ASL-SLAM: An asynchronous formulation of lines for SLAM with Event Sensors

Xiaoqi Nong

Submitted for the Degree of
Master of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

July 2022

Abstract

New bio-inspired sensors that measure brightness changes per-pixel have the potential to become a novel solution to the pose estimation problem. The sensor generates a stream of events that represent the position and the polarity of intensity change. This means that it can still work well under the low light condition, even with unstable movement. Currently, event-based vision sensors output compressed digital data in the form of events, reducing latency and having higher temporal range than conventional image-based methods. These event-based cameras encode visual information in an extremely efficient manner in terms of data reduction and energy consumption. This is especially important in the field of localization, where responsiveness is one of the most significant properties.

In this thesis, we explore approaches to perform feature detection directly on the event stream without intermediate event accumulation. Event flows are obtained to track lines. We introduce ASL-SLAM, the first line-based SLAM system operating directly on asynchronous event streams. This approach maximizes the advantages of the event information generated by a bio-inspired sensor. We estimate the local Surface of Active Events (SAE) to get the space-time planes associated with each incoming event in the event stream. Then the edges and their motion are recovered by our line extraction algorithm. We show how the inclusion of event-based line tracking significantly improves performance compared to state-of-the-art frame-based SLAM systems. The approach is evaluated on publicly available datasets. The results show that our approach is particularly practical with poorly textured frames. We also experimented with challenging illumination situations, including low-light and high motion blur scenarios. We show that our approach with an event-based camera has natural advantages and provides up to 85% reduction in error when performing SLAM under these conditions compared to the traditional approach.

Key words: Event Camera, SLAM, Visual Odometry, Surface of Active Event.

Email: xn00046@surrey.ac.uk

WWW: http://www.feps.surrey.ac.uk/

Acknowledgements

I would first like to thank my supervisors Simon Hadfield and Richard Bowden, especially Dr Hadfield who was always there to support and suggest effective ideas of solving problems. He is more like a friend than a supervisor. I appreciate very much for giving me the opportunity to get to know the new technologies and sensors. I would like to thank everyone in CVSSP. I have a great time working with these friendly people and obtain many great memories. I would also thank my close friends Xihan Bian and Ye Ling, who also give me their hands both on my daily life and the my study.

Finally, I would like to give a very big appreciation to my family for their continuous love and support over the years. Even when times were tough, they are always on my side and send me their love at all costs. I would not have got this far in life if all of them were not part of my life, and so I am eternally grateful.

Declaration

This thesis and the work to which it refers are the results of my own efforts. Any ideas, data, images or text resulting from the work of others (whether published or unpublished) are fully identified as such within the work and attributed to their originator in the text, bibliography or in footnotes. This thesis has not been submitted in whole or in part for any other academic degree or professional qualification. I agree that the University has the right to submit my work to the plagiarism detection service TurnitinUK for originality checks. Whether or not drafts have been so-assessed, the University reserves the right to require an electronic version of the final document (as submitted) for assessment as above.

The work presented in this thesis is also present in the following manuscripts:

- ASL-SLAM: An asynchronous formulation of lines for SLAM with Event Sensors (Nong Xiaoqi, Simon Hadfield), In Proceedings of the International Conference on Industrial Application Engineering (ICIAE), 2022.

Signed:

Date: 29/09/2022

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, mobile robot technology has developed rapidly. The robot is capable of moving, automatic navigation, multi-sensor control, human-computer interaction, etc. The use of mobile robots has made a significant breakthrough along with the development of artificial intelligence. The application of mobile robots has transcended the industrial assembly lines, making unmanned factory or package delivery by a robot a reality. People are committed to the application of mobile robots in various scenarios, from indoor and outdoor motion robots to service robots and industrial robots.

When a mobile robot must solve any task, an essential precondition is to know where it is and what its surroundings are, and other problems(Fig.1.1). Due to the importance of the problem, researchers have explored using visual sensors, speech recognition sensors, olfactory sensors, lidar, ultrasonic sensors and more to achieve the localization and navigation tasks

In general, there are two kinds of localization problems. Relative (local) localization methods evaluate the position and orientation relative to some observed location (often the start pose). These techniques use the information provided by onboard sensors like INS (Inertial Navigation System), cameras, and laser and ultrasonic sensors. Methods that obtain the absolute position using beacons, landmarks or satellite-based signals are called absolute (global) localization, such as GPS (global position system) and GNSS (global navigation satellite system). This type of localization is relative to an external, often unobserved but universally agreed, frame of reference, which researchers and engineers have developed for many years.

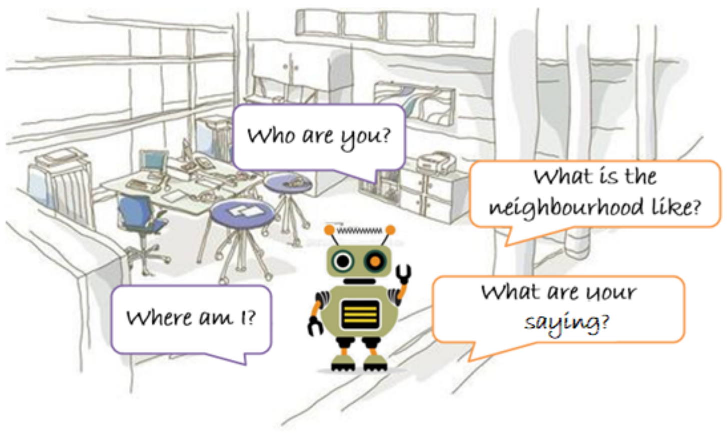


Figure 1.1: An Intelligent Robot in the Office

However, some new sensors have appeared in the past decade that have significantly changed the way we deal with many visual localization tasks. The emergence of cheap commercial depth sensors around 2010 led to massive growth in computer vision research. More recently, new dynamic vision sensors called event cameras are beginning to emerge, which are activity-driven. Techniques have been proposed to use these sensors for feature detection and tracking [46], 3D reconstruction [7], object recognition [11], simultaneous localization and mapping (SLAM) [1] and more. The advantages and disadvantages are summarized in Table.1.1 This thesis will attempt to introduce the first formalism for undertaking line-based projective geometry from event streams and demonstrate its applications to SLAM.

1.1 Localization and Mapping

As discussed, there are various methods and sensors that can be applied to indoor and outdoor localization. Here we will contrast the advantages and disadvantages of different approaches.

GPS was invented in the 1970s and becomes a very mature wireless navigation system. It calculates the distance from a collection of satellites and the Earth's ground point to get the absolute location. The accuracy of GPS is relatively high, but it is highly subject to signal strength, especially in indoor environments. A common alternative for absolute indoor localization is to use a map or floorplan of the building to get the absolute position.

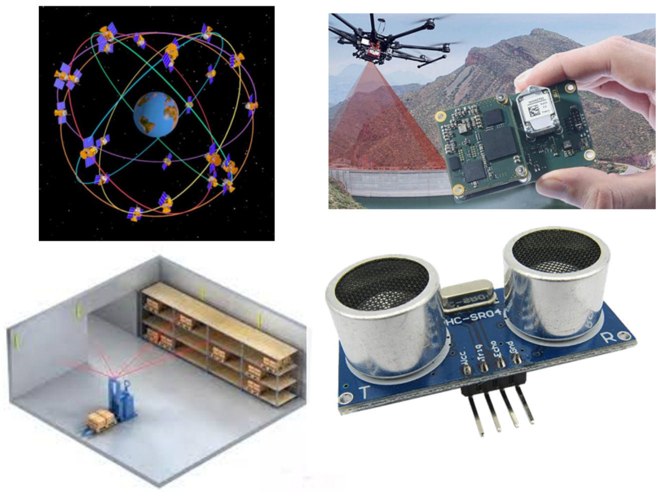


Figure 1.2: Sensors and techniques. The top left is GPS, the top right is IMU, the bottom left is LIDAR, the bottom right is ultrasonic sensor

LIDAR also achieves positioning by calculating the distance between the obstacles and itself. It has the merits of long detection distance, high measurement accuracy and high angular sensitivity. However, it usually needs a point cloud map to get the exact position and has a vital drawback of high cost when compared with other embedded sensors.

IMU(inertial measurement unit) has been a very active research subject as a small portable device in recent decades. It is a low-cost positioning system consisting of a 3-axis accelerometer, a 3-axis gyro, sometimes equipped with a magnetometer, combined to calculate the position and posture of the carrier. It can be applied with odometry algorithms, which are a type of relative localization method that only calculates the position change in a short time interval. Due to the high precision of inertial navigation, the system can undertake the attitude calculation and position evaluation based entirely on its measurements. It is widely used in aerospace technology. However, when the element is used in isolation for a period of time, the error will accumulate due to the integration, which is not conducive to the long-term positioning of the robot. An easy approach to this problem is to adopt high-precision inertial sensors, but the high cost limits its universality.

Camera is another popular solution for robots to estimate the position. In 2004, the Mars probe MER of NASA successful applied visual localization in the wild. Since then visual sensors have been widely admitted by researchers for their advantages of small volume and low power consumption. The visual positioning method gets the motion parameters of the mobile robot by obtaining an image sequence, extracting and describing the feature points in the image then performing feature matching across frames. It does not depend on other sensors and is unaffected by environmental complexity. Therefore it can be applied to a variety of unknown indoor environments. Traditional visual sensors such as monocular, binocular(stereo vision), RGBD (shown in Fig1.3) and omnidirectional cameras are commonly used in the estimation of the robot's pose and position. The stereo camera has two lenses, each of which has an image separately. Two images obtained means scale can be retrieved and more features for tracking, also the lenses need calibration[65]. The algorithm based on the depth camera or stereo camera is similar to the laser solution. The distance of obstacles can be calculated directly by collecting point cloud data. In contrast, the monocular method is a lower-cost solution and has low computational complexity, which is valuable for small robotics because it can work in real-time on limited compute hardware[15]. The monocular and fish-eye camera schemes use multiple frames to estimate the position and pose change, and the distance from the object is calculated by the cumulative position and pose change, then the location and map construction is carried out. However, scale uncertainty is unsolvable when camera motion is not constrained for monocular localization systems. Such as when an obvious change at the end of the road happens, it may lead to the wrong estimation of the scaling factor, which further affects the trajectory's estimation.

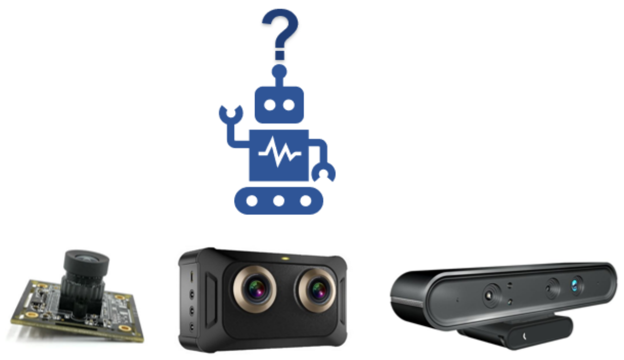


Figure 1.3: Monocular, Stereo and RGBD cameras

As discussed above, data from sensors need further process before getting the position. Normally it is the change in position over time that we estimate from motion sensors, odometry is a significant method for robots to estimate the position from its start point. Visual odometry(VO) is the process of estimating odometry using only a stream of images acquired from single or multiple cameras firstly proposed by [64]. VO is a pose estimation algorithm that can be used to localize robots, cars and other moving objects. The core of VO is that cameras' pose changes can be calculated from the difference between two images acquired from the input video. However, the drift accumulates as the VO runs. Then an expanding system appeared later, as one of the most advanced positioning technology in mobile robot research. As for VO, SLAM has not only localization but also mapping. It attempts to figure out the robot's motion in a completely unknown environment, and at the same time construct a map of that environment.

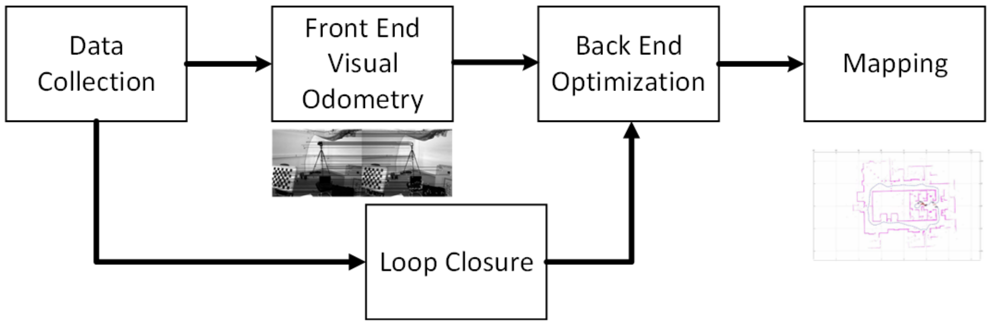


Figure 1.4: Diagram of a complete visual SLAM system and pose construction in the SLAM process

A classical visual SLAM system consists of five modules: Data collection, Visual odometry (Front End), Optimization (Back End), Loop closure and Mapping. In the beginning, the information obtained by the camera is read and pre-processed. Then the visual odometry calculates the orientation and position change during that period using a series of algorithms. The Back End optimizes the VO result with the information from loop closure to get the absolute position of the camera. To see if the robot has been in the same place before is a valuable supplement for a perfect SLAM system, it can improve the accuracy of the estimation. At last, a final map of the trajectory is built in the Mapping module. A complete visual SLAM system is shown in Figure.1.4. Nowadays, SLAM technology can be widely used not only for robots and unmanned aerial vehicles, but also for AR, VR and so on. One limitation of this purely visual approach was that motion can only be retrieved up to an unknown scale. Therefore many tasks like autonomous positioning, mapping, path planning can be realized by single or multiple sensors, such as laser SLAM and visual-inertial SLAM.

Unfortunately, visual algorithms lack robustness under fast motion or poor light conditions. Movement estimation can easily fail when visual tracking is lost. In addition, a major limitation of frame-based data acquisition is the computation cost. Many algorithms struggle to operate in complex and noisy scenes on embedded robotics hardware. Efforts have been made to improve the efficiency of these approaches. However, frame-based data acquisition places a fundamental limit on the computation cost. Standard cameras also can be affected by motion blur during fast movement. In contrast, event cameras do not suffer from motion blur and have greatly reduced data bandwidth [28], which means event cameras are promising for visual odometry or SLAM tasks. Here we conclude the advantages and disadvantages of different sensors or methods mentioned above in Table.1.1.

Table 1.1: Comparison of commonly used localization sensors

Localization	Sensors	Advantages	Disadvantages
Aboslute	GPS/GNSS	Provides both position and orientation using 3-axis accelerometer and gyroscope.	Unavailable in indoor, underwater, and closed areas.
Relative	INS	Provides absolute position with known value of error. No error accumulation over time.	Have long-term drift errors.
	Laser sensor	Has higher accuracy and scan rate Return the distance to a single point; Expensive	Reflection of signal wave is dependent on material or orientation of obstacle surface
	Ultrasonic sensor	Provides a scalar distance measurement from sensor to object. Inexpensive.	Suffer from interference if multiple sensors are used; Low angular resolution and scan rate.
	Optical camera	Stores a huge meaningful information. Provide high localization accuracy. Inexpensive solution.	Requires image-processing and data extraction techniques. High computational-cost.
	Monocular	Low cost and easy deployment. Light weight: good for small robotics. Simple calibration.	Suffer from image scale uncertainty.
	Stereo	Image scale and depth information is easy to be retrieved. Provide 3D vision	More expensive and needs more calibration effort. Difficult interfacing and synchronization.
	Event camera	High temporal resolution; low latency; high dynamic range; low power consumption.	Much more expensive; Unable to access the absolute brightness of single pixel.

1.2 Event Cameras

Thanks to the development of sensor technology, event-driven visual sensors are now available to solve the computational complexity problem. These sensors transmit active events on the event stream asynchronously [77] without the need to redundantly transmit duplicated pixels which have not changed. Event cameras output compressed visual data in the form of a visual event stream. The data provides an increased temporal resolution and lower latency compared to conventional images as shown in Fig.1.5. The red points represent the positive events and the blue points represent the negative events. The positive and the negative means the increase and the decrease of the brightness respectively.

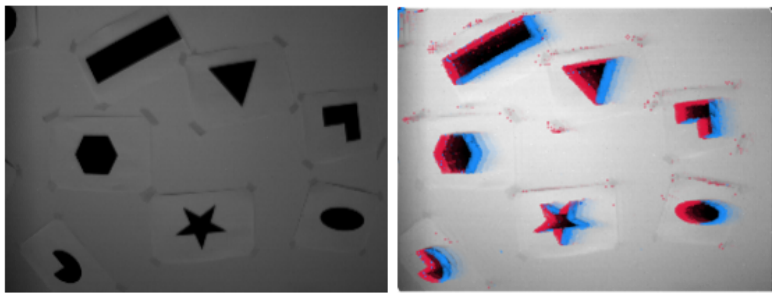


Figure 1.5: Events rendering on the frame and normal frame.

The event stream also records the timestamp with nanosecond precision, in addition to the position and polarity of the change. This stream is equivalent to a high-speed camera taking pictures at a rate of thousands of frames per second, with the additional benefit of far less redundant data [12]. Some event cameras also provide absolute intensity and very sensitive to light changes, which means it has better performance in a low-light environment than a traditional camera.

These characteristics make it possible to combine the benefits of traditional cameras with the unique properties of event-based sensors. This has massive potential for computer vision and high-speed robotics. In essence, visual information from event cameras is asynchronously acquired. Although it sends no information when there is no movement in the scene, it does not have to wait for the next frame before transmitting the signal when motion does occur. This paradigm shift (in Fig.1.6) means it has the advantages of high temporal resolution, low latency, high dynamic range and low power consumption.

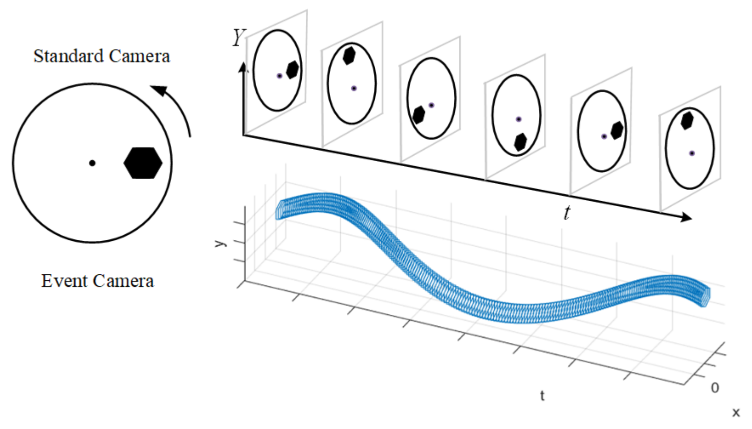


Figure 1.6: A comparison of a traditional frame-based output (top right) and asynchronous line detections. The scene comprises a black hexagon rotating on a disk. The normal camera outputs frames at a fixed rate, instead the event camera produces the stream of events responding to brightness changes from which asynchronous lines can be extracted.

As this technology is still in it's infancy, there are several types of event camera that can provide varying information. Fig.1.7 shows four platforms of event camera which are DVS, DAVIS, DVXplore and stereo event camera kit from left to right. The original and most basic event camera is DVS, normally with a resolution of 240×480. Then an alternative sensor was developed with an absolute intensity value. The Asynchronous Time Based Image Sensor (ATIS) has pixels that contain a subpixel of DVS[67] while another subpixel is triggered to read out the absolute intensity. The ATIS reaches an extensive, static dynamic range ($> 120dB$) but has a drawback that the area of the pixels is at least double the DVS pixels area. What is more, the time interval between two incoming events can be long so that the new events could interrupt the measurement of intensity in dark environments.

Another popular sensor Dynamic and Active Pixel Vision Sensor (DAVIS), combines a conventional active pixel sensor (APS) in the same pixel with DVS. A much smaller pixel size of ATIS makes it superior since the photodiode is shared and the readout circuit only increased about 5% compared to the DVS pixel area[10]. DVS events are analysed to trigger intensity (APS) frames on demand.However, the APS readout is limited at a dynamic range (55dB), which will result in information redundancy if the pixels do not change (as with a normal camera). Recently some new variants of the event camera were developed with other sensors fused such as 6-axis IMU in DVXplore. Two event cameras can also be integrated as one stereo platform to imitate traditional binocular cameras and applied to extensive research.

Figure 1.7: Four kinds of event camera platforms

1.3 Motivation and Aims

As a new type of sensor, there are still techniques that can be discovered and developed to take advantage of event cameras. Based on the background in traditional visual odometry, this thesis aims to investigate approaches to visual odometry and localization using event cameras. These techniques will operate directly on the event stream, rather than relying on traditional image-based intermediate representations.

In order to make full use of asynchronous data, similarities and differences have to be discovered between traditional cameras and event cameras. Traditional feature-based SLAM techniques may provide some clues to achieve with event streams while asynchronous event data has to be dealt with to get the simultaneous feature. However, an essential step in developing this new type of event-based SLAM is to create a comprehensive benchmark system that makes it possible to contrast both event and image-based SLAM under different conditions.
The objectives of this thesis are:

1. Create an event-camera SLAM benchmark using data with the following information: normal colour images, event data, ground truth position and covering a variety of different conditions.

2. Design a novel feature detection method based directly on the asynchronous event data.

3. Build a SLAM system for event-based cameras that uses the features developed in objective 2.

4. Conduct an extensive experimental evaluation compared with other SLAM methods using event features on publicly available datasets, demonstrating the validity and efficiency. Analyse the performance with other SLAM solutions.

1.4 Thesis Structure

In this thesis we tackle the problem of line-based SLAM with event cameras in natural scenes and arbitrary 6-DoF motion. This first chapter discussed briefly the development of robot localization, illustrating the methods that are commonly in use. It also detailed the advantages and disadvantages of state-of-the-art SLAM solutions using different sensing modalities. The event sensors were also introduced and compared with traditional cameras.

The second chapter discusses in depth previous VO or SLAM approaches. It also reviews the difference between frame-based methods and direct methods. We conclude by analysing the characteristics of feature points and reveal the limitation of traditional cameras. Then the application of event cameras is discussed.

In the third chapter we integrate the event stream alongside a low-speed visual point tracking in a framework inspired by PL-SLAM [68]. We detail the new general SLAM framework developed, which can estimate the camera motion by extracting line information from event streams. We make full use of asynchronous data to produce line feature tracks with a high temporal resolution by avoiding the need to accumulate events into frames. To achieve this, a line feature extraction approach that works directly on the event streams is designed.

Chapter 4 provides results based on the proposed algorithm. We propose a standard benchmark and give the performance evaluation of the ASL-SLAM in several scenes from different datasets. We compared our system with the current state-of-art frame-based SLAM methods, such as ORB-SLAM and PL-SLAM by employing their open-source implementations. An in-depth discussion of the results is also presented in this chapter. Finally the conclusion and future work have been described in chapter 5.

Contributions: In summary, our contributions are: 1: A novel asynchronous line detection method based directly on the event stream. 2: The first publicly available SLAM system built on event cameras, and the first SLAM system built on asynchronous lines. 3: An extensive experimental evaluation is conducted on asynchronous features compared with other SLAM methods on publicly available datasets, demonstrating that the system is computationally efficient, running in real-time on a standard CPU.

Chapter 2

Review of SLAM Technology and Event Cameras

SLAM has become a mature technology to achieve localization whether indoors or outdoors. Techniques below are some state-of-art SLAM methods [22] which a robot may use to estimate its position in an unknown environment and build a map of the environment simultaneously. The main difference between Visual Odometry (VO) and SLAM is that VO mainly concentrates on locally consistent estimates to incrementally calculate the path of the camera/robot pose. In contrast, SLAM techniques focus on obtaining global consistency of the camera/robot trajectory and map. As referred to in the previous chapter, VO can be an essential part of the visual SLAM system. According to the types of data processing, SLAM approaches can be categorized into filtering approaches (such as particle filter based SLAM [58] and EKF-SLAM [76]) and smoothing approaches (eg. RGB-D SLAM [38], Graph SLAM [78], Smoothing and Mapping [18]). It also can be divided into feature-based approaches and direct methods based on the type of data association. In addition, there is a long history of developing SLAM algorithms based on new sensors. The most recent challenge is applying SLAM to event cameras. Although there are some studies illustrating Event-based VO/SLAM, there is still significant scope for improvement. In this study, we mainly focus on feature-based SLAM using event features and traditional features.

This chapter will first look at the traditional technology using standard cameras, the difference between the frame-based method and the direct method, the specific problems that arise due to the limitation of traditional cameras and achievements made by the new event camera.

2.1 Feature-based VO/SLAM

VO is an effective non-contact positioning method and one of the most common visual navigation topics in robotics research, especially for indoor applications. Monocular vision has lower accuracy than binocular stereo vision does under certain scenes, such as indoor rooms with a small scope and low environment complexity [4]. However its computational complexity is lower, which helps achieve real-time positioning. As such it is more suitable for synchronous positioning and 3D online map creation in a complex environment [13].

In 2004, the first monocular VO running in real-time in a large-scale environment was proposed by Nister [64]. It detected Harris points using both single and stereo cameras. The results show competitive performance compared with INS and GPS methods. Feature tracking methods are used for the estimation of position change with the constraint of RANSAC [26]. Then the 3D to 2D camera-pose recovery is applied to update the camera pose.

2.1.1 Monocular Feature-based SLAM

Recently the most widely used feature-based approach is ORB-SLAM [61]. Under normal operating conditions, it can provide robust camera tracking and mapping. Subsequently, ORB-SLAM2 [62] was proposed to improve performance by using bundle adjustment. Both approaches rely on fast and continuous tracking of ORB features [73]. Many alternative feature detectors exist, for example, Harris points [34] are based on Moravec's corner detector when a non-corner region is defined to have no change in image intensities between adjacent regions in all directions. FAST (Features from Accelerated Segment Test) points [72] are known to be one of the most computationally efficient feature extraction methods. Then ORB method combines the FAST feature detector and BRIEF descriptor [14]. From this research, we summarize the performance and characteristics of some typical features which have been tested and contrasted in [37] [89] [40] as shown in Table 2.1.

From the table above, we can see that the FAST method has the best calculation speed as well as the most feature points while SURF and ORB have better rotation robustness, scale robustness

Table 2.1: Comparison of features

	FAST	Harris	SURF	SIFT	ORB
Calculation speed	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆	⋆⋆⋆
Feature Amount	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆⋆	⋆
Rotation robustness	⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆⋆
Scale robustness	⋆	⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆
Intensity robustness	⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆⋆

and intensity robustness. As a result of the relatively slow calculation speed, the SURF [8] and SIFT are less frequently applied to simultaneous localization, even if they are recognized as the most robust feature detectors but this comes at the expense of computation. In practice, it is not feasible to achieve real-time calculation of the SIFT features on the CPU [84]. Although the Fast and the Harris show stable performance in detection, the lack of descriptor means the features cannot be matched or tracked in SLAM. Even the ORB, detector requires almost 20ms to get a result. However, these traditional feature methods fail when the mobile platform comes to a poorly texture environment (such as a blank wall or empty hall) or suffers from motion blur. In these situations it is hard to get an accurate odometry estimation. These issues led researchers to explore a more robust representation such as edge features and line features [6].

2.1.2 Stereo VO/SLAM

Binocular stereo vision technology refers to two images taken by two cameras from different parallel positions at the same moment [42]. The parallax of the object is calculated by the similar triangle principle, and the 3D stereo information of the environment is recovered accordingly [39]. The study of binocular stereo vision technology has significant theoretical and practical value, and researchers have invested a lot of time in this field. Many feature-based approaches discussed in the previous part can also support stereo alternatives. The difference is that the depth can be calculated from the two cameras' parallax. It can be concluded that the binocular stereo vision system will have a more accurate scale and depth than the monocular system does. However, there are shortcomings that can not be ignored. First, binocular vision systems are computationally intensive and require high CPU processing speed

[31]. Additionally, it is difficult to design a low-cost FPGA or custom compute hardware to increase the computational efficiency of full parallel optimization [74]. Most mature binocular vision products use computing units with GPUs for parallel feature point calculation and stereo matching. Others optimize stereo matching algorithms and develop special FPGAs to complete the optimization [1]. Second, the binocular stereo alignment effect has a large impact on the accuracy of the solved distance, and a small matching error can lead to a huge error in the measured distance.

2.2 Direct VO/SLAM

The direct method is another branch of visual odometry, which differs significantly from feature-based approaches. These methods eschew feature tracking and essential matrix estimation. Instead, they start from a candidate motion which is optimised in order to best match the observations. It uses dense optical flow to calculate the change of brightness patterns from one frame to the next [83]. Not only can optical flow track the camera motion locally, Engel et.al [24] also built a consistent, large-scale map of the environment. Later they proposed Direct Sparse Odometry (DSO) [23] that can sample pixels from all image regions having intensity gradients. This method works directly on pixel intensities without traditional feature extraction and matching when estimating the pose.

Other optical flow based approaches are closer to traditional feature point tracking, and can be very computationally efficient. They retain the process of feature extraction in the first frame, but replace descriptor matching with optical flow tracking. [27] proposed a fast semi-direct monocular visual odometry (SVO) approach which works well on a micro aerial vehicle. During its localization process, the position change is firstly estimated by minimizing the intensity error of the same pixel between two frames. The optical flow and feature points are used to optimize the current position. Finally, it explicitly models outlier measurements to get 3D points using a probabilistic mapping method.

However, for optical flow methods to be accurate the camera motion needs to be slower or the frame rate must be higher to achieve higher accuracy. An alternative to feature point detection is edge or line detection. This is an efficient and highly parallelizable operation. Rather than detect and match large numbers of points, a small number of lines or edges are often detected and tracked in a direct manner. Edge-SLAM [57] detects edge points in frames and tracks them using optical flow for data association. This edge-based approach is shown to work well in both low-textured and highly-textured environments. In [53], direct lines are used to guide key point selection to increase efficiency. This idea is extended by using the stereo camera in [33]. The edge depth that the monocular method lacks is a valuable attribute to allow acceptable surface fitting [41]. All of these line-based approaches are shown to have high computational efficiency.

2.3 Improvement of SLAM Methods

Researchers have proposed some solutions to overcome the shortcomings of traditional SLAM. These mostly fall under one of three categories: multi-sensors fusion, deep learning and new sensors. Sensor fusion can compensate for the limitations of any individual sensor. Unlike industrial applications dealing with pure, representative and well-defined benchmarks, the camera-laser carrier system is a robust and accurate combination but requires advanced real-time techniques and algorithms to process dynamic unknown objects [30]. Kalman filtering was applied to fuse sonar and camera data which has significantly simplified the SLAM data association problems [43]. GPS can help SLAM to work in an outdoor environment despite complex surroundings and rapid motion [81]. The visual-inertial monocular system is a straightforward way to get accurate estimation at a relatively low cost[19] [50]. Thanks to the recent advances in GPU technologies, more attention can be paid to using deep learning to increase the SLAM accuracy. By training a convolutional network, DEMON [79] achieved not only depth and motion estimation. It is also successful in getting the surface normals, image optical flows and the predicted matches. Both SfM Learner [85] and Monodepth [32] use unsupervised convolutional neural networks to predict the depth of the input images(video).

2.4 Event-based Detection and Tracking

As outlined above, feature detection and tracking is the cornerstone of many SLAM algorithms. As such, it is natural that work has been undertaken in this area using event cameras to achieve asynchronous tracking with high processing speed and microsecond latency especially under high-dynamic sequences. How to detect features using event data becomes the fundamental question in the event-based visual odometry field. Researchers chose features and developed new methods aiming at dealing with event streams and image frames. Efforts have been put into trying to synthesize colour images from event data. In this case traditional features such as Harris can be detected on new event frames [80] [87]. Compared to synthesizing frames, [16] [52] have improved the normal features and detected them directly on event streams, plus [3] [59] achieved tracking on the event stream. [2] also proposed a novel local region descriptor of corner events and corresponding tracker. [46] can track different visual features in real time, which are capable of handling position variations, orientation and scale by the use of multiple pools of trackers. Another linkage of event camera and standard camera are proposey [20]. They extract the features on the normal frames, track them by event streams and update the features using new frames. However, it can be inferred that the traditional monocular camera and event camera need to be calibrated accurately when tracking and modifying the features. The import of the standard camera might reduce the dynamic range of the system, and cause the problem of calibration and synchronization.

Pure event-based feature detection and tracking are proposed in [17]. It intertwined the velocity vector and a bayesian description of the generative feature contours instead of aiming for particular predefined shapes such as corners which saves time on detection. Optical flow (OF) is widely used for feature tracking in event cameras as well as traditional cameras. Generally, the surface of active events(SAE) [9] also called the pixel map of the latest timestamps of the events, is used to compute the flow that is parallel to the brightness gradient. In this case Event lines can be segmented directly from the event streams. [71] uses iterative event-based weighted least squares fitting and optical flow to extract lines. More recent methods, such as [29], [5], [88], estimate the whole optical flow (i.e., both tangential and normal components). This full flow has more information than normal flow does at the expense of computation workload. [66] introduces an algorithm for spatio-temporal tracking that is suitable for the Dynamic Vision Sensor. In particular, it addresses the problem of multi-sensor tracking with high occlusions.

There are many works combing event cameras, standard cameras and IMUs (Inertial Measurement Units) to achieve higher accuracy in feature detection and tracking. [51] takes three kinds of data collected from the above sensors, detecting Harris points and Canny edges, using the estimated optical flow from event streams and the IMU data to update the position of line segments.

Table 2.2: Summary table of some feature detection and tracking methods in literature

Methods	Feature Detection	Feature Tracking	Assumption/Limitation
ACE [2]	Events	Normal frames	tracked features is relatively less and tracking time is longer
Arc* [3]	Events	Event stream	Use a Surface of Active Events
eFast [59]	Events	Normal frames	Detection quality is slightly worse
Clady [16]	Events	Matching on events	Rely on estimation of local velocities.
Fa-Harris [52]	Events	Intensity images	Detecting speed is slight lower
Lagorce [46]	Already known	Event stream	Feature's position has to be known beforehand.
Zhu [87]	Event frames	Event frames	Event density over a temporal window is used.
Vasco [80]	Event frames	None	Not computationally efficient.
Yan [20]	Standard frames	Event & Standard frames	Unstable in different scenes.

However, relying on the edge map from the Canny detector also results in unstable performance in different scenes. Similar to the traditional cameras, the appearance-based method can be also applied to events. Owing to its frame-free, asynchronous event nature, the DVS vision sensor is capable of monitoring a small moving particle seamlessly from one pixel to the next over a range of speeds, limited only by bandwidth [21]. [75] acquires a filtered or time-averaged version of the input event data by using the theory of continuous-time filter. The state of art methods for event feature detection and tracking are summarized in Table 2.2. We can see that tracking directly on event streams is still a challenge for further research.

2.5 Event-based VO/SLAM

As discussed above, there has been a great deal of work on feature point detection and tracking using event cameras. It is unsurprising that some of these techniques have been adapted for

VO/SLAM. Similar to the traditional camera, these techniques cover both monocular event-based VO and stereo event-based VO. For a single event camera, it has the same shortcomings as the traditional monocular camera. The scale can not be estimated, also meaning it needs supplements to get the estimation of depth. It is worth noting that event-only SLAM/VO techniques are very challenging to develop.

A groundbreaking work [44] achieved the first event-based visual odometry with the monocular DAVIS sensor. This work takes feature detections on the standard grayscale frames and tracks them asynchronously with event streams to calculate the 6-DOF motion estimate. [70] is able to reconstruct intensity images from the event stream through a combination of event-based feature tracking techniques and event-based 3D reconstruction techniques. Both approaches utilize the standard images or try to make standard images to get close to the existing VO/SLAM.

However, a stereo event camera can avoid the above issues. The work [86] performs VO with an event-based stereo camera instead of the monocular event camera. It uses time-surface maps to represent the 3D spatial event data. Then a spatio-temporal consistency is enforced from two time-surface maps, which builds the data association based on the events. As mentioned before, INS can also be helpful to event-based SLAM/VO. [90] fuses the event-based tracking algorithm with an inertial measurement unit(IMU), providing metric tracking of the 6-DOF pose.

All of these prior techniques focus on point-based features, which are sparse and hard to extract from event camera streams. Driven by this problem, we intend to develop an event feature-based SLAM system which can overcome the difficulty of the asynchronous data and utilize the advantages of the event camera in this study.

2.6 Event Datasets and Simulators

The world's first collection of event datasets is presented by [60]. It includes not only event data from DAVIS240C installed on high-speed robotics, but also intensity images, inertial measurements as well as the ground truth obtained from the motion-capture system. It provides various situations such as indoor simple shapes scenes, wall poster scenes even complex outdoor movement, which will contribute to the research of event-based visual odometry and SLAM.

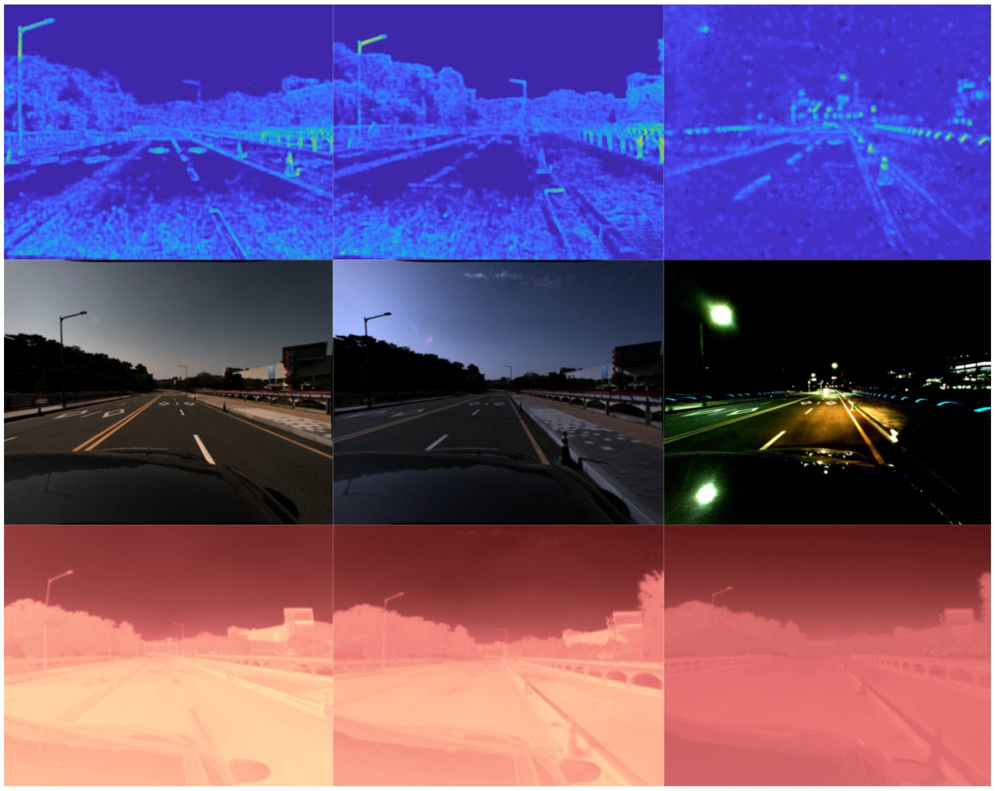


Figure 2.1: Samples of event edges (upper), RGB (centre), and thermal images (lower) from day1, day2 and night sequences, from left to right[48].

Recently, [48] has updated their datasets based on their previous ViVid dataset[49]. It expanded their experiment to driving sequences with more details on dataset statistics and characteristics. Compared to [60], it has more data from standard RGB cameras and depth cameras shown in Fig.2.1. This makes it possible to contrast the performance between event camera and other sensor types.

Since the event sensors are still scarce and expensive to get, [60] has also provided the event simulator which offers a good baseline. It generates the streams of events, intensity frames and depth maps which can be seen in Fig.2.2. A computer graphics software Blender is used to generate thousands of rendered images of the scene, which are then differenced in order to extract events. Later, they provide an enhanced open-source simulator for event streams. This was the first event camera simulator [69] able to generate large amounts of reliable events for complex scenes.

Building on this, the first differentiable simulator of event streams is introduced [63]. It can

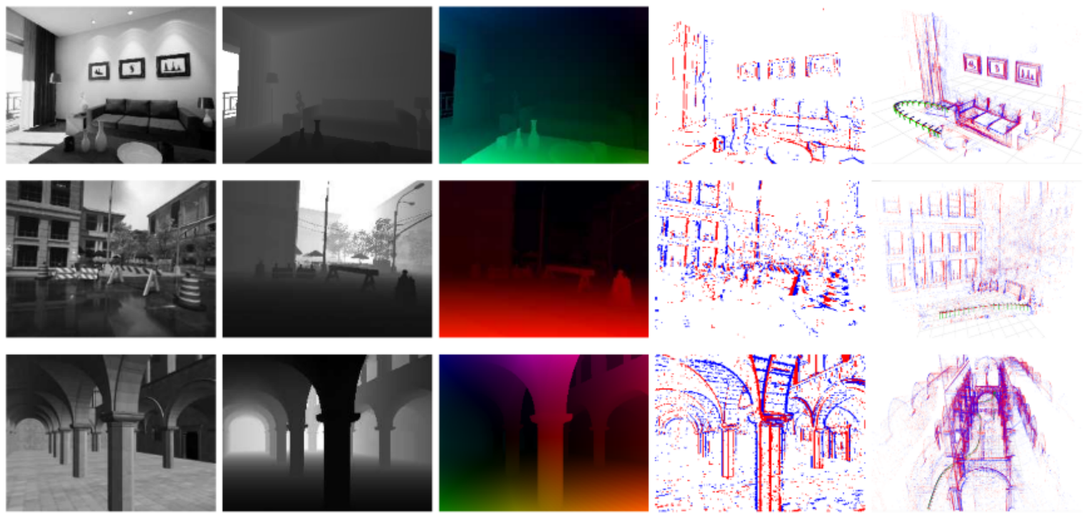


Figure 2.2: Example outputs from ESIM. From left to right: a preview of the scene, depth map, optical flow, events, 3D event point cloud with camera trajectory[69].

simply be applied to the original incoming events without any pre-processing. This can be helpful for event-based tracking and reconstruction of non-rigid objects in 3D, like hands and bodies.

In this thesis, a number of suitable VO datasets were explored. Our experiments will primarily focus on real-life data collected within the ViVid dataset as it enables us to compare fairly against monocular VO techniques.

Chapter 3

Feature Detector Background

The literature review has shown that in most visual localization, valid features first must be extracted from the local environment to describe the robot's surroundings. These markers may describe artificial characteristics or natural characteristics. The former are artificial patterns with special significance, such as QR codes, bar codes, numbers, Chinese characters, etc. The latter is the natural expression of non-artificial objects in the real environment, such as table corners, windows, chairs, fans, etc. In this chapter we will formalize and contrast some typical features that are commonly used in visual localization technology, for both RGB and event cameras.

3.1 Traditional Feature Detection and Tracking

First we will formalize two traditional feature point detection methods using frame-based cameras. We will specifically focus on detailing the frame-based feature detection algorithms which are necessary to understand the subsequent event-based detectors. All these discussions below will make it easier to understand the event-based features.

3.1.1 Harris Feature Detector

One of the most widely used classical corner detectors is the Harris corner detector, the idea behind it is to detect points based on the intensity variation in a local neighbourhood. Harris [35] identified a corner when the intensity gradient in two perpendicular directions is larger than any neighbouring region. Let the image be a scalar function $I : \Omega \to \mathbb{R}$. The process can be summarised in 5 steps: First the horizontal and vertical gradients images $I(x, y)$ must be computed by convoluting I with a derivative kernel:

$$\begin{aligned} I_x &= I \otimes (-1, 0, 1) = \frac{\partial I}{\partial x} \\ I_y &= I \otimes (-1, 0, 1)^T = \frac{\partial I}{\partial y} \end{aligned} \tag{3.1}$$

Next the autocorrelation matrix M is computed: $M = \begin{bmatrix} A(x) & C(x) \\ C(x) & B(x) \end{bmatrix}$:

$$\begin{aligned} A(x) &= g(I_x^2) = I_x^2 \otimes \omega \\ C(x) &= g(I_y^2) = I_y^2 \otimes \omega \\ B(x) &= g(I_{x,y}) = I_x I_y \otimes \omega \end{aligned} \tag{3.2}$$

where the function ω allows selecting the support region that is typically defined as a rectangular or Gaussian function. Next the corner response value R is computed:

$$\begin{aligned} R(x) &= det(M) - \alpha(trace(M))^2 \\ S_H &= \{x : R(x) > t\} \end{aligned} \tag{3.3}$$

S_H represents the corner set. After computing non-maximum suppression, output corners will be selected. The parameter α is a constant that can affect the performance of the corner detector. The corner response value R will decrease when the value of α increases and the number of detected corner points will also be reduced making the detector more selective.

3.1.2 FAST Feature Detector

In contrast to the linear algebra approach employed by Harris, the traditional FAST corner detector applies a segment test to candidate points. The rule is to detect a feature point at position p, if there exists a set of n contiguous pixels in the circle with radius r that are all darker than I_p minus a threshold t, or all brighter than $I_p + t$. Fig. 3.1 shows the idea where pixels on the circle are the pixels used in the corner detection. The 2D pixel offsets belonging to this circular domain are grouped into the set $O = O_1, O_2..O_{16}$.

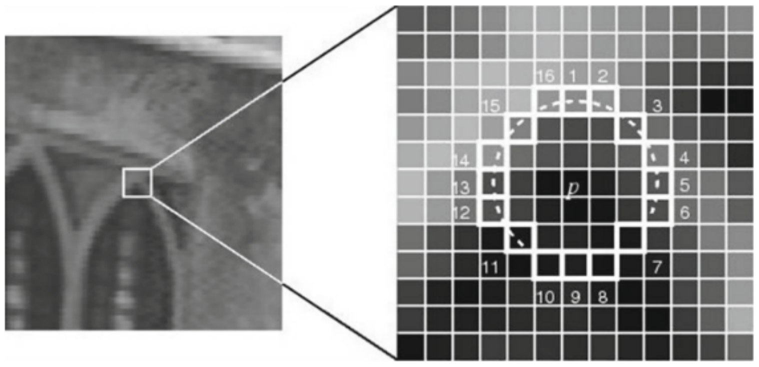


Figure 3.1: Feature detection in an image patch using the FAST detector[72]

The pixel at position p is added to the set of FAST corners if it obeys one of these two conditions:

$$S_f(x,y) = \begin{cases} 1, & \exists i,j \quad I_p + t < min(O_i..O_j) \\ 1, & \exists i,j \quad I_p < max(O_i..O_j - t) \\ 0, & otherwise \end{cases} \tag{3.4}$$

From the above process, the non-feature points can be easily screened out which will help significantly increase the speed of feature detection. However, this also results in very dense features. An algorithm of non-maximum suppression is used to improve its robustness.

3.1.3 Lucas–Kanade Optical Flow Tracking Algorithm

Regardless of which of the previous feature detectors is used, the resulting feature detected in the previous step is generally fed to the next feature tracking step. A common approach to feature tracking uses the Lucas–Kanade (LK) [56] method. It assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighbourhood. Even for event-based feature points, we can find the corresponding pixel on the normal image frame.

To calculate the optical flow, every moving point q must satisfy:

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{3.5}$$

where q_1,q_2,...,q_n are the pixels inside the window, and $I_x(q_i)$,$I_y(q_i)$,$I_t(q_i)$ are the partial derivatives of the image I with respect to position x, y and time t, evaluated at the point q_i and at the current time. So V_x and V_y can be calculated by the linear equations:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_k(q_i) \\ -\sum_i I_y(q_i)I_k(q_i) \end{bmatrix} \tag{3.6}$$

Now that the motion of the detected feature points is known, it can be used to estimate the movement of the camera. The core of event-based VO is calculating relative position by the relative displacement of feature points between two images. Once we have point-correspondences, the geometric relationships between adjacent frames can be described by the essential matrix:

$$E_{t,t-1} \simeq = \widehat{T}_{t,t-1} R_{t,t-1} \tag{3.7}$$

$\widehat{T}_{t,t-1}$ and $R_{t,t-1}$ are the translation matrix and the rotation matrix respectively from time $t-1$ to t. Then RANSAC [25] will be used to find five feature correspondences between two successive frames to estimate motion accurately. The RANSAC is an iterative algorithm. At every iteration, it randomly samples five points from a set of correspondences, estimates the Essential Matrix, and then checks if the other points are inliers when using this essential matrix. Let the pose of the camera be denoted by R_k, T_t. The trajectory can be tracked using the following equation:

$$\begin{aligned} R_t &= R_{t,t-1} R_{t-1} \\ T_t &= T_{t,t-1} + t_{t-1} R_{t,t-1} \end{aligned} \tag{3.8}$$

3.2 Event-based Feature Detection

The asynchronously and continuously updated output of an event camera is basically different from grayscale images. Thus accordingly changes are necessary for making a corner detector work on the event stream. As shown in equations 3.1 - 3.4, both traditional FAST and Harris corners are calculated over a local neighbourhood. However, in an event stream the definition of a local neighbourhood is unclear. Fortunately, for event cameras, the lack of redundant data makes it fast to exhaustively check the events, despite the lack of efficient convolution operations. This check is performed at the moment the event occurs asynchronously. As a replacement for the local neighbourhood, we use the concept of the Surface of Active Events (SAE), which is similar to an elevation map. Since visual information is indexed by time, the event corner can be assumed to operate on the SAE $\mathcal{S} : \sum (x, y, p) \mapsto t$. The SAE maps the position on the image plane to the timestamp of the latest event, where t is the most recent event triggered at the pixel location (x, y). The local neighbourhood in $\mathcal{S}$ of the new event (i.e., the temporal ordering of the latest triggered events in the neighbouring pixels), is used to test whether the new event is a corner. It is worth noting that the event camera can only detect moving corners. Static corners will not create any intensity changes so no events will occur. This is generally acceptable for dynamic tasks like VO.

3.2.1 eHarris Event Corner Detector

The event-based Harris Detector binarizes the SAE by the latest N events. They defined a local spatial window (W) which is L pixels wide, using the binarized surface $S_b(x, y)$ instead of the 2D image $I(x, y)$ in the traditional Harris detector. The binarized SAE is defined as

$$S_b(x,y) = \begin{cases} 1, (x,y)\ is\ one\ of\ the\ N\ most\ recent\ events \\ 0, otherwise \end{cases} \tag{3.9}$$

Fig. 3.2 shows an example of the binarized SAE when the local window includes an edge (a) and a corner (b), the black patterns correspond to “1” in the binary local map where an event occurred. The local change in contrast is high in (b) along the two major axes, compared to (a) which is high along one direction.

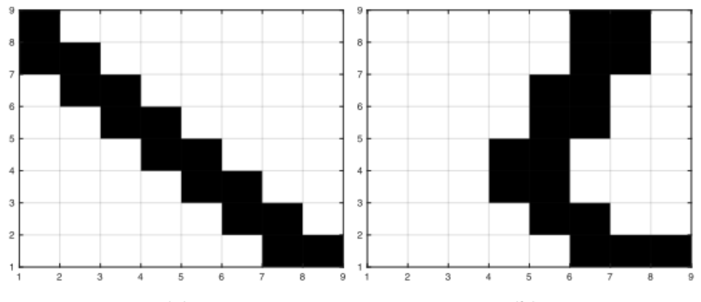


Figure 3.2: A visualization of the asynchronous local contrast map[80].

Then compute the gradients of the obtained binary surface $\Delta S_b = [\frac{\partial S_b}{\partial x}, \frac{\partial S_b}{\partial y}]$. These are substituted in place of Ix and Iy in equation (2), and then the algorithm proceeds as for the frame-based harris detector.

3.2.2 eFAST Event Corner Detector

Similar to e-Harris, eFAST [59] has enabled traditional FAST features to be computed from event streams. It compares the timestamps of the most recent event of the pixels on two circles around the current event. As a result of the latest event is identified as the centre pixel of the local neighbourhood it always has the highest timestamp on the SAE. The whole process can be described as: firstly a circle that is similar to traditional FAST (see Fig. 3.3) is proposed:

$$C^r = \{C_0^r, C_1^r ... C_n^r\}, n = (r+1)^2 - 1 \tag{3.10}$$

where C^r is the set of pixels with radius r from the centre pixel. C_0^r is the pixel directly above the centre pixel on the circle, with subscripts of each pixel increasing in a clockwise manner around the circle.

$$|P - C_i^r| = r \tag{3.11}$$

Assume indices are periodic such that: $C_{n+1}^r = C_0^r$. A potential feature point is detected if there exist two indices i and j, $(i, j \in (0, n))$, the timestamp of which satisfy:

$$min\{t_{C_i^r} ... t_{C_j^r}\} > max\{t_{C_j^r} ... t_{C_i^r}\} \tag{3.12}$$

if $|i - j| > T_r$, where T_r is the pre-set threshold time.

Unlike the frame-based FAST detector, e-FAST uses two circles C_1 (inner circle) and C_2 (outer circle) of radius of r_1 and r_2 around the latest event on SAE $\mathcal{S}$. Both circles must contain a contiguous block of recent events that are used to decide whether it is an event corner.

In Fig. 3.3, for circle C^1 (red), a segment of length between 3 and 6 (pixels) ($3 <= T_{r1} <= 6$)is required, for C^2, a segment of length between 4 and 8 ($3 <= T_{r2} <= 6$) is needed. A corner is classified when both circles need to meet the condition above.

3.2.3 Arc* Event Corner Detector

Inspired by the FAST event corner detector(eFAST), Arc* is presented as a means to detect corners much faster. Firstly, it proposed an event filter that is more strict to limit the timestamp

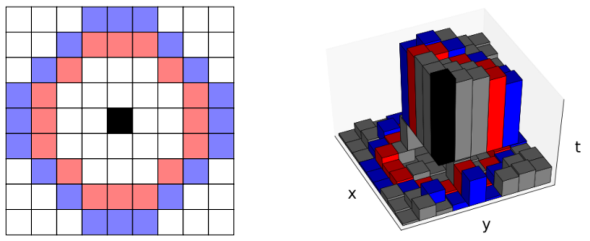


Figure 3.3: The inner (red) C_1 circle and outer (blue) C_2 circle around the latest event (black) and visualization of the SAE around the latest event (black) and of the circles used for the timestamp comparison[59].

range of the SAE, such that $\mathcal{S} : \sum (x, y, p) \mapsto (t_b, t_l)$. The value of t_l is always updated when the newest event happens, $t_l \leftarrow t$, as in the standard SAE. Meanwhile, the reference time t_b is only updated if the time since the last event at that location exceeds k (such that $t_b \leftarrow t$ if $t > t_l + k$), or if the polarity of the latest event triggered in the same location differs from the polarity of the incoming one.

After this filter, arc* proceeds similarly to the original eFAST algorithm selecting a subset of the circles around the newest event, and searching for a contiguous circular arc whose angle is within a pre-defined range. However, eFAST fails to detect some corners such as in Fig. 3.4 where the angle of the arc of the newest elements is now over 180°. Then Arc* has considered both situations and optimized the basic constraint with an equation 3.12 and additional equation 3.13 (one represents the corner on the right and the other represents the corners on the left):

$$max\{t_{C_i^r}...t_{C_j^r}\} < min\{t_{C_j^r}...t_{C_i^r}\} \tag{3.13}$$

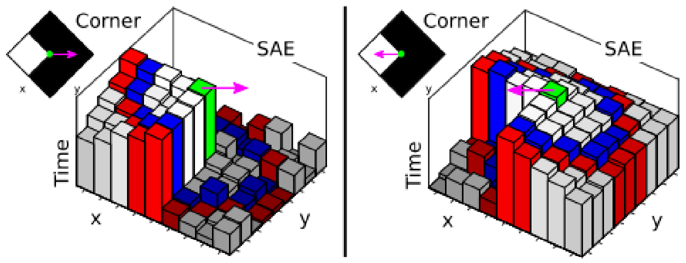


Figure 3.4: Two different motion directions of the same corner[3]

3.2.4 The Fa-Harris Event Corner Detector

In the previous section, both the eFAST and the eHarris detector construct and maintain a local SAE with a constant size for each pixel in the image plane with a size up to M×N, which means there are up to M×N local SAEs to be calculated. Representing this necessitates a complex data structure which is time-consuming[52]. A single global SAE (G-SAE) with size M×N is proposed which saves the newest timestamp of the event at the corresponding pixel. When a new event occurs, G-SAE will be updated at the corresponding position of the event. Then the local SAE centred on the event will be extracted from G-SAE (9×9 size in the algorithm). The local SAE is then used for corner candidate selection and refinement. The events of two polarities are handled on two different G-SAEs respectively. This greatly accelerates the speed of the algorithm.

3.3 Conclusion

In this chapter we discussed some traditional features and event features. A comparative evaluation of these different techniques will be provided in chapter 5. However, it is obvious that many event features are derived and developed from classic methods, simply exchanging the pixels for events. Because events are not synchronised into frames, the techniques must also be adapted to operate across the time dimension. The results in [59] show that asynchronous feature detection methods have better real-time performance than those which accumulate events into artificial frames. Based on this observation we propose to detect our line features by fitting event planes in the continuous event stream.

Chapter 4

Asynchronous Line SLAM

We have seen that feature-based tracking is still the most common form of SLAM. There have also been a huge number of frame-based and event-based feature-point detectors proposed. The extraction speed and reduction rate differ greatly between these techniques. Unlike traditional monocular or stereo SLAM, event-based SLAM has several additional steps making the feature extraction more complex. As we discussed in last chapter, event based features are rare, especially in low-textured scenes with limited motion. Therefore, to address objective 3, a new general SLAM framework will be developed which can estimate the camera motion by extracting line information from event streams, as shown in Fig. 4.1. The pipeline of our system is inspired by the integrated points and lines tracking of the PL-SLAM, but using our asynchronously extracted event lines with dense image based feature points. The main idea behind the algorithm is to identify the coherent space-time event surfaces using the local SAE. These event surfaces are then used to define both the line features in the image and their motion. Local Bundle Adjustment is applied to optimize the pose of all lines after obtaining the initial line feature set, find further correspondences can be established by projecting the local map onto the image.

4.1 Event Camera Feature Detection

Before discussing feature extraction for event cameras, we must provide basic background on how event cameras themselves function. The event camera has a higher temporal resolution

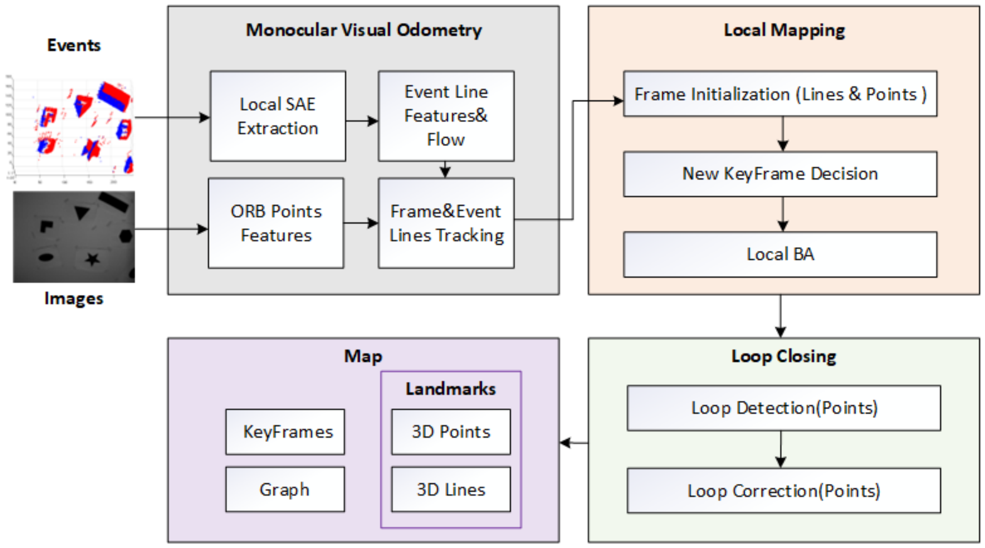


Figure 4.1: The scheme of ASL-SLAM is composed of three main threads: Data Extraction & Tracking, Local Mapping and Loop Closing. Event lines are extracted from the fitted plane. Both lines and points are tracked between frames. Then camera position is estimated and the new keyframes are chosen. Then, the latest keyframe information is added to the map and optimized with BA during Local Mapping. The Loop Closing process keeps optimizing the map and poses with the bag of words.

because it responds to intensity changes in the environment independently and asynchronously for each pixel. Then the output of the event sensor is the representation of intensity change (log intensity) and the position of the corresponding pixel. The log intensity has ON/OFF polarity which indicates an increase or decrease of brightness.

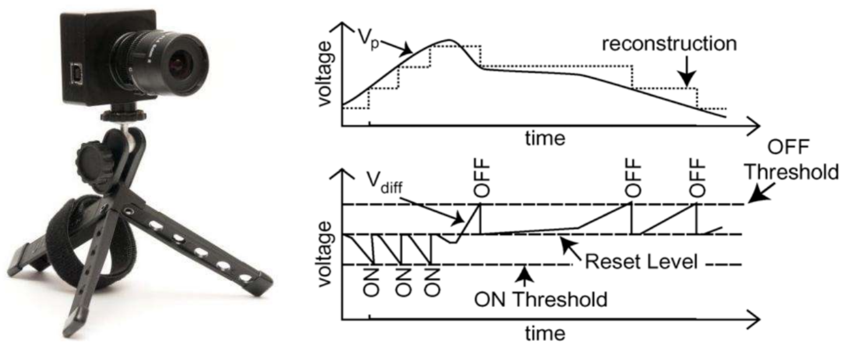


Figure 4.2: Event camera and the principle of ON and OFF spike generation for DVS pixels[9]

The DVS is shown on the left in Fig. 4.2. The top right shows the details of triggered events, and the evolution of the pixel's voltage V_p. This can be roughly interpreted as the intensity of light received by the pixel over time. It shows the corresponding generation of ON (voltage increases above change threshold) and OFF (voltage decreases) events, from which the change of V_p can be reconstructed. The event camera captures the events with a pixel array before formulating the event stream through the peripheral circuit [54] and outputs the stream using a shared digital output bus. This route will take advantage of a kind of address event representation (AER) readout, which grants a faster read speed [55]. The flow chart in Fig. 4.3 below explains how light is converted into event data:

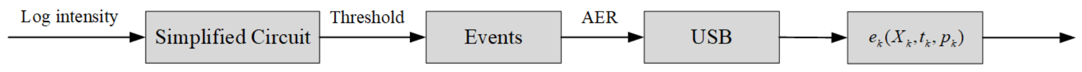


Figure 4.3: Working Process of Event Camera

Specifically, at time t_k we define the brightness change between the latest event and current event at pixel $X_k = (x_k, y_k)^T$ as:

$$\Delta L(X_k, t_k) = L(X_k, t_k) - L(X_k, t_k - \Delta t_k) \tag{4.1}$$

where Δt_k is the time interval since the last event.

An event $e_k[x_k, y_k, t_k, p_k]$ is then launched when $\Delta L(x_k, y_k, t_k)$ exceeds a contrast threshold C (with $C > 0$):

$$\Delta L(x_k, y_k, t_k) = p_k C \tag{4.2}$$

where the polarity $p_k \in \{+1, -1\}$ indicates whether the brightness increased or decreased. Here, a generator that has on-chip digitally-programmed bias can produce the pixel bias currents. It set the speed and threshold voltages of a change detector on the right in Fig 3, and also decides the contract sensitive threshold C.

4.1.1 Plane Fitting and Filtering

Firstly we define the event stream as the set of all events $\mathcal{E} = \{e_0, e_1 ... e_N\}$, where $e_i \in \mathbb{R}^3$. Each event is constructed as $e_i = [x, y, t, p]$, where $[x, y]$ is the position of the event in the event frame coordinates at time t. For every incoming event, we define an event neighbourhood which includes all events with a similar time and position. This SAE neighbourhood is used to provide an estimation of the orientation and amplitude of the event motion as shown in Fig. 4.4.

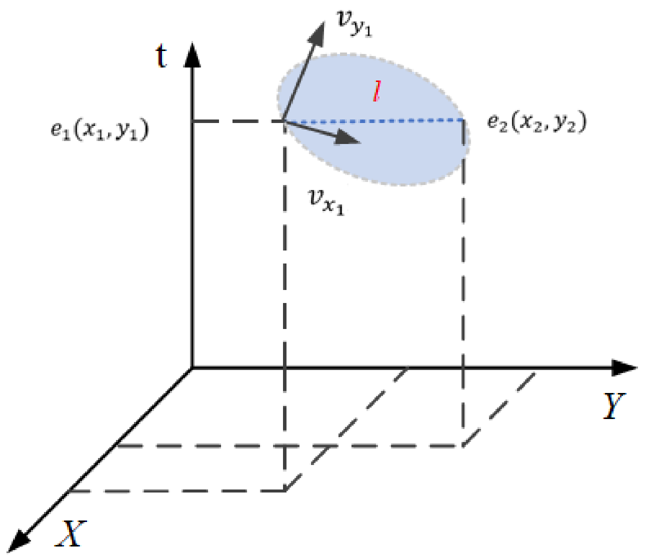


Figure 4.4: An event line is extracted from the SAE with line orientation and amplitude of motion defined by the orientation of the SAE. The extremities of Line l are determined by the inlier points within the local SAE neighbourhood.

A spatio-temporal neighbourhood of events Ω_e is defined as the set of all events falling within a window of size $2L \times 2L \times 2\Delta T$ centred on event e:

$$\begin{aligned}\Omega_{e_i} = \{e_j,\ where\ x_j \in (x_i - L, x_i + L),\\ y_j \in (y_i - L, y_i + L),\ t_j \in (t_i - \Delta T, t_i + \Delta T)\}\,.\end{aligned} \tag{4.3}$$

We parametrize a space-time plane as $\beta = [a, b, c, d]$. It is noted that for any event e_i which lies on this plane, the following equality must satisfied:

$$\beta[x, y, t, 1]^T = 0. \tag{4.4}$$

Here we use the Singular Value Decomposition (SVD) method to fit the plane to minimize the error function 4.4, with the constraint:

$$a^2 + b^2 + c^2 + d^2 = 1 \tag{4.5}$$

For each incoming event, we optimize the plane β to extract the local Surface of Active Events. The optimization process is defined as:

$$\beta^* = \underset{\beta}{argmin} \sum_{e_j \in \Omega_{e_i}} \left|\beta[x_j, y_j, t_j, 1]^T\right|^2. \tag{4.6}$$

Once the initial plane candidate has been computed, we extract the set of inlier events from the neighbourhood as:

$$\Omega^*_{e_i} = \{e_j,\ where\ |\beta[x_i, y_j, t_j, 1]^T| < \lambda_1,\ \forall e_j \in \Omega_{e_i}\}. \tag{4.7}$$

We then repeat equation 4.6 to refine the plane parameters on the inlier events set. We define the update size as $\epsilon = ||\beta - \beta^*||$, and iterate equations 4.6 and 4.7 until $\epsilon < \lambda_2$, at which point the local SAE has converged. It is worth noting that for the current event e_i, the inlier neighbourhood $\Omega^*_{e_i}$ should also contain itself, this means the error of the current point $_i$ needs to be smaller than λ_1. If this condition is not satisfied the plane candidate is discarded.

4.1.2 Event Line Flow

It can be inferred that all the events on the line should have the same velocity over a short enough timeframe, and the velocity remains constant during this time interval. This makes it possible to track the lines without resorting to appearance based descriptor matching which is challenging for event cameras. The velocity $[v_x, v_y]$ can be obtained from the local SAE via:

$$[v_x, v_y] = \left[\frac{-ac}{(a^2+b^2)/\Delta t}, \frac{-bc}{(a^2+b^2)/\Delta t}\right], \tag{4.8}$$

where Δt is the time interval between two frames. After obtaining the parameters of the synchronized 2D lines, and their velocities, we compute the data association with the 3D line segments currently contained in the map.

4.2 Asynchronous Line Extraction and Synchronisation

After obtaining the plane, we then extract the lines using the current event. We synchronize the lines to the frames and each other, so that batch data association can be achieved and there will be enough information to reliably constrain the odometry. Although the event information is asynchronous and discrete, the space-time planes computed above provide a continuous representation of the feature over time. This makes it possible to extract the corresponding visual line at any required timestamp. As such we can synchronize the asynchronous event lines with the visual feature points extracted from the frame-based camera. Based on equation 4.4, we use a homogeneous definition of line l_i extracted around the neighbourhood of event e_i:

$$l_i : a_i x + b_i y + c_i t_f + d = 0, \tag{4.9}$$

where t_f is the timestamp of the frame which the line is being synchronized to. This line describes the intersection of the SAE plane β with a horizontal plane at time t_f. In this thesis we set the length of every line for a fixed value 10, so the 2D coordinates of end points can be obtained by the current event and the gradient of the line function.

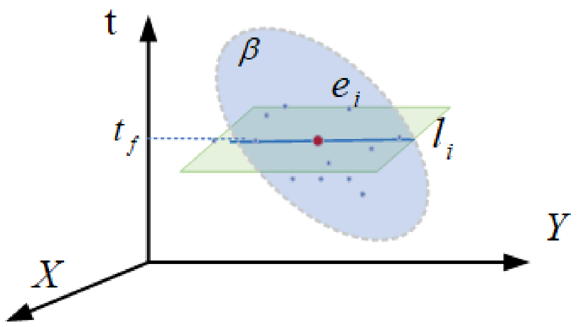


Figure 4.5: The intersection of β with a t_f plane.

4.2.1 Synchronous Data Association with Asynchronous Lines

After obtaining the parameters of the synchronized 2D lines, and their velocities, we compute the data association with the 3D line segments currently contained in the map. The 3D end points of these line segments are projected into the image space. $M, N \in \mathbb{R}^3$ are the extremities of the 3D line segment. As in [33] we define, the algebraic point-line error E_{pl} for a 3D point as:

$$E_{pl}(\theta, M_{t_f}, l_i) = (l_i)^T \pi(\theta, M_{t_f}) \tag{4.10}$$

where the pose parameters $\theta = \{R, t\}$ include the rotation and translation parameters which align the camera and the world coordinate systems. Furthermore, the algebraic line segment error E_l is defined as the sum of squares of the two event-line errors for the 3D line segment endpoints:

$$E_l(\theta, M_{t_f}, N_{t_f}, l_i) = E_{pl}(\theta, M_{t_f}, l_i)^2 + E_{pl}(\theta, N_{t_f}, l_i)^2 \tag{4.11}$$

We associate the newly synchronized event lines with the closest line segment in the current map $\mathcal{M}$, assuming the algebraic error is less than λ_3:

$$l_i \leftrightarrow \begin{cases} \underset{[M_{t_f}, N_{t_f}] \in \mathcal{M}}{argmin} \quad E_l(\theta, M_{t_f}, N_{t_f}, l_i), & if \ E_l(\theta, M_{t_f}, N_{t_f}, l_i) < \lambda_3 \\ \emptyset, & otherwise \end{cases} \tag{4.12}$$

4.3 Local Bundle Adjustment with Points and Synchronized Lines

Once the synchronized event lines are associated with the line segments in the map, we combine them with frame-based feature point correspondences. To get the optimized results of the camera pose, a bundle adjustment (BA) of the local map is used. This constrains θ to an $SE(3)$ pose for each keyframe (KF). After we associate the data with the i^{th} keyframe, let $X_{ij} \in \mathbb{R}^3$ be the generic j^{th} point of the map. The projection error e_{ij} represents the 2D distance between the observation x_{ij} of the i^{th} keyframe:

$$e_{ij} = x_{ij} - \pi(\theta_j, X_{ij}) \tag{4.13}$$

For the event lines defined in the previous section we use the same error function as that used for data association. Namely, we define the error function as the distances between the projected endpoints of the 3D line and its corresponding infinite line in the 2D image plane:

$$e_{ik} = \begin{bmatrix} (I_{ik})^T \pi(\theta_j, M_j) \\ (I_{ik})^T \pi(\theta_j, N_j) \end{bmatrix}, \tag{4.14}$$

where M_j and N_j refer to the 3D endpoints of the line segments in the world coordinate frame. The coefficients of detected line I_{ik} estimated by Eq.(**??**) refers to the 2D line in the corresponding i^{th} keyframe. Because of the unknown camera pose and the noise of the observation, the problem is defined as a joint optimization over the camera pose θ, map points $\mathcal{X}$ and map lines $\mathcal{M}$:

$$\theta^*, \mathcal{X}^*, \mathcal{M}^* = \underset{\theta, \mathcal{X}, \mathcal{M}}{argmin} \sum_{i=0}^{K} \left[\sum_{j=0}^{P} e_{ij}^T u_{e_{ij}}^{-1} e_{ij} + \sum_{k=0}^{I} e_{ik}^T u_{e_{ik}}^{-1} e_{ik} \right], \tag{4.15}$$

where K, P, I represent the number of keyframes, points and lines respectively. $u_{e_{ij}}^{-1}$ and $u_{e_{ik}}^{-1}$ are the covariance matrices for the key points and line endpoints. Given this definition the problem can be iteratively solved by a Damped Newton method which is more robust than standard Gaussian Newton methods.

4.4 Mapping With Points and Asynchronous Lines

It is known that current frame-based SLAM methods sometimes fail when running in low texture environments which lack key points. Rather than using points, event lines are more robust for map initialization between two frames. We make an assumption that the camera's angular acceleration is small between two time steps. This means for three consecutive poses, the angular velocity (i.e. the rotational transition between each camera view) is constant. Thus the three camera orientations can be represented as $R_{t-1} = R^T$, $R_t = I$, and $R_{t+1} = R$, and I is the 3×3 identity matrix. After accounting for the change in camera orientation, the three observations of the line should be colinear with each other. We can quantify this by checking that the cross product of any two of the lines must also be perpendicular to the third line. This constraint can be written as:

$$I_t^T((R_{t-1}I_{t-1}) \times (R_{t+1}l_{t+1})) = 0. \tag{4.16}$$

Moreover, for small rotations R, it can be approximated as:

$$R = \begin{pmatrix} 1 & -r_3 & r_2 \\ r_3 & 1 & -r_1 \\ -r_2 & r_1 & 1 \end{pmatrix} \tag{4.17}$$

For this parametrization, a polynomial solver which produces up to eight solutions is applied to solve three quadratic equations with three unknowns r_1, r_2 and r_3 after we have three matched lines. For each rotation candidate we can estimate the corresponding t using the trifocal tensor equations [36]. To minimize 4.16, one out of the eight possible solutions is selected. Once the rotation has been solved, two corresponding pairs are selected to establish line correspondences. These line matches are sufficient for independently constraining the translation elements using the trifocal tensor equations [45].

Chapter 5

Evaluation

5.1 Dataset and Parameter Settings

To realize Objective 3, we undertake a performance evaluation of ASL-SLAM in several scenes from two different datasets. As we focus on low latency event features, the event-line detectors mentioned in the last chapter is firstly compared with other event feature points. We also compare our full SLAM system with current popular frame-based SLAM methods, such as ORB-SLAM and PL-SLAM by employing their open-source implementations. We also compare the computation time of the detection and tracking for the asynchronous lines with other event feature algorithms. All experiments were conducted with an Intel Core i7-8700K ($12\times$@ 3.70GHz), Ubuntu 18.04, ROS melodic.

5.1.1 Dataset and Parameter Settings

In order to quantify the efficiency of the asynchronous line extraction we use the event camera datasets proposed by [60]. It was generated using a DAVIS240C and also provides the images, events, IMU estimations, and calibration with different sensors as well as ground truth obtained by a motion-capture system. We use the shape rosbag which has the rotation and translation of some typical shapes in order to show the detection results more clearly. In this part, we set $\lambda_1 = 0.001$, $\lambda_2 = 0.05$, $\Delta T = 0.01$s. The four approaches work directly on event streams under two different scenes.

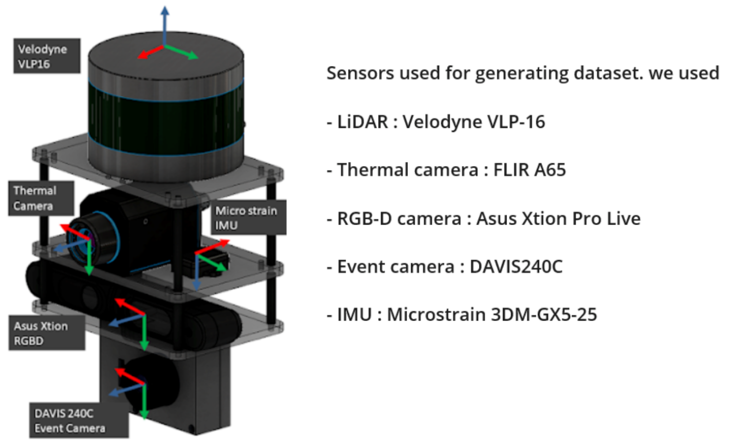

Velodyne VLP16
Thermal Camera
Micro strain IMU
Asus Xtion RGBD
DAVIS 240C Event Camera
Sensors used for generating dataset. we used
- LiDAR : Velodyne VLP-16
- Thermal camera : FLIR A65
- RGB-D camera : Asus Xtion Pro Live
- Event camera : DAVIS240C
- IMU : Microstrain 3DM-GX5-25

Figure 5.1: Platform structure of the VIVID system

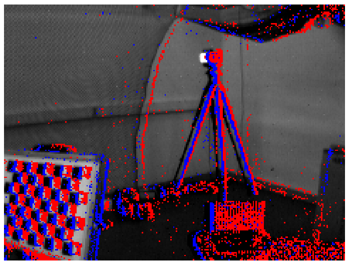
Our second experiment explores the robustness of asynchronous line extraction in varying illumination conditions. To achieve this we use the ViViD (Vision for Visibility Dataset) [49] that captured unconventional visual data obtained from varying lighting and motion conditions. The dataset provides normal and poor illumination sequences which are called local light and global light, captured by RGB-D camera, and event data using DAVIS240C as shown in Fig. 5.1. They also provide sequences under robust motion and unstable motion. These four combinations of data are ideal for our evaluation making it possible to test the robustness of the SLAM system based on line features. The resolution of the RGBD image is 640×480.

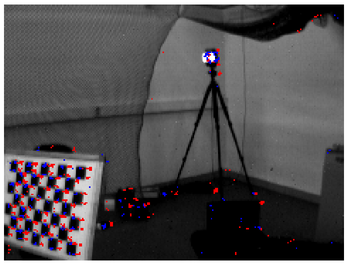
5.1.2 Asynchronous Feature Extraction Benchmark

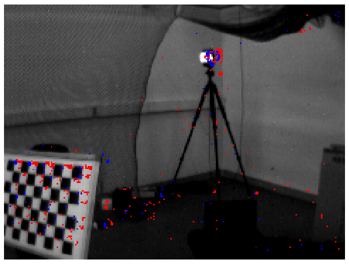
Some researchers have compared the performance of a small number of event corner detectors under some simple texture environments such as shapes, boxes and posts to test the detection quality[52]. As described above we choose a pair of datasets with more complicated indoor scenes including robust motion using local or global light. To make the code run easily on the dataset, we use the Robotic Operating System (ROS) platform that can easily work with several asynchronous processes together.

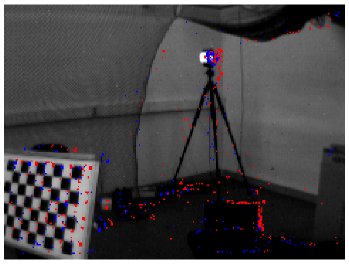
The corners detected under two light conditions by the frame-based and the event-based methods are shown respectively below. The event points are shown on the image from the event camera (which is 240×180 pixel greyscale) while the traditional features are shown on the normal camera. First we compare the existing event features mentioned in Section 3.2 with our ASL extractor. We synthesized events and detected features within $30ms$ on the intensity images

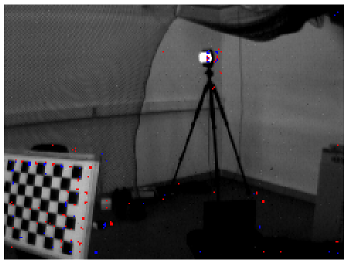
(a) Normal global image

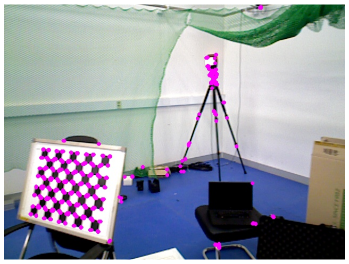

(b) Global events

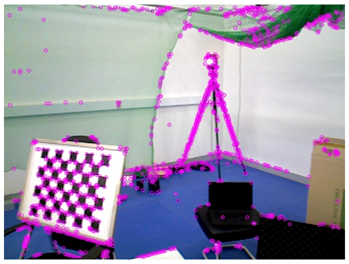

(c) eHarris global event corners


(d) eFast global event corners


(e) Arc* global event corners


(f) Fa-Harris global event corners


(g) Global Harris corners


(h) Global Fast corners

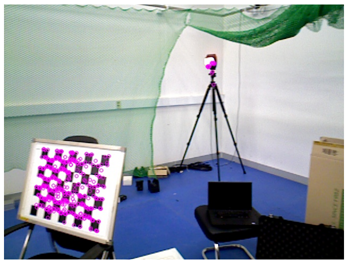
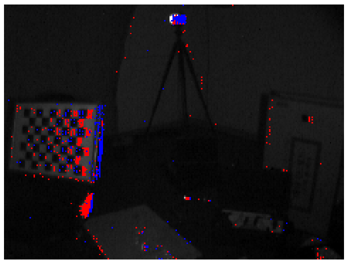

(i) Global ORB corners

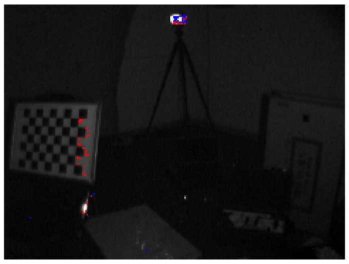
Figure 5.2: 5.2(a) shows the normal local image captured from RGB camera, 5.2(b)) shows the events that event cameras detected,5.2(c)- 5.2(g) depict the corner-events detected (blue and red dots) using eHarris, eFast, Arc* and Fa-Harris corner detector algorithms respectively. 5.2(g)- 5.2(i) depict the traditional corner detector of Harris, Fast and ORB respectively, all the detection processes are under the global light environment.

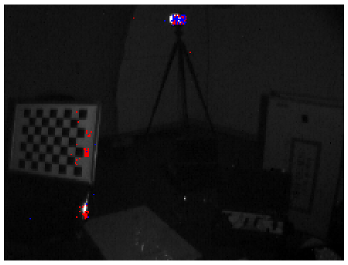
for visualization. The colours represent the different polarities of events and event corners, red and blue mean positive and negative polarity respectively. Fig.5.2 and Fig.5.3 are the corner detection results under the global light condition and local light condition respectively for the 4 feature detectors.

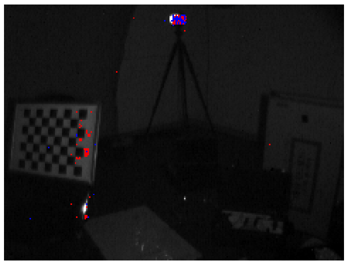
We can tell obviously that the figures in Fig.5.2 are brighter than the figures in Fig.5.3, which

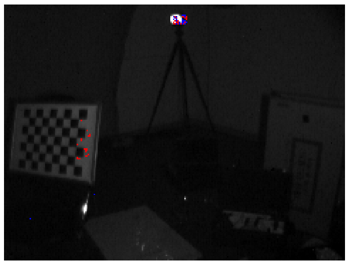
(a) Normal local image

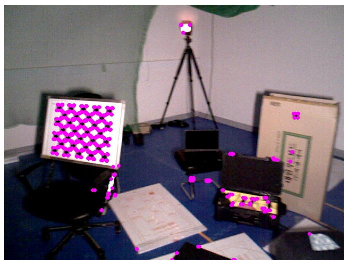

(b) Local events

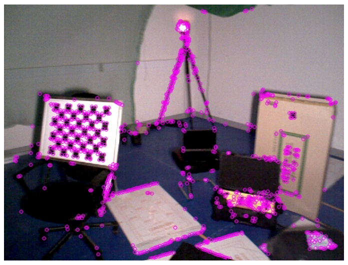

(c) eHarris local event corners


(d) eFast local event corners


(e) Arc* local event corners


(f) Fa-Harris local event corners


(g) Local Harris corners


(h) Local Fast corners

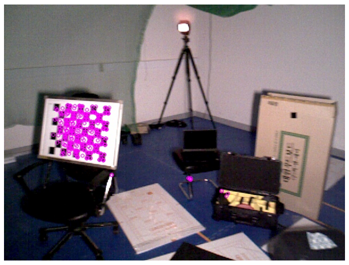

(i) Local ORB corners

Figure 5.3: 5.3(a) shows the normal local image captured from RGB camera, 5.3(b) shows the events that event cameras detected, 5.3(c) - 5.3(f) depict the corner events detected (blue and red dots) using eHarris, eFast, Arc* and Fa-Harris corner detector algorithms respectively. 5.3(g) - 5.3(i) depict the traditional corner detector of Harris, Fast and ORB respectively, all the detection processes are under the local light environment.

demonstrates that the event camera can still work well under low light environments. Fig.5.2(b) - 5.2(f) and Fig.5.3(b) - 5.3(f) indicate that all of the four methods have significantly reduced the number of events under both local and global illumination, compared to the original event stream. All of the four methods have detected the corners on the checkerboard and tripod, which means the intensity of these places has changed markedly. Moreover, of the three traditional corner detectors, the Fast detector has detected much more features primarily along the edges of the scene. The ORB detector has detected the least feature points which are located in the place where the brightness change is greater on the image.

Table 5.1.1: Performance of event feature detectors

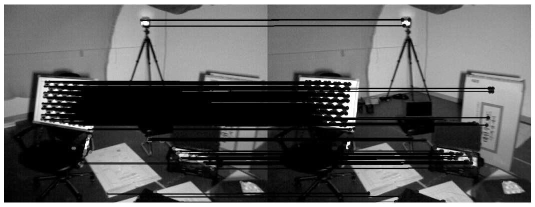
Scene	Algorithm	eHarris	Fa-harris	eFast	Arc*	ASL
Indoor Robust Local	Reduction rate (%)	85.66	93.30	78.16	77.90	93.14
	Time per event (us/ev)	5.92	0.90	1.88	0.19	2.32
	Max. event rate (Mev/s)	0.17	1.11	0.53	5.29	10.71
Indoor Robust Global	Reduction rate (%)	94.24	98.3	94.19	91.60	95.52
	Time per event (us/ev)	4.23	0.46	1.50	0.12	1.78
	Max. event rate (Mev/s)	0.24	2.19	2.00	8.05	13.36

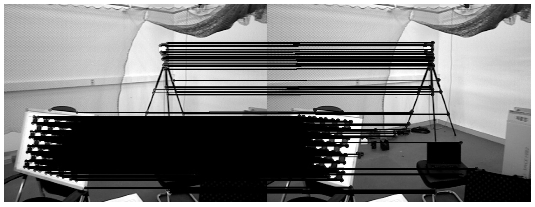
Table 5.1.1 illustrates the average processing time in µs of one single event and the maximal event bandwidth in Millions of events per second (Meps) . It also shows the Reduction rate, which explains the number of detected features over the number of input events. According to the results, our method shows reliable and consistent performance under the different types of motion, and it can deal with the highest incoming event rate among the 5 methods. It may be that the reason the max event rate stays highest with our method is that, the lines in the neighbourhood of the incoming events are being dealt with together in our method. In addition, the number of unique lines is lower than corners meaning that we will deal with less information than keypoint-based SLAM does. The time per event performance shows that it can detect line features (which contain at least 4 events) with a similar level of efficiency. The number of lines is less than feature points meaning that we will deal with less information than traditional keypoints-based SLAM.

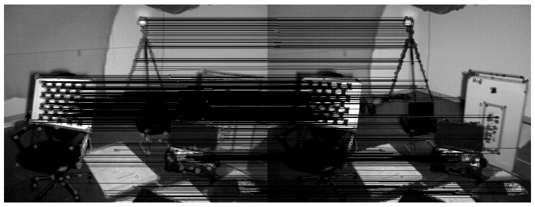
The above results have focused primarily on the efficiency characteristics of different feature detectors. Unfortunately it's hard to evaluate the reliability of the different feature detectors in isolation. Therefore we will use the following tracking and SLAM evaluations to determine the quality of the different detectors.

5.1.3 Feature Tracking Results

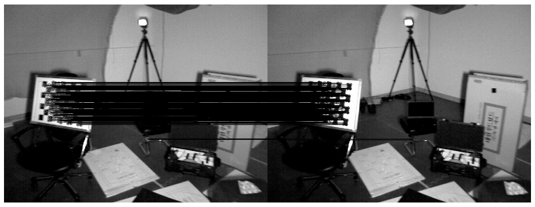
Unfortunately there are no absolute evaluation criteria to identify the quality of detected feature points, as there is no ground truth for what is and is not interesting. Instead we attempt to evaluate the quality of the detected features, in an application specific manner, by testing their suitability for tracking. To achieve this, traditional points and lines, event corners and our proposed event lines acquired from the previous experiment can be used to match and track camera motion.

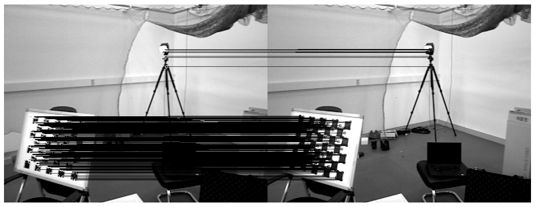

(a) Harris corner local tracking

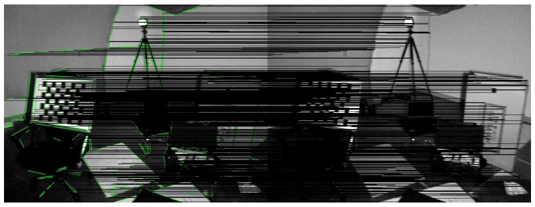

(b) Harris corner global tracking


(c) Fast corners local tracking

(d) Fast corners global tracking


(e) ORB corners local tracking


(f) ORB corners global tracking


(g) LSD lines local tracking

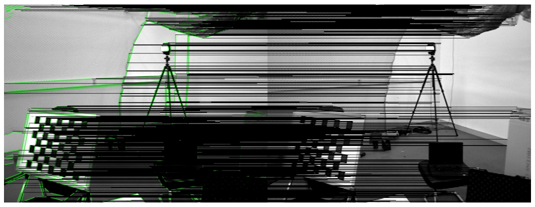

(h) LSD lines global tracking

Figure 5.4: Feature tracking using traditional frame-based feature

In this part, three traditional feature detectors, four event-based corner detectors as well as our asynchronous event line will be compared. We use LK optical flow [56] method to track frame-based feature points directly. For event features we followed a similar approach, projecting the features to the RGB camera and tracking according to optical flow. The features are then used to estimate the epipolar geometry between the views via RANSAC. Because the events occur continuously in time, we use all event corners that happen in $30ms$ between two frames and compare the tracking result. For PL-SLAM features we track the extremities of line-features and calculate the line distance to get good track.

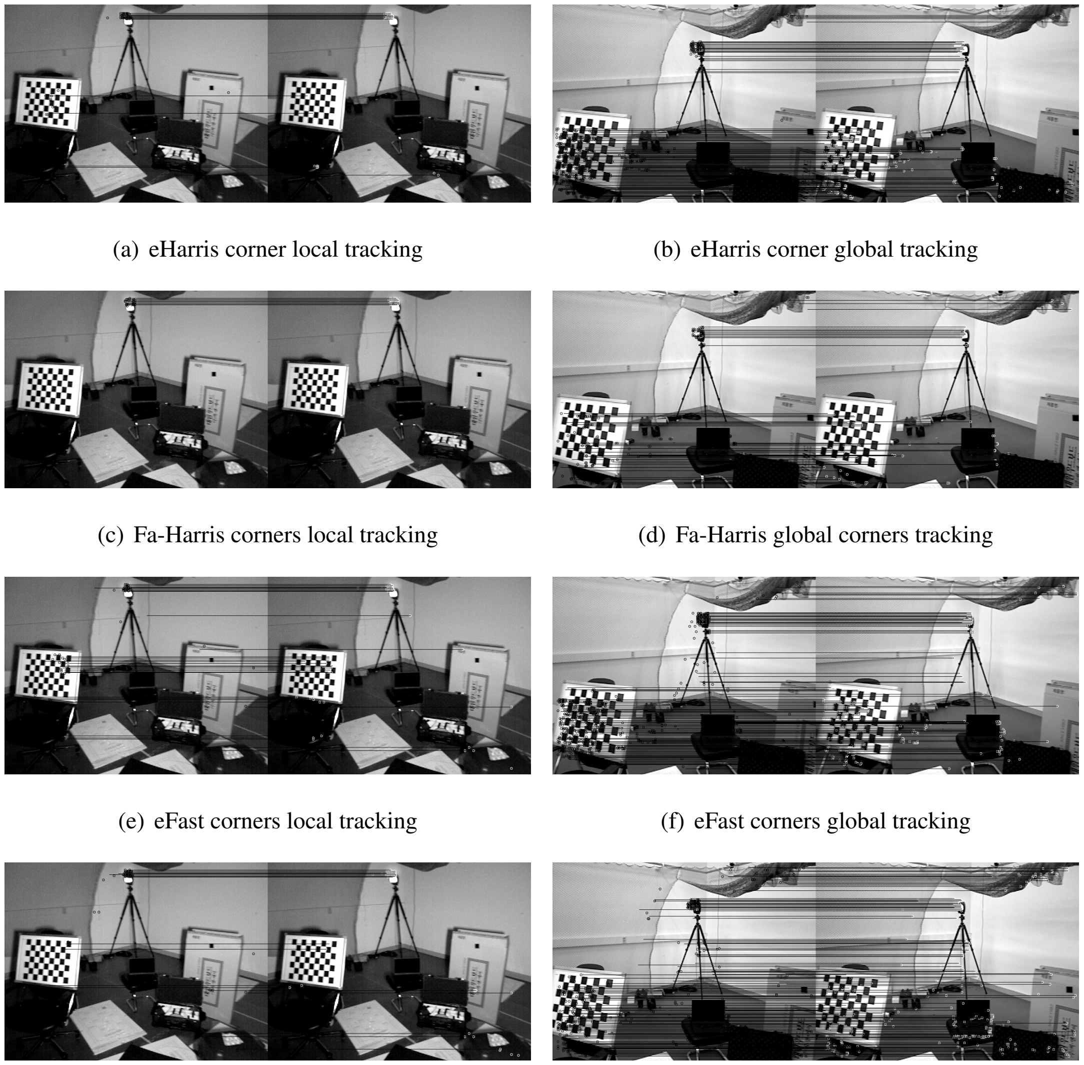

(a) eHarris corner local tracking
(b) eHarris corner global tracking
(c) Fa-Harris corners local tracking
(d) Fa-Harris global corners tracking
(e) eFast corners local tracking
(f) eFast corners global tracking
(g) Arc* corners local tracking
(h) Arc* corners global tracking

Figure 5.5: Feature tracking using event-based feature

Fig. 5.4 and Fig. 5.5 show the performance of the traditional feature and event feature tracking under two different light condition referred to the previous experiment. Because of the real-time detecting and tracking environment, both images are changed into grayscale pictures to process the tracking tasks. Here we still show the traditional points on colour images and event features on grayscale images in order to distinguishing them. Two continuous frames are used, features(circles) are detected in the previous frame (left image) and corresponding points (blue circle) are tracked on the next frame (right image). We can see that the successfully tracked inlier points are largely correctly matched. We also note that some circles have not been linked which

Table 5.1.2: Feature tracking results based on frames and events

Feature Types		Local Light Detection	Local Light Tracking	Global light Detection	Global light Tracking
Traditional Features	Harris	3153	84.64% Successfully tracked	3298	85.29% Successfully tracked
	Fast	628	85.82% Successfully tracked	687	85.58% Successfully tracked
	ORB	463	85.10% Successfully tracked	498	87.35% Successfully tracked
	PL	372	89.92% Successfully tracked	400	94.50% Successfully tracked
Event Feature	eHarris	158	81.01% Successfully tracked	511	89.62% Successfully tracked
	Fa-Harris	62	88.71% Successfully tracked	92	85.87% Successfully tracked
	eFast	242	88.43% Successfully tracked	528	93.18% Successfully tracked
	Arc*	257	74.31% Successfully tracked	565	98.58% Successfully tracked
	ASL	78	97.43% Successfully tracked	136	100% Successfully tracked

means the RANSAC has effectively removed the wrong or badly tracked points. The Harris corner detector has detected much more corners than other traditional feature detectors because there are many close features that have been kept. Moreover, we can see that the Fast method has detected the most features while Harris has detected some incorrect points in untextured regions (although some of these are nevertheless tracked successfully).

Fig. 5.5 shows the results of event corner tracks. It is worth noting that the detected event features do not lie in areas that we would normally expect to contain feature points. This is because the events are aggregated over a period of time from a moving camera. Projecting these points to either the first or last image frame is a simplification. The detection and tracking performance is shown in Table 5.1.2. The Fa-Harris method has detected and tracked the least number of points. We can see that the tracking rate of line features is higher than point features. Additionally, the number of line features is less than points, which can save on computational cost. It is interesting to note that eHarris and Arc* are are some of the best performing under global lighting with near 100% tracking inliers. However, under local light tracking reduces drastically compared to the other features. This suggests that the interest points targeted by these detectors are susceptible to lack of contrast. Except that other event tracking results show higher tracking rate than traditional features. The results indicate that although it is possible to do traditional image-based tracking, for event-based feature detectors, this may not be optimal. In particular because there is not a good correspondence between the high frequency events, and the images, the points tracked do not exactly match those detected.

5.1.4 SLAM performance

To evaluate the robustness of the event-based SLAM algorithm, we still use the VIVID dataset which can provide a comprehensive evaluation of our method against other approaches. The image frames from the four rosbags of the VIVID dataset are shown in Fig. 5.6, we can see that the two global light rosbag have brighter illumination than the local light ones. At the same time, the frames from the dataset of unstable motion are more blurred than in robust motion which means tracking may be lost when using the traditional keypoints tracking method. Since there is no open source pipeline for event-based VO/SLAM, we choose to compare against the highly effective ORB-SLAM2 (which is equivalent to our system without the asynchronous lines) and a state-of-art line-based method PL-SLAM. We use the monocular demos to compare fairly with our method. It is known that traditional feature-based SLAM would suffer significant

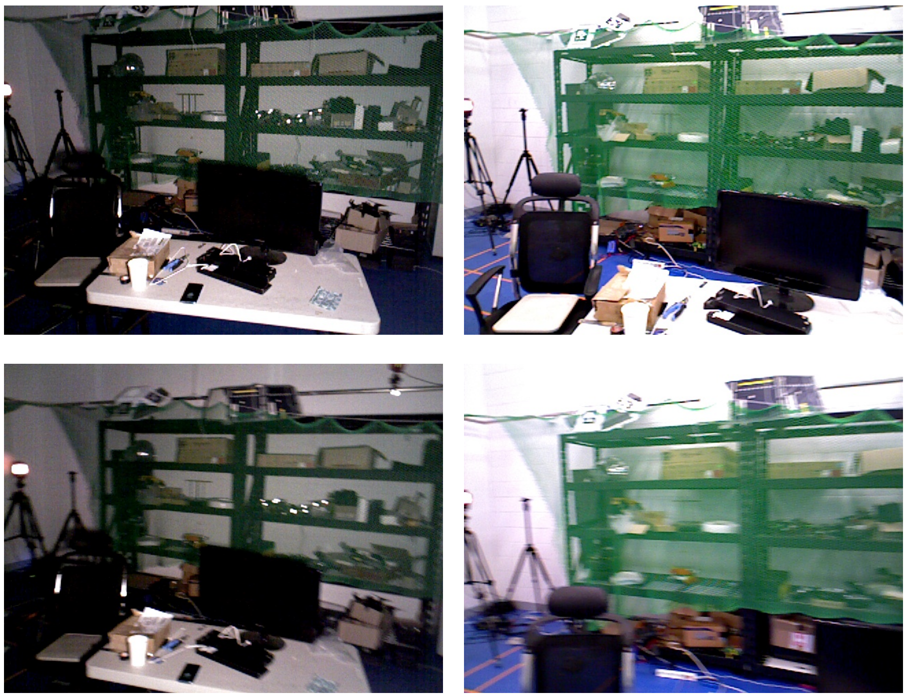


Figure 5.6: Frames of four different experimental conditions.The top left is local light and robust motion, the bottom left is global light and robust motion, and top right is global light and unstable motion, bottom right is local light and unstable motion.

degradation from motion blur or low light conditions, which will result in tracking failure. We evaluated our ASL system with the other two methods on the four sequences. The results are shown in Table 5.1.3 and 5.1.4.

First we calculate the running time of each method under four scenarios, where tracking performance is shown below in Table 5.1.3. It is obvious that our ASL method significantly outperforms the frame-based baseline ORB-SLAM2 and line-based PL-SLAM. The tracking time has decreased by more than 20% compared to the other two methods. It can be inferred that there are more features for ORB-SLAM2 to deal with. We can see that under the global light condition, all three methods spent more time tracking which means there are more features when it is brighter in the room. This also shows that our methods will not be affected by the light change of the environments

Then we used the metrics for the Absolute Trajectory Error(ATE) which is provided by the evaluation script of the benchmark. Before computing the error, all trajectories are aligned

Table 5.1.3: Comparison of tracking time for each method

Scene	Tracking Time (ms)	ORB-SLAM2	PL-SLAM	ASL-SLAM
Local Robust	Median	0.0616	0.0629	**0.0456**
	Mean	0.0627	0.0642	**0.0496**
Global Robust	Median	0.0662	0.0706	**0.0481**
	Mean	0.0661	0.0702	**0.0521**
Local Unstable	Median	0.0553	0.0488	**0.0381**
	Mean	0.0568	0.0503	**0.0390**
Global Unstable	Median	0.0501	0.0564	**0.0437**
	Mean	0.0499	0.0572	**0.0445**

using a similarity warp and scaled by calculating the Euclidean distance between the estimated trajectory and the ground truth. We evaluated the ATE by calculating its RMS which is illustrated in Table. 5.1.4.

Table 5.1.4: Absolute Trajectory Error(RMS) [t:mm]

	ORB-SLAM2	PL-SLAM	ASL-SLAM
Local Robust	53.54	54.73	**45.38**
Global Robust	**24.11**	34.69	27.96
Local Unstable	526.53	498.72	**321.3**
Global Unstable	324.50	306.97	**46.80**

We can see from the Table. 5.1.4, our method outperforms all others, except for global robust conditions. In particular our method shows robust performance under challenging light and motion conditions. It can be inferred that the efficiency of our method results in less error than PL-SLAM does. Though the ORB-SLAM has the best performance under the global robust condition under any challenging scenario, it has an error almost twice that of the ASL-SLAM.

Fig. 5.7 displays the absolute trajectory error of each method. For the two figures on the left, we can see that the distance error of our method has a lower peak and changes smoothly. It can be inferred from the right bottom figure that, there are violent motion changes around 5s and 15s, our ASL methods show lower error compared to the other two methods, especially for ORB-SLAM2, which has a significant increase in error at that point. However, events can be

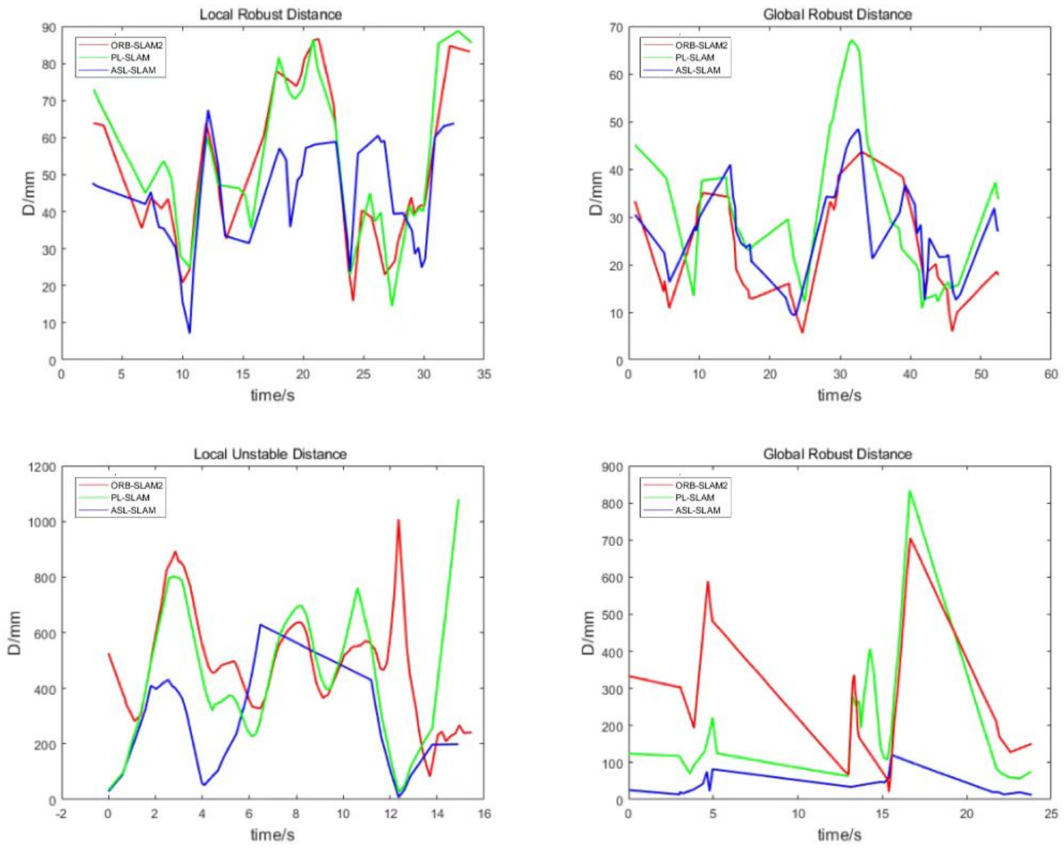


Figure 5.7: Absolute Trajectory Error between Estimation and Ground Truth.

obtained as long as the brightness has changed, this also means the event lines can be detected no matter how the camera moves. In this case, our ASL-SLAM can provide a more stable and robust localization result.

5.1.5 Result Summary

Our results show that the proposed ASL-SLAM architecture is able to operate using low-latency line features with high accuracy and data efficiency. Our method consistently outperforms the traditional feature point-based approaches when subject to significant motion-blur or low-textured scenes with a low feature count. We also demonstrate our approach has a good performance in low-light situations, where the traditional approaches are prone to failure. Additionally, the pipeline is evaluated with the VIVID dataset and showed consistent improvement compared to the current competing methods.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In the work of this thesis, we proposed ASL-SLAM, an event-based visual SLAM approach that covers both feature points and event lines simultaneously. Pure event-based SLAM has some limitations, so we presented an alternative way to take the advantage of both an event camera and traditional camera. Their different properties (high dynamic range/ lack of motion blur and distinctive visual descriptors respectively) lead to a strong combined system.

In the first chapter we introduced the general problem of positioning and localization. They are mainly divided into two areas: relative localization and global localization. Many sensors are commonly used during the positioning process such as GPS, cameras, lidars and IMUs. However in this thesis we focused on the application of traditional cameras and event cameras.

For the second chapter, we generally introduced some recent SLAM/VO technologies and their application to event cameras. Classic SLAM technologies can be divided into feature-based methods and direct methods. The former is dependent significantly on the feature detector and matcher. The latter relies on data association and can save time on feature detection and matching, but the effect of motion blur is exacerbated. There are still open questions around the best way to use event cameras for SLAM, and there is currently no definition for event-based line tracking.

In the third chapter we discussed and analysed the characteristics and differences between traditional feature detection methods and the original event feature detection methods in detail. This laid the groundwork to formalize our own novel contribution and also formed the baselines for evaluation in later chapters.

In Chapter 4 we introduced the structure of ASL-SLAM. Firstly we proposed our event-line detection method based on the SAE. We then synchronised our continuous event lines with the traditional feature points to recover the 3D pose. Next we use BA to optimize our camera pose by combining the frame-based feature points with our event lines. Finally after the correspondences of points and lines are established, the camera rotation matrix can be recovered and the map updated. For this chapter we tested our technique using a simple dataset of different shapes.

For the final chapter we undertook an evaluation of the outcomes from the other chapters. First we compared our line features with other frame-based features and event-features. We then run a complete SLAM system under different environments with local and global light conditions. The results show that our ASL-SLAM architecture is able to operate with low latency line features at high accuracy and data efficiency that can be suitable for industrial environments. It consistently outperforms the traditional feature point-based approaches in the presence of motion blur or low-textured scenes. We also demonstrate our approach performs well in low-light situations, where traditional approaches are prone to failure. On the VIVID dataset, this led to consistent improvement of up to 85% reduction in error compared to the current competing methods. For future work, further exploration of asynchronous line-based SLAM and incorporation of higher dimensional geometric primitives like planes could be fruitful.

6.2 Future Work

In this thesis we used both event information and traditional feature points to achieve SLAM. Further exploration of the asynchronous line feature based SLAM such as pure event-based SLAM with event line tracking may be valuable. It is possible to take advantages of event flow to track lines directly in the event volume. Since the event camera can directly respond to the edges of the scene, it would be easier to detect certain primitives such as corner and lines.

In addition the incorporation of higher dimensional geometric primitives like planes could be applied to our SLAM algorithm. It can be inferred that, just as a line in the image becomes a plane in the 3D event volume; a 2D region in the image is a volume in the event cloud. Therefore if a specific volume and the image frame are matched among the event points, the camera's pose can be obtained. We would expect that similarly to the findings of this thesis, these ”event region primitives” would be significantly sparser than lines, but correspondingly more robust and informative.

6.3 Directions for the Field

Event camera as a low latency high dynamic visual sensor has wide potential in fields such as computer vision, robotics, aerospace technology etc. The fundamental problem of pure event-based SLAM is how to track the events. The lack of any descriptors for data association makes this a very challenging problem. One potential avenue may be [82] which takes the relationship of events, lines and ego-motion of the camera to formulate a continuous constraint.

Another way to get a more accurate result is to fuse other sensors such as IMU. It is common to use this simple and low-cost sensor in SLAM research.[47] proposed a IMU-DVS SLAM system by optimizing the camera pose. The inertial representation was associated with each event to minimize the distance between points and lines. In future research it may be possible to use IMU measurements to estimate motion priors for higher level primitives. This may assist with the event based line/region tracking problem. If these questions can be addressed in future research, event based SLAM has incredible potential to revolutionise one of the most fundamental aspects of robotics.

Bibliography

[1] Mohamed Abouzahir, Abdelhafid Elouardi, Rachid Latif, Samir Bouaziz, and Abdelouahed Tajer. Embedding slam algorithms: Has it come of age? *Robotics and Autonomous Systems*, 100:14–26, 2018.

[2] I. Alzugaray and M. Chli. Ace: An efficient asynchronous corner tracker for event cameras. In *2018 International Conference on 3D Vision (3DV)*, pages 653–661, 2018.

[3] I. Alzugaray and M. Chli. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robotics and Automation Letters*, 3(4):3177–3184, 2018.

[4] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1–26, 2016.

[5] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 884–892, 2016.

[6] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer vision and image understanding*, 100(3):416–441, 2005.

[7] Alexis Baudron, Winston Wang, Oliver Cossairt, and Aggelos Katsaggelos. E3d: Event-based 3d shape reconstruction. *arXiv preprint arXiv:2012.05214*, 2020.

[8] H. Bay. Surf: Speeded up robust features. *Computer Vision & Image Understanding*, 110(3):404–417, 2006.

[9] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–417, 2013.

[10] R. Berner, C. Brandli, Minhao Yang, Shih Chii Liu, and T. Delbruck. A 240×180 10mw 12us latency sparse-output vision sensor for mobile applications. In *Vlsi Circuits*, 2013.

[11] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 491–501, 2019.

[12] C Brandli, R Berner, M Yang, S.-C Liu, and T Delbruck. A 240x180 130db 3us latency global shutter spatiotemporal vision sensor. 49(10):2333–2341, 2014.

[13] Shuhui Bu, Yong Zhao, Gang Wan, and Zhenbao Liu. Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4564–4571. IEEE, 2016.

[14] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[15] Manmohan Chandraker and Shiyu Song. Real-time monocular visual odometry. 2015.

[16] Xavier Clady, Sio Hoi Ieng, and Ryad Benosman. *Asynchronous event-based corner detection and matching*. Elsevier Science Ltd., 2015.

[17] Laurent Dardelet, Sio-Hoi Ieng, and Ryad Benosman. Event-based features selection and tracking from intertwined estimation of velocity and generative contours. *arXiv preprint arXiv:1811.07839*, 2018.

[18] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[19] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.

[20] Yan Dong and Tao Zhang. Standard and event cameras fusion for feature tracking. In *2021 International Conference on Machine Vision and Applications*, pages 55–60, 2021.

[21] David Drazen, Patrick Lichtsteiner, Philipp Häfliger, Tobi Delbrück, and Atle Jensen. Toward real-time particle tracking using an event-based dynamic vision sensor. *Experiments in Fluids*, 51(5):1465, 2011.

[22] Hugh F Durrantwhyte and Tim Bailey. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics & Amp Amp Automation Magazine*, 13(2):99 – 110.

[23] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[24] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[25] Martin A. Fischler. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Readings in Computer Vision*, pages 726–740, 1987.

[26] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[27] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics & Automation*, 2014.

[28] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based Vision: A Survey. pages 1–25, 2019.

[29] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2018.

[30] Maria C Garcia-Alegre, David Martin, D Miguel Guinea, and Domingo Guinea. Real-time fusion of visual images and laser data images for safe navigation in outdoor environments. *Sensor Fusion-Foundation and Applications, Published*, 2011.

[31] Riccardo Giubilato, Sebastiano Chiodini, Marco Pertile, and Stefano Debei. An evaluation of ros-compatible stereo visual slam methods on a nvidia jetson tx2. *Measurement*, 140:161–170, 2019.

[32] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3828–3838, 2019.

[33] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuniga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez. Pl-slam: A stereo slam system through the combination of points and line segments. *IEEE Transactions on Robotics*, 35(3):734–746, 2019.

[34] C. G. Harris and M. J. Stephens. A combined corner and edge detector. 1988.

[35] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[36] Richard I Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[37] M Hassaballah, Aly Amin Abdelmgeid, and Hammam A Alshazly. Image features detection, description and matching. In *Image Feature Detectors and Descriptors*, pages 11–45. Springer, 2016.

[38] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[39] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics research*, pages 235–252. Springer, 2017.

[40] Şahin Işık. A comparative evaluation of well-known feature detectors and descriptors. *International Journal of Applied Mathematics Electronics and Computers*, 3(1):1 – 6, 2015.

[41] Juan Jose Tarrio and Sol Pedre. Realtime edge-based visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 702–710, 2015.

[42] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *2010 ieee intelligent vehicles symposium*, pages 486–492. IEEE, 2010.

[43] Lindsay Kleeman. Advanced sonar and odometry error modeling for simultaneous localisation and map building. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 699–704. IEEE, 2003.

[44] Beat Kueng, Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 16–23. IEEE, 2016.

[45] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1381–1393, 2011.

[46] Xavier Lagorce, Cedric Meyer, Sio Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Transactions on Neural Networks & Learning Systems*, 26(8):1710–1720, 2015.

[47] Cedric Le Gentil, Florian Tschopp, Ignacio Alzugaray, Teresa Vidal-Calleja, Roland Siegwart, and Juan Nieto. Idol: A framework for imu-dvs odometry using lines. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5863–5870, 2020.

[48] Alex Junho Lee, Younggun Cho, Youngsik Shin, Ayoung Kim, and Myung Hyun. Vivid++: Vision for visibility dataset. *IEEE Robotics and Automation Letters*, 2022.

[49] Alex Junho Lee, Younggun Cho, Sungho Yoon, Youngsik Shin, and Ayoung Kim. ViViD : Vision for Visibility Dataset. In *ICRA Workshop on Dataset Generation and Benchmarking of SLAM Algorithms for Robotics and VR/AR*, Montreal, May. 2019. Best paper award.

[50] 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, 2015.

[51] Kaiyue Li, Dianxi Shi, Yongjun Zhang, Ruoxiang Li, Wei Qin, and Ruihao Li. Feature tracking based on line segments with the dynamic and active-pixel vision sensor (davis). *IEEE Access*, 7:110874–110883, 2019.

[52] Ruoxiang Li, Dianxi Shi, Yongjun Zhang, Kaiyue Li, and Ruihao Li. Fa-harris: A fast and asynchronous corner detector for event cameras. pages 6223–6229, 09 2019.

[53] Shi-Jie Li, Bo Ren, Yun Liu, Ming-Ming Cheng, Duncan Frost, and Victor Adrian Prisacariu. Direct line guidance odometry. In *2018 IEEE international conference on Robotics and automation (ICRA)*, pages 1–7. IEEE, 2018.

[54] 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*, 43(2):566–576, 2008.

[55] Shih-Chii Liu, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas. *Event-based neuromorphic systems*. John Wiley & Sons, 2014.

[56] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[57] Soumyadip Maity, Arindam Saha, and Brojeshwar Bhowmick. Edge slam: Edge points based monocular visual slam. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2408–2417, 2017.

[58] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.

[59] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast Event-based Corner Detection. 1:1–11, 2019.

[60] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.

[61] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[62] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[63] Jalees Nehvi, Vladislav Golyanik, Franziska Mueller, Hans-Peter Seidel, Mohamed Elgharib, and Christian Theobalt. Differentiable event stream simulator for non-rigid 3d tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1302–1311, 2021.

[64] D. Nister, O. Naroditsky, and J. R. Bergen. Visual odometry. In *IEEE Computer Society Conference on Computer Vision & Pattern Recognition*, 2004.

[65] David Nistér, Oleg Naroditsky, and James R. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2010.

[66] E. Piatkowska, A. N. Belbachir, S. Schraml, and M. Gelautz. Spatiotemporal multiple persons tracking using dynamic vision sensor. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 35–40, 2012.

[67] C Posch, D. M 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*, 46(1):p.259–275, 2011.

[68] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508, 2017.

[69] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In *Conference on Robot Learning*, pages 969–982. PMLR, 2018.

[70] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide 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.

[71] David Reverter Valeiras, Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Event-based line fitting and segment detection using a neuromorphic visual sensor. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1218–1230, 2019.

[72] E. ROSTEN. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, 2006.

[73] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, 2012.

[74] Camilo Sánchez-Ferreira, Jones Y Mori, Mylène CQ Farias, and Carlos H Llanos. A real-time stereo vision system for distance measurement and underwater image restoration. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 38(7):2039–2049, 2016.

[75] C. Scheerlinck, N. Barnes, and R. Mahony. Asynchronous spatial image convolutions for event cameras. *IEEE Robotics and Automation Letters*, 4(2):816–822, 2019.

[76] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. *Machine Intelligence & Pattern Recognition*, 5(5):435–461, 1988.

[77] Lea Steffen, Daniel Reichard, Jakob Weinland, Jacques Kaiser, Arne Roennau, and Rüdiger Dillmann. Neuromorphic stereo vision: A survey of bio-inspired sensors and algorithms. *Frontiers in neurorobotics*, 13:28, 2019.

[78] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[79] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.

[80] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based harris corner detection exploiting the advantages of event-driven cameras. In *IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2016.

[81] Lijun Wei, Cindy Cappelle, Yassine Ruichek, and Frédérick Zann. Gps and stereovision-based visual odometry: Application to urban scene mapping and intelligent vehicle localization. *International Journal of Vehicular Technology*, 2011, 2011.

[82] Peng Xin, Xu Wanting, Yang Jiaqi, and Kneip Laurent. Continuous event-line constraint for closed-form velocity initialization. *arXiv preprint arXiv:2109.04313*, 2021.

[83] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 34(9):1744–1757, 2012.

[84] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015.

[85] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[86] Yi Zhou, Guillermo Gallego, and Shaojie Shen. Event-based stereo visual odometry. *arXiv preprint arXiv:2007.15548*, 2020.

[87] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[88] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*, 2018.

[89] YANG Sheng. ZHU Chaozheng, HE Ming. Survey of monocular visual odometry. *Computer Engineering and Applications*, 54(007):20–28,55, 2018.

[90] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5391–5399, 2017.