://en.wikipedia.org/wiki/Von_Neumann_architecture (accessed September 2017)

[2]Zeroth (software): https://en.wikipedia.org/wiki/Zeroth_(software) (accessed September 2017)

[3]Graph computing for machine intelligence with Poplar: https://www.graphcore.ai/posts/graph-computing-for-machine-intelligence-with-poplar (accessed September 2017)

hand shaking, typical 3D motion, walking, moving while looking down at the ground, and driving in a road vehicle at different speeds; and different indoor and outdoor scenes with different proximity to objects under various lighting conditions. Although these are just several examples and different event rates can be obtained with different bias settings, we believe that they are representative enough of most scenarios that our methods aim to cover.

As shown in the table, besides a few extreme cases such as datasets 4 with rapid rotating motion, 8 with rapid shaking motion, and 20 with an artificially textured spinning object at high speed, none of the measured maximum event rates exceed 400keps. Moreover, unrealistic dataset 20 is the only one that exceeds the 1Meps maximum bandwidth of the DVS128 event camera, indicating that saturating an event camera to the maximum event rate is relatively rare. We therefore safely set the *practical* maximum event rate to 400keps, and in our serial processing aim to process each event within 3μs to achieve real-time processing in the following methods.

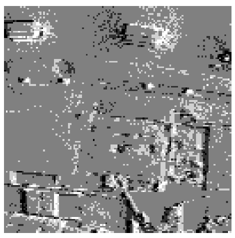
5.2.3 Processing Time of the Method in Chapter 4

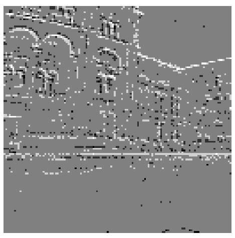
Before we present a real-time implementation of simultaneous mosaicing and 3-DoF event camera pose estimation which can run in real-time on a standard PC up to the *practical* maximum event rate 400keps in Section 5.3, we first investigate the current processing time of the method described in Chapter 4. As mentioned, the previous method works in *near* real-time on a standard PC with a small number of particles and a low log intensity map resolution due to the very high computational cost of the tracking and reconstruction components running on a CPU. The high quality results shown in Section 4.5, however, were generated using a high number of particles at high template resolution and were unable to be processed in real-time.

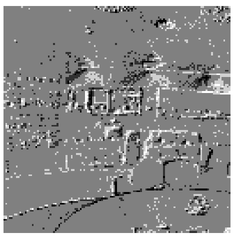
Figure 5.2 (a) and (b) show the current processing time required for the estimation and reconstruction components with respect to different numbers of particles and map resolutions respectively — measured by a time measurement software function running on a 3.33GHz quad-core computer, and ignoring its overhead. As shown, even with as few as 10 particles, the current processing time of particle filter based tracking and pixel-wise EKF based gradient estimation barely meets the 3μs real-time processing requirement, and the Poisson solver-based log intensity reconstruction method shows a very slow reconstruction update rate which tends to cause poor tracking, leading to data association errors, and corruption of the other part of the estimation process.

Table 5.1: Maximum and average event rates (eps) measured with different motions, speeds, scenes and lighting conditions — all of the datasets are captured using a DVS128 with fixed bias settings. As shown, the maximum event rates do not exceed 400keps except in a few extreme cases such as datasets 4 and 8 with extremely rapid motion and dataset 20 with an artificially textured moving object moving at high speed. Also note that except the dataset 20, none of the datasets exceed the 1Meps maximum bandwidth of the DVS128 device used in our work. All of the datasets listed here are visualised in the form of accumulated events in Figure 5.1.

dataset				time duration	# of events	**events per second (eps)**	
no.	motion	scene	lighting			max.	avg.
1	rotating at slow speeds	indoor	room lighting	64.41 sec	2.93M	88.98k	44.71k
2	rotating at slow speeds	outdoor	day	59.16 sec	1.82M	89.18k	30.47k
3	rotating at different speeds	indoor	room lighting	71.41 sec	6.58M	188.90k	91.06k
4	rotating at fast speeds	indoor	room lighting	30.08 sec	8.02M	689.80k	260.40k
5	shaking at different speeds	indoor, close-up	HDR	16.86 sec	327.62k	28.53k	18.51k
6	shaking at different speeds	indoor, desk	room lighting	17.34 sec	1.21M	146.00k	66.68k
7	shaking at different speeds	indoor, close-up	room lighting	23.65 sec	3.66M	208.20k	150.30k
8	shaking at fast speeds	indoor	room lighting	2.86 min	16.13M	750.40k	93.61k
9	natural 3D motion	outdoor	dusk	46.84 sec	4.46M	143.40k	93.86k
10	natural 3D motion	indoor	room lighting	59.14 sec	6.83M	169.00k	113.20k
11	natural 3D motion	indoor, desk sized	room lighting	4.78 min	29.44M	194.80k	102.30k
12	walking	indoor	room lighting	4.08 min	19.17M	281.10k	78.27k
13	walking	outdoor	day	29.71 min	69.18M	374.90k	38.89k
14	walking	outdoor	day	10.73 min	65.66M	397.01k	100.45k
15	ground moving at fast speeds	indoor	room lighting	71.84 sec	4.62M	301.30k	65.79k
16	driving at max. 30 mph	outdoor, road	night	7.65 min	13.90M	161.90k	29.67k
17	driving at max. 30 mph	outdoor, road	sunny	8.79 min	25.81M	189.70k	48.13k
18	driving at max. 40 mph	outdoor, road	dusk	10.51 min	31.82M	200.00k	50.09k
19	looking at a spinning fan	indoor, close-up	room lighting	44.85 sec	6.30M	301.70k	138.70k
20	looking at a textured spinning fan	indoor, close-up	room lighting	33.47 sec	32.78M	**2.15M**	994.00k

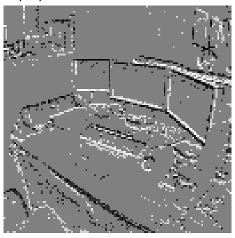

(a) dataset no. 1


(b) dataset no. 2

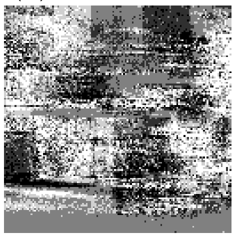

(c) dataset no. 3

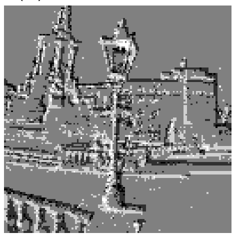
(d) dataset no. 4

(e) dataset no. 5


(f) dataset no. 6

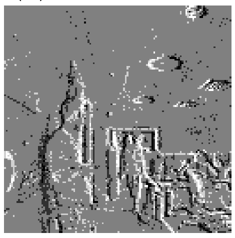
(g) dataset no. 7

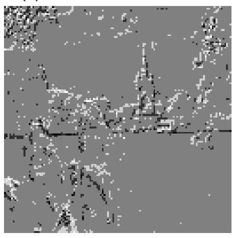

(h) dataset no. 8

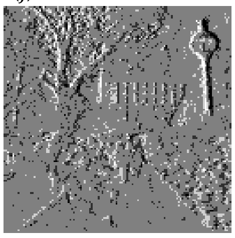

(i) dataset no. 9

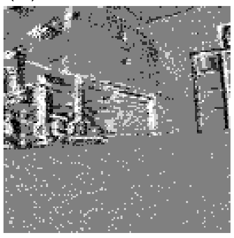
(j) dataset no. 10

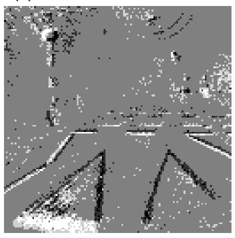
(k) dataset no. 11

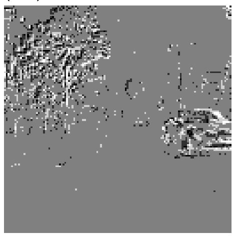

(l) dataset no. 12

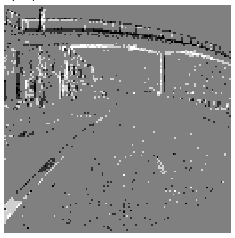

(m) dataset no. 13

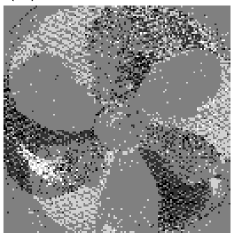

(n) dataset no. 14


(o) dataset no. 15


(p) dataset no. 16


(q) dataset no. 17


(r) dataset no. 18


(s) dataset no. 19

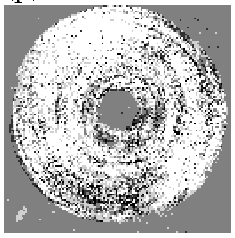
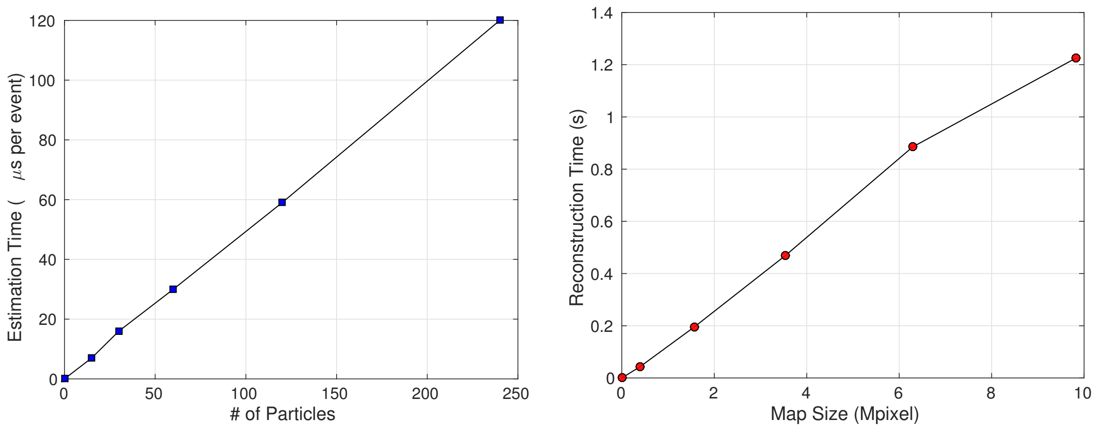

(t) dataset no. 20

Figure 5.1: Visualisation of accumulated events within a 33ms time interval for all of the datasets used in Table 5.1. White and black pixels represent positive and negative events respectively and grey areas represent no event.



(a) estimation processing time per event (b) reconstruction processing time

Figure 5.2: Processing time of the method described in Chapter 4: (a) The processing time per event in microseconds plotted with respect to the number of particles used in particle filter-based tracking and the pixel-wise EKF-based gradient estimation components showing that even with as few as 10 particles, it barely meets the 3μs real-time processing requirement. (b) The processing time in seconds plotted with respect to different map resolutions of the Poisson solver-based log intensity reconstruction method running in parallel showing a very slow reconstruction update rate which tends to cause poor tracking, leading to data association errors, and a corruption of the other part of the estimation process.

Although the current implementation is not optimised for speed, it is clear that using the method described in Chapter 4, which relies on slow particle filter based tracking and Poisson reconstruction running on a CPU, achieving real-time performance is not feasible. We initially attempted to speed it up by using the OpenMP (Open Multi-Processing) API[4], which supports multi-platform shared memory parallel programming, or by harnessing the parallel compute power of a GPU while minimising algorithmic changes. It was however soon realised that both software optimisation attempts were not sufficient because of the limited hardware resources and the overhead of data transfer between CPU and GPU memory. We therefore decided to adopt significant algorithmic changes as described in the following section.

[4]OpenMP (Open Multi-Processing) API: www.openmp.org (accessed September 2017)

5.3 Real-Time Mosaicing and 3-DoF Tracking with an Event Camera

In this section, we present a real-time implementation of simultaneous mosaicing and 3-DoF event camera tracking whose overall structure and functionality is the same as the one described in Chapter 4, but substantial speedup has been achieved via adapting a computationally efficient EKF based estimation method for tracking instead of the previous particle filter based one that approximates the multimodal posterior distribution of the Bayes filter by a finite number of particles. We also replace the Poisson solver based log intensity reconstruction running on a CPU with a parallelisable primal-dual method which is implemented in NVIDIA's CUDA to harness the parallel compute power of a GPU. The third main element of computation, the log intensity gradient estimation method, is the same as described in Section 4.4.1. The experimental results in Section 5.4 show that these algorithmic changes also increase the quality of estimation and reconstruction as they guarantee a higher fidelity of the independence assumption on which our approach of interleaved probabilistic filters and separated log intensity reconstruction heavily rely.

5.3.1 EKF-Based 3-DoF Camera Pose Estimation

We first explain the tracking component of our method which is based on the EKF and provides an event-by-event updated estimate of the rotational location of the event camera with respect to the scene mosaic based on the assumption that the current reconstructed log intensity is correct. It estimates the global 3-DoF camera rotation motion over time with its state $\mathbf{x} \in \mathbb{R}^3$, which is a minimal representation of the camera orientation c with respect to the world frame of reference w, and updates a covariance matrix $\mathtt{P}_{\mathbf{x}} \in \mathbb{R}^{3\times 3}$ representing uncertainty. The state vector is mapped to a member of the Lie group $\mathbf{SO}(3)$ by the matrix exponential map [155]:

$$\mathtt{R}_{wc} = \exp\left(\sum_{i=1}^{3} \mathbf{x}_i \mathtt{G}_i\right) , \tag{5.1}$$

where $\mathtt{G}_i$ are the Lie group generators for $\mathbf{SO}(3)$ as in Equation (4.4), and $\mathtt{R}_{wc}$ is a 3×3 orthonormal rotation matrix.

At initialisation, the initial pose is set to a nominal value with absolute certainty (e.g. $\mathbf{x}_0 = \mathbf{0}_{3\times 1}$ and $\mathtt{P}_{\mathbf{x}_0} = \mathtt{0}_{3\times 3}$ as shown in Table 5.2), and whenever a new event arrives, the state vector and the uncertainty covariance matrix evolve in the standard EKF prediction and measurement update steps as follows.

Motion Prediction

We again use a 3-DoF constant position (random walk) model for motion prediction, where the predicted mean rotation at any given time remains constant while the variance of the prediction is proportional to the time interval as shown in Figure 4.5:

$$\mathbf{x}^{(t|t-\tau)} = \mathbf{x}^{(t-\tau|t-\tau)} + \mathbf{n} \;, \tag{5.2}$$

$$\mathtt{P_x}^{(t|t-\tau)} = \mathtt{P}_{\mathbf{x}}^{(t-\tau|t-\tau)} + \mathtt{P}_{\mathbf{n}} \;, \tag{5.3}$$

where each component of $\mathbf{n}$ is independently Gaussian distributed in all three axes i.e. $\mathbf{n}_i \sim \mathcal{N}(0, \sigma_i^2 \tau)$, and $\mathtt{P}_{\mathbf{n}} = \mathrm{diag}(\sigma_1^2 \tau, \sigma_2^2 \tau, \sigma_3^2 \tau)$. The noise is the predicted change the current rotation might have undergone since the previous event was received — all the parameters used in the experiments are given in Table 5.2. We also use the same time interval notation τ and τ_c as before, which are the time elapsed since the most recent previous event from *any pixel* and at *the same pixel* respectively.

Measurement Update

We calculate the value of a measurement $z_{\mathbf{x}}$ given an event $\mathbf{e}(u, v)$, the current camera pose estimate $\mathtt{R}_{wc}^{(t)}$, the previous pose estimate $\mathtt{R}_{wc}^{(t-\tau_c)}$ where the previous event was received at *the same pixel*, and a reconstructed image-like log intensity map, by taking the log intensity difference (i.e. prediction $h_{\mathbf{x}}$) between two corresponding points $\mathbf{p}_m^{(t)}$ and $\mathbf{p}_m^{(t-\tau_c)}$:

$$z_{\mathbf{x}} = \pm C \;, \tag{5.4}$$

$$h_{\mathbf{x}}(\mathbf{x}^{(t|t-\tau)}) = \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) - \mathtt{I}_l \left(\mathbf{p}_m^{(t-\tau_c)} \right) \;. \tag{5.5}$$

Here $\pm C$ is the known event threshold — its sign is decided by the polarity of the event (i.e. $+C$ for a positive event, and $-C$ for a negative one), $\mathtt{I}_l$ is a log intensity value based on a reconstructed log intensity map, and $\mathbf{p}_m$ is obtained as:

$$\mathbf{p}_m = \begin{pmatrix} \theta \\ \phi \end{pmatrix} = \begin{pmatrix} \tan^{-1}\left(p_{w1}/p_{w3}\right) \\ \tan^{-1}\left(p_{w2}/\sqrt{p_{w1}^2 + p_{w3}^2}\right) \end{pmatrix} , \tag{5.6}$$

where:

$$\mathbf{p}_w = \begin{pmatrix} p_{w1} \\ p_{w2} \\ p_{w3} \end{pmatrix} = \mathtt{R}_{wc}\mathtt{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} . \tag{5.7}$$

Basically, what we want to do is to find the current camera orientation which best predicts the log intensity change between the current time of the new event and the last time there was an event at this pixel.

In the standard EKF framework, the camera pose estimate and its uncertainty covariance matrix are updated by the standard equations at every event using:

$$\mathbf{x}^{(t|t)} = \mathbf{x}^{(t|t-\tau)} + \mathbf{W}_{\mathbf{x}}\nu_{\mathbf{x}} , \tag{5.8}$$

$$\mathtt{P}_{\mathbf{x}}^{(t|t)} = \left(\mathtt{I}_{3\times3} - \mathbf{W}_{\mathbf{x}} \frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \right) \mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} , \tag{5.9}$$

where the innovation $\nu_{\mathbf{x}}$ is:

$$\nu_{\mathbf{x}} = z_{\mathbf{x}} - h_{\mathbf{x}}(\mathbf{x}^{(t|t-\tau)}) , \tag{5.10}$$

the innovation covariance $S_{\mathbf{x}}$ is:

$$S_{\mathbf{x}} = \frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} \left(\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \right)^{\top} + \mathtt{N}_{\mathbf{x}} , \tag{5.11}$$

and the Kalman gain $\mathbf{W}_{\mathbf{x}}$ is:

$$\mathbf{W}_{\mathbf{x}} = \mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} \left(\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \right)^{\top} {S_{\mathbf{x}}}^{-1} . \tag{5.12}$$

The measurement uncertainty $\mathtt{N}_{\mathbf{x}}$ is a scalar variance $\sigma_{\mathbf{x}}^2$ — all the parameters used in the experiments are given in Table 5.2, and the important Jacobian $\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}}$, the partial differentiation of the measurement function with respect to changes in camera pose, is derived as:

$$\begin{aligned}
\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} &= \frac{\partial}{\partial \mathbf{x}^{(t|t-\tau)}} \left(\mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) - \mathtt{I}_l \left(\mathbf{p}_m^{(t-\tau_c)} \right) \right) , && (5.13)\\
&= \frac{\partial}{\partial \mathbf{x}^{(t|t-\tau)}} \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) - 0 , && (5.14)\\
&= \frac{\partial}{\partial \mathbf{p}_m^{(t)}} \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) \frac{\partial \mathbf{p}_m^{(t)}}{\partial \mathbf{x}^{(t|t-\tau)}} , && (5.15)\\
&= \frac{\partial}{\partial \mathbf{p}_m^{(t)}} \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) \frac{\partial \mathbf{p}_m^{(t)}}{\partial \mathbf{p}_w^{(t)}} \frac{\partial \mathbf{p}_w^{(t)}}{\partial \mathbf{x}^{(t|t-\tau)}} , && (5.16)\\
&= \frac{\partial}{\partial \mathbf{p}_m^{(t)}} \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) \frac{\partial \mathbf{p}_m^{(t)}}{\partial \mathbf{p}_w^{(t)}} \frac{\partial \mathbf{p}_w^{(t)}}{\partial \mathtt{R}_{wc}} \frac{\partial \mathtt{R}_{wc}}{\partial \mathbf{x}^{(t|t-\tau)}} , && (5.17)
\end{aligned}$$

where $\frac{\partial}{\partial \mathbf{p}_m^{(t)}} \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right)$ is:

$$\frac{\partial}{\partial \mathbf{p}_m^{(t)}} \mathtt{I}_l \left(\mathbf{p}_m^{(t)} \right) = \begin{bmatrix} \mathtt{I}_{lx} & \mathtt{I}_{ly} \end{bmatrix} , \tag{5.18}$$

$\frac{\partial \mathbf{p}_m^{(t)}}{\partial \mathbf{p}_w^{(t)}}$ is:

$$\frac{\partial \mathbf{p}_m^{(t)}}{\partial \mathbf{p}_w^{(t)}} = \begin{bmatrix} \frac{p_{w_3} W}{2\pi ({p_{w_1}}^2 + {p_{w_3}}^2)} & 0 & -\frac{p_{w_1} W}{2\pi ({p_{w_1}}^2 + {p_{w_3}}^2)} \\ -\frac{p_{w_1} p_{w_2} H}{\pi ({p_{w_1}}^2 + {p_{w_2}}^2 + {p_{w_3}}^2) \sqrt{{p_{w_1}}^2 + {p_{w_3}}^2}} & \frac{\sqrt{{p_{w_1}}^2 + {p_{w_3}}^2} H}{\pi ({p_{w_1}}^2 + {p_{w_2}}^2 + {p_{w_3}}^2)} & -\frac{p_{w_2} p_{w_3} H}{\pi ({p_{w_1}}^2 + {p_{w_2}}^2 + {p_{w_3}}^2) \sqrt{{p_{w_1}}^2 + {p_{w_3}}^2}} \end{bmatrix} , \tag{5.19}$$

and $\frac{\partial \mathbf{p}_w^{(t)}}{\partial \mathtt{R}_{wc}}$ is:

$$\frac{\partial \mathbf{p}_w^{(t)}}{\partial \mathtt{R}_{wc}} = \begin{bmatrix} p_{w_1} & 0 & 0 & p_{w_2} & 0 & 0 & p_{w_3} & 0 & 0 \\ 0 & p_{w_1} & 0 & 0 & p_{w_2} & 0 & 0 & p_{w_3} & 0 \\ 0 & 0 & p_{w_1} & 0 & 0 & p_{w_2} & 0 & 0 & p_{w_3} \end{bmatrix} , \tag{5.20}$$

and $\frac{\partial \mathtt{R}_{wc}}{\partial \mathbf{x}^{(t|t-\tau)}}$ is:

$$\frac{\partial \mathtt{R}_{wc}}{\partial \mathbf{x}^{(t|t-\tau)}} = \begin{bmatrix} 0 & \mathtt{R}_{1,3} & -\mathtt{R}_{1,2} \\ -\mathtt{R}_{1,3} & 0 & \mathtt{R}_{1,1} \\ \mathtt{R}_{1,2} & -\mathtt{R}_{1,1} & 0 \\ 0 & \mathtt{R}_{2,3} & -\mathtt{R}_{2,2} \\ -\mathtt{R}_{2,3} & 0 & \mathtt{R}_{2,1} \\ \mathtt{R}_{2,2} & -\mathtt{R}_{2,1} & 0 \\ 0 & \mathtt{R}_{3,3} & -\mathtt{R}_{3,2} \\ -\mathtt{R}_{3,3} & 0 & \mathtt{R}_{3,1} \\ \mathtt{R}_{3,2} & -\mathtt{R}_{3,1} & 0 \end{bmatrix} . \tag{5.21}$$

5.3.2 Primal-Dual Reconstruction

We now explain the new fast log intensity reconstruction component of our real-time mosaicing and 3-DoF tracking method with a single event camera. Basically, we replace the Poisson solver based method running on a CPU described in Chapter 4 with a parallelisable primal-dual method which is implemented in NVIDIA's CUDA to harness the parallel compute power of a GPU.

Specifically, we define our convex minimisation function as:

$$\min_{\mathtt{I}_l} \left\{ \int_\Omega ||\mathbf{g}(\mathbf{p}_m) - \nabla \mathtt{I}_l(\mathbf{p}_m)||^h_{\epsilon_d} + \lambda ||\nabla \mathtt{I}_l(\mathbf{p}_m)||^h_{\epsilon_r} \mathrm{d}\mathbf{p}_m \right\} . \tag{5.22}$$

Here the data term represents the error between estimated gradients $\mathbf{g}(\mathbf{p}_m)$ and those of a reconstructed log intensity $\nabla \mathtt{I}_l(\mathbf{p}_m)$, which we would like to minimise, and the regularisation term enforces smoothness, both under a robust Huber norm. To minimise Equation (5.22), we use a primal-dual algorithm following [4, 27, 181] which guarantees its optimal convergence and is easily parallelisable. To arrive at its primal-dual form, we use duality principles replacing individual Huber norms of the equation by their convex conjugates using the Legendre-Fenchel transformation [69]:

$$\min_{\mathtt{I}_l} \max_{\mathbf{q}} \max_{\mathbf{p}} \left\{ \langle \mathbf{p}, \mathbf{g} - \nabla \mathtt{I}_l \rangle - \frac{\epsilon_d}{2} ||\mathbf{p}||^2 - \delta_{\mathbf{p}}(\mathbf{p}) + \langle \mathbf{q}, \nabla \mathtt{I}_l \rangle - \frac{\epsilon_r}{2\lambda} ||\mathbf{q}||^2 - \delta_{\mathbf{q}}(\mathbf{q}) \right\} . \tag{5.23}$$

Here $\mathbf{p}$ and $\mathbf{q}$ are dual variables, $\langle y, x\rangle$ computes the inner product between the primal variable x and its dual y, the convex conjugate of the Huber norm in the region $||x||_1 > \epsilon$ is the indicator function in the dual variable:

$$\delta(y) = \begin{cases} 0 & \text{if } \epsilon < ||y||_1 \leq 1 \\ \infty & \text{otherwise} \end{cases}, \tag{5.24}$$

while its quadratic region for $||x||_1 \leq \epsilon$ results in a quadratic conjugate function within that region $\frac{\epsilon}{2}||y||^2$.

We can solve Equation (5.23) by maximising with respect to $\mathbf{p}$; by following iterate steps:

$$\mathbf{p}^{(n+1)} = \frac{\frac{\mathbf{p}^{(n)}+\sigma_{\mathbf{p}}(\mathbf{g}-\nabla \mathtt{I}_l)}{1+\sigma_{\mathbf{p}}\epsilon_d}}{\max\left(1, \left|\frac{\mathbf{p}^{(n)}+\sigma_{\mathbf{p}}(\mathbf{g}-\nabla \mathtt{I}_l)}{1+\sigma_{\mathbf{p}}\epsilon_d}\right|\right)}, \tag{5.25}$$

maximising with respect to $\mathbf{q}$ with steps:

$$\mathbf{q}^{(n+1)} = \frac{\frac{\mathbf{q}^{(n)}+\sigma_{\mathbf{q}}\nabla \mathtt{I}_l}{1+\frac{\sigma_{\mathbf{q}}\epsilon_r}{\lambda}}}{\max\left(1, \frac{1}{\lambda}\left|\frac{\mathbf{q}^{(n)}+\sigma_{\mathbf{q}}\nabla \mathtt{I}_l}{1+\frac{\sigma_{\mathbf{q}}\epsilon_r}{\lambda}}\right|\right)}, \tag{5.26}$$

and minimising with respect to $\mathtt{I}_l$ with steps:

$$\mathtt{I}_l^{(n+1)} = \mathtt{I}_l^{(n)} - \sigma_{\mathtt{I}_l}\left(\text{div}\mathbf{p}^{(n+1)} - \text{div}\mathbf{q}^{(n+1)}\right). \tag{5.27}$$

In our CUDA implementation, we initialise both dual variables as $\mathbf{p} = \mathbf{0}$ and $\mathbf{q} = \mathbf{0}$, and iterate the above steps until either the change of the reconstructed log intensity values becomes lower than a certain threshold $\Delta \mathtt{I}_l$, or the number of iterations reaches to a pre-set maximum number of iterations n_{max} — all the parameters used in the experiments are given in Table 5.2.

5.4 Evaluation and Results

Our algorithm runs in real-time on a standard PC with typical scenes and motion speeds, and we have conducted experiments both indoors and outdoors under different conditions including motion speeds, proximity to objects and illuminations. We recommend viewing our video[5] which illustrates all of the key results in a better form than still pictures and in real-time (also see Appendix A).

In all of the experiments, we have used a DVS128 camera from iniLabs[6] with 128×128 resolution, 120 dB dynamic range, and 15 microsecond latency, and communicated with a host computer using our own USB 2.0 driver. We applied the same bias settings to all of the experiments regardless of quite different experimental conditions especially lighting, and we expect even better results with optimised bias values under specific conditions. The camera has pre-calibrated intrinsics and all event pixel locations are pre-warped to remove radial distortion via the event camera calibration method described in Section 3.6. We run all of the experiments both on a standard desktop PC consisting of an NVIDIA GeForce GTX 680 GPU hosted by an Intel Xeon W5590 3.33GHz quad-core CPU, and an Apple MacBook Pro consisting of an NVIDIA GeForce GT 750M GPU hosted by an Intel i7 2.6Ghz dual-core CPU.

We present all of the parameters used in the experiments in Table 5.2. As shown, we have used the same values for most of the parameters across a wide range of experimental environments, except the ones (i.e. $\sigma_1, \sigma_2, \sigma_3$) relating to the motion estimation which are sensitive to different motion speeds.

5.4.1 Processing Time

The processing time per event of the EKF based 3-DoF camera tracking and the pixel-wise EKF based log intensity gradient estimation components is measured within a range of 1-2μs which is sufficient to meet the real-time requirement we discussed in Section 5.2 — measured by the same time measurement software function, and ignoring its overhead running on the same 3.33GHz quad-core computer. In contrast to the particle filter which requires monotonically increasing processing costs with respect to the number of particles being used, EKF based camera pose estimation requires almost constant processing time without compromising its performance.

The graph in Figure 5.3 also presents the processing time required for the primal-dual log intensity reconstruction with respect to different map pixel sizes running on a GPU (NVIDIA GTX 680 in this experiment), showing a significant speed up achieved (e.g. reduced from about 1.2s to about 41ms). Considering the high measurement rate of the event camera, it

[5]ETAM 2D: Real-Time Event-Based Tracking and Mapping: `https://youtu.be/z72lNV7idUs` (accessed September 2017)

[6]iniLabs Ltd: `www.inilabs.com` (accessed September 2017)

Table 5.2: Parameters used in the experiments

Parameter	Value	Reference
map size	2304×1152	
$\mathbf{x}_0$	$\mathbf{0}_{3\times1}$	Section 5.3.1
$\mathtt{P}_{\mathbf{x}_0}$	$\mathtt{0}_{3\times3}$	Section 5.3.1
$\mathbf{g}_0$	$\mathbf{0}_{2\times1}$	
$\mathtt{P}_{\mathbf{g}_0}$	$\begin{pmatrix} 1.0\times10^{-3} & 0 \\ 0 & 1.0\times10^{-3} \end{pmatrix}$	
$\sigma_1 \sim \sigma_3$	$2.0\times10^{-3} \sim 4.0\times10^{-3}$	Section 5.3.1
$\Delta\mathtt{I}_l$	1.0×10^{-8}	Section 5.3.2
n_{max}	100	Section 5.3.2
C	0.15	Section 5.3.1
$\sigma_{\mathbf{x}}$	1.0×10^{-2}	Section 5.3.1
γ	1.0×10^{-2}	
σ_t	1.0×10^{-4}	
σ_C	1.0×10^{-1}	
ϵ_d	0.1	Section 5.3.2
ϵ_r	0.1	Section 5.3.2
$\sigma_{\mathbf{p}}$	0.5	Section 5.3.2
$\sigma_{\mathbf{q}}$	0.5	Section 5.3.2
$\sigma_{\mathtt{I}_l}$	0.5	Section 5.3.2
λ	1.0×10^{-1}	Section 5.3.2

is, of course, a still relatively slow update rate especially with high template resolutions, but we have found that it is sufficient to guarantee a higher fidelity of the independence assumption which increases the quality of estimation and reconstruction as shown in the following sections.

5.4.2 Spherical Mosaicing

As in the previous chapter, we have first conducted spherical mosaicing in both indoor and outdoor scenes, but running in real-time on a standard PC. As shown in the following two figures, our algorithm is able to estimate scene log intensity gradients (shown in Figure 5.4 (a) and Figure 5.5 (a) — the colours and intensities represent the orientation and strengths of the estimated gradients of the scene) and reconstruct scene log intensity (shown in Figure 5.4 (b) and Figure 5.5 (b) — converted to intensity values for a visualisation pur-

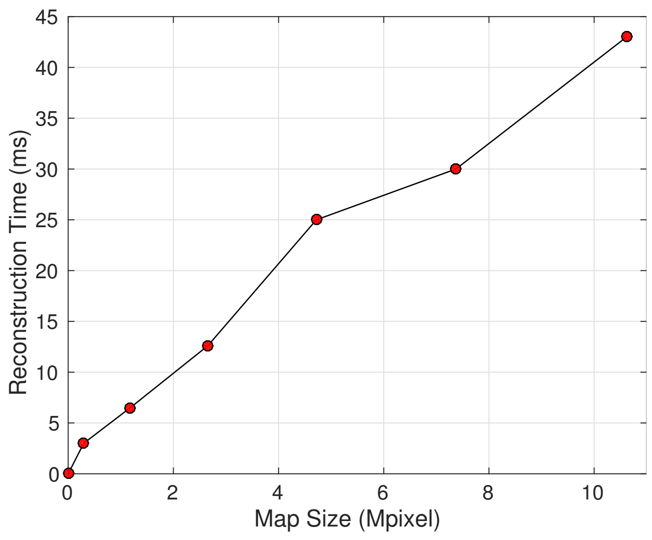


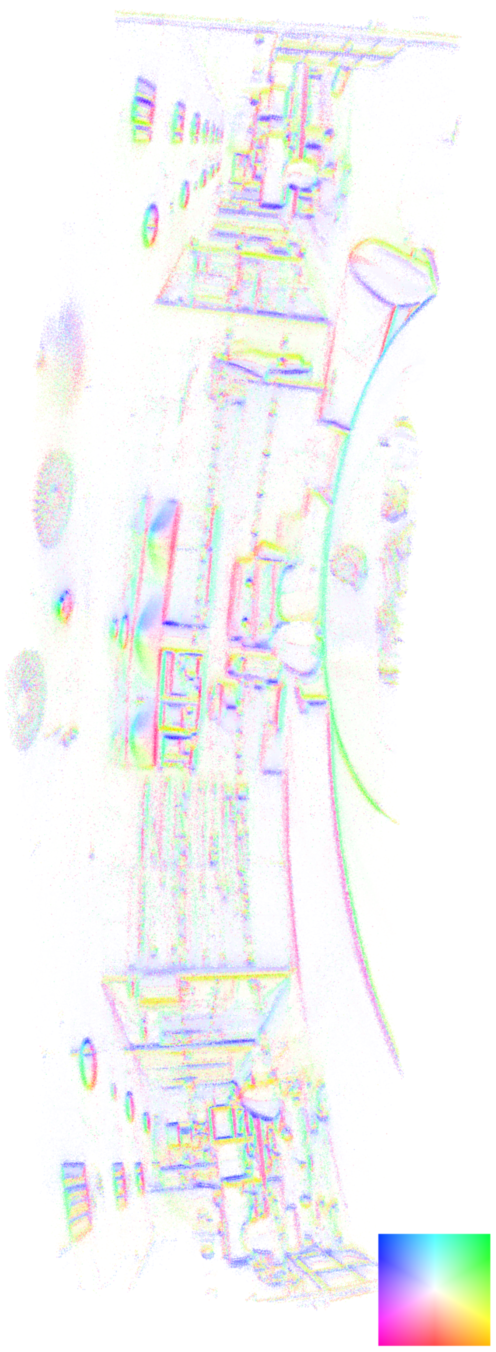
Figure 5.3: Processing time in milliseconds with respect to different map resolutions of the primal-dual log intensity reconstruction method running in parallel on a GPU, showing a significant speed up achieved which guarantees a higher fidelity of the independence assumption.

pose) while tracking the 3-DoF orientation of the event camera in real-time. The quality of tracking can also be measured by the sharp details of the reconstructions.

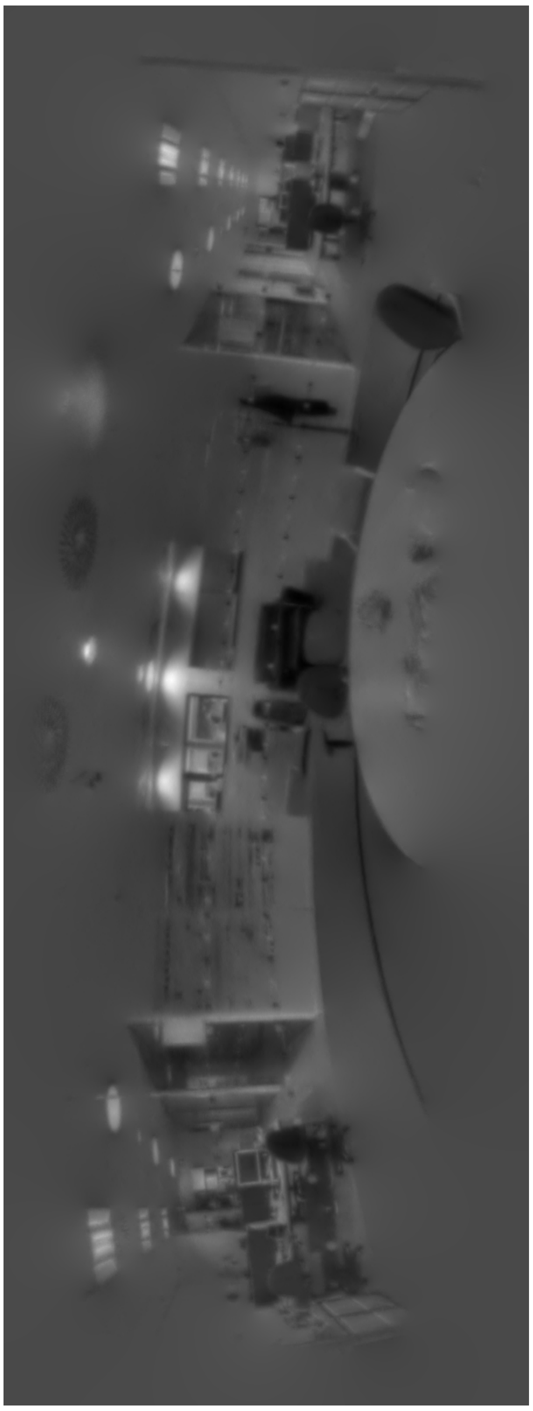
5.4.3 High Speed Tracking

The real-time processing capability has enabled us to easily evaluate our method against extremely rapid rotating motion. For instance, our method is able to reconstruct a high quality mosaic while tracking the event camera under vigorously oscillating motion up to about $\pm$ 10 rad s^{-1} as shown in Figure 5.6 — the graph in (a) plots angular velocity measurements from a gyroscope which was firmly attached to the event camera in this experiment showing its rapidly oscillating motion between 15 and 30 seconds, and (b) presents the high quality reconstruction with the overlaid FOV of the event camera (yellow box) and well aligned accumulated events within about 33ms (red and blue dots) showing the quality of our tracker. We again highly recommend viewing our video[7] which illustrates these high speed tracking results in a better form than still pictures and in real-time (also see Appendix A).

[7]ETAM 2D: Real-Time Event-Based Tracking and Mapping: `https://youtu.be/z72lNV7idUs` (accessed September 2017)

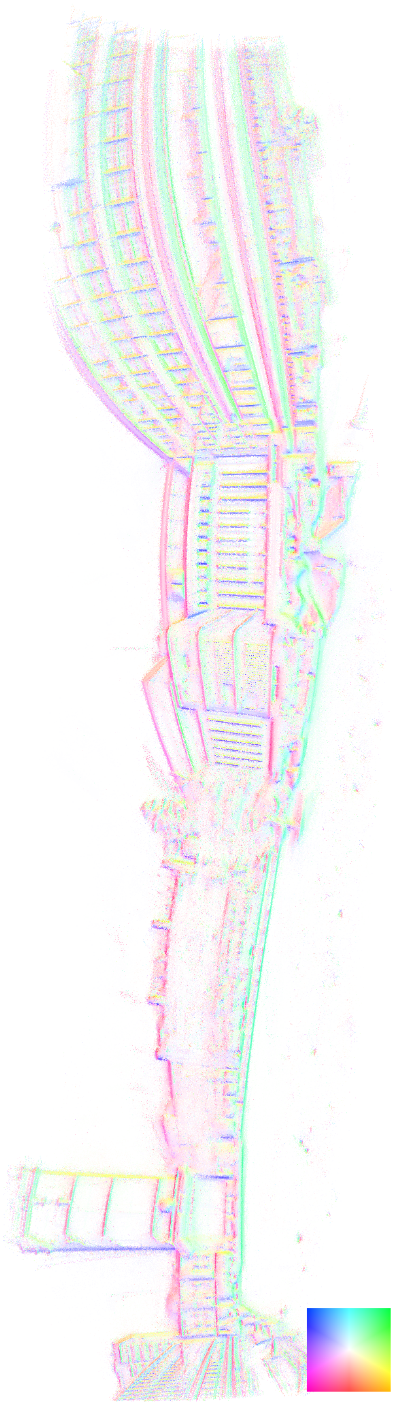


(a)



(b)

Figure 5.4: Spherical mosaicing — William Penney Laboratory at Imperial College London dataset: (a) estimated gradient map — the colours and intensities represent the orientation and strengths of the gradients of the scene (refer to the colour chart in the corner); (b) reconstructed log intensity spherical mosaicing (converted to intensity values for visualisation purposes).


(a)

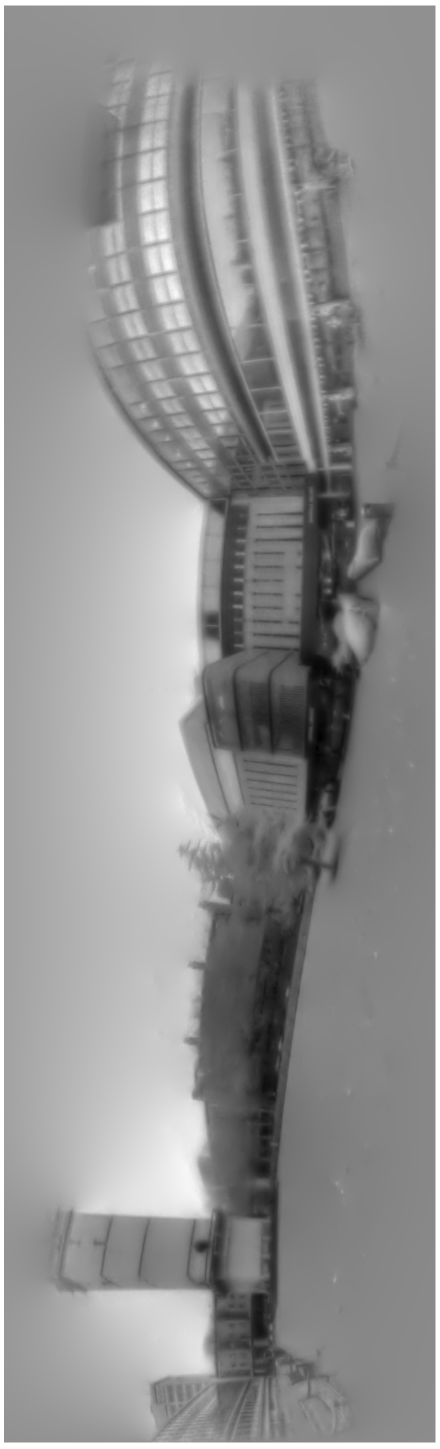
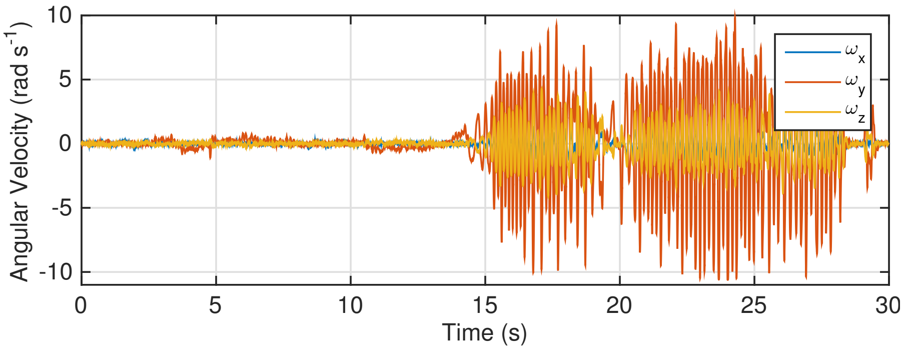
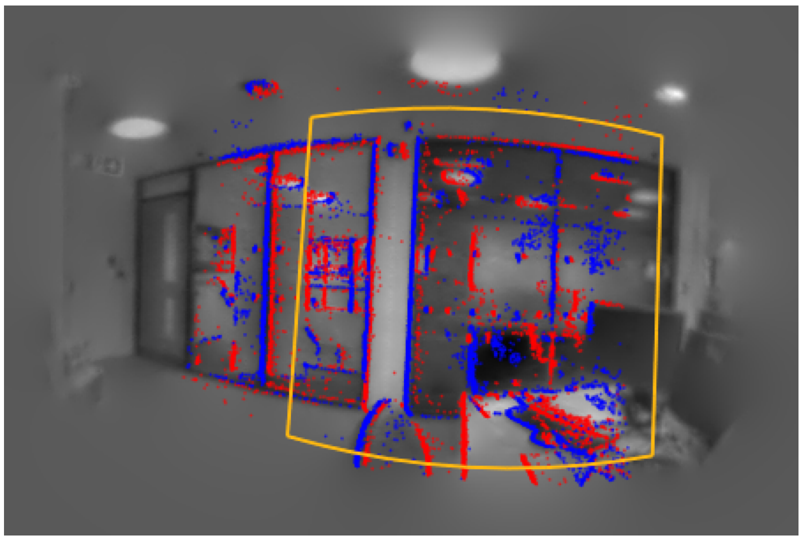

(b)

Figure 5.5: Spherical mosaicing — Queen's Lawn at Imperial College London dataset: (a) estimated gradient map — the colours and intensities represent the orientation and strengths of the gradients of the scene (refer to the colour chart in the corner); (b) reconstructed log intensity spherical mosaicing (converted to intensity values for visualisation purposes).



(a)



(b)

Figure 5.6: High speed tracking result: (a) angular velocity measurements from a gyroscope attached to the event camera in this experiment showing vigorously oscillating motion up to about $\pm$ 10 rad s^{-1}; (b) reconstructed mosaic with the overlaid event camera's FOV (yellow box) and accumulated events within about 33ms (red and blue dots) showing the quality of our tracker.

5.4.4 Qualitative Comparisons

Lastly, we conducted qualitative comparisons between the methods described in Chapter 4 and this chapter using three pre-recorded event camera datasets (i.e. office, lecture room, and outdoor) as shown in Figures 5.7, 5.8 and 5.9. In all cases, the speed up achieved by our new method also increases the quality of estimation and reconstruction as it guarantees a higher fidelity of the independence assumption which is key to our approach of interleaved probabilistic filters and separated log intensity reconstruction. The improvements can be observed in the higher image quality of the reconstructed mosaics.

5.5 Discussion and Summary

In this chapter, we have presented a real-time implementation of simultaneous mosaicing and 3-DoF camera pose estimation method whose overall structure and functionality is the same as the previous method, but a substantial speed up has been achieved via adopting a computationally efficient EKF based estimation method for tracking as well as parallelisable primal-dual log intensity reconstruction running on a GPU. The experimental results in Section 5.4 show that the processing time of the proposed method is sufficient to meet the real-time requirement we discussed in Section 5.2, and this speed-up also increases the quality of estimation and reconstruction as it guarantees higher fidelity of the independence assumption.

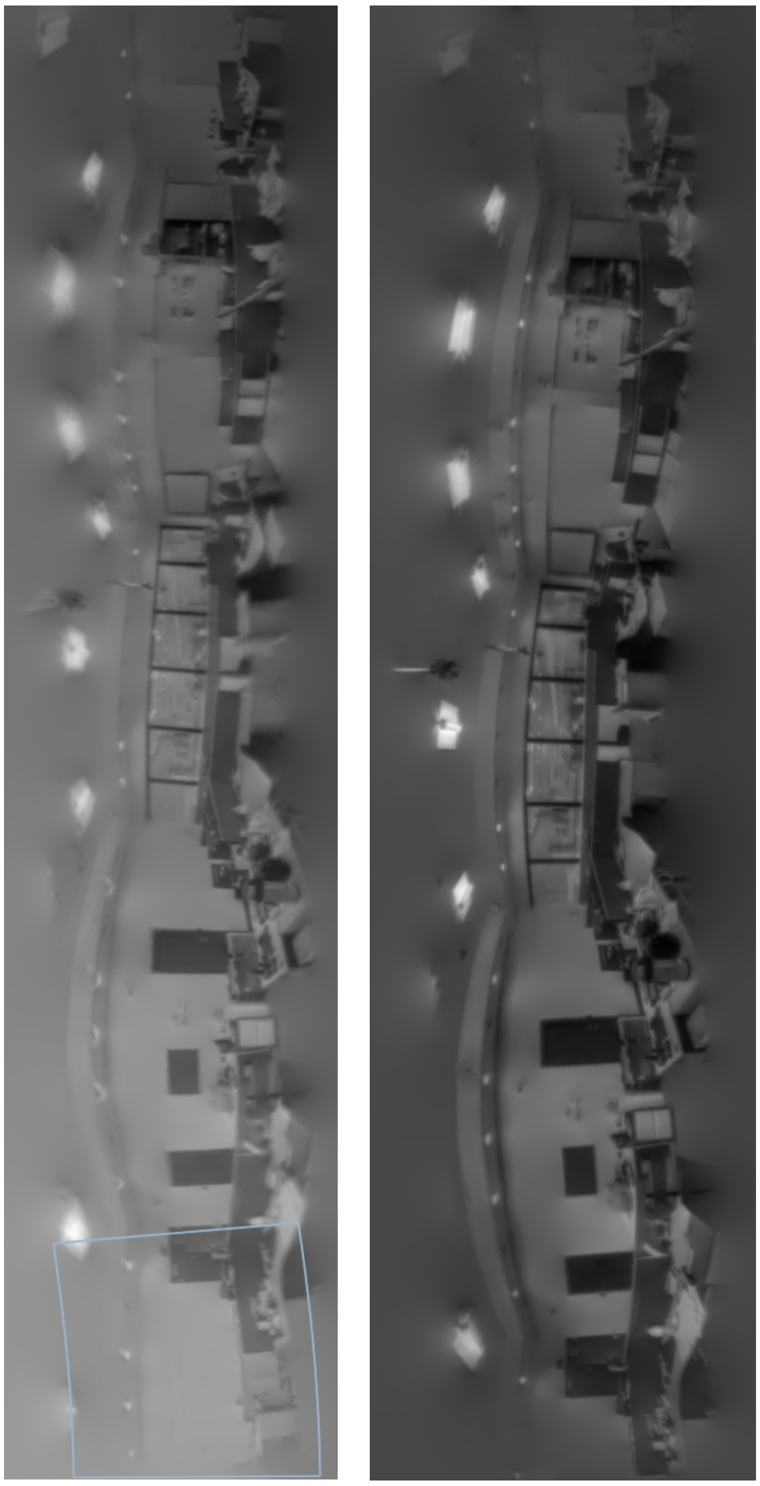


Figure 5.7: Office dataset qualitative comparison: (a) reconstructed intensity mosaic using the method described in Chapter 4 (performed off-line); (b) reconstructed intensity mosaic using the method described in this chapter showing sharper details (performed in real-time).

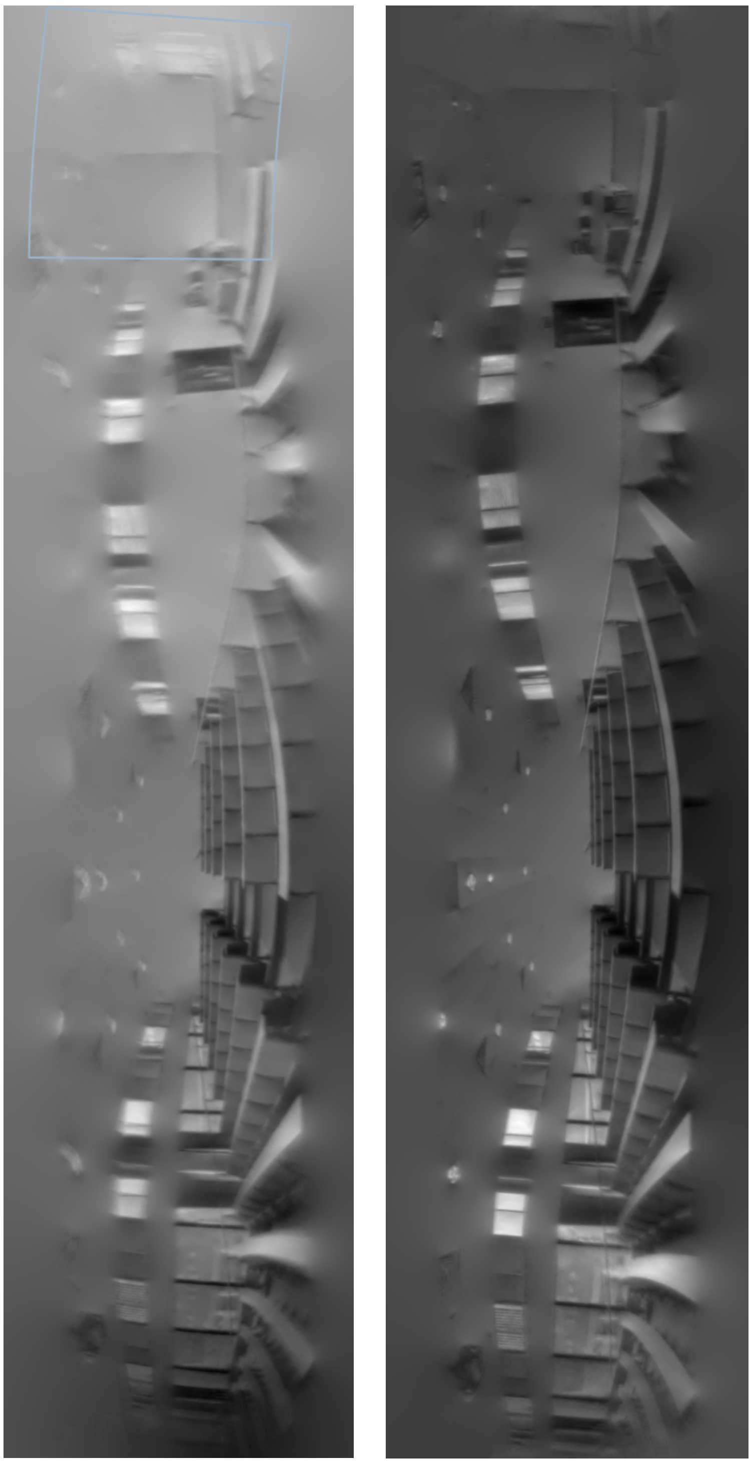
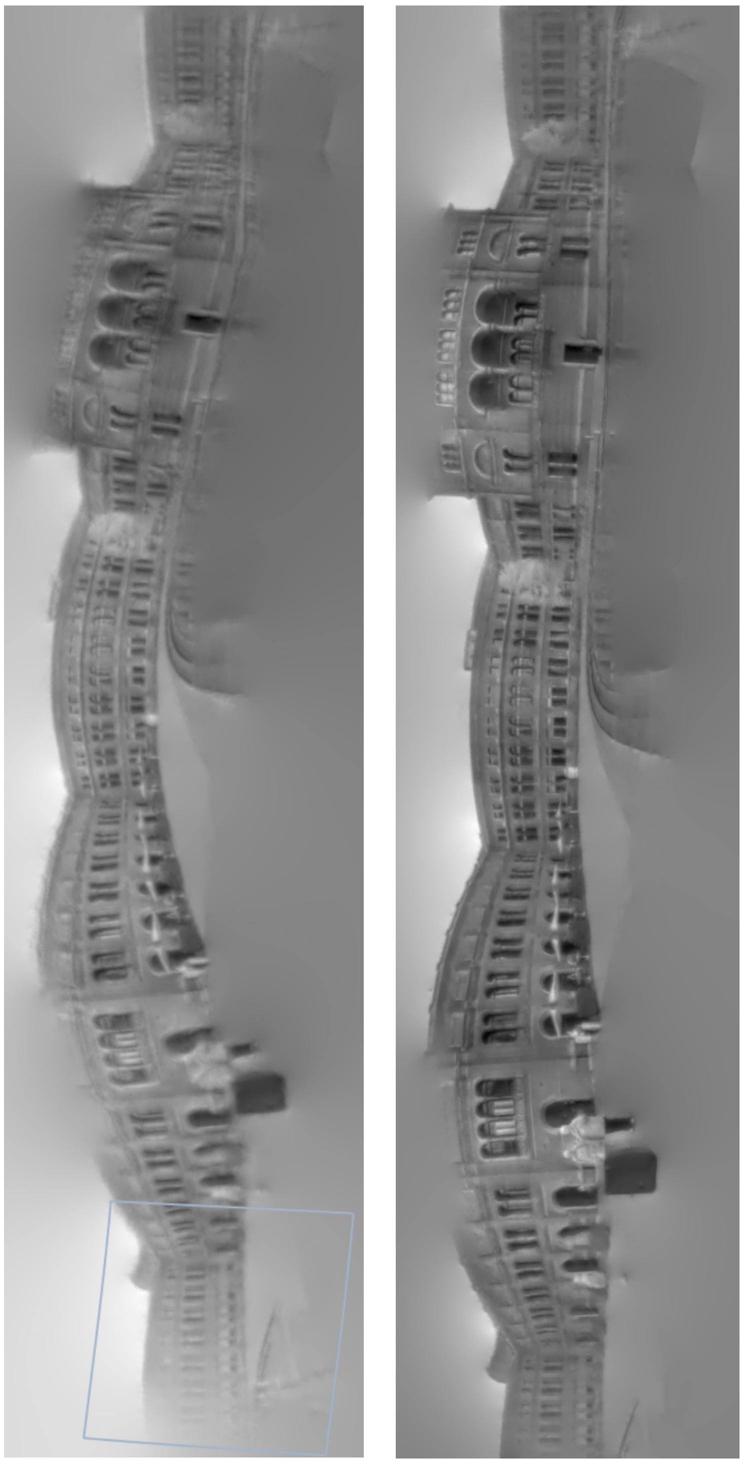

(a) (b)

Figure 5.8: Lecture room dataset qualitative comparison: (a) reconstructed intensity mosaic using the method described in Chapter 4 (performed off-line); (b) reconstructed intensity mosaic using the method described in this chapter showing sharper details (performed in real-time).



(a) (b)

Figure 5.9: Outdoor dataset qualitative comparison: (a) reconstructed intensity mosaic using the method described in Chapter 4 (performed off-line); (b) reconstructed intensity mosaic using the method described in this chapter showing sharper details (performed in real-time).

Chapter 6

Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera

Contents

6.1 Introduction

In this chapter, we extend the panoramic reconstruction and 3-DoF camera tracking methods described in the previous chapters to perform real-time 3D semi-dense reconstruction from a single hand-held event camera with no additional sensing. Our method works in unstructured scenes of which it has no prior knowledge. This work was inspired by a strong belief that depth estimation from a single moving event camera must be possible, because if the device is working correctly and recording all pixel-wise intensity changes then all of the information, that is important for localisation and mapping, present in a standard video stream must be available in principle, at least up to scale. In fact, the high temporal resolution and dynamic range of event pixels should remove the usual bounds on frame-rate and dynamic range of standard video frames. The essential insight to extending the previous simultaneous mosaicing and 3-DoF tracking methods towards getting depth from events is that once the camera starts to translate, if two pixels have the same intensity gradient then the one which is closer to the camera will move past the camera faster and therefore emit more events than the farther one. This is the essential mechanism built into our probabilistic filter for inverse depth.

As shown in Figure 6.1, our method takes inputs in the form of a stream of events from a freely moving event camera looking at a natural scene, and operates on an event-by-event basis to maximise the update rate with low latency. We again follow the many recent successful SLAM systems such as PTAM [84], DTAM [128], LSD-SLAM [57], and ORB-SLAM [125] which separate the tracking and mapping components based on the assumption that the current estimate from one component is accurate enough to lock for the purpose of estimating the other. The basic structure of the method relies on three decoupled probabilistic filters, shown as the blue rounded boxes, each estimating one unknown aspect of this challenging 3D SLAM problem: 6-DoF event camera global motion, scene log intensity gradient, and scene inverse depth. All of them are estimated relative to a virtual projective reference keyframe as shown in Figure 6.2, and each incoming event contributes to the estimation components while the event camera frame of reference is near to the keyframe. We also upgrade the log intensity gradient estimate for the keyframe into a log intensity image, allowing us to calculate the value of a measurement for the camera pose estimation as well as to recover a real-time video-like intensity sequence with spatial and temporal super-resolution from the low bit-rate input event stream. Also a textured semi-dense 3D point cloud can be generated from the keyframe with its associated reconstructed intensity and inverse depth estimates. To reconstruct super resolution scenes by harnessing the very high

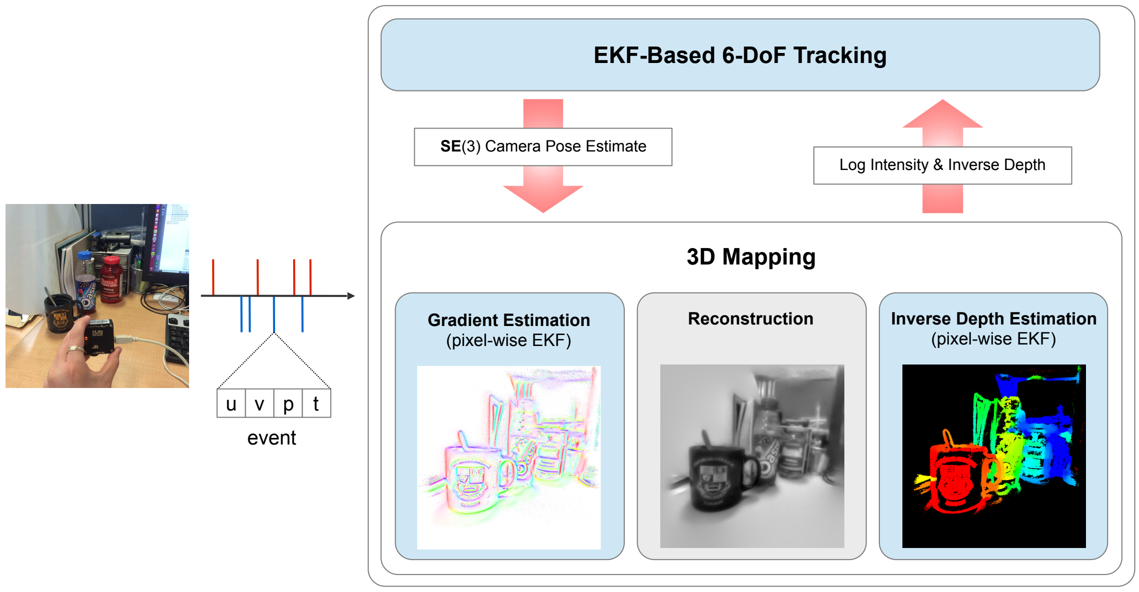

EKF-Based 6-DoF Tracking
SE(3) Camera Pose Estimate
Log Intensity & Inverse Depth
3D Mapping
Gradient Estimation (pixel-wise EKF)
Reconstruction
Inverse Depth Estimation (pixel-wise EKF)
u v p t
event

Figure 6.1: Method overview showing separation tracking and mapping: The method takes inputs in the form of a stream of events from a single hand-held, freely moving event camera looking at an unstructured scene, and is operating on an event-by-event basis. The basic structure of the method relies on three decoupled probabilistic filters as shown as blue rounded boxes, each estimating 6-DoF camera global motion, scene log intensity gradient, and scene inverse depth. The log intensity gradient is also in parallel upgraded into a full image-like intensity map as shown in the grey rounded box.

speed measurement property of the event camera which enables sub-pixel accurate camera tracking, we use a higher resolution for keyframes than for the low resolution sensor. We also use a wider *field of view* (FOV) to cover all pre-warped event pixel locations, especially those which are out of the event camera's FOV after the radial distortion compensation step (e.g. 576×576 instead of 128×128 as shown in Table 6.2).

We again do not use an explicit bootstrapping method as we have found that, starting from scratch, alternating estimation very often leads to convergence, though there are sometimes currently gross failures, and this is an important issue for future research.

6.2 Event Camera 6-DoF Tracking

We now explain one of the main parts of our proposed method, the 6-DoF event camera tracking component, whose job is to provide an event-by-event updated estimate of the position and orientation of the camera with respect to the reconstructed 3D map in the

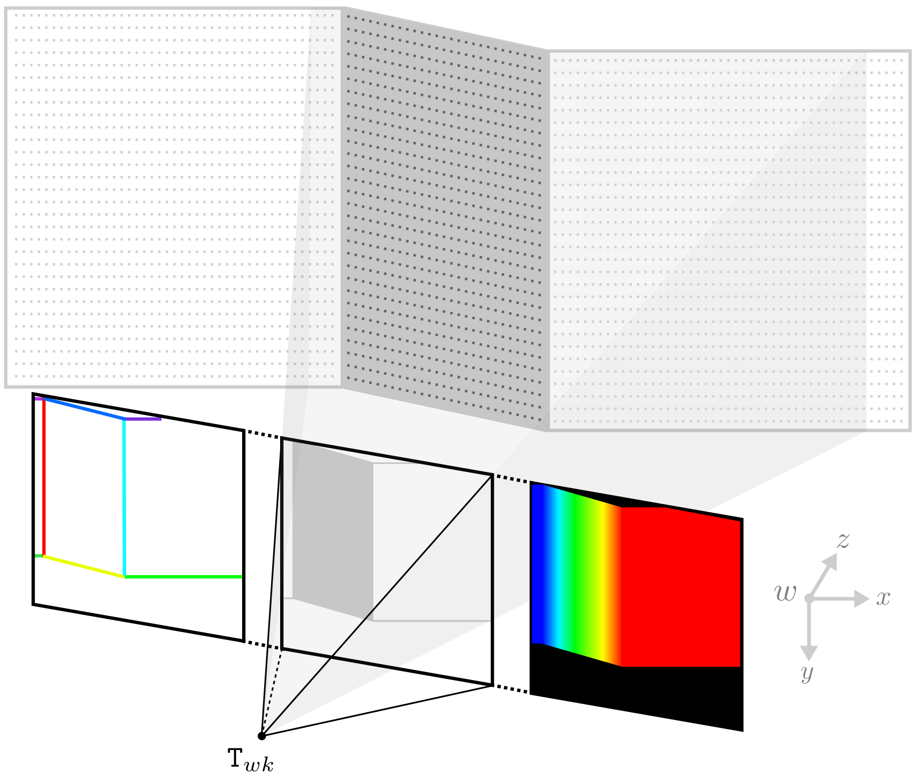


Figure 6.2: *Virtual keyframe*: a virtual projective reference frame whose pose is represented as a 3D rigid body transformation matrix $\mathtt{T}_{wk}$ with respect to the world frame of reference w. It consists of a log intensity gradient map, a log intensity map, and an inverse depth map. A textured semi-dense 3D point cloud can be also generated from the reconstructed intensity and inverse depth estimates. All of the probabilistic filtering components are estimated relative to this keyframe, and each incoming event contributes to the estimation components while the event camera frame of reference is near by the keyframe.

world frame of reference. The basic idea is, by assuming that the current log intensity and inverse depth estimates are correct, to find the new camera pose $\mathtt{T}_{wc}^{(t)}$ which best predicts a log intensity change consistent with the polarity of the event just received, as illustrated in Figure 6.3.

We use an EKF to estimate the global 6-DoF event camera motion over time with state $\mathbf{x} \in \mathbb{R}^6$ (i.e. translation in $\mathbb{R}^3$ and rotation in $\mathbb{R}^3$), which is a minimal representation of the camera pose c with respect to the world frame of reference w, and covariance matrix $\mathtt{P}_\mathbf{x} \in \mathbb{R}^{6\times 6}$. At initialisation, the initial pose is set to a nominal value with absolute certainty (e.g. $\mathbf{x}_0 = \mathbf{0}_{6\times 1}$ and $\mathtt{P}_{\mathbf{x}_0} = 0_{6\times 6}$ as shown in Table 6.2).

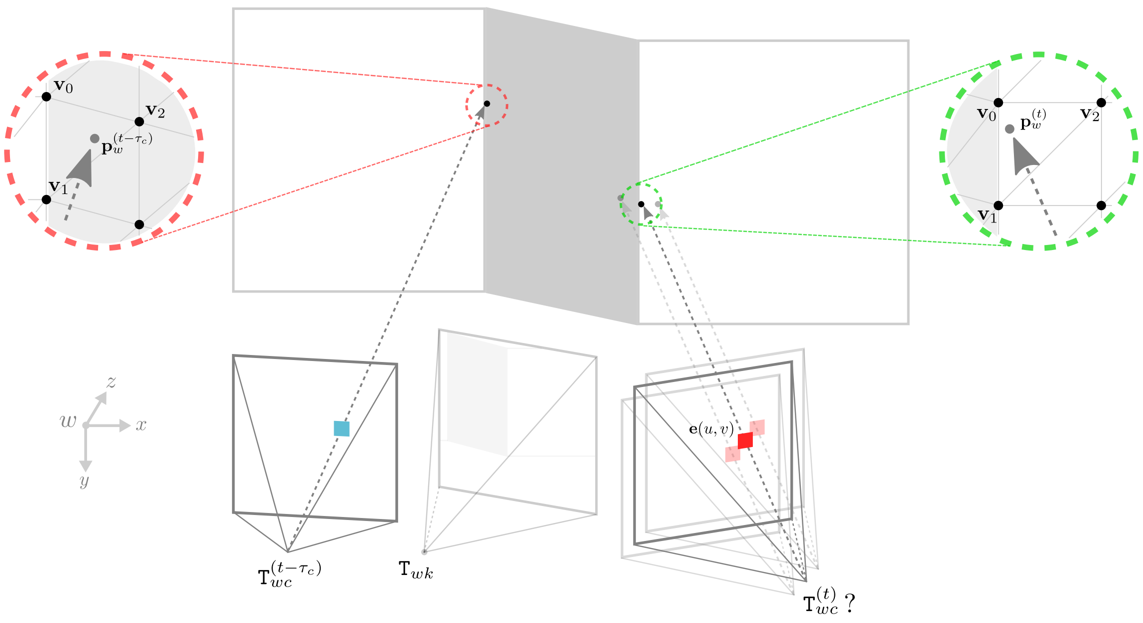


Figure 6.3: Camera pose estimation: based on the assumption that the current log intensity estimate (shown as the colours of the walls) and inverse depth estimate (shown as the geometry of the walls) are correct, we estimate the current camera pose most consistent with the predicted log intensity change since the previous event at the same pixel. Specifically, we first find two corresponding ray-triangle intersection points, $\mathbf{p}_w^{(t)}$ and $\mathbf{p}_w^{(t-\tau_c)}$, in the world frame of reference using the ray-triangle intersection method detailed in Section 6.2.1 to predict the value of a measurement. This is a log intensity difference between two points given an event $\mathbf{e}(u, v)$, the current keyframe pose $\mathtt{T}_{wk}$, the current camera pose estimate $\mathtt{T}_{wc}^{(t)}$, the previous pose estimate $\mathtt{T}_{wc}^{(t-\tau_c)}$ and the reconstructed log intensity and inverse depth keyframe. We update the camera pose estimate based on the standard EKF framework.

The state vector is mapped to a member of the Lie group $\mathbf{SE}(3)$, the set of 3D rigid body transformations, by the matrix exponential map [155]:

$$\mathtt{T}_{wc} = \exp\left(\sum_{i=1}^{6} \mathbf{x}_i \mathtt{G}_i\right) = \begin{pmatrix} \mathtt{R}_{wc} & \mathbf{t}_w \\ \mathbf{0}^\top & 1 \end{pmatrix}, \tag{6.1}$$

where $\mathtt{R}_{wc} \in \mathbf{SO}(3)$ is a 3×3 orthonormal rotation matrix representing rotation only point transfer between frames c and w, and $\mathbf{t}_w \in \mathbb{R}^3$ is a 3D translation vector in the world frame of reference w. The Lie group generators for $\mathbf{SE}(3)$ $\mathtt{G}_i$ are:

$$\mathtt{G}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathtt{G}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathtt{G}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$
$$\mathtt{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathtt{G}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathtt{G}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{6.2}$$

As every new event arrives, the state vector and its covariance matrix evolve in the standard EKF motion prediction and measurement update steps, which we explain in detail next.

6.2.1 Corresponding 3D Point Search

We again denote an event as $\mathbf{e}(u, v) = (u, v, p, t)^\top$, where u and v are pixel location, p is polarity and t is microsecond-precise timestamp as in the previous chapters. Our event camera has fixed pre-calibrated intrinsics matrix $\mathtt{K}$ and all event pixel locations are pre-warped to remove radial distortion via the event camera calibration method described in Section 3.6.

While conventional dense tracking methods such as DTAM [128] use whole image rendering to get correspondence, we do it the other way around using *ray casting* because we process event-by-event — when an event is received at a pixel location $\mathbf{p}_c = (u, v)^\top$ in the event camera frame of reference at $\mathtt{T}_{wc}$, we obtain a corresponding 3D point $\mathbf{p}_w$ in the world frame of reference w using ray casting as illustrated in Figure 6.4.

Specifically, we examine whether the event pixel's ray $\mathbf{r} = \mathtt{R}_{wc}\mathtt{K}^{-1}\dot{\mathbf{p}}_c + \mathbf{t}_w$ intersects a triangle formed by any three adjacent vertices $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$ in the 3D point cloud using the ray-triangle intersection algorithm proposed by Möller and Trumbore [117] which is a fast method for calculating a 3D intersection point without needing to compute the equation of the plane containing the triangle. The method takes as input the origin of a ray, i.e. the centre of the event camera frame of reference, the direction vector of the ray which is calculated using camera intrinsics for each pixel, and three adjacent vertices, and yields a vector $(l, a, b)^\top$, where l is the distance to the triangle from the origin of the ray and a, b

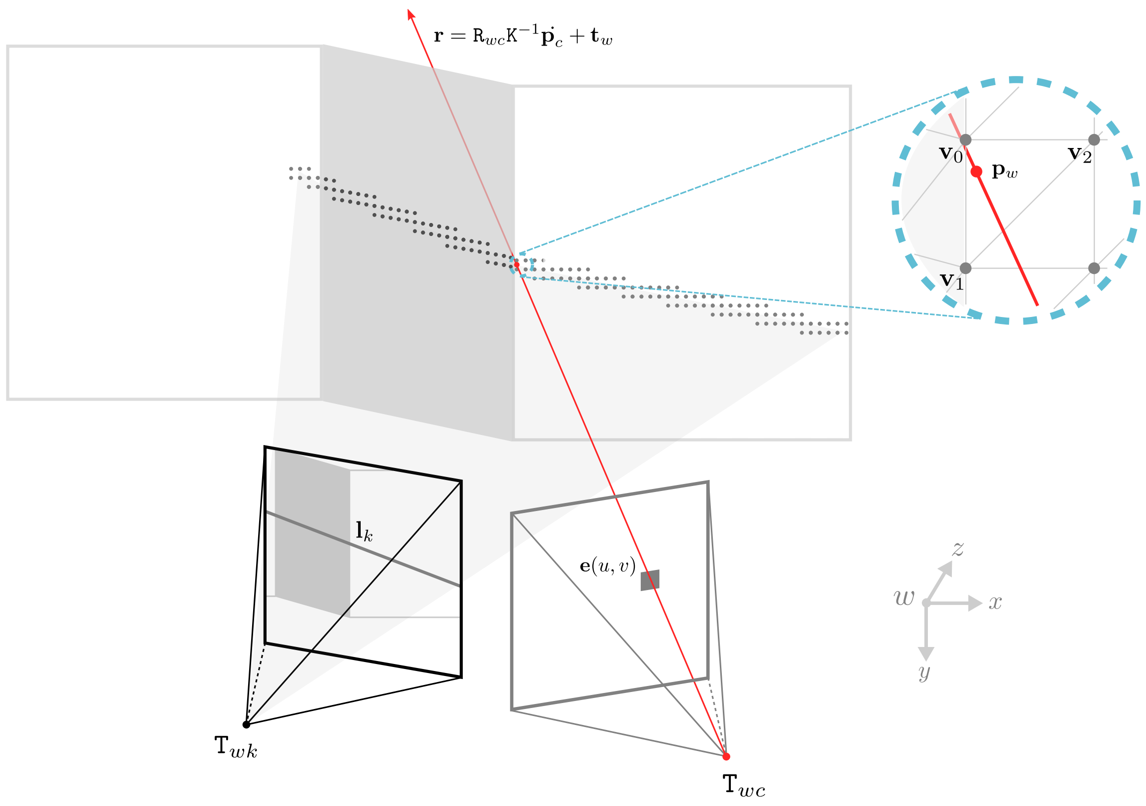


Figure 6.4: Corresponding 3D point search: when an event $\mathbf{e}(u, v)$ is received at a pixel location $\mathbf{p}_c = (u, v)^\top$ in the event camera reference of frame at $\mathtt{T}_{wc}$, we obtain the corresponding 3D point $\mathbf{p}_w$ in the world frame of reference w by examining whether the event pixel's ray intersects a triangle formed by any three adjacent vertices $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$ in the 3D point cloud using the ray-triangle intersection method [117]. We accelerate the computationally demanding ray casting method by restricting the number of search candidates to a subset of 3D points generated from pixels along the epipolar line $\mathbf{l}_k$ in the keyframe at $\mathtt{T}_{wk}$. Further speed-ups can be achieved by defining minimum and maximum depths and by starting from the depth of the previous intersection point based on the assumption of spatially smooth depth.

are the barycentric coordinates of the intersected point which is then used to calculate an interpolated log intensity.

The ray-triangle intersection 3D point search is however very computationally demanding especially if we inspect all possible triangles in the point cloud, and can be accelerated by restricting the number of search candidates to a subset of 3D points generated from pixels along the epipolar line $\mathbf{l}_k$ in the keyframe at $\mathtt{T}_{wk}$. Given the event pixel location $\mathbf{p}_c$, the current keyframe pose $\mathtt{T}_{wk}$, and the current event camera pose estimate $\mathtt{T}_{wc}$, the epipolar line $\mathbf{l}_k$ which encodes the line equation (i.e. $l_1 x + l_2 y + l_3 = 0$) in its 3-vectors form can be computed as [74, Chapter 9]:

$$\mathbf{l}_k = \mathtt{F}\dot{\mathbf{p}}_c \ , \tag{6.3}$$

where the fundamental matrix $\mathtt{F}$ is:

$$\mathtt{F} = \mathtt{K}^{-\top}\mathtt{R}_{kc}\mathtt{K}^{\top}\left[\mathtt{K}\mathtt{R}_{kc}^{\top}\mathbf{t}_{kc}\right]_{\times} \ , \tag{6.4}$$

and $[\cdot]_\times$ is a skew-symmetric matrix as follows [74, A4.5-p581]:

$$[\boldsymbol{\xi}]_\times = \begin{bmatrix} 0 & -\xi_3 & \xi_2 \\ \xi_3 & 0 & -\xi_1 \\ -\xi_2 & \xi_1 & 0 \end{bmatrix} \ . \tag{6.5}$$

Further speed-ups can be achieved by defining minimum and maximum depths which can make the search space smaller in some circumstances, and also by not starting from scratch (always starting from the minimum depth) but from the depth of the previous intersection point. This gives a higher chance of finding a corresponding point more quickly based on the spatially smooth depth assumption. In our current implementation, we use both tricks to keep the computation as low as possible.

6.2.2 Motion Prediction

We use a 6-DoF (translation and rotation) constant position motion model for motion prediction. The predicted camera pose at any given time remains constant while the variance of the prediction is proportional to the time interval as shown in Figure 4.5:

$$\mathbf{x}^{(t|t-\tau)} = \mathbf{x}^{(t-\tau|t-\tau)} + \mathbf{n} \ , \tag{6.6}$$

$$\mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} = \mathtt{P}_{\mathbf{x}}^{(t-\tau|t-\tau)} + \mathtt{P}_{\mathbf{n}} \ , \tag{6.7}$$

where each component of $\mathbf{n}$ is independent Gaussian noise in all six axes, i.e. $\mathbf{n}_i \sim \mathcal{N}(0, \sigma_i^2\tau)$, and $\mathtt{P}_{\mathbf{n}} = \mathrm{diag}(\sigma_1^2\tau, \ldots, \sigma_6^2\tau)$. The noise is the predicted change the current camera pose might have undergone since the previous event was received. We use the same time interval notation τ and τ_c as before, which are the time elapsed since the most recent previous event from *any pixel* and at *the same pixel* respectively.

6.2.3 Measurement Update

As shown in Figure 6.3, we calculate the value of a measurement $z_{\mathbf{x}}$ given an event $\mathbf{e}(u, v)$, the current keyframe pose $\mathtt{T}_{wk}$, the current camera pose estimate $\mathtt{T}_{wc}^{(t)}$, the previous pose estimate $\mathtt{T}_{wc}^{(t-\tau_c)}$ where the previous event was received at *the same pixel*, and a reconstructed image-like log intensity keyframe with inverse depth. We take the log intensity difference between two corresponding ray-triangle intersection points, $\mathbf{p}_w^{(t)}$ and $\mathbf{p}_w^{(t-\tau_c)}$:

$$z_{\mathbf{x}} = \pm C \; , \tag{6.8}$$

$$h_{\mathbf{x}}(\mathbf{x}^{(t|t-\tau)}) = \mathtt{I}_l\left(\mathbf{p}_w^{(t)}\right) - \mathtt{I}_l\left(\mathbf{p}_w^{(t-\tau_c)}\right) \; , \tag{6.9}$$

where:

$$\mathtt{I}_l\left(\mathbf{p}_w\right) = (1 - a - b)\mathtt{I}_l\left(\mathbf{v}_0\right) + a\mathtt{I}_l\left(\mathbf{v}_1\right) + b\mathtt{I}_l\left(\mathbf{v}_2\right) \; . \tag{6.10}$$

Here $\pm C$ is a known event threshold, its sign decided by the polarity of the event (i.e. $+C$ for a positive event, and $-C$ for a negative one), and $\mathtt{I}_l$ is a log intensity value based on the reconstructed log intensity keyframe. To obtain the corresponding 3D point location $\mathbf{p}_w$, a ray intersection point with respect to a triangle represented by three vertices $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$, in the world frame of reference, we use the ray-triangle intersection method detailed in Section 6.2.1. The method yields a vector $(l, a, b)^\top$, where l is the distance to the triangle from the origin of the ray and a, b are the barycentric coordinates of the intersected point which is then used to calculate an interpolated log intensity.

In the EKF framework, the camera pose estimate and its covariance matrix are updated by the standard equations at every event using:

$$\mathbf{x}^{(t|t)} = \mathbf{x}^{(t|t-\tau)} + \mathbf{W}_{\mathbf{x}}\nu_{\mathbf{x}} \; , \tag{6.11}$$

$$\mathtt{P}_{\mathbf{x}}^{(t|t)} = \left(\mathtt{I}_{6\times 6} - \mathbf{W}_{\mathbf{x}}\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}}\right)\mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} \; , \tag{6.12}$$

where the innovation $\nu_{\mathbf{x}}$ is:

$$\nu_{\mathbf{x}} = z_{\mathbf{x}} - h_{\mathbf{x}}(\mathbf{x}^{(t|t-\tau)}) \;, \tag{6.13}$$

the innovation covariance $S_{\mathbf{x}}$ is:

$$S_{\mathbf{x}} = \frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} \left(\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \right)^{\top} + \mathtt{N}_{\mathbf{x}} \;, \tag{6.14}$$

and the Kalman gain $\mathbf{W}_{\mathbf{x}}$ is:

$$\mathbf{W}_{\mathbf{x}} = \mathtt{P}_{\mathbf{x}}^{(t|t-\tau)} \left(\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} \right)^{\top} {S_{\mathbf{x}}}^{-1} \;. \tag{6.15}$$

The measurement uncertainty $\mathtt{N}_{\mathbf{x}}$ is a scalar variance $\sigma_{\mathbf{x}}^2$, and the important Jacobian $\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}}$, the partial derivative of the measurement function with respect to changes in camera pose, is derived as:

$$\begin{aligned}
\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} &= \frac{\partial}{\partial \mathbf{x}^{(t|t-\tau)}} \left(\mathtt{I}_l \left(\mathbf{p}_w^{(t)} \right) - \mathtt{I}_l \left(\mathbf{p}_w^{(t-\tau_c)} \right) \right) \;, && (6.16)\\
&= \frac{\partial}{\partial \mathbf{x}^{(t|t-\tau)}} \mathtt{I}_l \left(\mathbf{p}_w^{(t)} \right) - 0 \;, && (6.17)\\
&= \frac{\partial}{\partial \mathbf{x}^{(t|t-\tau)}} \left((1-a-b)\mathtt{I}_l \left(\mathbf{v}_0 \right) + a\mathtt{I}_l \left(\mathbf{v}_1 \right) + b\mathtt{I}_l \left(\mathbf{v}_2 \right) \right) \;, && (6.18)
\end{aligned}$$

where the barycentric coordinates a and b are computed as in [117]:

$$\begin{aligned}
a &= \frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))} ((\mathbf{o} - \mathbf{v}_0) \cdot (\mathbf{d} \times \mathbf{e}_2)) \;, && (6.19)\\
b &= \frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))} (\mathbf{d} \cdot ((\mathbf{o} - \mathbf{v}_0) \times \mathbf{e}_1)) \;, && (6.20)
\end{aligned}$$

when $\mathbf{o} = (o_0, o_1, o_2)^{\top}$ and $\mathbf{d} = (d_0, d_1, d_2)^{\top}$ are the origin and direction vectors of the ray (i.e. camera pose) intersected at $\mathbf{p}_w^{(t)}$, $\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0$, and $\mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$.

If we simply denote $((1-a-b)\mathtt{I}_l \left(\mathbf{v}_0 \right) + a\mathtt{I}_l \left(\mathbf{v}_1 \right) + b\mathtt{I}_l \left(\mathbf{v}_2 \right))$ as $\mathtt{I}_l$, then using the chain rule, $\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}}$ becomes:

$$\frac{\partial h_{\mathbf{x}}}{\partial \mathbf{x}^{(t|t-\tau)}} = \frac{\partial \mathtt{I}_l}{\partial \begin{bmatrix} a \\ b \end{bmatrix}} \frac{\partial \begin{bmatrix} a \\ b \end{bmatrix}}{\partial \begin{bmatrix} \mathbf{o} \\ \mathbf{d} \end{bmatrix}} \frac{\partial \begin{bmatrix} \mathbf{o} \\ \mathbf{d} \end{bmatrix}}{\partial \mathbf{x}^{(t|t-\tau)}} \ , \tag{6.21}$$

where $\frac{\partial \mathtt{I}_l}{\partial \begin{bmatrix} a \\ b \end{bmatrix}}$ is:

$$\frac{\partial \mathtt{I}_l}{\partial \begin{bmatrix} a \\ b \end{bmatrix}} = \begin{bmatrix} \mathtt{I}_l\left(\mathbf{v}_1\right) - \mathtt{I}_l\left(\mathbf{v}_0\right) & \mathtt{I}_l\left(\mathbf{v}_2\right) - \mathtt{I}_l\left(\mathbf{v}_0\right)) \end{bmatrix} \ , \tag{6.22}$$

$\frac{\partial \begin{bmatrix} a \\ b \end{bmatrix}}{\partial \begin{bmatrix} \mathbf{o} \\ \mathbf{d} \end{bmatrix}}$ is:

$$\frac{\partial \begin{bmatrix} a \\ b \end{bmatrix}}{\partial \begin{bmatrix} \mathbf{o} \\ \mathbf{d} \end{bmatrix}} = \begin{bmatrix} \frac{\partial a}{\partial \mathbf{o}} & \frac{\partial a}{\partial \mathbf{d}} \\ \frac{\partial b}{\partial \mathbf{o}} & \frac{\partial b}{\partial \mathbf{d}} \end{bmatrix} \ , \tag{6.23}$$

where $\frac{\partial a}{\partial \mathbf{o}}$ is:

$$\frac{\partial a}{\partial \mathbf{o}} = \left(\frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))} (\mathbf{d} \times \mathbf{e}_2) \right)^{\top} \ , \tag{6.24}$$

$\frac{\partial a}{\partial \mathbf{d}}$ is:

$$\frac{\partial a}{\partial \mathbf{d}} = \left(- \frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))} ((\mathbf{o} - \mathbf{v}_0) \times \mathbf{e}_2) + ((\mathbf{o} - \mathbf{v}_0) \cdot (\mathbf{d} \times \mathbf{e}_2)) \frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))^2} (\mathbf{e}_1 \times \mathbf{e}_2) \right)^{\top} \ , \tag{6.25}$$

$\frac{\partial b}{\partial \mathbf{o}}$ is:

$$\frac{\partial b}{\partial \mathbf{o}} = \left(-\frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))}(\mathbf{d} \times \mathbf{e}_1)\right)^\top , \tag{6.26}$$

and $\frac{\partial b}{\partial \mathbf{d}}$ is:

$$\frac{\partial b}{\partial \mathbf{d}} = \left(\frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))}((\mathbf{o} - \mathbf{v}_0) \times \mathbf{e}_1) + (\mathbf{d} \cdot ((\mathbf{o} - \mathbf{v}_0) \times \mathbf{e}_1))\frac{1}{(\mathbf{e}_1 \cdot (\mathbf{d} \times \mathbf{e}_2))^2}(\mathbf{e}_1 \times \mathbf{e}_2)\right)^\top , \tag{6.27}$$

and $\frac{\partial \begin{bmatrix} \mathbf{o} \\ \mathbf{d} \end{bmatrix}}{\partial \mathbf{x}^{(t|t-\tau)}}$ is:

$$\frac{\partial \begin{bmatrix} \mathbf{o} \\ \mathbf{d} \end{bmatrix}}{\partial \mathbf{x}^{(t|t-\tau)}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_2 & -d_1 \\ 0 & 0 & 0 & -d_2 & 0 & d_0 \\ 0 & 0 & 0 & d_1 & -d_0 & 0 \end{bmatrix} . \tag{6.28}$$

6.3 Gradient Estimation and Log Intensity Reconstruction

We now use the updated camera pose estimate to incrementally improve the estimates of the log intensity gradient at each keyframe pixel based on a pixel-wise EKF. Essentially, each new event which lines up with a particular keyframe pixel improves our estimate of its gradient in the direction parallel to the motion of the camera over the scene at that pixel while we learn nothing about the gradient in the direction perpendicular to camera motion. However, because of the random walk nature of our tracker which generates noisy motion estimates, we first apply a weighted average filter to the new camera pose estimate $\tilde{\mathbf{x}}$ as:

$$\mathbf{x}^{(t)} = (1 - \gamma)\mathbf{x}^{(t-\tau)} + \gamma\tilde{\mathbf{x}}. \tag{6.29}$$

where γ is a weight variable which controls the smoothness of estimated camera motion. In fact, a small γ value implicitly implements a strong accumulation of events, but our method still provides an event-by-event updated estimate of the position and orientation of the event camera as they arrive.

Note that the high average frequency of events relative to the dynamics of a hand-held camera strongly motivates the use of a stronger motion model (e.g. constant velocity or acceleration) [66], and this is an important issue for future research.

6.3.1 Pixel-Wise EKF-Based Gradient Estimation

Each pixel of the keyframe holds an independent gradient estimate $\mathbf{g}(\mathbf{p}_k) = (g_u, g_v)^\top$, consisting of log intensity gradients g_u and g_v along the horizontal and vertical axes in image space respectively, and a 2×2 uncertainty covariance matrix $\mathtt{P}_\mathbf{g}(\mathbf{p}_k)$. At initialisation, all gradients are initialised to zero with large variances like the ones used in the experiments shown in Table 6.2.

Based on the rapidity of events, we assume a linear change of gradient between two consecutive events at the same event camera pixel, and update the midpoint $\hat{\mathbf{p}}_k$ of the two projected points $\mathbf{p}_k^{(t)}$ and $\mathbf{p}_k^{(t-\tau_c)}$ as illustrated in Figure 6.5. We now define $z_\mathbf{g}$, a measurement of the instantaneous *event rate* at this pixel, and its measurement model $h_\mathbf{g}$ based on the brightness constancy equation $(\mathbf{g} \cdot \mathbf{m})\tau_c = \pm C$ [76], where $\mathbf{g}$ is a gradient estimate and $\mathbf{m} = (m_u, m_v)^\top$ is a motion vector — the displacement between two corresponding pixels in the current keyframe divided by the elapsed time τ_c:

$$z_\mathbf{g} = \pm \frac{C}{\tau_c} , \tag{6.30}$$

$$h_\mathbf{g} = (\mathbf{g}(\hat{\mathbf{p}}_k) \cdot \mathbf{m}) , \tag{6.31}$$

where:

$$\mathbf{m} = \frac{\mathbf{p}_k^{(t)} - \mathbf{p}_k^{(t-\tau_c)}}{\tau_c} . \tag{6.32}$$

Based on the standard EKF equations, the current gradient estimate and its uncertainty covariance matrix at that pixel are updated independently as:

$$\mathbf{g}(\hat{\mathbf{p}}_k)^{(t)} = \mathbf{g}(\hat{\mathbf{p}}_k)^{(t-\tau_c)} + \mathbf{W}_\mathbf{g}\nu_\mathbf{g} , \tag{6.33}$$

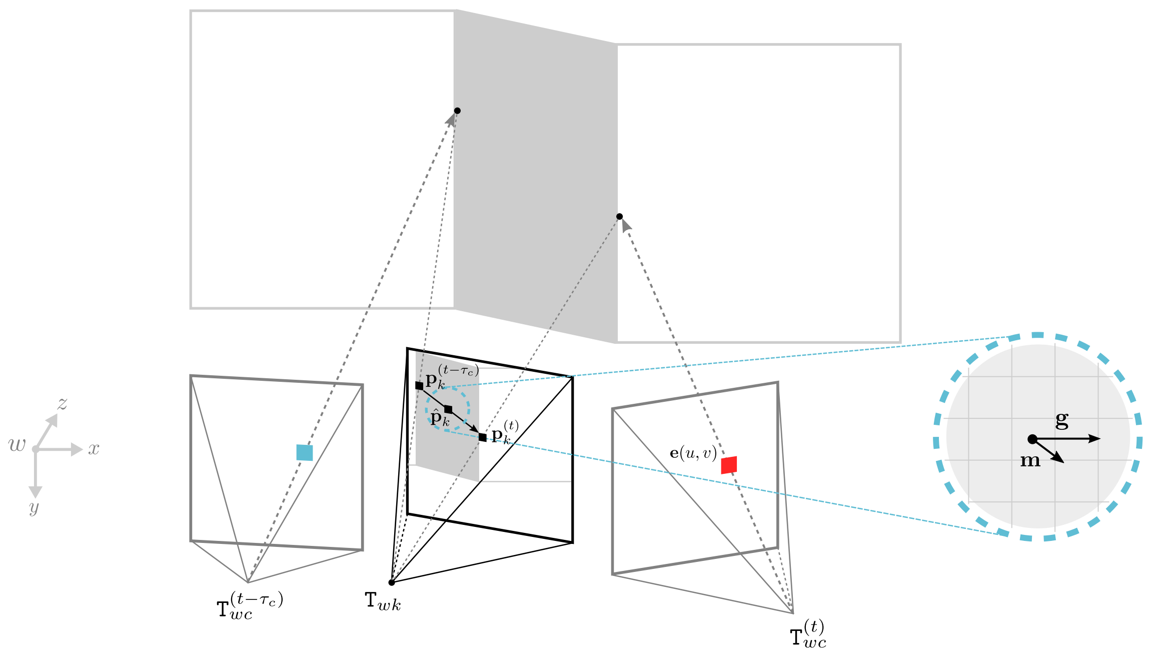


Figure 6.5: Gradient estimation: based on the assumption that the current camera pose estimates ($\mathtt{T}_{wc}^{(t)}$ and $\mathtt{T}_{wc}^{(t-\tau_c)}$) and inverse depth estimate (shown as the geometry of the walls) are correct, we first project two intersection points onto the current keyframe, $\mathbf{p}_k^{(t)}$ and $\mathbf{p}_k^{(t-\tau_c)}$. We then find a displacement vector between them, which is used to calculate a motion vector $\mathbf{m}$ to compute the value of a measurement ($\mathbf{g} \cdot \mathbf{m}$) at a midpoint, $\hat{\mathbf{p}}_k$, based on the brightness constancy and the linear gradient assumption. We update each independent gradient estimate based on the pixel-wise EKF framework. Essentially, each new event which lines up with a particular keyframe pixel improves our estimate of its gradient in the direction parallel to the motion of the camera over the scene at that pixel while we learn nothing about the gradient in the direction perpendicular to camera motion.

$$\mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_k)^{(t)} = \left(\mathtt{I}_{2\times 2} - \mathbf{W}_{\mathbf{g}} \frac{\partial h_{\mathbf{g}}}{\partial \mathbf{g}(\hat{\mathbf{p}}_k)^{(t-\tau_c)}} \right) \mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_k)^{(t-\tau_c)} , \tag{6.34}$$

where the innovation $\nu_{\mathbf{g}}$ is:

$$\nu_{\mathbf{g}} = z_{\mathbf{g}} - h_{\mathbf{g}} , \tag{6.35}$$

the innovation covariance $S_{\mathbf{g}}$ is:

$$S_{\mathbf{g}} = \frac{\partial h_{\mathbf{g}}}{\partial \mathbf{g}(\hat{\mathbf{p}}_k)^{(t-\tau_c)}} \mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_k)^{(t-\tau_c)} \left(\frac{\partial h_{\mathbf{g}}}{\partial \mathbf{g}(\hat{\mathbf{p}}_k)^{(t-\tau_c)}} \right)^{\top} + \mathtt{N}_{\mathbf{g}} \,, \tag{6.36}$$

and the Kalman gain $\mathbf{W}_{\mathbf{g}}$ is:

$$\mathbf{W}_{\mathbf{g}} = \mathtt{P}_{\mathbf{g}}(\hat{\mathbf{p}}_k)^{(t-\tau_c)} \left(\frac{\partial h_{\mathbf{g}}}{\partial \mathbf{g}(\hat{\mathbf{p}}_k)^{(t-\tau_c)}} \right)^{\top} S_{\mathbf{g}}^{-1} \,. \tag{6.37}$$

The Jacobian of the measurement function with respect to changes in gradient is:

$$\frac{\partial h_{\mathbf{g}}}{\partial \mathbf{g}(\hat{\mathbf{p}}_k)^{(t-\tau_c)}} = \frac{\partial}{\partial (g_u, g_v)^{\top}} (g_u m_u + g_v m_v) = (m_u, m_v), \tag{6.38}$$

and the measurement noise $\mathtt{N}_{\mathbf{g}}$ is:

$$\mathtt{N}_{\mathbf{g}} = \frac{\partial z_{\mathbf{g}}}{\partial C} \mathtt{P}_C \left(\frac{\partial z_{\mathbf{g}}}{\partial C} \right)^{\top} = \frac{\sigma_C^2}{\tau_c^2} \,, \tag{6.39}$$

where σ_C^2 is the sensor noise with respect to the event threshold.

However, once $\nu_{\mathbf{g}}$ and $S_{\mathbf{g}}$ have been computed, in order to be more robust to scene changes which violate the static scene assumption, a simple Mahalanobis distance D_M based outlier rejection is performed. The EKF update equations are employed only if D_M is within a confidence limit (e.g. 3σ), where D_M is:

$$D_M = \sqrt{\nu_{\mathbf{g}}^{\top} S_{\mathbf{g}}^{-1} \nu_{\mathbf{g}}} \,. \tag{6.40}$$

6.3.2 Log Intensity Reconstruction

Along with the pixel-wise EKF based gradient estimation method, we perform interleaved absolute log intensity reconstruction running on a GPU. As described in detail in Section 5.3.2, we define our convex minimisation function as:

$$\min_{\mathtt{I}_l} \left\{ \int_{\Omega} ||\mathbf{g}(\mathbf{p}_k) - \nabla \mathtt{I}_l(\mathbf{p}_k)||_{\epsilon_d}^{h} + \lambda ||\nabla \mathtt{I}_l(\mathbf{p}_k)||_{\epsilon_r}^{h} \mathrm{d}\mathbf{p}_k \right\} \,. \tag{6.41}$$

Here the data term represents the error between estimated gradients $\mathbf{g}(\mathbf{p}_k)$ and those of a reconstructed log intensity $\nabla \mathtt{I}_l(\mathbf{p}_k)$, and the regularisation term enforces smoothness, both under a robust Huber norm. This function can be written using the Legendre Fenchel transformation as follows [69]:

$$\min_{\mathtt{I}_l} \max_{\mathbf{q}} \max_{\mathbf{p}} \{ \langle \mathbf{p}, \mathbf{g} - \nabla \mathtt{I}_l \rangle - \frac{\epsilon_d}{2} ||\mathbf{p}||^2 - \delta_{\mathbf{p}}(\mathbf{p}) + \langle \mathbf{q}, \nabla \mathtt{I}_l \rangle - \frac{\epsilon_r}{2\lambda} ||\mathbf{q}||^2 - \delta_{\mathbf{q}}(\mathbf{q}) \} \ , \tag{6.42}$$

where we can solve by maximising with respect to $\mathbf{p}$:

$$\mathbf{p}^{(n+1)} = \frac{\frac{\mathbf{p}^{(n)} + \sigma_{\mathbf{p}}(\mathbf{g} - \nabla \mathtt{I}_l)}{1 + \sigma_{\mathbf{p}} \epsilon_d}}{\max\left(1, \left| \frac{\mathbf{p}^{(n)} + \sigma_{\mathbf{p}}(\mathbf{g} - \nabla \mathtt{I}_l)}{1 + \sigma_{\mathbf{p}} \epsilon_d} \right| \right)} \ , \tag{6.43}$$

maximising with respect to $\mathbf{q}$:

$$\mathbf{q}^{(n+1)} = \frac{\frac{\mathbf{q}^{(n)} + \sigma_{\mathbf{q}} \nabla \mathtt{I}_l}{1 + \frac{\sigma_{\mathbf{q}} \epsilon_r}{\lambda}}}{\max\left(1, \frac{1}{\lambda} \left| \frac{\mathbf{q}^{(n)} + \sigma_{\mathbf{q}} \nabla \mathtt{I}_l}{1 + \frac{\sigma_{\mathbf{q}} \epsilon_r}{\lambda}} \right| \right)} \ , \tag{6.44}$$

and minimising with respect to $\mathtt{I}_l$:

$$\mathtt{I}_l^{(n+1)} = \mathtt{I}_l^{(n)} - \sigma_{\mathtt{I}_l} \left(\mathrm{div}\mathbf{p}^{(n+1)} - \mathrm{div}\mathbf{q}^{(n+1)} \right) \ . \tag{6.45}$$

These steps are iterated until either the change of the reconstructed log intensity values becomes lower than a certain threshold $\Delta \mathtt{I}_l$, or the number of iterations reaches a pre-set maximum number of iterations n_{max}. We visualise the progress of gradient estimation and log intensity reconstruction over time during hand-held event camera motion in Figure 6.6 and Figure 6.7 respectively.

6.4 Inverse Depth Estimation and Regularisation

We use the same smoothed camera pose estimate from the weighted average filter described in Section 6.3 to incrementally improve the estimates of the inverse depth at each keyframe pixel based on another pixel-wise EKF. In order to get more accurate and consistent depth

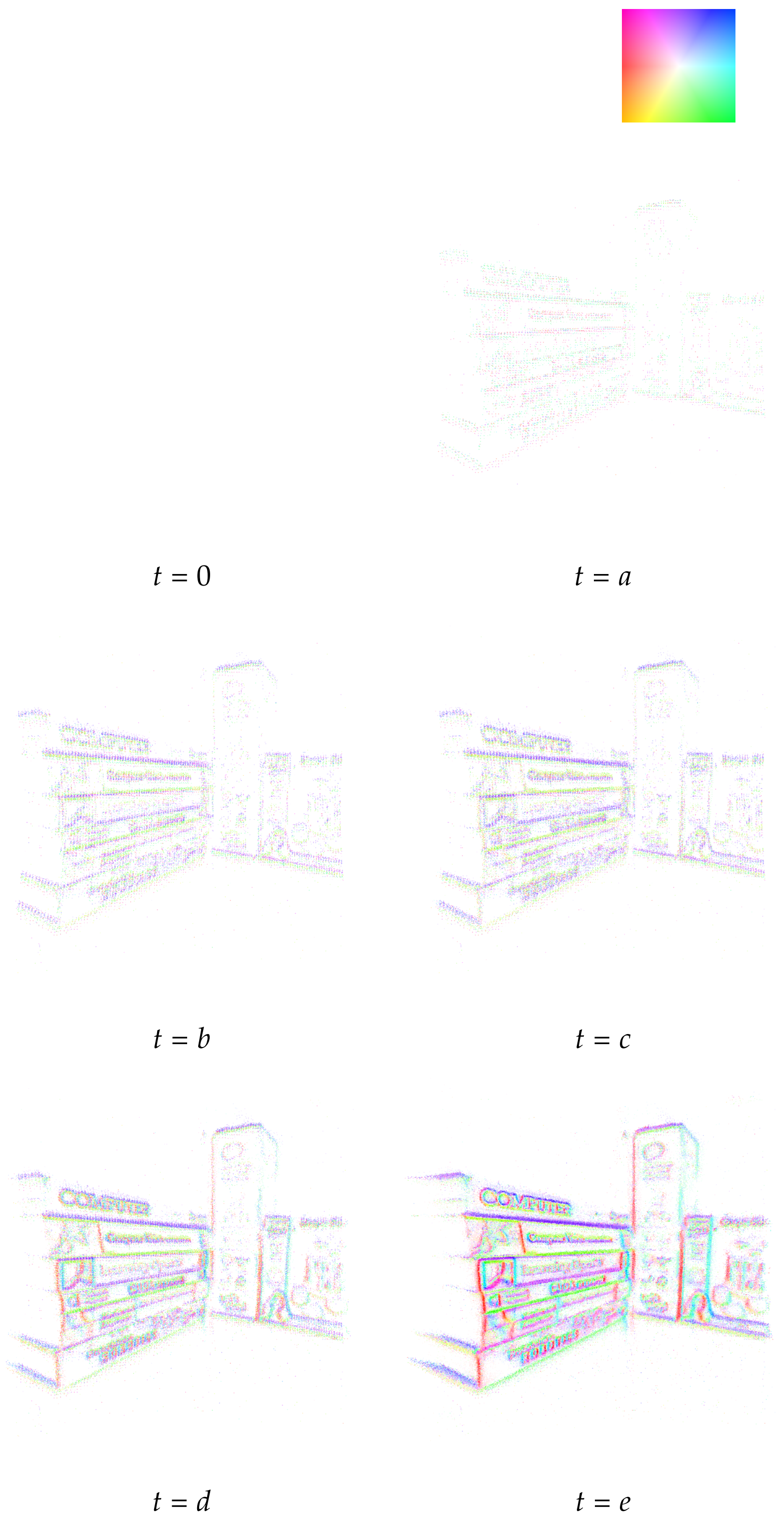


Figure 6.6: Typical temporal progression ($0 < a < b < c < d < e$) of gradient estimation as a hand-held camera browses a 3D scene. The colours and intensities represent the orientations and strengths of the gradients of the scene (refer to the colour chart in the top right).

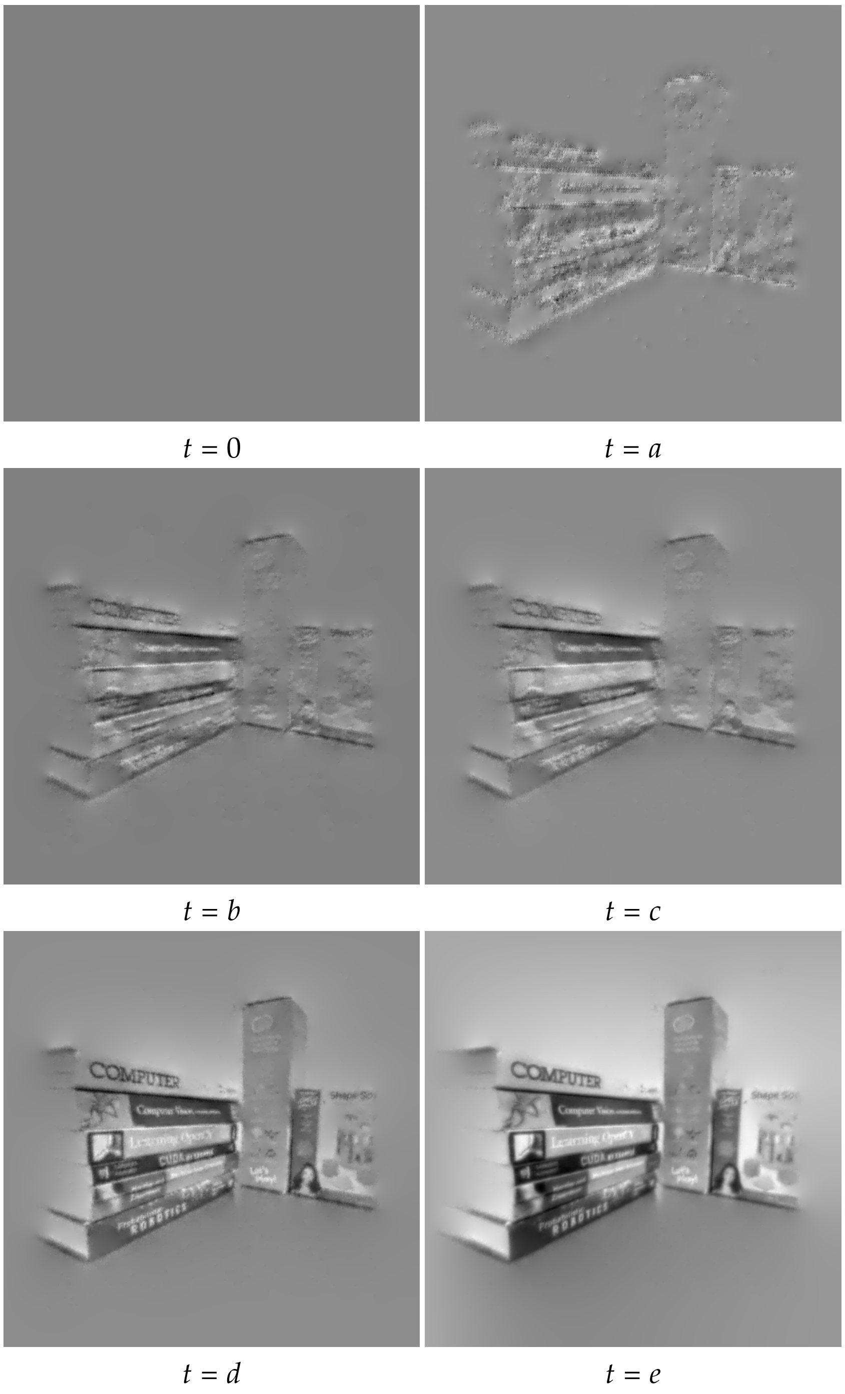


Figure 6.7: Typical temporal progression ($0 < a < b < c < d < e$) of log intensity reconstruction as a hand-held camera browses a 3D scene.

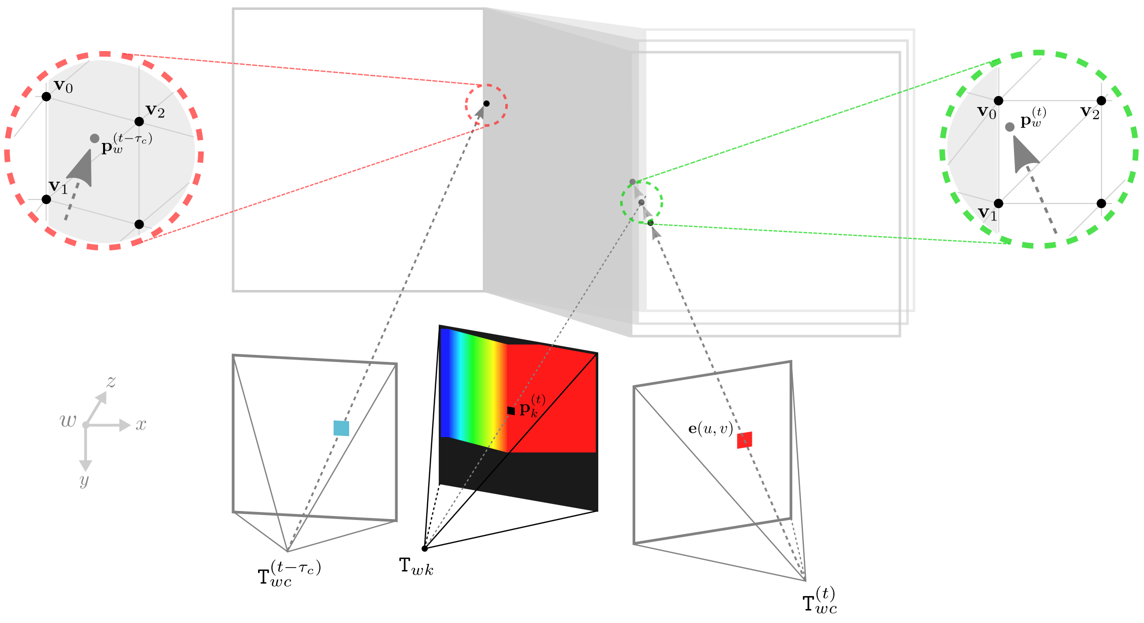


Figure 6.8: Inverse depth estimation: we assume that the current reconstructed log intensity (shown as the colour of the walls) and camera pose estimates ($\mathtt{T}_{wc}^{(t)}$ and $\mathtt{T}_{wc}^{(t-\tau_c)}$) are correct, and find the inverse depth most consistent with the predicted log intensity change since the previous event at the same pixel. Specifically, we first find two corresponding ray-triangle intersection points, $\mathbf{p}_w^{(t)}$ and $\mathbf{p}_w^{(t-\tau_c)}$, in the world frame of reference w using the ray-triangle intersection method detailed in Section 6.2.1. We then compute the value of a measurement, a log intensity difference between two points given an event $\mathbf{e}(u,v)$, the current keyframe pose $\mathtt{T}_{wk}$, the current camera pose estimate $\mathtt{T}_{wc}^{(t)}$, the previous pose estimate $\mathtt{T}_{wc}^{(t-\tau_c)}$ and the reconstructed log intensity and inverse depth keyframe. We update each independent inverse depth estimate based on the pixel-wise EKF framework.

estimation, we use the inverse depth parametrisation [119, 31] which is well suited to representing uncertainty using a Gaussian distribution over a huge range of depths, in particular for distant points, even at infinity (i.e. low parallax cases).

As illustrated in Figure 6.8, by assuming that the current camera pose estimates and reconstructed log intensity are correct, we find the inverse depth estimate which best predicts a log intensity change compared to the previous log intensity value recorded when the same event pixel got the previous event at pose $\mathtt{T}_{wc}^{(t-\tau_c)}$ consistent with the polarity of the event just received.

6.4.1 Pixel-Wise EKF-Based Inverse Depth Estimation

Each pixel of the keyframe holds an independent inverse depth state value $\rho(\mathbf{p}_k)$ with variance $\sigma^2_{\rho(\mathbf{p}_k)}$. At initialisation, all inverse depths are initialised to nominal values with large variances — the initial values ρ_0 and σ_{ρ_0} used in our experiments are shown in Table 6.2.

In the same way as in our tracking method, we calculate the value of measurement $z_{\boldsymbol{\rho}}$ which is the log intensity difference between two corresponding ray-triangle intersection points, $\mathbf{p}_w^{(t)}$ and $\mathbf{p}_w^{(t-\tau_c)}$, as shown in Figure 6.8:

$$z_{\boldsymbol{\rho}} = \pm C \ , \tag{6.46}$$

$$h_{\boldsymbol{\rho}} = \mathtt{I}_l\left(\mathbf{p}_w^{(t)}\right) - \mathtt{I}_l\left(\mathbf{p}_w^{(t-\tau_c)}\right) \ , \tag{6.47}$$

where:

$$\mathtt{I}_l\left(\mathbf{p}_w\right) = (1-a-b)\mathtt{I}_l\left(\mathbf{v}_0\right) + a\mathtt{I}_l\left(\mathbf{v}_1\right) + b\mathtt{I}_l\left(\mathbf{v}_2\right) \ . \tag{6.48}$$

Again here $\pm C$ is a known event threshold — its sign is decided by the polarity of an event (i.e. $+C$ for a positive event, and $-C$ for a negative one), and $\mathtt{I}_l$ is a log intensity value based on a reconstructed log intensity keyframe. To obtain a corresponding 3D point location $\mathbf{p}_w$, a ray intersection point with respect to a triangle represented by three vertices $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$, in the world frame of reference w, we use the ray-triangle intersection method detailed in Section 6.2.1, which yields a vector $(l, a, b)^\top$ where l is the distance to the triangle from the origin of the ray and a, b are the barycentric coordinates of the intersected point which is then used to calculate an interpolated log intensity.

In the EKF framework, we stack the inverse depths of all three vertices of the intersected triangle $\boldsymbol{\rho} = (\rho_{\mathbf{v}_0}, \rho_{\mathbf{v}_1}, \rho_{\mathbf{v}_2})^\top$ which contributed to the intersection 3D point and update them with their associated uncertainty covariance matrix $\mathtt{P}_{\boldsymbol{\rho}} = \mathrm{diag}\left(\sigma^2_{\rho_{\mathbf{v}_0}}, \sigma^2_{\rho_{\mathbf{v}_1}}, \sigma^2_{\rho_{\mathbf{v}_2}}\right)$ using the standard equations at every event as:

$$\boldsymbol{\rho}^{(t)} = \boldsymbol{\rho}^{(t-\tau_c)} + \mathbf{W}_{\boldsymbol{\rho}}\nu_{\boldsymbol{\rho}} \ , \tag{6.49}$$

$$\mathtt{P}_{\boldsymbol{\rho}}^{(t)} = \left(\mathtt{I}_{3\times 3} - \mathbf{W}_{\boldsymbol{\rho}}\frac{\partial h_{\boldsymbol{\rho}}}{\partial \boldsymbol{\rho}^{(t-\tau_c)}}\right)\mathtt{P}_{\boldsymbol{\rho}}^{(t-\tau_c)} \ , \tag{6.50}$$

where the innovation ν_ρ is:

$$\nu_\rho = z_\rho - h_\rho \ , \tag{6.51}$$

the innovation covariance S_ρ is:

$$S_\rho = \frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \mathtt{P}_\rho^{(t-\tau_c)} \left(\frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \right)^\top + \mathtt{N}_\rho \ , \tag{6.52}$$

and the Kalman gain $\mathbf{W}_\rho$ is:

$$\mathbf{W}_\rho = \mathtt{P}_\rho^{(t-\tau_c)} \left(\frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \right)^\top {S_\rho}^{-1} \ . \tag{6.53}$$

The measurement noise $\mathtt{N}_\rho$ is a scalar variance σ_ρ^2, and the important Jacobian $\frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}}$, the partial derivative of the measurement function with respect to changes in inverse depths, is derived as:

$$\begin{aligned}
\frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} &= \frac{\partial}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \left(\mathtt{I}_l \left(\mathbf{p}_w^{(t)} \right) - \mathtt{I}_l \left(\mathbf{p}_w^{(t-\tau_c)} \right) \right) \ , & (6.54) \\
&= \frac{\partial}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \mathtt{I}_l \left(\mathbf{p}_w^{(t)} \right) - 0 \ , & (6.55) \\
&= \frac{\partial}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \left((1-a-b)\mathtt{I}_l\left(\mathbf{v}_0\right) + a\mathtt{I}_l\left(\mathbf{v}_1\right) + b\mathtt{I}_l\left(\mathbf{v}_2\right) \right) \ . & (6.56)
\end{aligned}$$

Again, if we simply denote $\left((1-a-b)\mathtt{I}_l\left(\mathbf{v}_0\right) + a\mathtt{I}_l\left(\mathbf{v}_1\right) + b\mathtt{I}_l\left(\mathbf{v}_2\right) \right)$ as $\mathtt{I}_l$, then using the chain rule, $\frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}}$ becomes:

$$\frac{\partial h_\rho}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} = \frac{\partial \mathtt{I}_l}{\partial \begin{bmatrix} a \\ b \end{bmatrix}} \frac{\partial \begin{bmatrix} a \\ b \end{bmatrix}}{\partial \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}} \frac{\partial \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} \ , \tag{6.57}$$

where $\frac{\partial \mathtt{I}_l}{\partial \begin{bmatrix} a \\ b \end{bmatrix}}$ is the same as in Section 6.2.3:

$$\frac{\partial \mathtt{I}_l}{\partial \begin{bmatrix} a \\ b \end{bmatrix}} = \begin{bmatrix} \mathtt{I}_l\left(\mathbf{v}_1\right) - \mathtt{I}_l\left(\mathbf{v}_0\right) & \mathtt{I}_l\left(\mathbf{v}_2\right) - \mathtt{I}_l\left(\mathbf{v}_0\right)) \end{bmatrix} , \tag{6.58}$$

$\frac{\partial \begin{bmatrix} a \\ b \end{bmatrix}}{\partial \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}}$ is:

$$\frac{\partial \begin{bmatrix} a \\ b \end{bmatrix}}{\partial \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}} = \begin{bmatrix} \frac{\partial a}{\partial \mathbf{v}_0} & \frac{\partial a}{\partial \mathbf{v}_1} & \frac{\partial a}{\partial \mathbf{v}_2} \\ \frac{\partial b}{\partial \mathbf{v}_0} & \frac{\partial b}{\partial \mathbf{v}_1} & \frac{\partial b}{\partial \mathbf{v}_2} \end{bmatrix} , \tag{6.59}$$

where $\frac{\partial a}{\partial \mathbf{v}_0}$ is:

$$\begin{aligned} \frac{\partial a}{\partial \mathbf{v}_0} = \Bigg(&((\mathbf{o} - \mathbf{v}_0) \times \mathbf{d})\frac{1}{((\mathbf{d} \times \mathbf{e}_2) \cdot (\mathbf{v}_0 - \mathbf{v}_1))} + (\mathbf{d} \times \mathbf{e}_2)\frac{1}{((\mathbf{d} \times \mathbf{e}_2) \cdot (\mathbf{v}_0 - \mathbf{v}_1))} \\ &+ ((\mathbf{o} - \mathbf{v}_0) \cdot (\mathbf{d} \times \mathbf{e}_2))((\mathbf{d} \times (\mathbf{v}_0 - \mathbf{v}_1)) - (\mathbf{d} \times (\mathbf{v}_0 - \mathbf{v}_2)))\frac{1}{((\mathbf{d} \times \mathbf{e}_2) \cdot (\mathbf{v}_0 - \mathbf{v}_1))^2}\Bigg)^\top , \end{aligned} \tag{6.60}$$

$\frac{\partial a}{\partial \mathbf{v}_1}$ is:

$$\frac{\partial a}{\partial \mathbf{v}_1} = \left(-((\mathbf{o} - \mathbf{v}_0) \cdot (\mathbf{d} \times \mathbf{e}_2))(\mathbf{d} \times \mathbf{e}_2)\frac{1}{((\mathbf{d} \times \mathbf{e}_2) \cdot (\mathbf{v}_0 - \mathbf{v}_1))^2}\right)^\top , \tag{6.61}$$

$\frac{\partial a}{\partial \mathbf{v}_2}$ is:

$$\frac{\partial a}{\partial \mathbf{v}_2} = \Bigg(-((\mathbf{o}-\mathbf{v}_0)\times\mathbf{d})\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))} \\ -((\mathbf{o}-\mathbf{v}_0)\cdot(\mathbf{d}\times\mathbf{e}_2))(\mathbf{d}\times(\mathbf{v}_0-\mathbf{v}_1))\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))^2}\Bigg)^\top , \tag{6.62}$$

$\frac{\partial b}{\partial \mathbf{v}_0}$ is:

$$\frac{\partial b}{\partial \mathbf{v}_0} = \Bigg((\mathbf{d}\times(\mathbf{o}-\mathbf{v}_1))\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))} \\ +(\mathbf{d}\cdot((\mathbf{o}-\mathbf{v}_0)\times\mathbf{e}_1))((\mathbf{d}\times(\mathbf{v}_0-\mathbf{v}_1))-(\mathbf{d}\times(\mathbf{v}_0-\mathbf{v}_2)))\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))^2}\Bigg)^\top , \tag{6.63}$$

$\frac{\partial b}{\partial \mathbf{v}_1}$ is:

$$\frac{\partial b}{\partial \mathbf{v}_1} = \Bigg(((\mathbf{o}-\mathbf{v}_0)\times\mathbf{d})\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))} \\ -(\mathbf{d}\cdot((\mathbf{o}-\mathbf{v}_0)\times\mathbf{e}_1))(\mathbf{d}\times\mathbf{e}_2)\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))^2}\Bigg)^\top , \tag{6.64}$$

and $\frac{\partial b}{\partial \mathbf{v}_2}$ is:

$$\frac{\partial b}{\partial \mathbf{v}_2} = \Bigg(-(\mathbf{d}\cdot((\mathbf{o}-\mathbf{v}_0)\times\mathbf{e}_1))(\mathbf{d}\times(\mathbf{v}_0-\mathbf{v}_1))\frac{1}{((\mathbf{d}\times\mathbf{e}_2)\cdot(\mathbf{v}_0-\mathbf{v}_1))^2}\Bigg)^\top , \tag{6.65}$$

and $\frac{\partial \begin{bmatrix}\mathbf{v}_0\\ \mathbf{v}_1\\ \mathbf{v}_2\end{bmatrix}}{\partial \boldsymbol{\rho}^{(t-\tau_c)}}$ is:

$$\frac{\partial \begin{bmatrix}\mathbf{v}_0\\ \mathbf{v}_1\\ \mathbf{v}_2\end{bmatrix}}{\partial \boldsymbol{\rho}^{(t-\tau_c)}} = \begin{bmatrix} -\mathrm{R}_{wk}\begin{bmatrix}\mathbf{p}_k\\ 1\end{bmatrix}\frac{1}{\rho^2_{\mathbf{v}_0}} & \vec{0} & \vec{0} \\ \vec{0} & -\mathrm{R}_{wk}\begin{bmatrix}\mathbf{p}_k\\ 1\end{bmatrix}\frac{1}{\rho^2_{\mathbf{v}_1}} & \vec{0} \\ \vec{0} & \vec{0} & -\mathrm{R}_{wk}\begin{bmatrix}\mathbf{p}_k\\ 1\end{bmatrix}\frac{1}{\rho^2_{\mathbf{v}_2}} \end{bmatrix} . \tag{6.66}$$

As in gradient estimation, once ν_ρ and S_ρ have been computed, in order to be more robust to scene changes which violate the static scene assumption a simple Mahalanobis distance D_M based outlier rejection check is performed — the EKF update equations are employed only if D_M is within a confidence limit (e.g. 3σ), where D_M is:

$$D_M = \sqrt{\nu_\rho^\top S_\rho^{-1} \nu_\rho} \; . \tag{6.67}$$

6.4.2 Inverse Depth Regularisation

As a background process running on a GPU, we perform inverse depth regularisation on keyframe pixels whenever the change of the inverse depth estimates becomes higher than a certain threshold $\Delta\mathtt{D}_\rho$. We penalise deviation from a spatially smooth inverse depth map by assigning each inverse depth value the average of its neighbouring pixels in a given local window $\mathtt{W}_\rho$, weighted by their respective inverse variances as described in [56]. If a neighbour's inverse depth is different more than $2\sigma_\rho$, it is not taken into account during the regularisation step to preserve discontinuities due to occlusion boundaries.

We visualise the progress of inverse depth estimation and regularisation over time as event data is captured during hand-held event camera motion in Figure 6.9 and Figure 6.10. The parameters used in the experiments are given in Table 6.2.

6.5 Evaluation and Results

Our algorithm runs in real-time on a standard PC with typical scenes and motion speed, and we have conducted experiments both indoors and outdoors. We recommend viewing our video[1] which illustrates all of the key results in a better form than still pictures and in real-time (also see Appendix A).

In all experiments, we have used a DVS camera from iniLabs[2] with 128×128 resolution, 120 dB dynamic range, and 15 microsecond latency, and communicated with a host computer using our own USB 2.0 driver — readers can instead use the one provided by the manufacturer[3]. The camera has pre-calibrated intrinsics and all event pixel locations are pre-warped to remove radial distortion via the event camera calibration method described

[1]Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera: https://youtu.be/yHLyhdMSw7w (accessed September 2017)

[2]iniLabs Ltd: www.inilabs.com (accessed September 2017)

[3]jAER Open Source Project: https://github.com/SensorsINI/jaer (accessed September 2017)

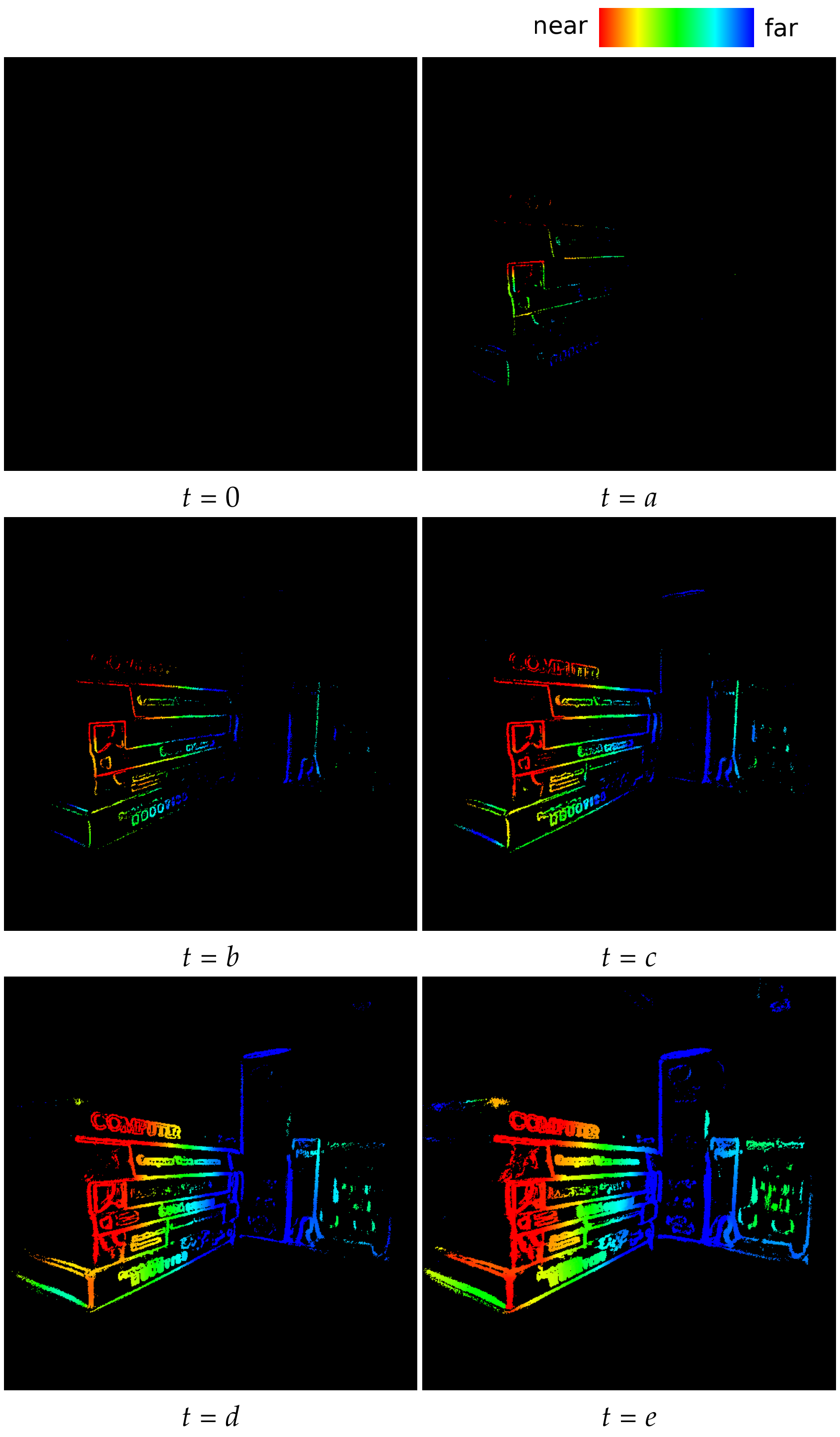


Figure 6.9: Typical temporal progression ($0 < a < b < c < d < e$) of inverse depth estimation and regularisation as a hand-held camera browses a 3D scene. The colours represent the different depths of the scene (refer to the colour chart in the top right).

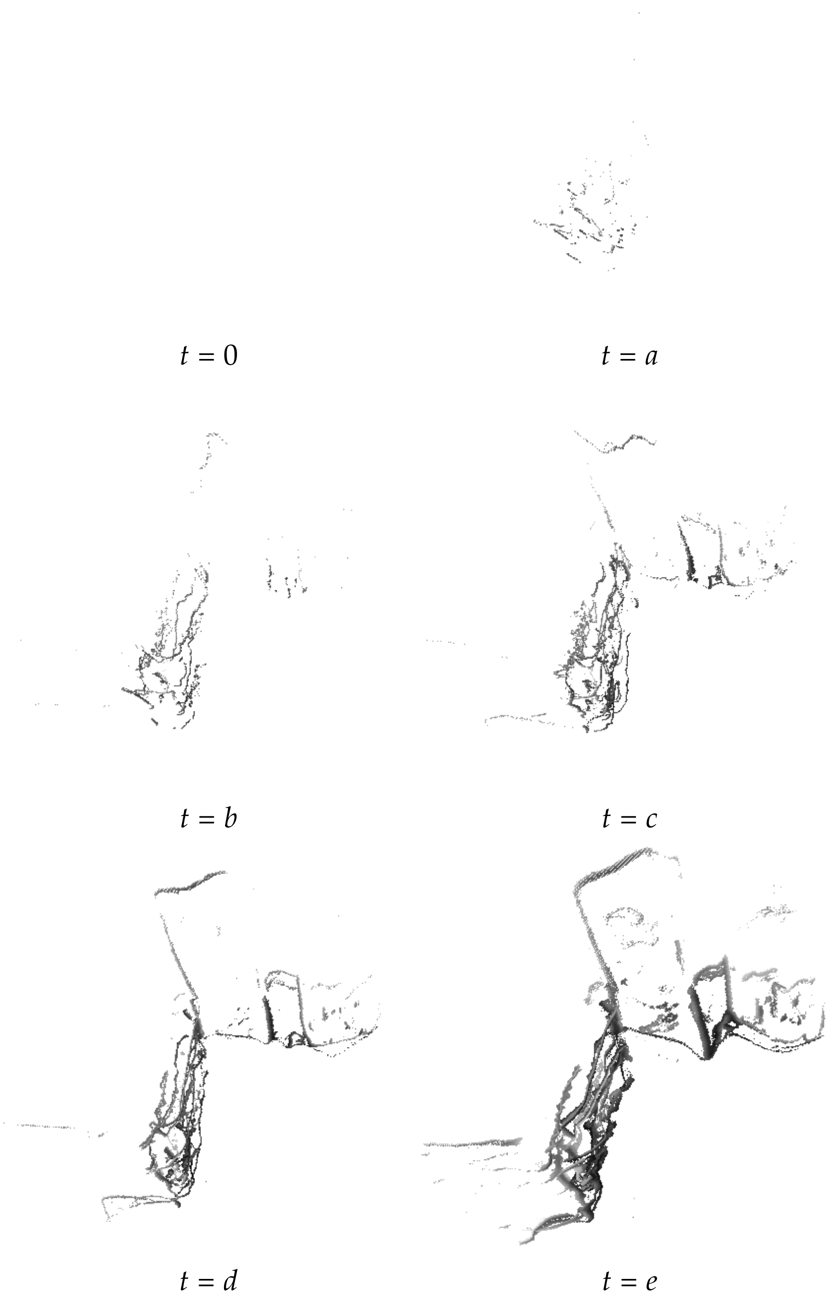


Figure 6.10: Typical temporal progression ($0 < a < b < c < d < e$) of a semi-dense 3D point cloud as a hand-held camera browses a 3D scene.

Table 6.1: Average processing time

Decoupled Work Package	Desktop	Laptop
tracking	6.27μs / event	4.33μs / event
mapping (gradient and inverse depth)	2.47μs / event	1.98μs / event
reconstruction and regularisation	8.09ms	9.92ms

in Section 3.6. We run both on a standard desktop PC consisting of an NVIDIA GeForce GTX 680 GPU hosted by an Intel Xeon W5590 3.33GHz quad-core CPU, and an Apple MacBook Pro consisting of an NVIDIA GeForce GT 750M GPU hosted by an Intel i7 2.6Ghz dual-core CPU. Table 6.1 shows the average processing time of the three decoupled work packages running on both machines — with our current *unoptimised* implementation, we can process about 200k to 300k events per second in real-time.

We present all the parameters used in the experiments in Table 6.2. As shown, we have used the same values for most of the parameters across a wide range of experimental environments, except the ones relative to the motion and inverse depth estimations which are sensitive to different motion speeds and scene structures.

6.5.1 Single Keyframe

We demonstrate the results from our algorithm as it tracks against and reconstructs a single keyframe in a number of different indoor and outdoor scenes. In Figure 6.11, for each scene we show column by column an image-like view of the event streams generated by accumulating events within a time interval, estimated gradient map, reconstructed intensity map with super resolution and high dynamic range properties, estimate depth map, and semi-dense textured 3D point cloud.

The 3D reconstruction quality is generally good, however we can see that there are sometimes poorer quality depth estimates near to occlusion boundaries and where not enough events have been generated.

6.5.2 Multiple Keyframes

We evaluated the proposed method on several trajectories which require multiple keyframes to cover. If the camera has moved too far from the current keyframe, we create a new keyframe from the most recent estimation results and reconstruction. To create a new keyframe, we project all 3D points based on the current keyframe pose and the estimated inverse depth

Table 6.2: Parameters used in the experiments

Parameter	Value	Reference
keyframe size	576×576	Figure 6.2
$\mathbf{x}_0$	$\mathbf{0}_{6\times1}$	Section 6.2
$\mathtt{P}_{\mathbf{x}_0}$	$\mathtt{0}_{6\times6}$	Section 6.2
$\mathbf{g}_0$	$\mathbf{0}_{2\times1}$	Section 6.3.1
$\mathtt{P}_{\mathbf{g}_0}$	$\begin{pmatrix} 2.5\times10^{-3} & 0 \\ 0 & 2.5\times10^{-3} \end{pmatrix}$	Section 6.3.1
$\Delta\mathtt{I}_l$	1.0×10^{-8}	Section 6.3.2
n_{max}	100	Section 6.3.2
ρ_0	$0.33 \sim 3.33$	Section 6.4.1
σ_{ρ_0}	$0.11 \sim 1.11$	Section 6.4.1
$\Delta\mathtt{D}_\rho$	1.0×10^{-5}	Section 6.4.2
$\mathtt{W}_\rho$ size	7×7	Section 6.4.2
$\sigma_1 \sim \sigma_3$	$1.5\times10^{-3} \sim 2.5\times10^{-3}$	Equation (6.7)
$\sigma_4 \sim \sigma_6$	$1.0\times10^{-3} \sim 2.5\times10^{-3}$	Equation (6.7)
C	0.15	Equation (6.8), (6.30), (6.46)
$\sigma_{\mathbf{x}}$	1.0×10^{-2}	Equation (6.14)
γ	1.0×10^{-3}	Equation (6.29)
σ_C	1.0×10^{-1}	Equation (6.39)
ϵ_d	0.1	Equation (6.43)
ϵ_r	0.1	Equation (6.44)
$\sigma_{\mathbf{p}}$	0.5	Equation (6.43)
$\sigma_{\mathbf{q}}$	0.5	Equation (6.44)
$\sigma_{\mathtt{I}_l}$	0.5	Equation (6.45)
λ	1.0×10^{-1}	Equation (6.44)
σ_ρ	$5.0\times10^{-2} \sim 1.0\times10^{-1}$	Equation (6.52)

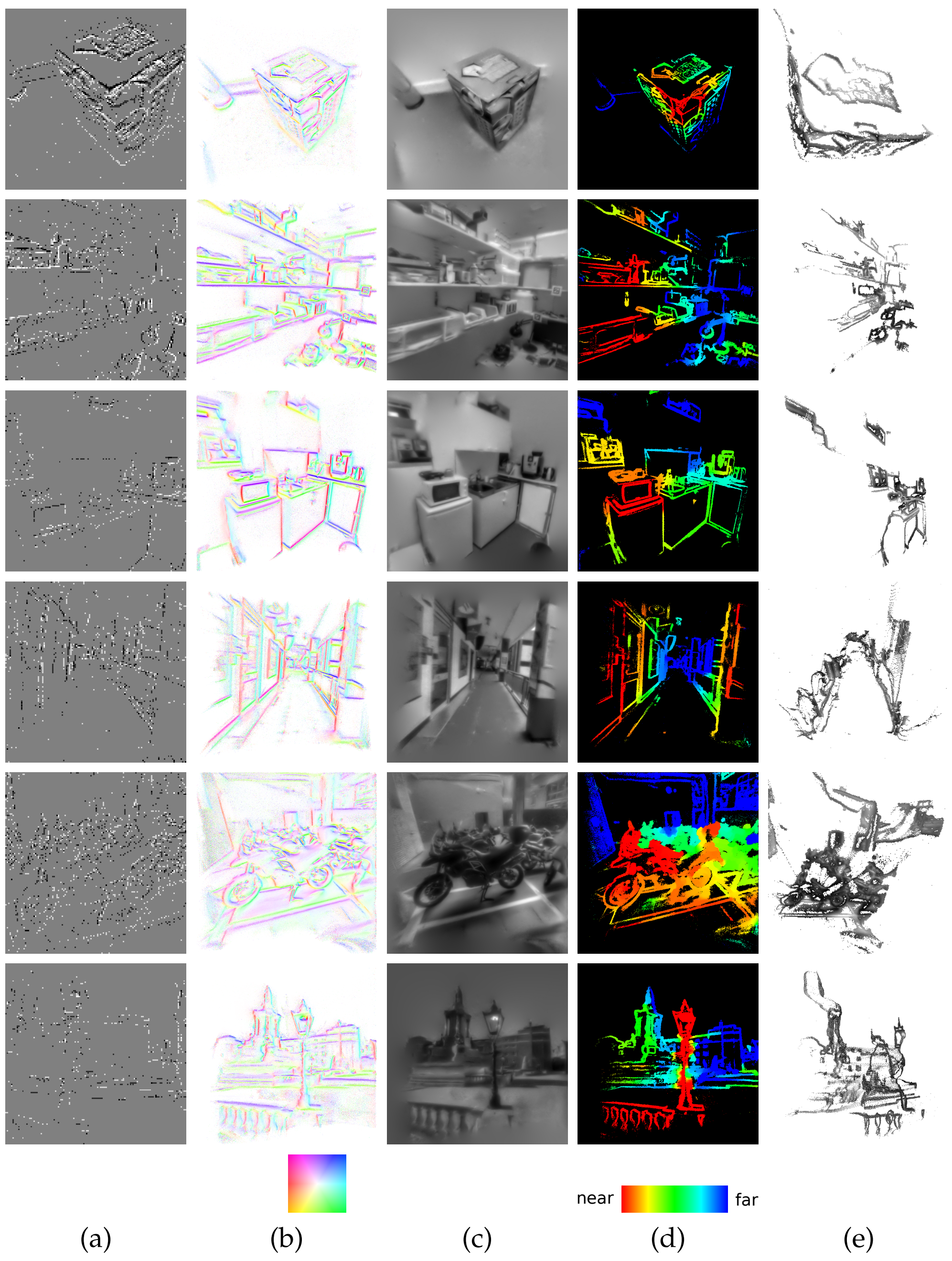


Figure 6.11: Demonstrations in various settings of the different aspects of our joint estimation algorithm. (a) Visualisation of the input event stream generated by accumulating events within a time interval — white and black pixels represent positive and negative events respectively; (b) estimated gradient keyframes — the colours and intensities represent the orientations and strengths of the scene gradients respectively; (c) reconstructed intensity keyframes with super resolution and high dynamic range properties; (d) estimated depth maps — the colours represent the different depths of the scene; (e) semi-dense textured 3D point clouds.

into the current camera pose, and propagate the current estimates and reconstruction only if they have high confidence in inverse depth.

Figure 6.12 shows one of the results in the form of a semi-dense textured 3D point cloud. This is built from the generated keyframes each consisting of reconstructed super-resolution and high dynamic range intensity and inverse depth map. The bright 3D coordinate axes represent the current camera pose along with the current image-like visualisation of the event stream, while the darker ones show all keyframe poses generated in this experiment with their associated intensity and depth map estimates.

6.5.3 Video Rendering

Using the method described in this chapter, we can turn an event camera into a high speed and high dynamic range artificial camera by rendering video frames based on ray casting as shown in Figure 6.13. Here we choose to render at the same low resolution as event camera input.

6.5.4 High Speed Tracking

We evaluated the proposed method on several trajectories which include rapid motion (e.g. a shaking hand). The graph in Figure 6.14 shows the estimated camera pose history, and the two groups of insets above and below show an image-like event visualisation, a rendered video frame showing the quality of our tracker, and a motion blurred standard camera video frame to make clear the rapid motion. Our current implementation is not able to process this very high event-rate (up to 1M events per second in this experiment) in real-time — with our current *unoptimised* implementation, we can process about 200k to 300k events per second in real-time as shown in Table 6.1.

6.6 Discussion and Summary

All the results presented here are qualitative, no benchmark experiments have been conducted, and we acknowledge that our work would certainly benefit from a wider range of evaluations. In this chapter, however, we have focused on demonstrating the core novelty of our approach in breaking through to get joint estimation of depth, 6-DoF motion and intensity from pure event data with general motion and unknown general scenes. To the best of our knowledge, there have been no previously published similar results and there are no alternative systems with which to compare our results. Certainly, if event cameras become

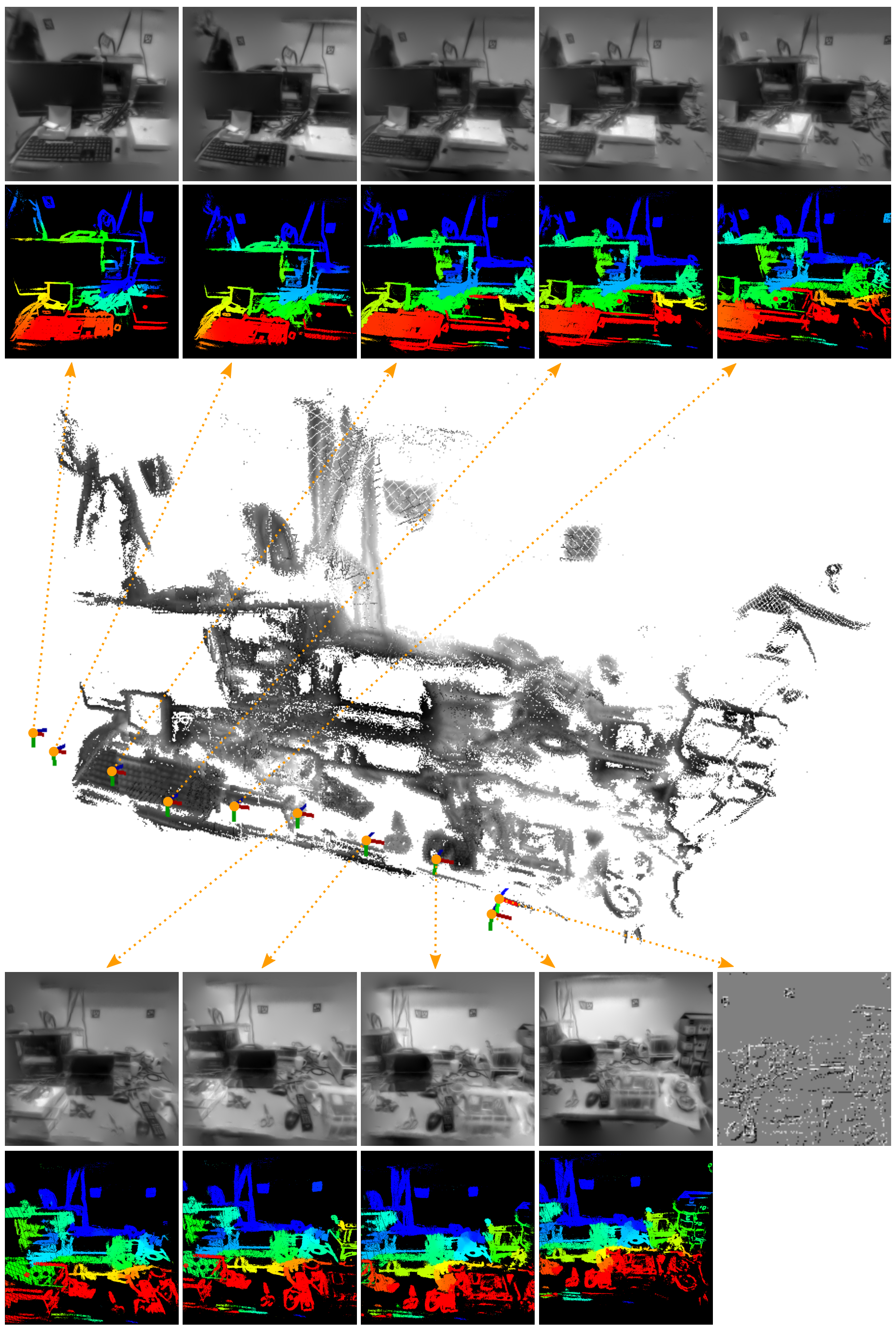


Figure 6.12: Semi-dense textured 3D point cloud of an indoor scene reconstructed from multiple keyframes. The bright RGB 3D coordinate axes represent the current camera pose along with the current image-like visualisation of the event stream, while the darker ones show all keyframe poses generated with their associated intensity and depth map estimates.

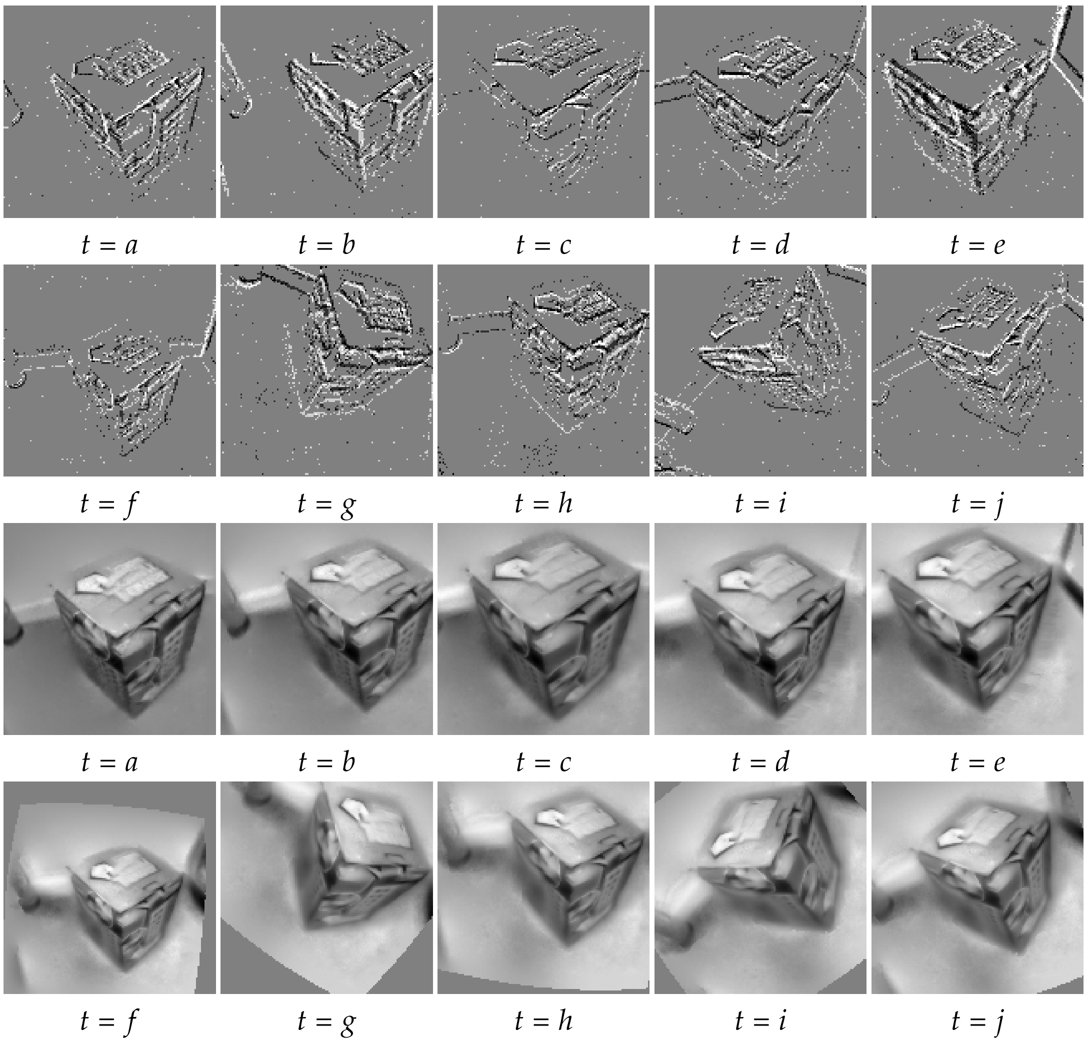


Figure 6.13: The method described in this chapter can convert a stream of events into HDR video frames at user-chosen time instances and resolutions by ray-casting the current reconstruction. This is the same scene as in the first row of Figure 6.11.

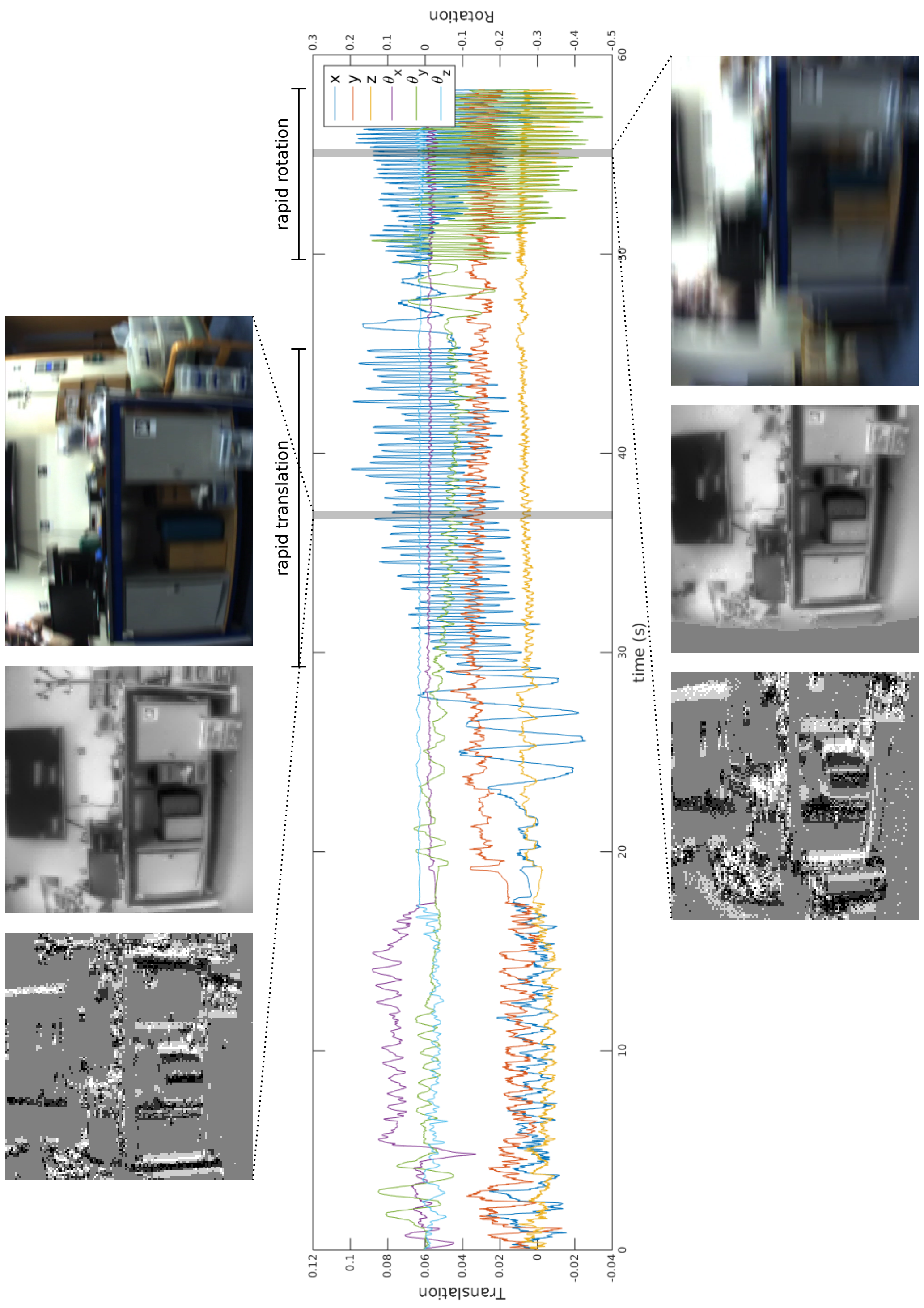


Figure 6.14: The graph shows the estimated camera pose history, and the two groups of insets above and below show an image-like event visualisation, a rendered video frame showing the quality of our tracker, and a motion blurred standard camera video frame to show the rapid motion (up to 5Hz of shake in this experiment).

well established as useful computer vision devices then an important piece of future work will be to design and release suitable comparative benchmarks.

Our approach of interleaved filters and separated intensity reconstruction makes many assumptions about independence that are certainly an approximation and thus constitute a weakness of our approach. Therefore, while we believe that it is remarkable that our approach of three decoupled filters, each of which operates as if the results of the others are correct, works at all, it is clear that we should try and consider those interactions as part of future work. In particular the relatively slow convergence of inverse depth estimates tends to cause poor tracking, leading to data association errors and a corruption of other parts of the estimation process. We can also see poorer quality depth estimates near occlusion boundaries and where not enough events have been generated. There are potential improvements in the inverse depth regularisation, and we will continue to work on improving it further by adapting more sophisticated methods (e.g. gradient-aware regularisation) in the near future.

Again, to the best of our knowledge, the method described in this chapter is the first 6-DoF tracking and 3D reconstruction method purely based on a stream of events with no additional sensing, and it runs in real-time on a standard PC. We hope this opens up the door to practical solutions to the current limitations of real-world SLAM applications.

Chapter 7

Conclusions

Contents

In this final chapter, we summarise the novel contributions presented in this thesis, and further discuss their current limitations and potential improvements. Finally, we share some ideas of future research to move towards more practical solutions for truly mass market SLAM products which are efficient and robust enough under very rapid motion and extreme lighting variation at always-on low power.

7.1 Contributions

In the previous chapters, we presented the first high performance real-time visual SLAM algorithms based on a single event camera, a paradigm shift in visual sensing, with a strong belief that such a next generation image sensor has great potential to solve the current limitations of real-time visual SLAM when integrated with novel event-based computer vision algorithms. Event sensors generate low bit-rate, information-rich data streams which are free of the redundancy of video while providing high dynamic range and high time resolution by reporting asynchronous intensity changes rather than full frames. Event sensors have no shutter or global exposure settings, which leads to a very high dynamic range (e.g. 130dB), and report events with almost continuous microsecond timestamps, allowing detailed observation of even spinning fan blades.

After introducing the event camera and some of the prerequisites required to work with it in Chapter 2 and Chapter 3 respectively, we showed for the first time that an event stream, with no additional sensing, can be used to track accurate camera rotation while building a persistent and high quality mosaic of a scene which is super-resolution accurate and has high dynamic range in Chapter 4. The method involves parallel camera rotation tracking and template reconstruction from estimated gradients, both operating on an event-by-event basis and based on probabilistic filtering to maximise the update rate with low latency. In Chapter 5, we then presented a real-time implementation whose overall structure and functionalities are the same, but where a substantial speed up was achieved by adapting a computationally efficient estimation method for tracking as well as a parallelisable log intensity reconstruction running on a GPU. The experimental results also showed that this speed-up increases the quality of estimation and reconstruction as it guarantees higher fidelity of the independence assumption.

Lastly, in Chapter 6, to move towards 3D visual SLAM, we proposed a method which can perform real-time 3D reconstruction from a single hand-held event camera with no additional sensing, and works in unstructured scenes of which it has no prior knowledge. It is based on three decoupled probabilistic filters, estimating 6-DoF camera motion, scene log intensity gradient and scene inverse depth relative to a virtual keyframe. We build a real-time graph of these keyframes to track and model over an extended local workspace. We also upgrade the gradient estimate for each keyframe into an intensity image, allowing us to recover a real-time video-like intensity sequence with spatial and temporal super-resolution from the low bit-rate input event stream. To the best of our knowledge, this was the first algorithm provably able to track a general 6D motion, along with reconstruction of arbitrary structure including its intensity and the reconstruction of grayscale video, that exclusively relies on event camera data.

All of the methods described in this thesis were able to harness the superior properties of the event camera such as high speed measurement, low latency, high dynamic range, and low data rate, and we hope our work opens up the door to practical solutions to the current limitations of real-world SLAM applications.

7.2 Discussion and Future Research

7.2.1 Limitations and Potential Improvements

Our approach of interleaved probabilistic filters and separated intensity reconstruction makes many assumptions about independence that are certainly an approximation and thus constitute a weakness of our approach. Therefore, while we are truly excited that the methods work, it is clear that we should try and consider those interactions as part of future work. In particular the relatively slow convergence of inverse depth estimates described in Chapter 6 tends to cause poor tracking, leading to data association errors and a corruption of other parts of the estimation process. We can also see poorer quality depth estimates near occlusion boundaries and where not enough events have been generated, and there is certainly a room for potential improvements in the inverse depth regularisation by adapting more sophisticated methods such as gradient-aware regularisation. Additionally, although the separation of the log intensity gradient and scene inverse depth estimation components was made considering fewer number of events caused by parallax while almost all events carry gradient information, the closely correlated nature of both visual quantities with respect to event rates provides a strong motivation of a joint estimation which could enable more consistent results.

All the implementations described within this thesis were able to run in real-time by harnessing the parallel processing power of a GPU, especially for log intensity reconstruction, which make them difficult to be deployed on computationally limited platforms. The clearest way to relieve the powerful hardware requirement is to directly track against estimated gradients from which we currently reconstruct log intensity values, so that we can remove the reconstruction component completely or at least execute it at a lower frequency. The high rate of event measurements relative to the dynamics of a hand-held camera also strongly motivates changing our tracking component to use a stronger motion model such as a constant velocity or constant acceleration which is more robust to significant hand-held jitter when shaking the camera [66]. We also do not use an explicit bootstrapping method as we have found that, starting from scratch, alternating estimation very often lead to convergence, though there are sometimes currently gross failures and this is an important issue for future research.

Lastly, all of our experimental results presented in this thesis are qualitative, no benchmark experiments have been conducted, and we acknowledge that our work would certainly benefit from a wide range of evaluations. In this work, however, we have focused on demonstrating the core novelty of our approaches in breaking through to get joint estimation of hand-held camera motion, intensity and depth from pure event data with arbitrary motion and unknown, unstructured scenes. To the best of our knowledge, there were no previously published similar results and there are no alternative systems with which to compare our results. Certainly, if event cameras become well established as useful computer vision devices then an important piece of future work will be to design and release suitable comparative benchmarks.

7.2.2 Sensor Fusion

Within this thesis, we only focused on single event camera SLAM solutions with no additional sensing, because we believe that first solving the hardest problem of not relying on other supplementary sensors will be useful on its own and provides the insights to make best use of additional measurements if they are available, as well as eliminating unnecessary extra complication including synchronisation and calibration problems to be solved. But we also see that investigating sensor fusion (e.g. with an inertial measurement unit or conventional image frames) will certainly result in a more accurate and robust system, aiding bootstrapping and the resolution of scale ambiguity similar to standard visual inertial methods [120, 80, 94, 16]. There are some early works on fusing the event camera with other complementary sensors such as of Censi and Scaramuzza [26] with a CMOS camera, Weikersdorfer *et al.* [170] with a RGB-D sensor, and Delbruck *et al.* [49] and Yuan and Ramalingam [177] with an IMU. More recently, Zhu *et al.* [180] presented the first algorithm to fuse a purely event-based tracking algorithm with an IMU, to provide accurate metric tracking of a camera's full 6-DoF pose. Rebecq *et al.* [140] also proposed an accurate keyframe-based, tightly-coupled visual-inertial odometry algorithm based on nonlinear optimisation, and showed their system works well even in challenging conditions by utilising the event camera's outstanding properties.

7.2.3 Towards Fully Event-Based SLAM Systems

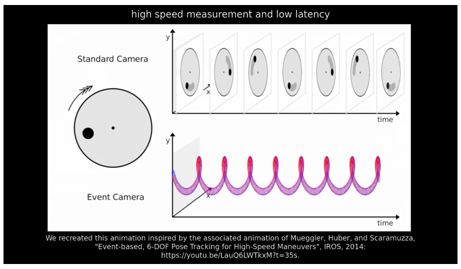
Despite the sparse and data-driven nature of event cameras, realising highly efficient SLAM systems while still relying on conventional communication protocols and traditional von Neumann computing architecture seems not feasible, and we expect special hardware to become available in the near future. For instance, neuromorphic processors or graph processors placed directly behind and connected in parallel to the pixels of event sensors, and incoming events wake up and activate local computation and message passing, while the majority of the processor remains in a sleep state to conserve power.

This concept resembles more closely biological vision systems where information is represented by means of data-driven pulsed messages, exchanged by nervous cells, and processed by sparse neural network architectures. As briefly discussed in Section 2.1, understanding how biology functions in such an efficient way will undoubtedly provide profound insights into new paradigms of sensing and processing. In the field of computational neuroscience, spike-based self motion estimation is well-studied. For instance, by investigating the optomotor response of the beetle Clorophanus, Hassenstein and Reichardt [75] presented the *elementary motion detector* (EMD) which detects the presence of motion in a specific direction by computing the correlation between the signal of one photoreceptor and its neighbouring photoreceptor's time-delayed signal. In 1985, Adelson and Bergen [1] and Watson and Ahumada [167] proposed similar models of how humans sense the velocity of moving images based on spatio-temporal frequency filtering. Simoncelli [151] formulated the motion estimation problem in a probabilistic Bayesian framework based on the brightness constancy constraint [76]. More recently, there are some early works on combining event cameras with novel computing architectures such as that of Martel *et al.* [108] with CPA [9, 23, 98], Orchard *et al.* [130] with SpiNNaker [63], and Samsung[1] with TrueNorth [111]. The unsolved challenges however are to map SLAM algorithms into message passing (belief propagation) and to obtain globally consistent estimates of non-local parameters such as ego-motion, and they remain as important subjects for future research.

[1]Samsung turns IBM's brain-like chip into a digital eye: https://www.cnet.com/news/samsung-turns-ibms-brain-like-chip-into-a-digital-eye (accessed September 2017)

Appendix A

Video Material

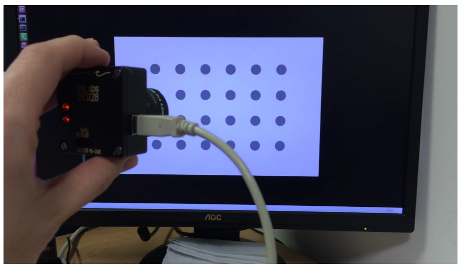


We recreated this animation inspired by the associated animation of Mueggler, Huber, and Scaramuzza, "Event-based, 6-DOF Pose Tracking for High-Speed Maneuvers", IROS, 2014: https://youtu.be/LauQ6LWTkxM?t=35s.

Event Camera vs Standard Camera

Detailed in Section 2.2. We recreated this animation inspired by the associated animation of [122]: youtu.be/LauQ6LWTkxM?t=35s.

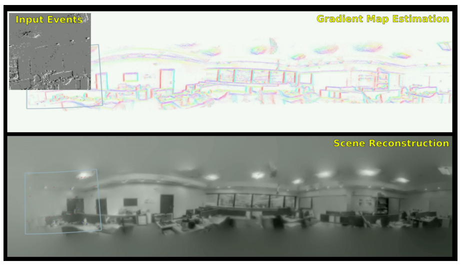
https://youtu.be/kPCZESVfHoQ



Event Camera Calibration

Detailed in Section 3.6.

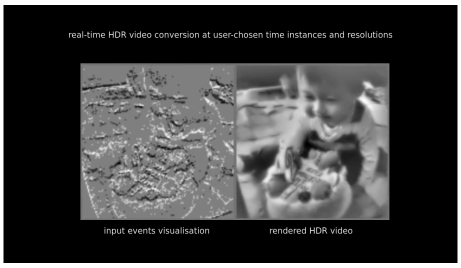
https://youtu.be/OK_m6OobntE



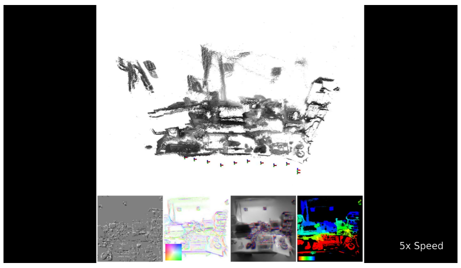
Simultaneous Mosaicing and Tracking with an Event Camera

Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoï Ieng and Andrew J. Davison. BMVC, 2014 [82]. Detailed in Chapter 4.

https://youtu.be/l6qxeM1DbXU


real-time HDR video conversion at user-chosen time instances and resolutions
input events visualisation
rendered HDR video

ETAM 2D: Real-Time Event-Based Tracking and Mapping
Detailed in Chapter 5.
https://youtu.be/z72lNV7idUs



Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera
Hanme Kim, Stefan Leutenegger and Andrew J. Davison. ECCV, 2016 [83]. Detailed in Chapter 6.
https://youtu.be/yHLyhdMSw7w

List of Figures

List of Tables

Bibliography

[1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2(2):284–299, 1985. 147

[2] A. Agrawal, R. Chellappa, and R. Raskar. An Algebraic Approach to Surface Reconstruction from Gradient Fields. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005. 78

[3] A. Agrawal, R. Raskar, and R. Chellappa. What is the Range of Surface Reconstructions from a Gradient Field. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006. 78

[4] Jean-François Aujol. Some First-Order Algorithms for Total Variation Based Image Restoration. *Journal of Mathematical Imaging and Vision*, 34(3):307–327, 2009. 96

[5] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó. The SLAM problem: a survey. In *Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, 2008. 19

[6] T. Bailey and H. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117, 2006. 19

[7] 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. 47

[8] 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. 41

[9] D. R. W. Barr and P. Dudek. APRON: A Cellular Processor Array Simulation and Hardware Design Tool. *EURASIP Journal on Advances in Signal Processing*, 2009. 40, 87, 147

[10] F. Barranco, C. Fermuller, Y. Aloimonos, and T. Delbruck. A Dataset for Visual Navigation with Neuromorphic Methods. *Frontiers in Neuroscience*, 10, 2016. 47

[11] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006. 47

[12] A. N. Belbachir, S. Schraml, M. Mayerhofer, and M. Hofstätter. A Novel HDR Depth Camera for Real-time 3D 360° Panoramic Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014. 43, 44

[13] R. Benosman, C. Clercq, X. Lagorce, S. Ieng, and C. Bartolozzi. Event-Based Visual Flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25:407–417, 2014. 40

[14] S. Betgé-Brezetz, P. Hébert, R. Chatila, and M. Devy. Uncertain Map Making in Natural Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1996. 11

[15] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Journal of Neural Networks*, 32:339–348, 2012. 38

[16] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015. 146

[17] K Boahen. Neuromorphic Chips. *Scientific American*, 2005. 28

[18] M. Bosse, P. Newman, J. J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas Framework for Scalable Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003. 13

[19] 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. 34, 35, 37, 86

[20] G. Brown. *The Energy of Life*. Free Press, New York, 1999. 28

[21] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32:1309–1332, 2016. 19

[22] S. J. Carey, D. R.W. Barr, and P. Dudek. Low power high-performance smart camera system based on SCAMP vision sensor. *Journal of Systems Architecture*, 59:889–899, 2013. 87

[23] S. J. Carey, A. Lopich, D. R. W. Barr, B. Wang, and P. Dudek. A 100,000 fps Vision Sensor with Embedded 535GOPS/W 256×256 SIMD Processor Array. In *Proceedings of the VLSI Circuits Symposium*, 2013. 40, 87, 147

[24] J. Carneiro, S. Ieng, C. Posch, and R. Benosman. Event-based 3D reconstruction from neuromorphic retinas. *Journal of Neural Networks*, 45:27–38, 2013. 43

[25] J. A. Castellanos. *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. PhD thesis, Universidad de Zaragoza, Spain, 1998. 11

[26] A. Censi and D. Scaramuzza. Low-Latency Event-Based Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 42, 146

[27] 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. 96

[28] 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)*, 42(10):2187–2196, 2007. 29

[29] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. "MFm": 3-D Motion From 2-D Motion Causally Integrated Over Time. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000. 12

[30] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from Motion Causally Integrated Over Time. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):523–535, 2002. 12

[31] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse Depth Parametrization for Monocular SLAM. *IEEE Transactions on Robotics (T-RO)*, 24(5):932–945, 2008. 127

[32] J. Civera, A. J. Davison, J. A. Magallón, and J. M. M. Montiel. Drift-Free Real-Time Sequential Mosaicing. *International Journal of Computer Vision (IJCV)*, 81(2):128–137, 2009. 66

[33] J. Civera, A. J. Davison, and J. M. M. Montiel. *Structure from Motion Using the Extended Kalman Filter*. Springer Tracts in Advanced Robotics (STAR), 2012. 19

[34] J. L. B. Claraco. *Contributions to Localization, Mapping and Navigation in Mobile Robotics*. PhD thesis, Universidad de Málaga, 2009. 74

[35] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardós. Mapping Large Loops with a Single Hand-Held Camera. In *Proceedings of Robotics: Science and Systems (RSS)*, 2007. 13

[36] R. T. Collins. A Space-Sweep Approach to True Multi-Image Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996. 45

[37] A. I. Comport, E. Malis, and P. Rives. Accurate Quadri-focal Tracking for Robust 3D Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007. 15

[38] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R.J. Douglas, and T. Delbruck. A pencil balancing robot using a pair of AER dynamic vision sensors. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2009. 38, 39

[39] 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. 39, 40

[40] A. J. Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford, 1998. 11, 12

[41] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003. 12

[42] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007. 12, 13

[43] T. Delbrck. Neuromorophic Vision Sensing and Processing. In *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference, Lausanne, Switzerland, September 12-15, 2016*, pages 7–14, 2016. 29

[44] T. Delbruck. Frame-free dynamic digital vision. In *Proceedings of International Symposium on Secure-Life Electronics*, pages 21–26, 2008. 56

[45] T. Delbruck. Fun with Asynchronous Vision Sensors and Processing. In *Computer Vision - ECCV 2012. Workshops and Demonstrations - Florence, Italy, October 7-13, 2012, Proceedings, Part I*, pages 506–515, 2012. 29

[46] T. Delbruck and M. Lang. Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*, 7(223), 2013. 38, 39

[47] T. Delbruck and P. Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2007. 38

[48] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch. Activity-Driven, Event-Based Vision Sensors. In *Proceedings of the International Symposium on Circuits and Systems*, pages 2426–2429, 2010. 29

[49] T. Delbruck, V. Villanueva, and L. Longinotti. Integration of Dynamic Vision Sensor with Inertial Measurement Unit for Electronically Stabilized Event-Based Vision. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014. 146

[50] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe. A review of recent developments in simultaneous localization and mapping. In *2011 6th International Conference on Industrial and Information Systems*, pages 477–482, 2011. doi: 10.1109/ICIINFS.2011.6038117. 19

[51] R. Douc, O. Cappé, and E. Moulines. Comparison of Resampling Schemes for Particle Filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69, 2005. 74

[52] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10:197–208, 2000. 74

[53] H. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006. 19

[54] E. Eade and T. Drummond. Scalable Monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 13

[55] E. Eade and T. Drummond. Monocular SLAM as a Graph of Coalesced Observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007. 13

[56] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013. 17, 132

[57] Jakob Engel, Thomas Schoeps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 17, 18, 66, 110

[58] R. Fattal, D. Lischinski, and M. Werman. Gradient Domain High Dynamic Range Compression. *ACM Transactions on Graphics (TOG)*, 21(3):249–256, 2002. 78

[59] G. D. Finlayson, S. D. Hordley, and M. S. Drew. Removing Shadows from Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2002. 78

[60] A. W. Fitzgibbon and A. Zisserman. Automatic Camera Recovery for Closed or Open Image Sequences. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 1998. 11

[61] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 18

[62] F. Fraundorfer and D. Scaramuzza. Visual Odometry Part II: Matching, Robustness, Optimization, and Applications. *IEEE Robotics & Automation Magazine*, 19:78–90, 2012. 19

[63] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The SpiNNaker Project. *Proceedings of the IEEE*, 102:652–665, 2014. 39, 87, 147

[64] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza. Event-based, 6-DOF Camera Tracking for High-Speed Applications. *arXiv preprint arXiv:1607.03468*, 2016. 42

[65] D. Gálvez-López and J. D. Tardós. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics (T-RO)*, 28(5):1188–1197, 2012. 14

[66] P. Gemeiner, A. J. Davison, and M. Vincze. Improving Localization Robustness in Monocular SLAM Using a High-Speed Camera. In *Proceedings of Robotics: Science and Systems (RSS)*, 2008. 121, 145

[67] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent. In *Proceedings of Robotics: Science and Systems (RSS)*, 2007. 13

[68] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2:31–43, 2010. 19

[69] 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. 96, 124

[70] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Real-Time Camera Tracking: When is High Frame-Rate Best? In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 21, 38, 61

[71] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014. URL `http://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html`. 61

[72] J. W. Harris and H. Stocker. *Handbook of Mathematics and Computational Science*. Springer-Verlag, 1998. 78

[73] L. A. Hart. *How the Brain Works*. Basic Books, New York, 1975. 28

[74] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 19, 52, 116

[75] B. Hassenstein and W. Reichardt. Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers Chlorophanus. *Zeitschrift für Naturforschung*, 11b:513–524, 1956. 147

[76] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981. 76, 121, 147

[77] S. Huang and G. Dissanayake. A critique of current developments in simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 13:1–13, 2016. 19

[78] Idaku Ishii, Yoshihiro Nakabo, and Masatoshi Ishikawa. Target tracking algorithm for 1 ms visual feedback system using massively parallel processing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1996. 21

[79] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970. 11

[80] E. S. Jones and S. Soatto. Visiual-inertial navigation, mapping and localization: A scalable real-time causal approach. *International Journal of Robotics Research (IJRR)*, 30 (4):407–430, 2011. 146

[81] D. Kim and E. Culurciello. A Compact-pixel Tri-mode Vision Sensor. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010. 29

[82] 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. 23, 45, 48, 149

[83] 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. 24, 46, 48, 150

[84] G. Klein and D. W. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 14, 15, 66, 110

[85] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-Latency Visual Odometry using Event-Based Feature Tracks. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2016. 45

[86] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 13, 14

[87] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie. Silicon Auditory Processors as Computer Peripherals. *IEEE Transactions on Neural Networks*, 4(3): 523–528, 1993. 54, 56

[88] J. H. Lee, T. Delbruck, M. Pfeiffer, P. K. J. Park, C.-W. Shin, H. Ryu, and B. C. Kang. Real-Time Gesture Interface Based on Event-Driven Processing from Stereo Silicon Retinas. *IEEE Transactions on Neural Networks and Learning Systems*, 25(12):2250–2263, 2014. 39

[89] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco. A Signed Spatial Contrast Event Spike Retina Chip. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010. 29

[90] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco. A 3.6μs Latency Asynchronous Frame-Free Event-Driven Dynamic-Vision-Sensor. *IEEE Journal of Solid-State Circuits (JSSC)*, 46(6):1443–1455, 2011. 34, 37

[91] J. J. Leonard. *Directed Sonar Sensing for Mobile Robot Navigation*. PhD thesis, University of Oxford, 1990. 11

[92] J. J. Leonard and Durrant H. Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 1991. 11

[93] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariance scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 47

[94] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2014. 146

[95] 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. In *International Image Sensor Workshop (IISW)*, 2015. 35

[96] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128$\times$128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits (JSSC)*, 43(2):566–576, 2008. 29, 31, 33, 35, 37, 73

[97] R. LiKamWa, Z. Wang, A. Carroll, F. X. Lin, and L. Zhong. Draining our Glass: An Energy and Heat Characterization of Google Glass. *arXiv preprint arXiv:1404.1320*, 2014. 22

[98] A. Lopich and P. Dudek. A General-purpose Vision Processor with 160$\times$80 Pixel-Parallel SIMD Processor Array. In *Proceedings of the Custom Integrated Circuits Conference*, 2013. 40, 87, 147

[99] S. J. Lovegrove and A. J. Davison. Real-Time Spherical Mosaicing using Whole Image Alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010. 66

[100] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 47

[101] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics (T-RO)*, 32(1):1–19, 2016. 19

[102] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333–349, 1997. 13

[103] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1981. 47

[104] M. Mahowald. *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. PhD thesis, California Institute of Technology, 1992. 28, 54, 56

[105] M. Mahowald and C. Mead. The Silicon Retina. *Scientific American*, 264(5):76–82, 1991. 28, 29

[106] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004. 47

[107] J. N. P. Martel and M. Cook. A Framework of Relational Networks to Build Systems with Sensors able to Perform the Joint Approximate Inference of Quantities. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on Unconventional Computing for Bayesian Inference*, 2015. 40

[108] J. N. P. Martel, M. Chau, P. Dudek, and M. Cook. Toward Joint Approximate Inference of Visual Quantities on Cellular Processor Arrays. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015. 40, 147

[109] N. Matsuda, O. Cossairt, and M. Gupta. MC3D: Motion Contrast 3D Scanning. In *Proceedings of the IEEE International Conference on Computational Photography (ICCP)*, 2015. 43, 44

[110] J. McCormac, A. Handa, A. J. Davison, and S. Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 19

[111] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha.

A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345:668–673, August 2014. 39, 87, 147

[112] M. Milford, H. Kim, S. Leutenegger, and A. J. Davison. Towards visual SLAM with event-based cameras. In *The Problem of Mobile Sensors: Setting future goals and indicators of progress for SLAM Workshop in conjunction with Robotics: Science and Systems (RSS)*, 2015. 24, 46

[113] M. Milford, H. Kim, M. Mangan, S. Leutenegger, T. Stone, B. Webb, and A. J. Davison. Place recognition with event-based cameras and a neural implementation of SeqSLAM. In *The Innovative Sensing for Robotics: Focus on Neuromorphic Sensors workshop at the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 24, 46

[114] M. J. Milford and G. Wyeth. Mapping a Suburb with a Single Camera using a Biologically Inspired SLAM System. *IEEE Transactions on Robotics (T-RO)*, 24(5):1038–1053, 2008. 14

[115] Y. Miyatani, S. Barua, and A. Veeraraghavan. Direct Face Detection and Video Reconstruction from Event Cameras. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016. 41

[116] A. Moini. *Vision Chips*. Kluwer Academic, Boston, UK, 2000. 29

[117] Tomas Möller and Ben Trumbore. Fast , Minimum Storage Ray / Triangle Intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997. 114, 115, 118

[118] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2002. 13

[119] J. M. M. Montiel, J. Civera, and A. J. Davison. Unified Inverse Depth Parametrization for Monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2006. 127

[120] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572. IEEE, 2007. 146

[121] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environement modelling. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 1989. 11

[122] 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. 31, 41, 42, 149

[123] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza. Towards Evasive Maneuvers with Quadrotors using Dynamic Vision Sensors. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*, 2015. 59

[124] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM. *International Journal of Robotics Research (IJRR)*, 2016. 47

[125] R. Mur-Artal, J. M. M Montiel, and J. D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics (T-RO)*, 31(5):1147–1163, 2015. 14, 66, 110

[126] 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. 15

[127] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. 16, 17

[128] 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. 15, 16, 66, 110, 114

[129] P. Newman. *On the Structure and Solution of the Simultaneous Localization and Map Building Problem*. PhD thesis, University of Sydney, 1999. 11

[130] G. Orchard, X. Lagorce, C. Posch, S. B. Furber, R. Benosman, and F. Galluppi. Real-time Event-driven Spiking Neural Network Object Recognition on the SpiNNaker Platform. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015. 147

[131] P. Pérez, M. Gangnet, and A. Blake. Poisson Image Editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318, 2003. 78

[132] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco. Mapping from Frame-Driven to Frame-Free Event-Driven

Vision Systems by Low-Rate Rate Coding and Coincidence Processing. Application to Feedforward ConvNets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35:2706 – 2719, 2013. 38

[133] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998. 11

[134] C. Posch. Bio-inspired vision. *Journal of Instrumentation*, 7(1), 2012. 29

[135] 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. 29

[136] 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. 32, 33, 37

[137] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, 2 edition, October 1992. 79

[138] H. Rebecq, G. Gallego, and D. Scaramuzza. EMVS: Event-based Multi-View Stereo. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 43, 46

[139] 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. 46

[140] H. Rebecq, T. Horstschaefer, and D. Scaramuzza. Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. 43, 146

[141] 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. 41

[142] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011. 14

[143] B. Ruckauer and T. Delbruck. Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor. *Frontiers in Neuroscience*, 10, 2016. 47

[144] S. Saeedi, M. Trentini, M. Seto, and H. Li. Multiple-Robot Simultaneous Localization and Mapping: A Review. *Journal of Field Robotics (JFR)*, 33(1):3–46, 2016. 19

[145] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. URL http://dx.doi.org/10.1109/CVPR.2013.178. 19

[146] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison. Dense planar SLAM. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014. 19

[147] D. Scaramuzza and F. Fraundorfer. Visual Odometry Part I: The First 30 Years and Fundamentals. *IEEE Robotics & Automation Magazine*, 18:80–92, 2011. 19

[148] S. Schraml, A. N. Belbachir, and H. Bischof. Event-Driven Stereo Matching for Real-Time 3D Panoramic Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 43, 44

[149] T. Serrano-Gotarredona and B. Linares-Barranco. A 128$\times$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. 35, 37

[150] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. 28

[151] E. P. Simoncelli. Local Analysis of Visual Motion. *The Visual Neurosciences*, pages 1616–1623, 2003. 147

[152] M. A. Sivilotti. *Wiring Considerations in Analog VLSI Systems, with Application to Field-Programmable Networks*. PhD thesis, California Institute of Technology, 1991. 54, 56

[153] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Uncertainty in Artificial Intelligence*, pages 435–461. Elsevier, 1988. 11

[154] B. Son, Y. Suh, S. Kim, H. Jung, J-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu. A 640$\times$480 Dynamic Vision Sensor with a 9μm Pixel and 300Meps Address-Event Representation. In *International Solid-State Circuits Conference ((ISSCC)*, 2017. 35, 36, 37

[155] J. Stillwell. *Naive Lie Theory*. Springer, 2008. 72, 74, 92, 113

[156] A. A. Stocker. An Improved 2D Optical Flow Sensor for Motion Segmentation. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2002. 29

[157] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-Time Monocular SLAM: Why Filter? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010. 14

[158] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale Drift-Aware Large Scale Monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010. 14

[159] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double Window Optimisation for Constant Time Visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 14

[160] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why filter? *Image and Vision Computing (IVC)*, 30(2):65–77, 2012. 14

[161] J. Stuehmer, S. Gumhold, and D. Cremers. Real-Time Dense Geometry from a Handheld Camera. In *Proceedings of the DAGM Symposium on Pattern Recognition*, 2010. 15

[162] R. Szeliski. Image Alignment and Stitching: A Tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006. 66, 68

[163] R. Szeliski and H. Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of SIGGRAPH*, 1997. 68

[164] J. J. Tarrio and S. Pedre. Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015. 18

[165] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Cambridge: MIT Press, 2005. 19

[166] 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. 29, 78

[167] A. B. Watson and A. J. Ahumada, Jr. Model of human visual-motion sensing. *Journal of the Optical Society of America A*, 2(2):322–341, 1985. 147

[168] D. Weikersdorfer and J. Conradt. Event-based Particle Filtering for Robot Self-Localization. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012. 41, 42, 45

[169] D. Weikersdorfer, R. Hoffmann, and J. Conradt. Simultaneous Localization and Mapping for event-based Vision Systems. In *International Conference on Computer Vision Systems (ICVS)*, 2013. 45, 46

[170] 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. 45, 46, 146

[171] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*, 2012. 16

[172] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626, December 2014. 16

[173] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015. 16, 17

[174] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John B. McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013. 16

[175] M. Yang, S.-C. Liu, and T. Delbruck. A Dynamic Vision Sensor With 1% Temporal Contrast Sensitivity and In-Pixel Asynchronous Delta Modulator for Event Envoding. *IEEE Journal of Solid-State Circuits (JSSC)*, 50(9):2149–2160, 2015. 35, 37

[176] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*, 1: 298–311, 2015. 19

[177] W. Yuan and S. Ramalingam. Fast Localization and Tracking using Event Sensors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016. 42, 146

[178] K. A. Zaghloul. *A Silicon Implementation of a Novel Model for Retinal Processing*. PhD thesis, University of Pennsylvania, 2001. 29

[179] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22:1330–1334, 2000. doi: 10.1109/34.888718. 59

[180] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based Visual Inertial Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 42, 146

[181] M. Zhu. *Fast numerical algorithms for total variation based image restoration*. PhD thesis, University of California at Los Angeles, 2008. 96