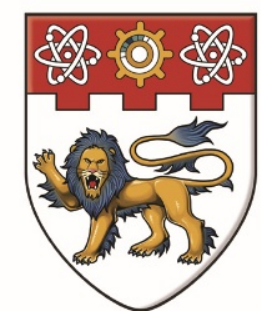


NANYANG TECHNOLOGICAL UNIVERSITY

ANALYSIS OF OBJECT AND ITS MOTION IN EVENT-BASED VIDEOS

SAJJAD SEIFOZZAKERINI

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2018

ANALYSIS OF OBJECT AND ITS MOTION IN EVENT-BASED VIDEOS

SAJJAD SEIFOZZAKERINI

School of Electrical and Electronic Engineering

A thesis submitted to Nanyang Technological University

in partial fulfillment of the requirements for the degree of

PhD of Engineering

2018

To my beloved sister who lived in ***peace***

and passed away in bigger ***peace***

and to my devoted parents who educated me

the ***peace*** *with this loss*

Acknowledgements

This thesis would not be possible if I did not have the help and advice of several people who contributed their valuable support in completion of this study.

First of all, I would like to express my sincere appreciation and gratitude to my supervisor, Associate Professor Mao Kezhi for giving me an opportunity to pursue my interest in research work under his valuable guidance. He is not only a supervisor but also a warm hearted friend that I always can count on his support and help.

It gives me immense pleasure to thank my mentor and co-supervisor, Doctor Yau Wei-Yun for his support, continuous guidance and astute criticism during this study. Nothing can gratify the invaluable time he generously dedicated to me despite his busy schedule.

My special thanks go to my Thesis Advisory Committee members, Professor Huang Guangbin, Associate Professor Wang Han and Dr. Li Xiaoli for their invaluable guidance and encouragement throughout my research.

Financial assistance provided by the Agency for Science, Technology and Research (A*STAR) in the form of Graduate Scholarship is thankfully acknowledgeable.

I would like to express my gratitude to the staffs in Institute for Infocomm Research (I2R), Ms. Janice Lee Sze Min and Ms. Felicia Chew for their effort on my attachment matters.

I cannot miss mentioning my home mates during last 4 years in Singapore, Amin Kianinejhad, Siavash Sakhavi, Hossein Dehghani Tafti and Ali Rajabipoor for making

my life during Ph.D. a very memorable one. Also, I should mention my Iranian friends in Singapore, whom their moral supports were a great help during this candidature: Ahmad Reza Shehabinia, Sajjad Maghareh, Mohammad Akbari, Said Arabnejhad, Said Montazeri, Hossein Nejati, Alireza Barzegar and all whom I might have forgotten to mention their names here.

Last but not least, I would like to thank my loving family, dear father and mother, nephew and brother in law as well as my lovely sister like whom there will never be another. Their prayer for me was what sustained me thus far.

Abstract

In the recent years, a new generation of cameras sensitive to pixel intensity variation rather than the traditional pixel intensity value has been introduced. These cameras called Dynamic Vision Sensors (DVSs) have recently attracted significant research interest. A conventional camera captures the intensity of all pixels in the sensor and generates an entire image to produce a frame. This is then repeated at a fixed rate to produce a video stream. Being totally different with the old frame-based videos, the captured videos of Dynamic Vision Sensors are polarized events, i.e. points in 3-D spatiotemporal space, indicating the polarity, location and time properties of the pixels with variable intensities. When there is a variation in a pixel intensity, a polarized event is created in the form of a vector with three elements (t, x, y). t shows the instance of the variation and (x, y) defines the position of the pixel. Moreover, the polarization of the event shows the direction of the change in the pixel intensity.

For event-based videos, some algorithms have been proposed for object tracking, optical flow extraction, human action recognition, etc. Still, there are many potential capabilities of these cameras that have not been used or explored. Extracting features of these videos requires a comprehensive understanding and novel procedures accordingly.

We began our research by focusing on the existing algorithms for DVS videos. We found out that motion analysis and feature extraction are trending topics in DVS videos and we began to work on these topics. This thesis introduces two different approaches to objects' motion in event-based videos. Hough transform and edge detection are also performed in event-based videos as two important methods of feature extraction.

This research work presents a novel framework for investigating the object's motion in event-based videos and extracting the edge information subsequently. In the event-based videos, the events normally occur in the moving edge areas. We consider the events as some points in the spatiotemporal space. Ignoring the noise, for each small spatiotemporal window in a moving edge area, we expect all events to be on a 3-D plane. The orientation of this plane depends on both edge direction and velocity. By approximating the object boundary as a series of linear elements, we derive a procedure based on principal component analysis to estimate their orientation and speed. According to the well-known aperture problem in machine vision, the velocity estimated at this stage is the normal portion of the actual velocity since any displacement along the edge orientation can not be recognized in a small spatiotemporal window.

The normal velocities are utilized in a larger window which covers a whole object to estimate its actual velocity. We define a cost function based on the difference between actual normal velocities and calculated normal velocities at the previous stage. Minimization of this cost function results in an estimation of the actual velocity which is a useful parameter in some applications e.g. object tracking. Moreover, we propose a procedure for localizing the edge based on regional exposure time and edge-dependent Gaussian filtering of the events. The regional exposure time is adjusted based on the normal velocities of edge pixels. This avoids any blurred edge which is a direct consequence of higher normal velocities. The orientation of Gaussian filter causes the maximum blurring effect along the edge direction and improve its connectivity for a better edge extraction.

Any kind of discontinuity in the object texture is appeared by the local variation of the pixel intensities. When the object is moving, these variations generate many unwanted events that we should tackle them as noise. Noise is another challenge of these videos that we can suppress by detecting the outliers in many stages of our algorithm.

Another approach to motion analysis is based on well-known Hough transform for

detecting straight lines. Hough transform has been widely used to detect lines in images captured by conventional cameras. We develop an event-based Hough transform and apply it to DVS output stream. The proposed algorithm is implemented in a spiking neural network to detect lines on DVS output. Spikes (events) from DVS are first mapped to Hough transform parameter space and then sent to corresponding spiking neurons for accumulation. A spiking neuron will fire an output spike once it accumulates enough input contributions and then reset itself. The output spikes of the spiking neural network represent the parameters of detected lines. An event-based clustering algorithm is applied on the parameter space spikes to segment multiple lines and track them. In our spiking neural network, a lateral inhibition strategy is applied to suppress noise lines from being detected. This is achieved by resetting a neuron's neighbors in addition to itself once the neuron fires an output spike. As an improvement to the work done, we deal with detecting small lines at the frame corners subsequently. In addition, the inhibitory window shape is optimized to suppress the lines which are close together in Cartesian space and are not necessarily close together in parameter space as assumed initially.

Finally, we perform many experiments for verification of our proposed algorithms. Some of them are performed on computer-generated videos while others performed on real DVS videos. The results show that our proposed methods have acceptable performances in recognizing the edges and estimating their velocities.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Notations

A. List of Acronyms

AAE	Average Angular Error
AEE	Average Endpoint Error
ANN	Artificial Neural Network
BCA	Brightness Consistency Assumption
BD	Backward finite Difference
CD	Central finite Difference
DAVIS	Dynamic and Active-pixel Vision Sensor
DS	Direction Selective
DVS	Dynamic Vision Sensor
FPGA	Field Programmable Gate Array
GT	Ground Truth
HT	Hough Transform
IF	Integrate and Fire
IMU	Inertial Measurement Unit
LIF	Leaky Integrate and Fire
LK	Lucas-Kanade
LMS	Least Mean Square
LP	Local Plane
PC	Personal Computer
PCA	Principal Component Analysis
RP	Refractory Period
SD	Standard Deviation

SF	Single Fit
SG	Savitzky-Golay
SN	Spiking Neuron
SNN	Spiking Neural Network
SVM	Support Vector Machine
VLSI	Very Large Scale Implementation

Chapter 1

Introduction

This chapter presents a background of Dynamic Vision Sensors, applications and capabilities of these imaging devices in machine vision. Then the objectives and main contributions of this research are highlighted. Thesis organization is the last section of this chapter.

1.1 Background and Motivations

Dynamic Vision Sensor (DVS) is a relatively new event-based video camera. Comparing to a conventional frame-based camera, DVS offers great advantages in terms of data rate, speed, and dynamic range [1]. Simulating biological retinas, DVS sensors are sensitive to the intensity change rather than the absolute intensity value. Conventional cameras synchronously capture frames (e.g. 30 frames per second) with each frame containing the intensity values of all the pixels. Pixels with little change are repeatedly reported in different frames, generating a lot of redundant information and seriously limiting the frame rate and temporal accuracy of the camera. In contrast, DVS sensors only report pixels with intensity changes and output them autonomously as an asynchronous stream of binary events. All the pixel circuits of the DVS work in parallel, and whenever there is an enough change in a pixel's intensity, the pixel

autonomously reports its position and the polarity of the intensity change as an event. Since redundant background information is discarded at the focal plane, we will have much less data volume as well as a very high temporal resolution. Moreover, due to logarithmic intensity change detection, DVS sensors offer a very high dynamic range, meaning it has no problem in capturing the scenes containing both very dark and very bright areas. These sensors have been used in many applications such as motion estimation [2], tracking [3–5], object recognition [6–8], and corner points detection [9].

1.2 Research Objectives

In this study, we primarily focus on the low-level image processing operations on event-based videos, especially for motion analysis and edge extraction of objects on DVS output streams. Note that low-level operations in this section refers to purely algorithmic operations (e.g. edge and line detection in image domain) and we do not concern any specific hardware design or running on a particular hardware implementation. The results of this study can be used in a vast variety of applications including, but not limited to, object recognition, object tracking, human action recognition, posture recognition, humanoid robots, surveillance systems, traffic control, etc. We begin this research by extracting motion flow (optical flow) vectors of objects in DVS videos. Using these vectors, we calculate the global velocity of objects and extract the edge information. Obviously, the algorithms proposed for these purposes need some mathematical processing to be done. Another strategy is to use a minimal processing power and exploit hardware capabilities instead to do a certain task. DVS output is well prepared to be injected into a Spiking Neural Network which can be implemented on the hardware without any further processing. In the rest of this study, we extract edge and motion information using a spiking neural network which is a different procedure compared to our initial work. An event-based version of well-known linear Hough Transform is introduced with a minimal processing demand. Based on the framework of this study, we briefly review optical flow, Hough transform and spiking

neural network in the literature.

1.3 Major Contributions

In this study, the object's motion is explored in a novel framework and the edge information is extracted in the event-based videos. Considering a moving line, its characteristics are extracted by principal component analysis of the generated events in 3-D spatiotemporal space. This method is applied in small windows on the object boundary to obtain the normal components of the motion vectors. Based on this information, the object global velocity is estimated and then a novel algorithm is proposed for extracting the clear boundary of the object. The key contributions of this part are twofold. Firstly, we provide a fundamental analysis of the motion and edge information that can be estimated from an event-based camera such as DVS. Secondly, we propose algorithms that can provide accurate motion analysis and edge localization for an event-based camera.

Subsequently we perform line detection based on an event-based Hough transform algorithm where the major contribution includes: 1) proposing a fully event-driven SNN-based algorithm for fast line detection, 2) incorporating local lateral inhibition in the SNN for noise line suppression, and 3) applying event-based clustering on SNN output spikes to achieve efficient multiple line tracking. Then we investigate the effect of the inhibitory window shape on the result and introduce an optimal window according to the area between two lines. Moreover, the threshold values of the membrane potential (MP) are regulated for detecting the lines in the frame corners where they have smaller lengths.

1.4 Thesis Organization

This thesis includes six chapters organized as follows:

- **Chapter 1** presents a summarized background of Dynamic Vision Sensor and its applications. The output format of DVS and main characteristics are discussed. Then the research objectives and major contributions of this study are highlighted. The chapter concludes with the whole organization of this thesis.

- **Chapter 2** presents an overview of Neuromorphic engineering and systems including Neuromorphic vision systems e.g. Dynamic Vision Sensor and its applications. Then the optical flow problem is investigated in both conventional form and recent event-based form. Two more topics which are the primary scopes of this study are highlighted including Hough transform and spiking neural network.

- **Chapter 3** introduces a novel framework to analyze the motion of objects in event-based videos. The effect of spatial or temporal quantization is discussed in details and normal motion flow vectors and edge orientations are calculated based on the Principal Component Analysis (PCA) of the generated events in DVS output.

- **Chapter 4** describes the global velocity of an object and the relation with the normal flow vectors estimated in the previous chapter. These vectors are utilized subsequently to extract edge information in two steps for assuring maximum connectivity and minimum thickness.

- **Chapter 5** summarizes conventional Hough transform for line detection. Then a novel event-based version of this transform is proposed in a spiking neural network. The inhibitory lateral connections are used for suppressing redundant lines. These connections, as well as spiking neurons characteristics, are optimized for better results.

- **Chapter 6** concludes the thesis with a summary of research work performed in this thesis as well as some recommendations for future work. Followed by this chapter are the List of authors publications and Bibliography related to this research work.

1.5 Methods Evaluation

This study includes nine experiments for evaluating the efficacy of algorithms proposed in this thesis. Some of them are performed on artificially generated videos while others performed on real DVS videos. The objective of these experiments are as follows:

- **Experiment I:** Showing the accuracy of normal flow vectors and comparison with other existing methods
- **Experiment II:** Finding an optimal size of the spatio-temporal window used in PCA based on the curvature of objects in the video
- **Experiment III:** Showing the accuracy of calculated global velocity and detected edges
- **Experiment IV:** Showing the accuracy of calculated global velocity and normal motion flow vectors
- **Experiment V:** Showing the efficacy of event-based Hough transform on artificially generated videos
- **Experiment VI:** Showing the efficacy of event-based Hough transform on real DVS videos
- **Experiment VII:** Showing the efficacy of event-based Hough transform on real DVS videos for detecting parallel lines
- **Experiment VIII:** Showing the efficacy of revised event-based Hough transform on artificially generated videos for detecting small lines
- **Experiment IX:** Showing the efficacy of revised event-based Hough transform on real DVS videos for detecting the lines in close proximity

The association between the chapters of the thesis and the author publications is as follows[1].

Table 1.1: The association between thesis chapters and author publications[1]

Chapter number	Author publication number
3	[3]
4	[2],[3]
5	[1], [2], [4]

[1]Refer to page 123 for Author List of Publications.

Chapter 2

Literature Review

This chapter presents an overview of Neuromorphic engineering and systems like Dynamic Vision Sensors and some existing methods on event-based video processing. Optical flow is explored in both conventional form and new event-based form. Two more topics including spiking neural network and Hough transform are highlighted as main scopes of this research work.

2.1 Neuromorphic Engineering

Living organisms have been an essential source of inspiration for human-made technology and development. They have had direct/indirect influences on researchers when they were solving problems or designing artificial systems. Simply we can express that Neuromorphic engineering is a field for implementing neural systems of living creatures especially human or animals on the electronic circuits. This implementation can be done in different levels e.g. about human vision system, a simple neuron, many neurons as a vision sensor, the whole visual system, the brain area for vision or even the whole brain. The term 'Neuromorphic' was used by Carver Mead for the first time in the late 1980s, although the idea goes back before the term.

2.1.1 Background

According to the complexity Neuromorphic systems usually have, they are implemented in analog or digital Very Large Scale Implementation (VLSI). The initial implementation of a neuron with a capacitor and resistor was reported in 1907 [10]. Since then, more complex models of a neuron or sensory surfaces have been introduced by advances in electronic circuits. These advances were utilized more in communications and digital calculations and less in other fields. Hodgkin and Huxleys work in 1952 on dynamics of cells might be the first basis for mathematical models of a cell function and its equivalent electrical circuit [11]. Fitz Hug utilized this model and implemented on a computer to see the behavior of a neuron [11]. Then some simple equivalent circuits were introduced for neuron model [12], [13], [14], [15]. On the other hand, first large scale Neuromorphic circuit was an electrical simulation of pigeon retina in 1968 [16]. [17] discussed a fairly large electrical model for the cochlea with a transmission line connecting 178 different sections and every section included four capacitors and two inductors. Although such large models are useful for simulating and testing real neural systems, they are not appropriate to be used in equipment. The method of using many discrete parts and constructing a more complex system has two issues, complexity and cost. Building such systems for research purposes in the laboratory is reasonable, although they are not economically appropriate for general applications that need a large number of such systems.

2.1.2 Recent Developments

In 1989, an analog field-effect transistor was considered as a mimic of an actual neuron membrane since its current is exponentially related to its voltage [18], [19]. This finding opened the door to those wanted to artificially implement biological neurons in VLSI hardware and resulted in the low-cost design of Neuromorphic chips in large numbers. It still was a long wait from the time one designed a Neuromorphic circuit until the implemented device was delivered from a foundry. Many novel Neu-

romorphic circuits implemented in analog VLSI are proposed in [20]. Some of them are simulating the sensory systems while others mimic just a single neuron or small parts of cortex [21]. We will review the existing models of a single neuron in Section 2.2.

Recent advances in VLSI technology have made such implementations very easy and cheap. As a result many Neuromorphic chips have been proposed for visual ([1], [22], [23], [24], [25], [26], [27]), olfactory ([28], [29], [30]), auditory ([31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42]) and tactile systems ([43], [44], [45], [46]). We should highlight that the term 'Neuromorphic' is usually used just for systems work in real time and the elements operate in parallel. We will discuss more on Neuromorphic vision systems in Section 2.4 since they are the primary interest of this study.

2.2 Artificial Neuron

The simplest model proposed as an artificial neuron is integrate-and-fire neuron which is widely used in many applications. A comprehensive review of the integrate-and-fire neuron is performed in [47]. IF neuron is the least complex model of a real neuron lacking many actual characteristics. The synapses are considered as electrical currents into the neuron. These currents increase the membrane potential continuously like charging a capacitor with an electrical current. When the membrane potential exceeds some threshold, the output fires and generates a spike. The inputs of a neuron are modelled as a Poisson process in time. Then the spikes temporal distribution in output is calculated mathematically. Rapid spike generation in the neuron output and fairly slow integration of synaptic inputs are two essential features of actual neural processing. The model is simple enough to provide analytical answers to some questions. On the other hand, it suffers from many shortcomings caused by excessive simplifications of the model. The firing mechanism and spatial structure are two aspects which have been neglected in IF model. Therefore using

the results of this modeling as a guide to real neural behavior is usually questionable. Some improvements for IF model are discussed that enhance the model applicability and similarity to actual neurons and at the same time increase the complexity of mathematical equations. Despite shortcomings IF neuron has, it is widely used in many applications and arguments such as explaining the control mechanism of the brain [48] and network stability [49].

Among recent works, [50] presents the simplest possible model that combines the exact behavior of most real neurons likes spiking, bursting, etc and the low processing demand of IF neuron. The model contains just two equations with only one nonlinear term as seen in Figure 2.1. If second equation is removed, the model misses its bursting capability and if the nonlinear term is neglected, the model is equivalent to the resonate-and fire model. Figure 2.1 shows the equations of this model and its responses to a step function for different values of parameters. Moreover, the paper shows how to create a spiking neural network that simulates mammalian cortex. According to the simplicity of this model, a thalamocortical network can be implemented by tens of thousands of neurons with $1ms$ resolution in a normal $1GHz$ PC.

[51] introduces many models of spiking neurons and compares them with respect to biologically plausibility and cost of implementation. 20 different neuro-computational properties are considered for a model to be biologically plausible as shown in Figure 2.2. This paper discusses which model is the best and concludes that it depends on the application in which a model is used. If the purpose is studying a real neuron behavior under specific conditions, we should use HodgkinHuxley model which is the most biologically plausible. Although a small number of neurons can be run online due to the computational complexity of this model. On the other hand, if the goal is simulating a large number of neurons in real time, the most appropriate model is the integrate-and-fire model which is the most computationally optimal, even though it does not show very basic characteristics of real neurons. The advantages of IF model is that it is linear and as a result, mathematical analysis of complex structures of these neurons is easily possible. The quadratic IF model has a low

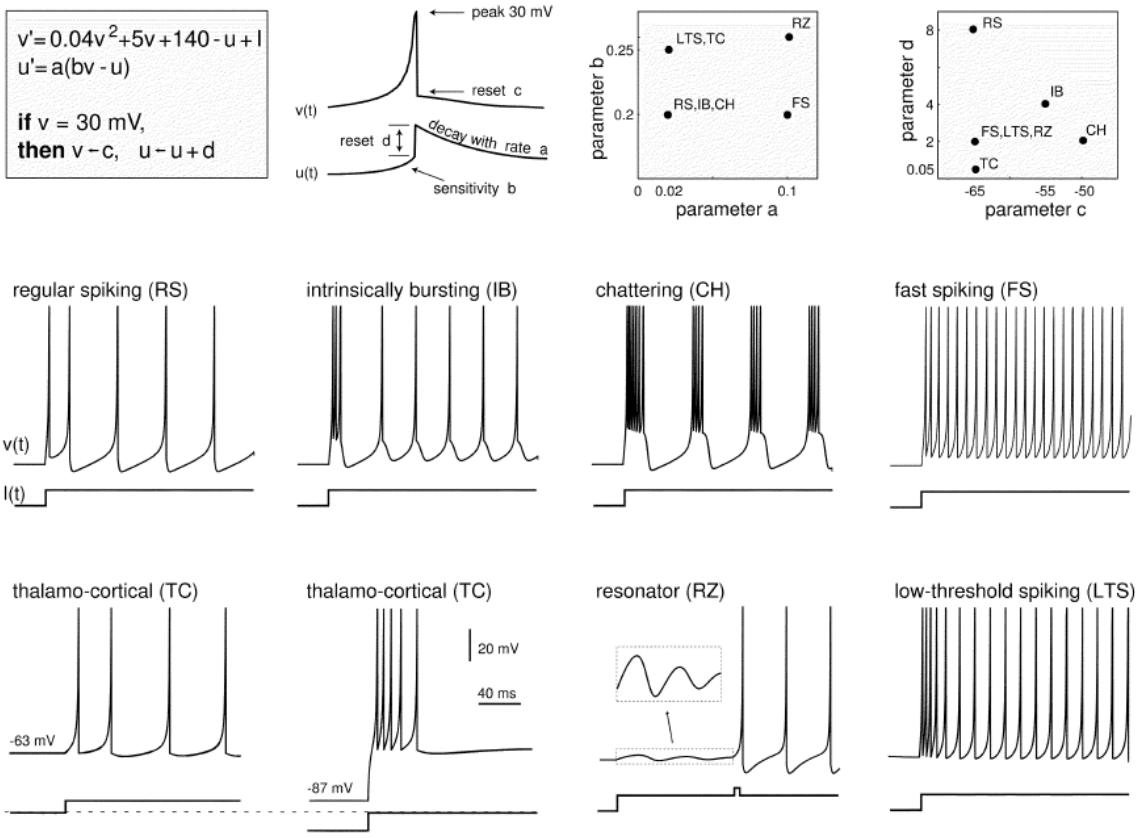


Figure 2.1: Equations of Izhikevich model and their all possible responses to a step function for different values of parameters a, b, c, d. This model can exhibit almost all firing patterns in the output

complexity near to IF neuron as well as more biologically plausibility, although it can not show spike frequency adaptation. Another finding is that when neurons are polychroniously grouped, the number of patterns that network could memorize exceeds the number of neurons or even the number of synapses significantly which can result in an unprecedented memory capacity.

2.3 Spiking Neural Network

Spiking Neural Network (SNN) is the third generation of Artificial Neural Network (ANN) models. Compared to conventional ANN, SNN is more biologically plausible since it incorporates spike times into computational models which mimic the information processing in the biological neural system. Recent advances in VLSI technology have solved the spiking neurons' implementation issues [52], [53] that we faced previously because of their higher computational complexity compared to conventional artificial neurons, and thus have largely boosted the SNN research and development. SNN have been used for many tasks such as learning [54], [55], [56] and classification [57], [58], [8]. Among the various spiking neuron models proposed in the literature [50], [59], the most popular one is Leaky Integrate-and-Fire (LIF) neuron model [47] due to its simplicity. Since the output of DVS can be considered as some spikes in time, it is highly consistent with any spiking neural network input and combination of them has been a trending research interest.

2.4 Neuromorphic Vision

The first Neuromorphic visual system implemented in VLSI was introduced in [27]. The paper represents a silicon retina consists of two layers. The lower layer is an array of 48×48 photo receptors (which have near-logarithmic responses to light) while the upper layer is a horizontal resistive layer which simulates the plexiform layer of a real retina. The system outputs the difference between a pixel intensity and the weighted

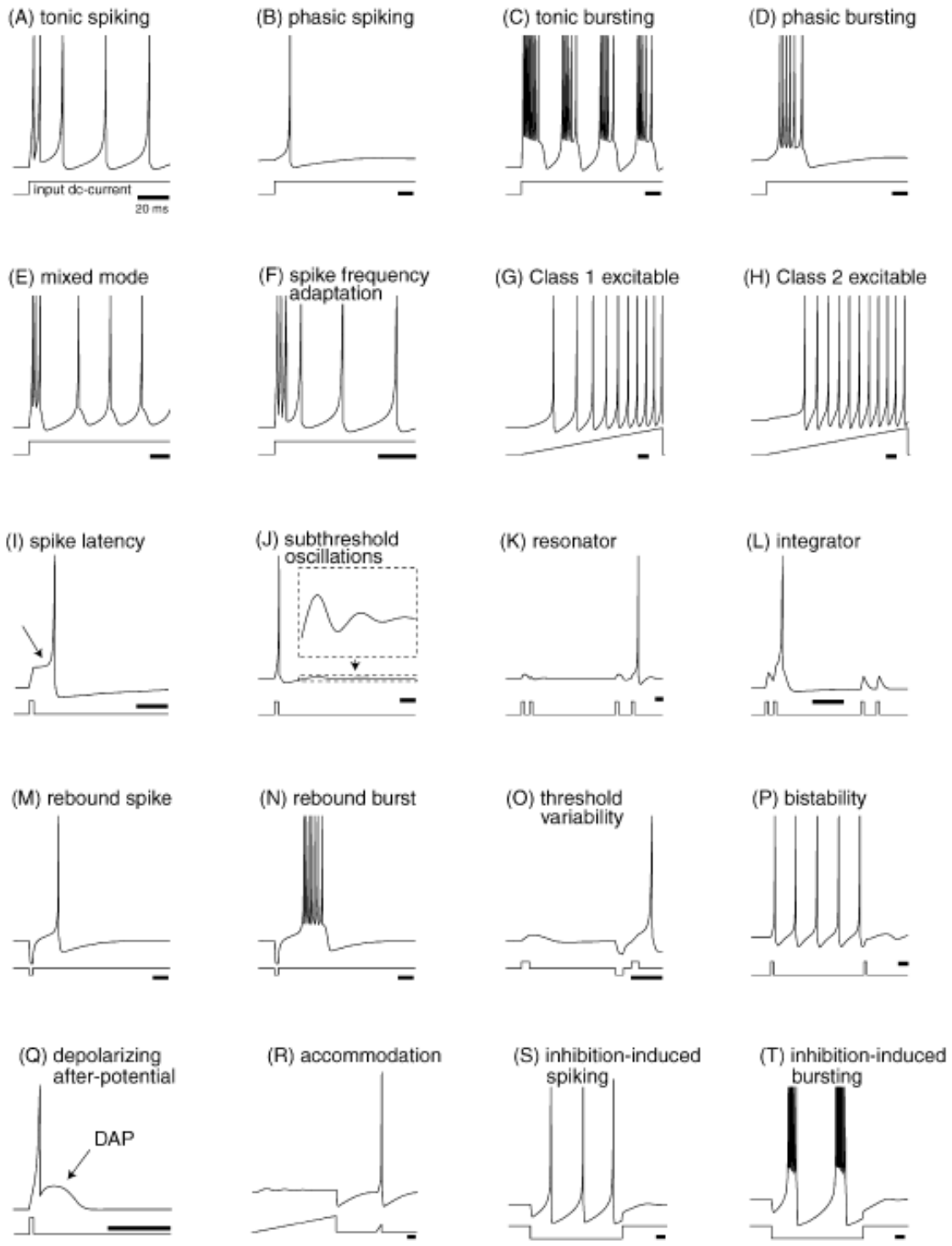


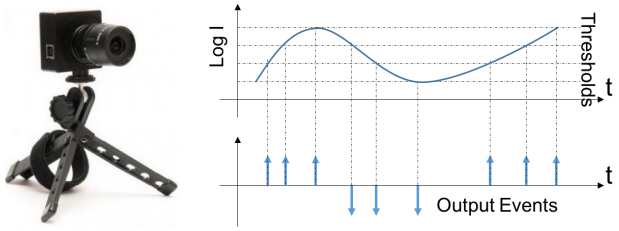
Figure 2.2: 20 different neuro-computational properties of real neurons. These patterns are used for evaluation of models with respect to their biologically plausibility

average intensity of all neighboring pixels. As a result, the system response for a plain mono-color surface is zero while the response for a static edge is spatial derivative similar to a real retina. This idea has been developed further and subsequently, better transducers were introduced in [26] and [25]. A Neuromorphic retina can be developed for a special application by adding an additional component or performing further processing in the device. Depth detector [22], moving object detector and localizer [4], time to collision detector [24] and motion detector system of flies [23] are some examples of such Neuromorphic retinas. One of the Neuromorphic vision devices which has recently attracted many researchers interests is Dynamic Vision Sensor (DVS) [60]. This study is performed on captured videos by Dynamic Vision Sensors. The following presents an overview of these imaging devices as well as their applications.

2.4.1 Dynamic Vision Sensors

Dynamic Vision Sensor (DVS) is the latest Neuromorphic implementation of a real visual system. Currently they are available in two different resolutions 128×128 ([61], [62], [63], [1]) or 240×180 ([64], [65]). Conventional cameras capture a series of frames which have information about all the pixels. Therefore there is a lot of information in each frame most of which is redundant. The high volume of information in each frame prevents the camera to have a better time resolution [66]. On the other hand, DVS captures only the changes in pixels intensity and as a result, it generates less redundant data. This data is transmitted serially using Address Event Representation (AER) protocol. Moreover, due to logarithmic intensity change detection, DVS sensors offer a very high dynamic range, meaning it has no problem in capturing the scenes containing both very dark and very bright areas [63].

The main characteristic of DVSs is high temporal resolution (usually 1 microsecond). These cameras are event based; meaning that when there is a variation in a pixel intensity, a polarized event is created in the form of a vector. The vector of each event has three elements (x, y, t); The coordinate (x, y) defining position of the pixel



(a) DVS (b) DVS working principle

Figure 2.3: DVS used in this study and its working principle

while the term t showing instance of the variation. In addition, the event polarization represents the change direction of the pixel intensity.

Dynamic Vision Sensors are sensitive to intensity change, more specifically, to intensity logarithmic change. Let us consider the logarithm of a pixel intensity as shown in Figure 2.3(b). Once the logarithmic intensity change is larger than a predefined threshold, a positive or negative event will be generated depending on the direction of the change (dark-to-bright or bright-to-dark). Figure 2.3(a) shows the DVS used in this study which has a 128×128 spatial resolution and $1\mu s$ temporal accuracy. According to the characteristic of a logarithmic function which is more sensitive to small values, DVS is also more sensitive to darker areas. A small change in low intensity (dark pixel) can cause a significant change in the intensity logarithm and generate an event subsequently. As a result, there will be more noise events in darker areas and it is verified by actual experiments as well.

The DVS we use (Figure 2.3(a)) has the latency of $15\mu s$ in hardware level. However events are received with much delay because of USB interface mechanism which sends data in blocks. The USB delay does not affect the content of DVS output data. Nevertheless it is reflected on the receiving delay. As a result user is assured about the accuracy of received data from DVS [1].

Referring to Figure 2.4 one black square is moving on a white background. When the motion is horizontal, two parallel vertical lines of events occur. In contrast for the vertical movement, lines of the events are horizontal. It is Noted that the black pixels

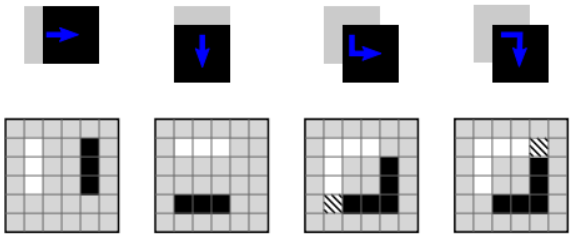


Figure 2.4: The generated events by a moving simple square

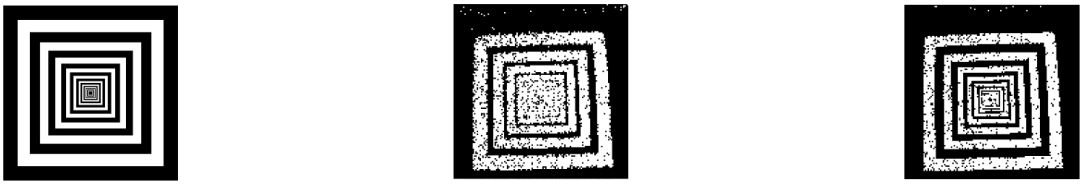
show the negative events while white pixels show the positive events. For diagonal motion, a complete square of events occur. The hatched pixel represents a positive event following a negative one [67].

2.4.2 Focusing a DVS

In conventional cameras, before capturing any video the distance between image sensor and camera lens should be adjusted. Without correct setting of the focal length, blurred videos will be captured by camera. The same issue exists in DVS, but the meaning of a blurred video is different in DVS comparing to that of conventional cameras. Figure 2.5 describes the difference between in focus and out of focus DVS.

A DVS generates events only if there is variation in the pixels intensity. It can happen by a moving object or changing the pixels intensities themselves. To properly focus a camera, moving object is not a good choice. As a result continuous change in pixels intensities are needed. For this purpose, a simple tool may be conventional monitors which have brightness oscillation at a high frequency about hundreds of hertz. A normal eye is unable to recognize this oscillation while a DVS can easily detect it [67].

As depicted in Figure 2.5(a), a focusing pattern is displayed in a conventional monitor. The distances between squares are logarithmic. The inner squares are nearer than the outers. DVS captures a video from the monitor as shown in Figure 2.5(b). The black areas don't generate any event since their intensity is always zero.



(a) Focusing pattern (b) Out of focus DVS (c) In focus DVS

Figure 2.5: A pattern for adjusting the distance between image sensor and DVS lens

However the white areas generate many events by monitor brightness change at high frequency. Using a simple trick, a continuous stream of events is obtained without moving DVS or any other object. At this stage, the focal length can be adjusted for having a good capturing result. When the DVS is in focus, the captured pattern will be as shown in Figure 2.5(c) having the maximum number of visible squares in the frame.

2.4.3 DVS Applications

Although DVS devices are relatively new, they have many applications in machine vision, robotics, high speed tracking and surveillance. In the following we mention some algorithms proposed for different objectives.

In a study, a DVS has been exploited as a traffic camera over the highways where tracking a vehicle and estimating its velocity are two important issues. Knowing the accurate position of DVS over the highway and the captured velocity in the video, its geometric problem is solved and the vehicle actual velocity is correctly estimated [68].

Having high temporal resolution, a DVS can be used as a goalkeeper as well. Tracking fast moving objects e.g. a ball is hardly possible by conventional cameras. A ball has significant displacement within time period between two consequent frames in conventional cameras. An event-based algorithm has been introduced for tracking circles with varying diameters [3], [69]. Similarly another method has been proposed for tracking micro-particles [70].

In addition, a tracking algorithm [4] has been suggested based on a Gaussian modeling of objects [71]. Subsequently this algorithm has been generalized for part-based shapes based on minimizing an energy function defined by parts locations and distances [5].

Object recognition [6], [7], [8] and gesture recognition [72] are other applications of DVS devices. The frames are divided into sub-frames and the numbers of events in these sub-frames are given as inputs to an ordinary neural network which is able to distinguish between simple actions.

Robot self-localization is the next area where DVS devices can be used successfully. A DVS is mounted on a robot and the coordinate of the robot is estimated accurately based on the processing of captured events [73], [74], [75]. Moreover DVSs have been used for terrain map reconstruction [76].

The corner points of an object are the intersecting points of two edges in different directions. For a corner point, the velocity components along the perpendicular directions to two intersecting edges are extracted. These components can be used to recognize the corner points and their actual velocity [9].

In [77], [78] and [75] some methods have been introduced for event-based visual flow extraction. They are accurate in detecting the normal velocity of the objects in DVS videos. The following section discusses more in this area which is one of primary purposes of this study.

2.5 Conventional Optical Flow

Optical flow is an important problem in machine vision which is widely used in many applications e.g. object tracking. In the conventional form of this problem, it is assumed that for a special point of the scene, the intensity value will be unchanged regardless of objects or camera motion which is called Brightness Consistency Assumption or BCA. This problem is equivalent of finding pixels with the same inten-

sities in successive frames of the video. In a video, intensity is a function of three parameters x, y, t. The coordinate (x, y) is the pixel location while the term t is the time or frame number. BCA enforces the intensity variation to be zero. As a result:

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = \begin{pmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} + \frac{\partial I}{\partial t}$$

$$(\nabla I)^T \mathbf{V} + \frac{\partial I}{\partial t} = 0 \tag{2.1}$$

Equation (2.1) is the fundamental equation of the optical flow problems. Having an equation with two unknown variables v_x and v_y, it can not be solved uniquely and therefore another equation is needed. There are many methods for considering other reasonable equations or assumptions. One is Lucas-Kanade method [79] which assumes a fixed displacement for all n pixels within a window. As a result, equation (2.1) is valid for all n pixels with the same velocity $\mathbf{V}$. This leads to n equations and two variables which will be solved by the least squares criterion.

Some groups of methods for optical flow calculation was proposed including differential techniques ([80], [81], [82], [83]), region-based matching ([84], [85], [86], [87]), energy-based methods ([88], [89], [90], [91], [92]), phase-based techniques ([93], [94]). However, there was not a comprehensive comparison between these methods until 1994 when Barron et al. provided some artificial and real data with known ground truth and reported the accuracy of existing methods on them [95]. This was a beginning for quantitative analysis of methods and improving their accuracy. This data set was standard until the more challenging Middlebury benchmark was introduced in 2011 [83]. Both data sets have helped to explore difficult areas of optical flow calculation and improve the accuracy significantly so far. A more recent work by Sun et al. [96] has performed an extensive review of existing methods. They believe that most of the methods try to solve the main optical flow Equation 2.1. However many of them can not be run online on a video stream according to high processing demand they have.

2.6 Optical Flow for Event-Based Videos

The problem in DVS is the lack of intensity absolute values. As a result, none of the intensity partial derivatives e.g. ∇I nor $\frac{\partial I}{\partial t}$ can be calculated which is necessary for estimating the conventional optical flow vectors. In DVS, only the sign of $\frac{\partial I}{\partial t}$ is known whether it is positive or negative and obviously it is not enough for estimating the optical flow by this method. There are some groups of methods in the literature that we review in following sections. Four methods are based on the conventional form of the optical flow (Lukas-Kanade variants) and four other methods are based on local plane fitting on the latest events. Another method is Direction Selective filter which is limited to four major orientations and as a result, it has a limited accuracy for flow directions [97].

2.6.1 Direction Selective Filter

This method was proposed by T. Delbruck in 2007 for the first time as an inspiration of biological V1 direction selective cells [98], [99]. The idea behind is that the events occurring along an edge are almost synchronous; i.e. the temporal distance of events along an edge is very small.

In Direction Selective (DS) algorithm, a 2-D array is used for storing the time stamps of latest events at different pixels. Two separate arrays are considered for positive and negative events. By receiving any event, the time difference between the incoming event and latest events of all neighboring pixels along a direction is calculated. This procedure is carried out on neighboring windows of five pixels for four major directions $(0^o, 45^o, 90^o, 135^o)$. As the pixels along a moving edge generate almost synchronous events, one of four possible directions (which is the nearest to correct edge orientation) will have the minimum average of time differences. In this procedure, the events older than some threshold e.g. $100ms$ are not counted in calculating the average time differences. There are eight more 2-D arrays correspond to four possible directions and two types of edge (positive/negative). By finding the

best matching direction, the orientation time stamp (which is the same as incoming event time stamp) is stored in the corresponding 2-D array.

The next step is to find the velocity of the detected edge. For every orientation event (which is represented by a number in one of eight 2-D arrays), all neighboring numbers along the perpendicular direction to the actual edge are examined. The neighboring window is considered 5×5 pixels and the numbers older than a threshold e.g. $100ms$ are neglected. The numbers located at one side of the edge (toward which the edge is moving) are usually too old and as a result, they are ignored. Alternatively, two numbers at the other side are used for calculating the speed of detected edge. Simply the reciprocal of the average temporal difference of these numbers will be the edge velocity in pixels per second. The velocity magnitude is reported as well as a number between $0 - 7$ showing the velocity orientation among eight possible directions.

2.6.2 Lukas-Kanade Variants

In section 2.5, we mentioned that to solve the optical flow problem in the conventional form, we need to solve both BCA and Lukas-Kanade equations simultaneously using LMS method. In conventional image processing, to achieve this simultaneous solution, we use the spatial and temporal derivatives of image intensity. However, these values are not defined in an event-based output stream. In the literature, there are four variations of techniques to estimate these derivatives and then solve the optical flow problem using conventional techniques. In this group of solutions for optical flow problem in event-based videos, the first step was taken by Benosman et al. [77]. The histogram of events in a small window during a short period of time is used to estimate the spatial and temporal derivatives, which are then used to solve the optical flow problem. The methods to estimate these derivatives from histogram of events, can be categorized in four groups, briefly discussed in the following.

2.6.2.1 Backward finite difference

The first method is to estimate the intensity partial derivatives by the differences of pixels activities [77]. A pixel activity is defined as the summation of all events at that pixel within a time period e.g. $\Delta t = 50\mu s$.

$$\frac{\partial I(x,y,t)}{\partial x} \approx \sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x-1,y,t)$$

$$\frac{\partial I(x,y,t)}{\partial y} \approx \sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x,y-1,t)$$

$$\frac{\partial I(x,y,t)}{\partial t} \approx \frac{1}{\Delta t} \sum_{t-\Delta t}^{t} e(x,y,t)$$

2.6.2.2 Second order temporal derivative

The second method is to use the second order difference of pixels activities for temporal derivative [100].

$$\frac{\partial I(x,y,t)}{\partial t} \approx \frac{1}{\Delta t}(\sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-2\Delta t}^{t-\Delta t} e(x,y,t))$$

2.6.2.3 First order central finite difference

Previous two methods are biased toward $x-1$ and $y-1$ which is verified by experiments as well. Using central finite differences as spatial derivatives remove this bias.

$$\frac{\partial I(x,y,t)}{\partial x} \approx \frac{1}{2}(\sum_{t-\Delta t}^{t} e(x+1,y,t) - \sum_{t-\Delta t}^{t} e(x-1,y,t))$$

$$\frac{\partial I(x,y,t)}{\partial y} \approx \frac{1}{2}(\sum_{t-\Delta t}^{t} e(x,y+1,t) - \sum_{t-\Delta t}^{t} e(x,y-1,t))$$

2.6.2.4 Second order central finite difference

The fourth method is the second order of accuracy for calculating central finite differences as spatial derivatives.

$$\frac{\partial I(x,y,t)}{\partial x} \approx \frac{2}{3}(\sum_{t-\Delta t}^{t} e(x+1,y,t) - \sum_{t-\Delta t}^{t} e(x-1,y,t)) - \ldots$$
$$-\frac{1}{12}(\sum_{t-\Delta t}^{t} e(x+2,y,t) - \sum_{t-\Delta t}^{t} e(x-2,y,t))$$
$$\frac{\partial I(x,y,t)}{\partial y} \approx \frac{2}{3}(\sum_{t-\Delta t}^{t} e(x,y+1,t) - \sum_{t-\Delta t}^{t} e(x,y-1,t)) - \ldots$$
$$-\frac{1}{12}(\sum_{t-\Delta t}^{t} e(x,y+2,t) - \sum_{t-\Delta t}^{t} e(x,y-2,t))$$

2.6.2.5 Savitzky-Golay filter

Lukas-Kanade methods are based on derivatives of pixels activity as discussed in previous sections. The pixels activity is taken in small temporal windows and since the events are sparse, they are highly sensitive to noise. Smoothing the pixels activity function by fitting a low-order polynomial to neighboring pixels makes the function more robust to noise. One method for smoothing is using Savitzky-Golay (SG) [101] filter which estimates a two-dimensional function as follows:

$$f(x,y) = \sum_{p=0}^{n} \sum_{q=0}^{n-p} a_{pq} x^p y^q$$

This fitting is usually performed to a 3×3 window using least squares method and polynomials are confined to the same order in both x and y directions for simplicity. The coefficients of first order polynomials immediately show the gradients of the surface. As a specific example, if we solve the problem for the first order of fitting in 3×3 windows, the gradients in x and y directions will be as follows:

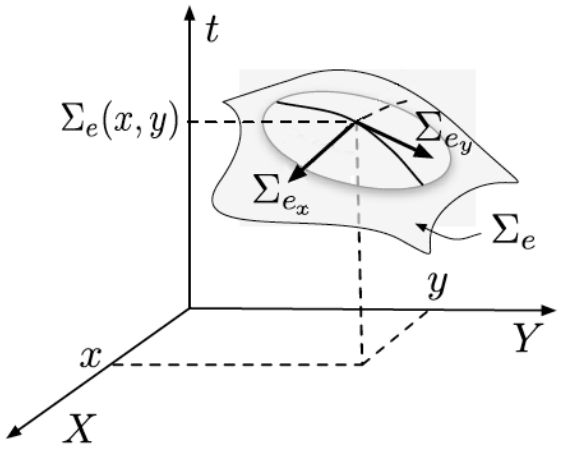


Figure 2.6: Active events surface $t = \sum_e(x, y)$ in spatio-temporal space. $\sum_{e_x}$ and $\sum_{e_y}$ are partial derivatives of this surface with respect to x and y.

$$\begin{bmatrix} +\frac{1}{6} & +\frac{1}{6} & +\frac{1}{6} \\ 0 & 0 & 0 \\ -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} \end{bmatrix} \qquad \begin{bmatrix} -\frac{1}{6} & 0 & +\frac{1}{6} \\ -\frac{1}{6} & 0 & +\frac{1}{6} \\ -\frac{1}{6} & 0 & +\frac{1}{6} \end{bmatrix}$$

We need to calculate these coefficients just once for a specific order of polynomials and window size. The results of this method will be presented in Tables 3.2 and 3.1 as LK_{SG}.

2.6.3 Local Plane Fits

The next group of solutions for this problem was introduced in 2014 [78]. As stated before, every event has the location and time information. The time parameter can be considered as a function of the events' location. This function is defined as $t = \sum_e(\mathbf{P})$ where $\mathbf{P}$ is the location vector of the events. If events are represented in 3-D spatiotemporal space, they form a surface as seen in Figure 2.6.

Considering an event on the surface $\sum_e$, a small area around this event on the surface can be realized as a small plane. As a result, the events within a small spatiotemporal window are supposed to be on a plane. Keeping only the first term

of the Taylor series, there is a linear equation between t and $\mathbf{P}$.

$$\Delta t \approx \nabla(\sum_{e}).\Delta\mathbf{P} = \frac{\partial \sum_e}{\partial x}\Delta x + \frac{\partial \sum_e}{\partial y}\Delta y$$

The above equation describes a plane in 3-D spatiotemporal space. In other words, the problem is equivalent to finding a plane $ax + by + ct + d = 0$ tangent to events surface at a point. The inverse slope of this plane along x or y axes are the velocity components along those. This is accomplished by an iterative procedure on all events inside a small spatiotemporal window. The objective is fitting a plane using the least squares regression. In every iteration, those events further away from the plane than $10ms$ are neglected and the algorithm continues with the remaining events. Every iteration makes the parameters (a, b, c, d) more accurate and the algorithm stops running when the Euclidean distance of these parameters (a, b, c, d) in two successive iterations becomes less than 0.01. Finally, the edge velocity in x and y direction will be as follows:

$$v_x = \frac{\partial x}{\partial t} = \frac{-c}{a} \qquad\qquad v_y = \frac{\partial y}{\partial t} = \frac{-c}{b}$$

Above formulations are correct unless the edge is along either x or y axes. For a vertical/horizontal edge, one of the parameters (a, b) is zero and above formulations result in infinite speed along the edge which is obviously not true. One can set a threshold for parameters (a, b); i.e. if either a or b is less than a threshold, the corresponding velocity is neglected and considered as zero.

2.6.3.1 Orthogonality constraints

We face the issue of infinity speed along the edge because events surface is not a one-to-one function and as a result, not invertible [100]. The edge velocity magnitude is reciprocal of the surface gradient magnitude $(\frac{1}{g})$ while the edge velocity direction is the same with gradient orientation. Therefore edge velocity vector can be calculated

based on the gradient magnitude and direction as follows:

$$\mathbf{g} = \begin{bmatrix} \sum_{e_x} \\ \sum_{e_y} \end{bmatrix} = \begin{bmatrix} \frac{\partial t}{\partial x} \\ \frac{\partial t}{\partial y} \end{bmatrix} = \frac{-1}{c} \begin{bmatrix} a \\ b \end{bmatrix} \quad \rightarrow \quad g = \frac{\sqrt{a^2 + b^2}}{c}$$

$$\mathbf{v} = \frac{1}{g}\frac{\mathbf{g}}{g} = \frac{\mathbf{g}}{g^2} = \frac{-c}{a^2 + b^2} \begin{bmatrix} a \\ b \end{bmatrix} \tag{2.2}$$

We should mention that by this method, only the normal components of the visual motion flow to the object boundary are extracted [75].

2.6.3.2 Single fit

The problem of finding surface parameters (a, b, c, d) using an iterative algorithm and improving the results step-by-step is a bit challenging. If the initial values considered for (a, b, c, d) are not true enough, then the correct events will be ignored wrongly as too far points from the plane and the parameters will not be revised correctly in the iterations.

An alternative method which solves the above-mentioned issue is to utilize all events simultaneously and fit the plane based on the Least Mean Squares method. As a result, the best-fitted plane is obtained instantaneously by considering all events inside a spatiotemporal window and this method does not remove the events which are too far from the fitted plane.

2.6.3.3 Savitzky-Golay filter

For smoothing the noisy surface of latest events in a spatiotemporal window, a two-dimensional Savitzky-Golay filter is applied again. For the first order of this filter, the following plane is obtained.

$$t(x, y) = a_{00} + a_{10}x + a_{01}y$$

Using Equation 2.2, the flow vector is directly calculated by substituting $(a_{10}, a_{01}, -1)$ for (a, b, c).

Sometimes the spatiotemporal window contains some old events belonging to an object previously passed from the same location. This method, as well as the previous method (single fit), do not offer any way for ignoring such events. One solution is estimating the average derivatives in x and y directions. This method is repeated over x values or y values of events and calculate the derivatives for valid pairs of events. The final gradient will be the average of these individual derivatives. It can be shown that this method is equivalent to the first order Savitzky-Golay filter with the benefit of ignoring invalid events, at the cost of more computations.

2.7 Hough Transform

Hough Transform (HT) is a well-known method in computer vision to efficiently identifies lines in images. The first appearance of HT was in a patent application by Paul V C Hough in 1962, for machine analysis of bubble chamber photographs [102] with the name "Method and Means for Recognizing Complex Patterns". Then Hough Transform was proposed in 1972 as a feature extraction (especially line detection) method in computer vision [103]. Through years of application to a wide range of pattern recognition applications in more than 2500 research articles, resulted in the method to be popularly called Hough Transform (HT), and it is frequently used today. The main idea of this method is first transforming every point from the conventional Cartesian coordinates to the parameter space, in which every point defines a specific shape, and then finding local maximums in the parameter space to obtain the shape parameters through a voting procedure. The dimension of the parameter space depends on the shape that we want to extract and its complexity. A line can be uniquely defined by two parameters and therefore the parameter space for detecting lines is two dimensional. Three parameters (x and y positions of center and radius) can define a circle on a plane, and thus the parameter space for detecting

circles is three dimensional. Hough transform can also be used for detecting arbitrary shapes [104]. Hough transform can be accelerated by managing the needed memory size or limiting the search area in the parameter space. The problem of dimensionality can be reduced by some tricks [105]. Using line detection as an example, since the line direction is perpendicular to the image gradient vector, the line slope can be obtained immediately according to the gradient vector and a two-dimensional Hough transform is then converted to a simpler one-dimensional problem. Another idea for accelerating the algorithm is using coarse-to-fine searching in the parameter space and is named hierarchical Hough transform [106]. In this section, we briefly discussed on the literature of Hough Transform.

It is probable that the popularity of HT usage in pattern recognition and image processing is rooted in that a significant portion of man-made and natural objects has a circular profile like disc, coin, button, celestial body, biscuit, and biological part. Moreover, the oblique projection of these objects into 2D space can be used for orientation estimation, as the projections of circles result in an elliptical pattern. These properties render HT a beneficial image preprocessing for a variety of applications in pattern recognition and object detection fields [107].

The parameterization of circle used in [102, 108] is:

$$f_c(x, y) = (x - a)^2 + (y - b)^2 - r^2 = 0$$

where (a, b) denote the center and r is the radius of the circle. Thus, each pixel in the image plane is transformed to a cone in a 3-D parameter space, also known as the Circle HT (CHT) [109]. It can be shown that by coding orientation and distance information as a complex phase, one can improve position accuracy [109].

The general equation of an ellipse is:

$$x^2 + b'y^2 + 2d'xy + 2e'x + 2g'y + c' = 0$$

where b', c', d', e', g' are constant coefficients normalized with respect to the coefficient

of x^2. Based on the ellipse equation, to detect an elliptical object using HT, a 5-D accumulator array is needed, parameterizing location (co-ordinates of center), shape and orientation of an ellipse.

With the great potential to define shapes, HT has been shown to perform as a matching filter for several applications [110, 111]. A template matching filter is applied to image domain, based on corresponding image points. In many cases the complete set of corresponding points does not exist in the image, adding unnecessary calculations on these template points, with no effect on the retrieving the matching degree between the model and the image. Using HT, a shape can be represented by a list of boundary points, a.k.a. templates, and similar shapes have similar templates with addition of translation, rotation and scaling [104]. HT then always assumes a match between given basic template point and a selected image point and then calculates the transformation parameters which connect them.

For the image matching application, scalable translation invariant rotation-to-shifting (STIS) signature was proposed by Pao et al. [112] where rotation in image space corresponds to circular shifting of the signature space. Using the signature space, a matching merely needs computing a 1-D correlation of the reference template with the test template.

2.7.1 Standard Hough Transform disadvantages

Standard Hough Transform (SHT) is a powerful method that is robust against missing data and discontinuity on the curve [102,108], but SHT has several drawbacks:

- Time and memory relation with curve parameters: The computation time and memory requirements grow exponentially with the number of curve parameters because n parameters, each resolved into m quantized intervals (bins), require an n-D accumulator of mn elements.
- Time and memory relation with quantization: For high accuracy of localization, finer parameter quantization is required, that entails higher processing time

and memory requirements. There is no clear-cut way to optimize the trade-off between accuracy and computation cost for an image.

- Uniform parameter quantization: A uniform quantization of the parameter space means a non-uniform precision in detecting the curve in the image space.
- Peak Spreading: Due to various sources of error, votes near the true parameter vector will increase, leading to peak spreading, which hampers precise detection of maxima in the accumulator.
- One transformation process per feature: For the detection of each new feature, a separate transformation process must be initiated.
- Blind voting: Irrespective of noise, voting contribution of each pixel in the parameter bins is the same (blind voting), resulting in reduction of detection accuracy.
- End points: SHT cannot automatically detect the end points of line segments.

To overcome these limitations, modifications have been made in one or more stages. Some of these algorithms have introduced features that are significantly distinct from the SHT, including Generalized Hough Transform (GHT) [104], Probability based HT [113], Randomized HT [114], and Monte Carlo HT [115]. In this text we briefly overview GHT due to its relation to our proposed methods.

2.7.2 Generalized Hough Transform (GHT)

An initial approach towards a generalized version was made by Merlin and Farber [116] by assuming the target object to be the result of translation of the model object. This idea was extended by Ballard [104] into the Generalized HT (GHT), a generalization of SHT, that includes translation, rotation, and scaling of the model. GHT is a two-phase learning-detection process to detect non-parametric curves. In the first phase, learning, R-table is constructed from a model object. Then by fixing a

reference point and using it as the origin, a polar co-ordinate system is established in the model object, and the R-table stores the polar coordinates of all template points. Finally, each row at the R-table is indexed by the gradient directions of the edge points on the template. In the second phase, detection, an accumulator is constructed via a 2D array, a.k.a. the Hough Counting Space (HCS) or parameter space. Matching gradients directions at each edge points and the corresponding R-table entries will add a vote to the hypothetical reference point in the HCS. The highest number of votes casted to a cell in the accumulator array (and its corresponding reference point), determines the image pattern matched to the model.

Compared to [116], local information of object edge points is incorporated in GHT, and allow faster and more accurate execution. The local information properties were extended in [117], allowing application of stronger constraints for matches. These extensions include contrast, position and curvature of contour points. Further optimization of the computation is also proposed by Leavers in Dynamic GHT (DGHT) [118], using available information on the distribution of feature points.

While GHT retains the robustness of SHT, it does not solve all SHT drawbacks:

- GHT cannot detect end points of line segments.
- Parallel processing of GHT requires a large number of elements.
- Brute force search is usually applied when orientation and scale of a new shape is unknown
- Due to brute force search, arbitrary shape extraction under similarity or affine transformations leads to 4D and 6D accumulator spaces, with $O(n^4)$ and $O(n^6)$ complexities, respectively.
- Unlike rigid objects, GHT cannot adequately handle flexible shapes that are usually found in nature such as leaves or animals.
- The conventional GHT cannot detect perspective transformation of planar shapes. Most images of real world objects are undergone perspective transformation.

To tackle the dimensionality problem of GHT, Tsai proposed two-stage voting [119]. The first stage of the voting process finds the matching points with the same concavity and radii, to estimate the rotation angle of the object w.r.t the model. The second stage then matches points having the same radii, concavity and rotational angles to find the centroid of the object. Another method proposing multi-staged GHT is affine GHT in [120]. At the first stage, candidate points are selected by applying 2D HT. At the subsequent second stage, a 4-D HT is applied to determine the remaining four parameters. Moreover, Adaptive HT [121] is proposed for efficient estimation of the 4-D HT at the second stage. The same concept of two-stage GHT [119], but on scale and orientation is proposed in Scale and Orientation Invariant GHT (SOIGHT) in [122], where the accumulator array stores scale and orientation pairs for the points.

GHT can be extended to recognize articulated objects. One approach is using reference frames at joints [123]. Another approach is a modified GHT, HT for Natural Shapes (HNS), that updates votes of all points on a line segment [124]. This approach can be used for natural shape recognition. An extension to HNS is proposed in [125] that allows template matching from a single sample of the object.

To achieve invariance to perspective transformation, the method proposed in [126] uses a Perspective-Reference table, to store exhaustive list of perspective transformation information of the model.

2.7.3 Digital Hough Transform (DHT)

The original SHT variants work in analog mode and do not consider the location quantization due to image digitalization [127]. Digital Hough Transform (DHT) techniques consider different aspects of using SHT in the digital domain.

When an analog straight line is digitized, it forms a digital straight line (DSL). The Analytic Hough Transform (AHT) [128] works on a set of pixels and determines if an analog straight line can be obtained in digital format, using that set of pixels.

Chord property, the basis of these techniques, determines if and only if a digital arc S is a DSL [129]:

Let p, q be two points of a digital image subset S. pq^{-} is the continuous line joining p and q. pq^{-} lies near S if for every real point (x, y) on pq^{-}, a point (i, j) exists on S such that $max(|i - x|, |j - y|) < 1$. S has the chord property if for every p, q on S, the chord pq^{-} lies near S.

In AHT [128], authors proposed a non-uniform space quantization, partitioning the space into four sub-spaces, called Analytic Hough Region (AHR). The core concept used in AHT is described originally in [130], that shows HT of a set of points will overlap on a small polygon in one of the four AHRs, if the points are collinear with a set of lines. AHT used slope-intercept parameterization is used to determine a set of points satisfy this property. The Inverse AHT (IAHT) can be then used to convert the overlapping polygons into a pair of convex hulls in image space. If a line passes through all the pixels connected with the parameter space, this line will pass through the two convex hulls. Based on this, IAHT can be used to generate a pair of geometric boundaries in the image space, from the overlapping polygons in the HT space.

The effects of line quantization and the sensitivity of the resulting DHT has been discussed in detail and formulated in [131]. In another evaluation, Analog HTs and different varieties of DHT has been compared by Kiryati et al. [127]. Kiryati et al. has compared the number of accumulators in DHT, determined by digital image resolution, and Analog HT, determined by parameter specifications that ultimately fixes the resolution.

2.7.4 Problem of dimensionality

HT performs mapping of each pixel in HT space, and is therefore a time-consuming process. Moreover, the computation required to calculate HT, increases exponentially with number of curve parameters. The main proposed solution to address high dimensional computation is under divide and conquer category, and either through

calculating HT in sub-images, or parameter space decomposition.

Several authors have used dividing the original image into sub-images [132–138] and applied the divide and conquer technique. The proposed methods range from single level application of HT to the sub-images under the constraint of the curve passing through a set of pixels [137], or multi-level application of HT, first on sub-images, and then evaluating the contribution of each sub-image to the HT space of the original image [135], or parallel running of matching sub-images of template and the image, using master-slave technique [138]. A performance boosting technique that can be applied to all the above sub-image-based techniques is to avoid processing of relatively empty block. One proposed technique is thresholding blocks based on the block gradient magnitudes, and applying HT only to those that pass the threshold [133]. Another technique for example, estimates the contribution of each sub-image to the HT of a specific target region, and avoids calculating HT for the ones with low contribution. Fast HT [139, 140] is another example that thresholds blocks based on the votes associated with each block in contributing to the HT space, with each block divided to sub-blocks to create low to high resolution application of HT, based on the required calculation for a certain special resolution in the final HT.

Parameter space decomposition is the other group of divide-and-conquer methods to overcome the problem of dimensionality. For example, for detecting a curve, instead of using full HT parameter space, Pao et al [112] proposed decomposition of the parament space into translation, rotation and intrinsic spaces, and then start with searching for the matching orientation of the curve in the rotation space, followed by determining intrinsic curve parameters and translation of the curve in the transformed space.

Parameter space decomposition is a popular method for reducing the dimensionality of calculating HT, in detecting circles and ellipses. Particularly, because of the geometric properties in circle and ellipse, geometric constraints can be used to avoid exhaustive search and reduce computational complexity. The parameter space decomposition in these cases usually start with finding the center of circle or ellipse, and

then a guided search for candidate points that satisfy the related geometric equation. For example, in [141,142], the search for center point of the target ellipse is performed using a 2D array, searching for lines joining two set of points with parallel tangent, followed by a 1D array to search for candidate points of the ellipse. This method is further developed in [143] for partially occluded ellipses, where the dimensionality reduction is obtained at the expense of high storage space. In [143], after finding the candidate points in the second stage, the remaining 3 parameters of the ellipse (major and minor axis length and the orientation) are estimated using the following equation:

$$x + by\frac{dy}{dx} + d(x\frac{dy}{dx} + y) = 0$$

Another example of using geometric constraint is presented in [144, 145], where the center of ellipse is found by intersecting two lines, one line, $L1$, crossing two points on the ellipse (with non-parallel tangents), and the second line, $L2$, connecting midpoint of $L1$ and the intersection tangents of the two points on the ellipse. It can be shown that the center of the ellipse lies on $L2$.

To provide a robust detection in the presence of noise and occlusion, the symmetry in the ellipse is usually used as another geometric constraint. Methods such as in [146], tackle the problem of dimensionality by first applying geometric symmetry constraint to group all feature points into different possible ellipses, and then following by searching these groups for sets to satisfy geometric properties of the ellipse to find the remaining 3 parameters.

Detecting circles are usually on the basis of circles being special cases of ellipses. For example the two-staged parameter space decomposition of finding the center first and then other parameters is shown in [147], using two sets of 2D accumulators. The geometric symmetry is also used in [148], applied on gradient pair vectors, and in [149] using the geometric constraint that any chord passes through the circle center.

Other examples of parameter space decomposition are use of mean squares error (MSE) to estimate one parameter at a time [150], dividing the image into sub-images based on convexity of the ellipse shape [151], finding The Fuzzy Cell HT [152] and

Randomized Fuzzy Cell HT [146].

Chapter 3

Normal Motion Analysis in Event-Based Videos

Motional analysis of objects in a video is an important subject in machine vision since it is a useful tool to understand the video and extract the information we are interested in. Therefore there are a large number of studies on different aspects of optical flow in the literature. On the other hand, optical flow extraction is relatively new in event-based videos. In this chapter, we propose a novel framework to understand objects' motion in DVS videos. An algorithm is proposed for calculating flow vectors as well as edge orientations based on principal component analysis of generated events in small spatiotemporal windows. Effects of spatial and temporal quantization are investigated by proposing a model for events generation. Results of this chapter will be a basis for further processing in next chapter.

3.1 Introduction

Optical flow extraction in event-based videos is a relatively new field of researchers' interest. There are two different approaches in the literature. First group of algorithms called Lucas-Kanade (LK) variants try to solve the conventional equation of

the optical flow (Equation 2.1) by estimating derivatives of intensity using generated events. On the other hand, second group of algorithms are based on Local Plane (LP) fits in spatio-temporal space. Our proposed method in this chapter can be considered in the second group. Our goal in proposing the current framework, is to establish an analytical framework to assess the results of optical flow for DVS output streams. Considering the two main categories of estimating the optical flow in DVS systems, the Local Plane Fit (LPF) category has a distinct advantage over the Lukas-Kanade (LK) category. In the LPF category, PCA/SVD can be readily applied. As PCA is a well-established analytical approach, using the LPF category and applying PCA allow us to provide a much deeper analytical framework. In comparison, LK category does not allow direct application of PCA. On the other hand, LPF methods in the literature use Least Mean Square (LMS) which approximates PCA answer with lower computational complexity, although it does not allow deeper analytical assessment of results.

When an object is moving in a video, there are many pixels with varying intensities and they can generate many events accordingly. For a static background and a fixed DVS, these events are mostly limited to object areas and if objects have monotonous textures, they are limited to edge areas. Any event occurring in the background due to the change of the lighting or shading or any event inside the object caused by a discontinuity of object texture is considered as a noise unless a clear edge can be detected in the form of a line or curve. The objective is to find all lines and their properties within small windows to reconstruct the whole boundary of objects.

3.2 Events in spatiotemporal space

Each event has four parameters including the time, x-position, y-position and polarization. The last parameter showing the direction of the intensity change is not important in the edge recognition. Negative events can represent an edge in the same manner as positive events can. As a result, the polarization of events is ignored in this

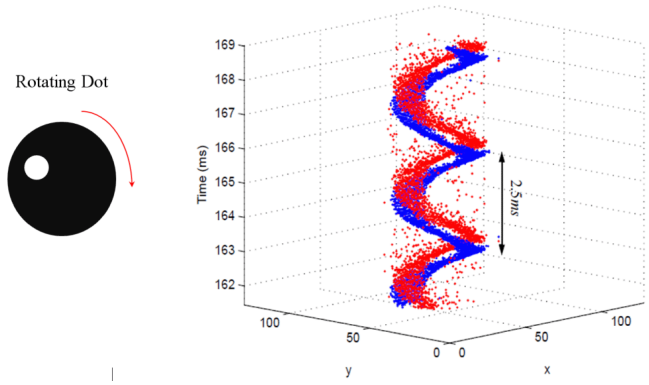


Figure 3.1: DVS output for a rotating black disk with a white dot in spatiotemporal space

study and the other three parameters can be depicted in 3-D spatiotemporal space. Figure 3.1 shows the generated events by a DVS capturing a black rotating disk with a white dot on the disk. As seen in the figure, the time duration of a turn is 2.5 ms because the disk is rotating at the frequency of 400 Hz. The blue points represent the positive events while the red points represent the negative ones [7]. Again it is emphasized that the polarization of events is not important in the edge detection problem.

Small parts of an object boundary can be assumed as linear elements; Meaning that we can consider the object boundary as short moving lines in small spatial windows. The events are generated randomly on these short lines. Knowing that a moving line with a fixed velocity spans a plane in the spatiotemporal space, the events are supposed to be on this plane in any small window on the boundary. The characteristics of this plane represent the edge direction and velocity at the center point of the window.

Figure 3.2 shows a simple straight line while vector $\hat{\mathbf{n}} = [a, b]^T$ is the unit normal vector to this line ($\left\|\hat{\mathbf{n}}\right\|^2 = a^2 + b^2 = 1$). For any point $[x, y]^T$ on this line:

$$ax + by = \begin{bmatrix} a \\ b \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix} = \left\|\hat{\mathbf{n}}\right\| \left\| \begin{bmatrix} x \\ y \end{bmatrix} \right\| \cos(\varphi) = c$$

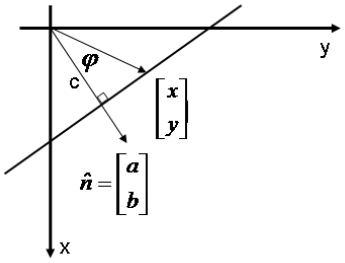
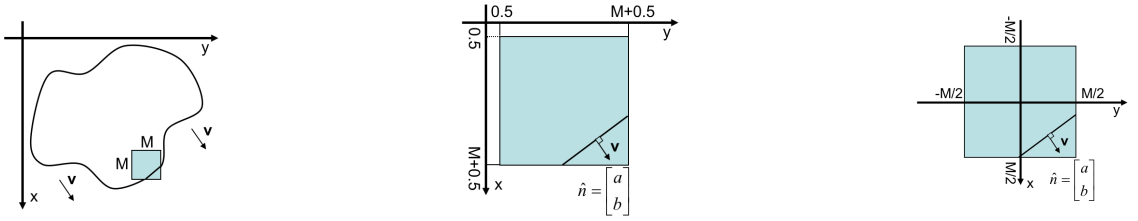


Figure 3.2: The normal distance of every point on the line from the origin is c



(a) An object is moving at velocity $\mathbf{v}$

(b) A part of the edge in a $M \times M$ window

(c) The window after removing bias values

Figure 3.3: A moving shape and a small part of its boundary in a window

Where c is the normal distance of the line from the origin. Assume that the line is moving with a velocity $\mathbf{v}$. Therefore $c = vt + c_0$, giving a general equation of a moving line as $ax + by = vt + c_0$.

As shown in figure 3.3(a), a small $M \times M$ window of the object boundary is chosen where the velocity $\mathbf{v}$ is perpendicular to the edge direction ($\mathbf{v} = v\hat{\mathbf{n}}$). Figure 3.3(b) shows this window where a small part of the boundary is a straight line moving at velocity $\mathbf{v}$ with vector $\hat{\mathbf{n}} = [a, b]^T$ the normal to the edge direction. The events occur randomly on this line at random instances. Removing the bias values of both x and y, results in figure 3.3(c) giving the moving line equation as:

$$ax + by = vt \tag{3.1}$$

3.3 Events analysis in continuous space

In this stage, it is considered that all three parameters, t, x and y are continuous. Let $\mathbf{Q} = [T, X, Y]$ be a random vector with three random variables T, X and Y. Referring to equation (3.1), the relationship among these random variables is given by:

$$T = \frac{aX + bY}{v} \tag{3.2}$$

X and Y are chosen randomly within a window. For square windows, they can be any value in the interval $[\frac{-M}{2}, \frac{M}{2}]$. For circular windows, they can be any value in the interval $[\frac{-M}{2}, \frac{M}{2}]$ with a further constraint that $X^2 + Y^2 \leq (\frac{M}{2})^2$.

Computing the covariance matrix of the random vector $\mathbf{Q}$ (note that $\mathbf{Q}$ is a zero-mean random vector) gives:

$$C = cov(Q) = E\{Q^T Q\} = E\{\begin{bmatrix} T \\ X \\ Y \end{bmatrix} \begin{bmatrix} T & X & Y \end{bmatrix}\} = E\{\begin{bmatrix} T^2 & TX & TY \\ XT & X^2 & XY \\ YT & YX & Y^2 \end{bmatrix}\}$$

Both square and circular windows are symmetric with respect to x and y axes. Therefore X and Y are uncorrelated and independent with the same variances.

$$E\{XY\} = E\{YX\} = 0$$

$$E\{X^2\} = E\{Y^2\} = \sigma^2$$

$$E\{XT\} = E\{TX\} = E\{X\frac{aX + bY}{v}\} = \frac{aE\{X^2\} + bE\{XY\}}{v} = \frac{a\sigma^2}{v}$$

$$E\{YT\} = E\{TY\} = E\{Y\frac{aX+bY}{v}\} = \frac{aE\{YX\}+bE\{Y^2\}}{v} = \frac{b\sigma^2}{v}$$

$$E\{T^2\} = E\{(\frac{aX+bY}{v})^2\} = \frac{a^2E\{X^2\}+b^2E\{Y^2\}+2abE\{XY\}}{v^2}$$

$$E\{T^2\} = \frac{a^2\sigma^2+b^2\sigma^2}{v^2} = (a^2+b^2)\frac{\sigma^2}{v^2} = \frac{\sigma^2}{v^2}$$

$$C = \begin{bmatrix} \frac{\sigma^2}{v^2} & \frac{a\sigma^2}{v} & \frac{b\sigma^2}{v} \\ \frac{a\sigma^2}{v} & \sigma^2 & 0 \\ \frac{b\sigma^2}{v} & 0 & \sigma^2 \end{bmatrix} = \sigma^2 \begin{bmatrix} \frac{1}{v^2} & \frac{a}{v} & \frac{b}{v} \\ \frac{a}{v} & 1 & 0 \\ \frac{b}{v} & 0 & 1 \end{bmatrix} \tag{3.3}$$

Performing PCA on this covariance matrix gives the eigenvalues λ_i and eigenvectors w_i. For a positive definite matrix like a covariance matrix, eigenvectors build an orthogonal basis for data (events here). Eigenvalues represent the distribution (variance) of events along the corresponding eigenvector direction.

$$Cw_i = \lambda_i w_i \ , \ ||w_i|| = 1 \rightarrow |C-\lambda I| = 0 \rightarrow \begin{vmatrix} \frac{\sigma^2}{v^2}-\lambda & \frac{a\sigma^2}{v} & \frac{b\sigma^2}{v} \\ \frac{a\sigma^2}{v} & \sigma^2-\lambda & 0 \\ \frac{b\sigma^2}{v} & 0 & \sigma^2-\lambda \end{vmatrix} = 0$$

$$\lambda(\lambda-\sigma^2)(\lambda-\sigma^2(1+\frac{1}{v^2}))=0 \rightarrow \begin{cases} \lambda_1=0 & \rightarrow \quad w_1=\frac{1}{\sqrt{1+v^2}}\begin{bmatrix} v \\ -a \\ -b \end{bmatrix} \\ \lambda_2=\sigma^2 & \rightarrow \quad w_2=\begin{bmatrix} 0 \\ b \\ -a \end{bmatrix} \\ \lambda_3=\sigma^2(1+\frac{1}{v^2}) & \rightarrow \quad w_3=\frac{1}{\sqrt{1+v^2}}\begin{bmatrix} 1 \\ av \\ bv \end{bmatrix} \end{cases} \tag{3.4}$$

Equations (3.4) indicate that if both time and location parameters are continuous, the covariance matrix will have one zero and two non-zero eigenvalues. The eigenvector corresponding to the middle eigenvalue denotes the edge direction while the edge velocity can be estimated using the eigenvectors corresponding to zero or the largest eigenvalue.

The covariance matrix and its eigenvalues and eigenvectors can be calculated based on the generated events inside a window. As a result it is assumed that three eigenvectors $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ correspond to three eigenvalues $(\lambda_1 = 0 < \lambda_2 < \lambda_3)$ are known and we want to estimate the edge velocity and direction based on these values (w_{ij} denotes the jth element of $\mathbf{w}_i$).

$$\begin{cases} \tan(\theta^*)=\dfrac{w_{23}}{w_{22}}=\dfrac{-a}{b} \\ \mathbf{v}^*=\dfrac{-w_{11}}{w_{12}^2+w_{13}^2}\begin{bmatrix} w_{12} \\ w_{13} \end{bmatrix}=\dfrac{1}{w_{31}}\begin{bmatrix} w_{32} \\ w_{33} \end{bmatrix}=v\begin{bmatrix} a \\ b \end{bmatrix} \\ \hat{\mathbf{n}}^*=\dfrac{-1}{\sqrt{w_{12}^2+w_{13}^2}}\begin{bmatrix} w_{12} \\ w_{13} \end{bmatrix}=\dfrac{1}{\sqrt{w_{32}^2+w_{33}^2}}\begin{bmatrix} w_{32} \\ w_{33} \end{bmatrix}=\begin{bmatrix} a \\ b \end{bmatrix} \end{cases} \tag{3.5}$$

Equations (3.5) state that how the edge direction and velocity can be estimated

based on the covariance matrix of the events in a window. $\mathbf{v}^*$ and θ^* are the estimated velocity and angle between the edge direction and x axis. Moreover $\hat{\mathbf{n}}^*$ denotes the estimated unit normal vector to the edge. The eigenvectors should be normalized such that their first elements become positive ($w_{11} > 0$ and $w_{31} > 0$) and $\mathbf{w}_2$ becomes equal to $\mathbf{w}_3 \times \mathbf{w}_1$.

3.3.1 Events analysis in discrete space

In real cameras, both time and location are discrete, i.e. there is a finite resolution on location and time. For the DVS used in this study, frame resolution is 128×128 pixels (relatively low resolution) and time accuracy is $1\mu s$ (very high accuracy). Normally some quantization error is added when the location or time are quantized. Here a sensible modeling of this quantization error is proposed. Although its verification is beyond of this study and it demands a comprehensive investigation of DVS behavior in different conditions. The purpose of this modeling is just to show we will have a zero-mean noise on the time parameters of the events when the time and location become discrete.

Referring to figure 3.4, line L is moving at velocity $\mathbf{v}$. Let us consider a 1×1 window W centered at pixel O(x,y). This window covers interval $[x - 0.5, x + 0.5] \times [y - 0.5, y + 0.5]$ of the space and $[a, b]^T$ is the unit normal vector to line L.

- At instance $t - \frac{\tau}{2}$, line L approaches F, the first point of window W.
- At instance t, line L arrives at O, the center point of window W.
- At instance $t + \frac{\tau}{2}$, line L passes R, the last point of window W.

Assuming the time parameter is continuous, let us consider an event e occurs at instance $t + \delta$ on line L and inside window W, where the nearest pixel to this event is O(x,y). As a result, this event is reported as $[t + \delta, x, y]^T$ while the relationship between x, y and t is $t = \frac{ax+by}{v}$ based on equation (3.1). The term δ can be any value in interval $[-\frac{\tau}{2}, \frac{\tau}{2}]^T$.

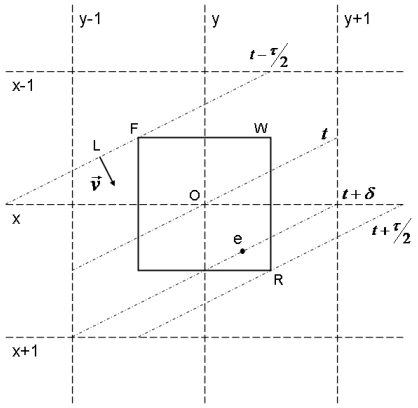


Figure 3.4: The effect of location discretization. $[a, b]^T$ is a unit normal vector to line L

Since $[a, b]^T$ is a unit normal vector to line L ($a^2 + b^2 = 1$) and window W is 1×1, the displacement of line L from instance $t - \frac{\tau}{2}$ to instance $t + \frac{\tau}{2}$ is $|a| + |b|$. The time duration of this displacement is τ and the line velocity is v. Therefore:

$$\tau = \frac{|a| + |b|}{v} \tag{3.6}$$

Based on this modeling, let $\mathbf{Q} = [T, X, Y]$ be a random vector with three random variables T, X and Y. If X and Y are discrete, then:

$$T = \frac{aX + bY}{v} + \delta$$

δ is another zero-mean random variable such that:

$$E\{\delta\} = 0 \qquad -\frac{\tau}{2} < \delta < \frac{\tau}{2}$$

If the time parameter becomes discrete as well, with similar reasoning, the interval for δ will become wider as much as τ^* which is the time accuracy of DVS.

$$E\{\delta\} = 0 \qquad -\frac{\tau+\tau^*}{2} < \delta < \frac{\tau+\tau^*}{2}$$

If $\frac{\tau^*}{\tau} \ll 1$, we can ignore the time accuracy τ^* which is $1\mu s$ for the used DVS in this study. Based on equation (3.6):

$$\frac{\tau^*}{\tau} = \frac{\tau^* v}{|a|+|b|} \leq \tau^* v \ll 1 \rightarrow v \ll \frac{1}{\tau^*} = 10^6 \rightarrow v < 10^5$$

If velocity $v < 10^5$, the temporal accuracy τ^* can be ignored in the calculations; i.e. to ignore the DVS temporal accuracy, a one-pixel displacement of the edge should take a time more than $10\mu s$ which is usually true. The high temporal accuracy of the DVS is such that the time element can be considered as a continuous parameter in most cases.

X and Y have integer values chosen randomly within a window and the values depend on the window shape. The covariance matrix is calculated for $\mathbf{Q}$ which is a zero-mean random vector.

$$C = cov(Q) = E\{Q^T Q\} = E\{\begin{bmatrix} T^2 & TX & TY \\ XT & X^2 & XY \\ YT & YX & Y^2 \end{bmatrix}\}$$

Both square and circular windows are symmetric with respect to x and y axes. Therefore X and Y are uncorrelated and independent random variables with the same variance value σ^2.

$$E\{XY\} = E\{YX\} = 0$$

$$E\{X^2\} = E\{Y^2\} = \sigma^2$$

$$E\{XT\} = E\{TX\} = E\{X(\frac{aX+bY}{v} + \delta)\} = \frac{a\sigma^2}{v}$$

$$E\{YT\} = E\{TY\} = E\{Y(\frac{aX+bY}{v} + \delta)\} = \frac{b\sigma^2}{v}$$

$$E\{T^2\} = E\{(\frac{aX+bY}{v} + \delta)^2\} =$$

$$= \frac{a^2\sigma^2 + b^2\sigma^2}{v^2} + \sigma_d^2 = \frac{\sigma^2}{v^2} + \sigma_d^2$$

$$C = \begin{bmatrix} \frac{\sigma^2}{v^2} + \sigma_d^2 & \frac{a\sigma^2}{v} & \frac{b\sigma^2}{v} \\ \frac{a\sigma^2}{v} & \sigma^2 & 0 \\ \frac{b\sigma^2}{v} & 0 & \sigma^2 \end{bmatrix} = \sigma^2 \begin{bmatrix} \frac{1+\nu}{v^2} & \frac{a}{v} & \frac{b}{v} \\ \frac{a}{v} & 1 & 0 \\ \frac{b}{v} & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$\nu = (\frac{v\sigma_d}{\sigma})^2$$

Comparing matrices in equations (3.3) and (3.7), it is observed that they are the same except in the discrete space there is a small disturbance on the time variance. Referring to the proposed modeling, σ_d is proportional to the inverse velocity and as a result ν is mainly affected by σ. If $\nu = (\frac{v\sigma_d}{\sigma})^2 \ll 1$, it can be ignored more reliably in the calculations and it is possible by increasing σ which is proportional to the window size. On the other hand, assumption of the edge as a series of linear elements is not valid for large windows and there is an optimum window size based on the edge curvature.

The eigenvalues and eigenvectors of matrix (3.7) are calculated as follows.

$$C\mathbf{w}_i = \lambda_i \mathbf{w}_i \ , \ ||\mathbf{w}_i|| = 1$$

$$|\lambda I - C| = (\lambda - \sigma^2)(\lambda^2 - \sigma^2\lambda(1 + \frac{1+\nu}{v^2}) + \frac{\nu\sigma^4}{v^2}) = 0$$

The parameters Δ and ν^* are defined for simplifying the results.

$$\Delta = 1 - \frac{4\nu v^2}{(1+v^2+\nu)^2} \qquad \nu^* = \frac{\nu}{1+v^2}$$

$$\Delta = 1 - \frac{4\nu^*}{(1+\nu^*)^2}\frac{v^2}{1+v^2}$$

Here are the eigenvalues and eigenvectors, both accurate values and first order approximations:

$$\begin{cases} \lambda_1 = \frac{\sigma^2}{2}(1+\frac{1+\nu}{v^2})(1-\sqrt{\Delta}) \approx \sigma^2\nu^* \\ \lambda_2 = \sigma^2 \\ \lambda_3 = \frac{\sigma^2}{2}(1+\frac{1+\nu}{v^2})(1+\sqrt{\Delta}) \approx \sigma^2(1+\frac{1+\nu^*}{v^2}) \end{cases} \tag{3.8}$$

$$\begin{cases} \mathbf{w}_1 = \frac{\begin{bmatrix} v(1-\frac{\lambda_1}{\sigma^2}) \\ -a \\ -b \end{bmatrix}}{\sqrt{v^2(1-\frac{\lambda_1}{\sigma^2})^2+1}} \approx \frac{1}{\sqrt{1+v^2}} \begin{bmatrix} v(1-\frac{\nu^*}{1+v^2}) \\ -a(1+\frac{\nu^*}{1+\frac{1}{v^2}}) \\ -b(1+\frac{\nu^*}{1+\frac{1}{v^2}}) \end{bmatrix} \\ \mathbf{w}_2 = \begin{bmatrix} 0 \\ b \\ -a \end{bmatrix} \\ \mathbf{w}_3 = \frac{\begin{bmatrix} v^2(\frac{\lambda_3}{\sigma^2}-1) \\ av \\ bv \end{bmatrix}}{v\sqrt{v^2(\frac{\lambda_3}{\sigma^2}-1)^2+1}} \approx \frac{1}{\sqrt{1+v^2}} \begin{bmatrix} 1+\nu^*\frac{v^2}{1+v^2} \\ av(1-\frac{\nu^*}{1+v^2}) \\ bv(1-\frac{\nu^*}{1+v^2}) \end{bmatrix} \end{cases} \tag{3.9}$$

Comparing with results (3.4), results (3.8) and (3.9) indicate that if both time and location parameters are discrete, the covariance matrix will have three non-zero eigenvalues. The middle eigenvalue and eigenvector are the same as those in the continuous space. The middle eigenvector is used to estimate the edge direction. Moreover the first and last eigenvalues and eigenvectors in discrete space have extra disturbance terms compared to continuous space.

The covariance matrix and its eigenvalues and eigenvectors can be calculated based on the generated events inside a window. As a result it is assumed that three eigen-

vectors $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ correspond to three eigenvalues $(\lambda_1 < \lambda_2 < \lambda_3)$ are known and similar to continuous space, we want to estimate the edge velocity and direction based on these values (w_{ij} denotes the jth element of $\mathbf{w}_i$).

$$\begin{cases} \tan(\theta^*) = \dfrac{w_{23}}{w_{22}} = \dfrac{-a}{b} \\ \mathbf{v}_1^* = \dfrac{-w_{11}}{w_{12}^2 + w_{13}^2} \begin{bmatrix} w_{12} \\ w_{13} \end{bmatrix} = v(1 - \dfrac{\lambda_1}{\sigma^2}) \begin{bmatrix} a \\ b \end{bmatrix} \approx v(1 - \nu^*) \begin{bmatrix} a \\ b \end{bmatrix} \\ \mathbf{v}_3^* = \dfrac{1}{w_{31}} \begin{bmatrix} w_{32} \\ w_{33} \end{bmatrix} = \dfrac{1}{v(\frac{\lambda_3}{\sigma^2} - 1)} \begin{bmatrix} a \\ b \end{bmatrix} \approx v(1 - \nu^*) \begin{bmatrix} a \\ b \end{bmatrix} \end{cases} \tag{3.10}$$

θ^* is the estimated angle between the edge direction and x axis while $\mathbf{v}_1^*$ and $\mathbf{v}_3^*$ are the estimations of the velocity vectors based on the first and third eigenvectors and they are equal in the first order approximations ($\mathbf{v}_1^* \approx \mathbf{v}_3^* = \mathbf{v}^*$). Both estimations are accurate in the direction, although they are biased in the magnitude. Keeping the first order approximation of disturbance ν, it is concluded that vectors magnitudes are ν^* percent less than the actual value.

$$\frac{\left\| \mathbf{v} - \mathbf{v}^* \right\|}{\left\| \mathbf{v} \right\|} = \nu^* = \frac{\nu}{1 + v^2} \tag{3.11}$$

Equation (3.11) states that decreasing ν (by increasing e.g. window size) results in a better estimation of edge velocity. However window size can be increased only where the edge curvature is low.

3.4 Some points about PCA of the events

In this section, some important details are mentioned about last two sections.

1. The term σ^2 is used as the variance of random variables X and Y within a window.

- For continuous random variables distributed uniformly inside a $M \times M$ square window:

$$\sigma^2 = \int_0^M (x - \bar{X})^2 f_X(x) dx =$$

$$= \int_0^M (x - \frac{M}{2})^2 \frac{1}{M} dx = \frac{M^2}{12}$$

- For continuous random variables distributed uniformly inside a $M \times M$ circular window:

$$\sigma^2 = \int_0^{\frac{M}{2}} \int_0^{2\pi} (x - \bar{X})^2 f_X(x) r d\theta dr =$$

$$= \int_0^{\frac{M}{2}} \int_0^{2\pi} (r\cos\theta)^2 \frac{1}{\pi(\frac{M}{2})^2} r d\theta dr = \frac{M^2}{16}$$

- For discrete random variables distributed uniformly inside a $M \times M$ square window:

$$\sigma^2 = \sum_{k=1}^{M} (k - \bar{k})^2 p(k) =$$

$$= \sum_{k=1}^{M} (k - \frac{M+1}{2})^2 \frac{1}{M} = \frac{M^2 - 1}{12}$$

A reasonable conclusion is that σ is proportional to the window size.

2. As stated before, the events in a small spatio-temporal window are supposed to be on a plane. The objective of the PCA is finding the properties of this events plane.

   - The first eigenvector is normal to the plane and the corresponding eigenvalue shows the deviation of events from the plane. Therefore this eigenvalue illustrates the noise power of the events.
   - The second eigenvector, i.e. the minor axis of the events plane shows the edge direction. This vector is horizontal in the spatiotemporal window, meaning that the time element of this vector is zero. Only this eigenvector

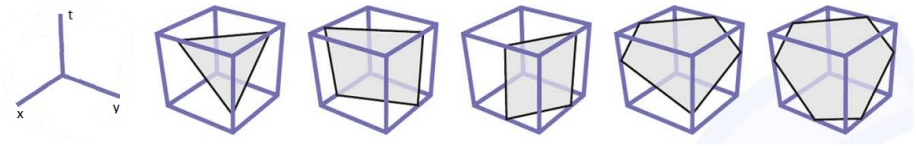


Figure 3.5: Different shapes of the intersection between the events plane and a cubic spatiotemporal window

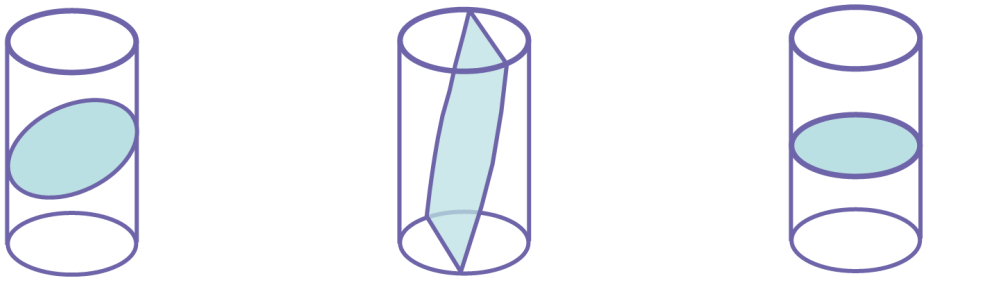
is not affected by the quantization error.

- The third eigenvector, i.e. the major axis of the events plane represents the movement direction of the edge in spatiotemporal window.

3. For the square windowing of the events, they will be inside a cube in the spatiotemporal space. The intersection of the events plane and a cubic window can be different shapes as shown in figure 3.5. Therefore the minor and major axes of these complex shapes of the intersection may be lost. In this case, the second and third eigenvectors will not be valid in the way we explained before. A circular window can somewhat solve this problem.

4. For the circular windowing of the events, they will be inside a cylinder in the spatiotemporal space as seen in figure 3.6. The intersection between the events plane and a cylindrical window is an ellipse as depicted in figure 3.6(a).
   If the edge velocity is low, the slope of the events plane will be high and the intersection may be a cropped ellipse (Figure 3.6(b)). Even in this case the minor and major axes of the events plane are usually valid.
   If the edge velocity is very high, the slope of the events plane will be low and the shape of the intersection changes to a circle. Referring to figure 3.6(c), only in this case the minor and major axes of the events plane may be lost and therefore the second and third eigenvectors will not be valid in the way we explained before.

5. As mentioned before, sometimes the minor and major axes of the events plane are lost specially for square windowing of the events and two last eigenvectors are not valid accordingly. In this case, the first eigenvector is still valid since it



(a) For normal velocity (b) For low velocity (c) For high velocity

Figure 3.6: The intersection between events plane and a cylindrical spatiotemporal window

shows the normal direction to the events plane. In addition the time element of the second eigenvector is zero, i.e. it is horizontal in the spatiotemporal space. Knowing the first eigenvector, the second can be estimated more robustly since there is only one horizontal direction which is perpendicular to the first eigenvector as well.

The eigenvectors of a positive-definite matrix e.g. a covariance matrix are orthonormal and therefore the cross product of the first two eigenvectors is an estimation of the third.

Due to above explanations, the normal vector of the events plane is the most important parameter of that and any other parameter can be calculated based on this normal vector.

3.5 Adaptive estimation of the events plane

The normal vector to the events plane defines all properties of the plane. Let us consider some events in a spatiotemporal window as seen in figure 3.7. All the events are supposed to be on a 3-D plane. Because of the noise, some small deviations from the plane may exist on the events coordinate. We are interested in finding the vector $\mathbf{w}_1$ which is normal to the events plane. This vector corresponds to the first eigenvector in the PCA of the events. Assuming the mean point O of the events

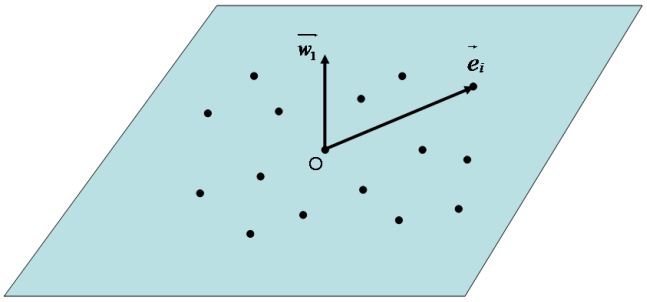


Figure 3.7: Estimating normal vector based on the events

as the origin of the coordinates, this set of events has zero mean value. The events can be considered as some vectors since they have three parameters t, x and y. The mapping of any event $\mathbf{e}_i$ within a window on the vector $\mathbf{w}_1$ should be zero or a small value since $\mathbf{e}_i \perp \mathbf{w}_1$.

This problem can be considered as an adaptive filtering problem. Referring to figure 3.8, event $\mathbf{e}_i$ is the input of an adaptive filter with characteristic vector $\mathbf{w}$. The objective of adaptive filtering problem is finally finding $\mathbf{w} = \mathbf{w}_1$. The output of this filter, $d_i^* = \mathbf{w}.\mathbf{e}_i = \mathbf{w}^T\mathbf{e}_i$, is supposed to be zero or a small value when $\mathbf{w}$ approaches $\mathbf{w}_1$ and $d_i = 0$ is the desired output. The output d_i^* is compared with desired value d_i and then error value $er_i = d_i - d_i^*$ is calculated.

The variance of er is minimized as its power value to tune the filter $\mathbf{w}$ and approach $\mathbf{w}_1$.

$$E\{er\} = E\{(d - d^*)\} = E\{-\mathbf{w}^T\mathbf{e}\} = -\mathbf{w}^T E\{\mathbf{e}\} = 0$$

$$var(er) = E\{(er - E\{er\})^2\} = E\{(-d^*)^2\} = E\{d^* d^{*T}\} = E\{\mathbf{w}^T\mathbf{e}\mathbf{e}^T\mathbf{w}\}$$

$$var(er) = \mathbf{w}^T E\{\mathbf{e}\mathbf{e}^T\}\mathbf{w} = \mathbf{w}^T R_e \mathbf{w}$$

$$\mathbf{w}_1 = \arg\min_{\mathbf{w}}\{var(er)\} = \arg\min_{\mathbf{w}}\{\mathbf{w}^T R_e \mathbf{w}\} \qquad \|\mathbf{w}\|^2 = 1$$

(R_e is the covariance matrix of the events)

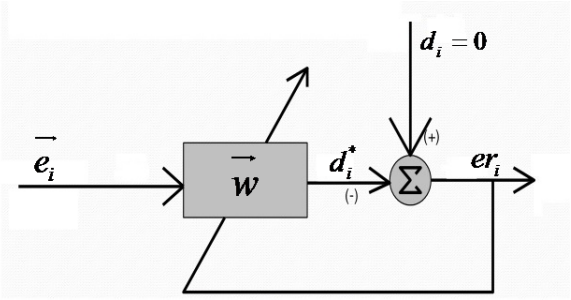


Figure 3.8: Problem of finding first eigenvector as an adaptive filtering problem

The exact solution of this minimization problem leads to the PCA of the events as discussed before. We use the steepest descent algorithm to minimize the above expression iteratively. $\mathbf{w}$ should move against the gradient vector of the expression to approach the minimum point $\mathbf{w}_1$ gradually.

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \mu\nabla(\mathbf{w}^T R_e \mathbf{w}) = \mathbf{w}_{old} - 2\mu R_e \mathbf{w} = \mathbf{w}_{old} - 2\mu E\{\mathbf{e}\mathbf{e}^T\}\mathbf{w}$$

The LMS algorithm suggests that we can substitute $E\{\mathbf{e}\mathbf{e}^T\}$ with $\mathbf{e}_i\mathbf{e}_i^T$ [153].

$$\mathbf{w}_{new} = \mathbf{w}_{old} - 2\mu\mathbf{e}_i\mathbf{e}_i^T\mathbf{w} = \mathbf{w}_{old} - 2\mu d_i^*\mathbf{e}_i \tag{3.12}$$

The update equation 3.12 for $\mathbf{w}$ is the final LMS algorithm. μ is the step size and defines the convergence rate. It is proven that the convergence is guaranteed by $\mu < \frac{1}{\lambda_{max}}$. Moreover the maximum convergence speed is achieved when $\mu = \frac{1}{\lambda_{min}+\lambda_{max}}$ [153]. λ_{min} and λ_{max} are the smallest and largest eigenvalues of the events covariance matrix R_e. In every step, $\mathbf{w}$ should be normalized after updating its value. The norm of $\mathbf{w}$ should remain one in all stages since without this constraint, the obvious solution for the minimization problem is $\mathbf{w} = 0$.

A similar algorithm has been introduced with a different explanation in [78]. It is equivalent to LMS algorithm except that in [78] the events within a spatiotemporal window are iteratively used many times during many epochs in the steepest descent

algorithm. Moreover, there is a noise suppression mechanism that whenever an event is far from the events plane, it is considered as noise and ignored.

The LMS algorithm is online and computationally efficient. PCA algorithm should wait for some events while $\mathbf{w}$ is updated based on LMS algorithm whenever an event occurs in the spatiotemporal window. PCA is more computationally complex than LMS since calculating eigenvalues and eigenvectors demands a high processing power. An important issue in LMS is defining the step size μ. Large μ has less inertia to the last events and subsequently converges fast. However, it may have more error on the final convergence point compared to small μ. There are some optimized methods that initialize μ with a large value and then reduce it gradually by approaching the minimum point [153]. As a result, a fast and accurate convergence is achieved by using large and small step sizes during the process.

3.6 Implementation

Algorithm 1 Normal motion analysis of an event-based video

1. **initialization:**
   $frame \leftarrow$ All events within interval $[T - \frac{\Delta T(i,j)}{2}, T + \frac{\Delta T(i,j)}{2}]$
   $M \leftarrow$ Window size
   $P \leftarrow$ Maximum size of noise blocks
2. **Cleaning:**
   Remove any $isolated - part$ from $frame$ if $area \leq P$
3. **for** each non-zero $pixel(i, j)$ of $frame$ **do**
   Get $M \times M$ window of events centered at $pixel(i, j)$
   Perform PCA of the events and obtain eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$
   Calculate velocity $\mathbf{v}(i, j)$ of $pixel(i, j)$ based on the eigenvectors
   end for

Let us analyze an event-based video at the time instance T (Algorithm 1). All

events are captured within the time interval $[T - \frac{\Delta T}{2}, T + \frac{\Delta T}{2}]$. ΔT can be thought of as the time of a frame in the conventional videos. There are some small isolated parts which are most likely noise. All of these parts smaller than a predefined number of pixels e.g. P=4 are removed. PCA of the events is performed to obtain the event plane and subsequently edge directions and velocities.

3.7 Experiment I

Bodo Rueckauer and Tobi Delbruck have recently analyzed the existing methods of event-based normal flow extraction on 5 different videos including 2 synthetic videos and 3 real streams [97]. They have used the relative AEE (Average Endpoint Error) and AAE (Average Angular Error) for comparing the results with the ground truth. We have also applied our algorithm on these videos which are available online. We just consider the edge pixels with a displacement at least 1/4 of the window size during the accumulation period. The accumulation time is assumed 100ms for "rotDisk" (fast), 1000ms for "RotatingBar" (slow) and 400ms for other objects while the window size is considered 16 for real videos (with larger objects) and 8 for synthetic videos (with smaller objects). The AEE_{rel} and AAE results are shown in tables 3.1 and 3.2. Comparing our method with 9 different methods in the literature, our method has good result especially on detecting the flow direction. We use PCA for finding the edge information and then estimate the flow direction which is presumably perpendicular to the edge. Moreover we calculate the exact direction rather than mapping the edge on some discrete directions. LK variants display higher error on synthetic videos while LP variants including PCA result in poor accuracy on "translSin" which has no clear edge. Moreover LP variants show higher error on rotational objects (where the edge plane is not flat) except PCA which uses circular windowing. When the far events from the edge plane are not removed, a corner point can be also a challenging problem as observed in "translSqure" video even though it is noise-free.

The exact complexity of the method can be hardly obtained since it is highly

dependent on many parameters e.g. window size, accumulation time and especially the captured video. If there is enough memory, the algorithm can be implemented such that the window shape does not influence the processing power demand. The video "translBoxes" is used for investigating the algorithm complexity. If there are N points inside a spatiotemporal window, $6(2N-1)$ floating point operations (flops) are needed for calculating the covariance matrix. The experiment shows that we need at least 1.8K flops per event for calculating the covariance matrices in "translBoxes". This is almost fixed even if the accumulation time changes. By decreasing the accumulation time, the number of covariance matrices increases which affects the total complexity of the algorithm. In this experiment, we need to calculate eigenvectors about $60K$ times. Each time needs $O(n^3)$ around 40 flops. Therefore, we need $2.4M$ flops in total. Since the total number of events is $220K$, 10 flops per event are needed by this algorithm. This number can be increased significantly if the accumulation time is decreased. In summary, the processing demand of this algorithm is around $2K$ flops per event. As a result, the algorithm can be run in real time only for videos of very simple shapes like a line.

3.8 Conclusion

In this chapter, we exploited the principal components analysis of the events to extract the events plane properties in 3-D spatiotemporal space. We investigated the time and location discretization effects on the final results and showed that the spatial quantization error is more important than the temporal quantization error. The problem was also defined as an adaptive filtering problem and its solution was discussed. We will use the results of this chapter to find objects global velocity and extract edge information in next chapter.

Table 3.1: Relative Average Endpoint Error in percent and its standard deviation for 9 existing methods and our proposed method (PCA)

$AEE_{rel}[\%]$	translSqu	rotBar	translSin	rotDisk	translBoxes
LK_{BD}	123.43± 11.04	414.95± 767.50	57.85± 16.37	76.65± 31.87	116.95± 15.57
LK_{CD1}	12.94± 36.16	197.47± 386.69	57.16± 25.55	54.46± 34.89	58.13± 28.86
LK_{CD2}	32.87± 24.43	183.69± 276.91	37.54± 21.52	64.53± 24.52	72.54± 27.70
LK_{SG}	65.08± 21.08	326.77± 253.31	32.50± 26.98	49.83± 27.64	35.62± 24.68
LP_{orig}	0.00± 0.00	175.08± 460.93	62.82± 48.67	60.71± 61.76	37.93± 35.15
LP_{robust}	0.00± 0.00	91.61± 278.97	59.45± 37.49	50.57± 37.08	36.13± 27.80
LP_{SF}	6.41± 15.13	39.08± 63.84	69.62± 33.17	58.99± 37.87	33.49± 18.01
LP_{SG}	0.00± 0.00	114.56± 341.18	54.41± 51.27	78.02± 281.03	66.74± 44.78
DS	1.06± 8.99	40.37± 54.65	62.92± 60.50	44.18± 34.91	30.73± 34.06
PCA	7.57± 17.22	3.89± 1.91	33.61± 20.21	18.03± 9.27	18.68± 11.97

Table 3.2: Average Angular Error and its standard deviation for 9 existing methods and our proposed method (PCA)

$AAE[\circ]$	translSqu	rotBar	translSin	rotDisk	translBoxes
LK_{BD}	135.77± 31.45	73.24± 56.52	20.35± 16.46	51.71± 45.47	108.07± 28.67
LK_{CD1}	7.05± 20.05	29.68± 20.54	21.72± 35.31	19.93± 21.35	28.98± 31.27
LK_{CD2}	9.38± 19.31	32.85± 21.17	13.33± 15.18	18.72± 19.00	36.07± 36.69
LK_{SG}	11.48± 8.80	16.75± 8.26	9.56± 15.90	6.37± 6.53	12.13± 19.84
LP_{orig}	0.00± 0.00	17.54± 21.56	38.93± 61.85	28.06± 33.39	9.78± 32.96
LP_{robust}	0.00± 0.00	9.56± 29.97	37.72± 55.76	22.30± 32.70	9.46± 22.67
LP_{SF}	2.39± 8.98	6.81± 24.34	43.99± 48.52	23.39± 32.12	9.27± 13.11
LP_{SG}	0.00± 0.00	8.72± 17.92	28.39± 42.78	18.77± 31.74	13.96± 23.62
DS	0.63± 5.28	6.88± 7.18	32.82± 56.67	16.62± 20.36	12.01± 31.91
PCA	3.71± 8.56	1.24± 0.81	8.20± 21.77	6.54± 4.55	6.39± 7.50

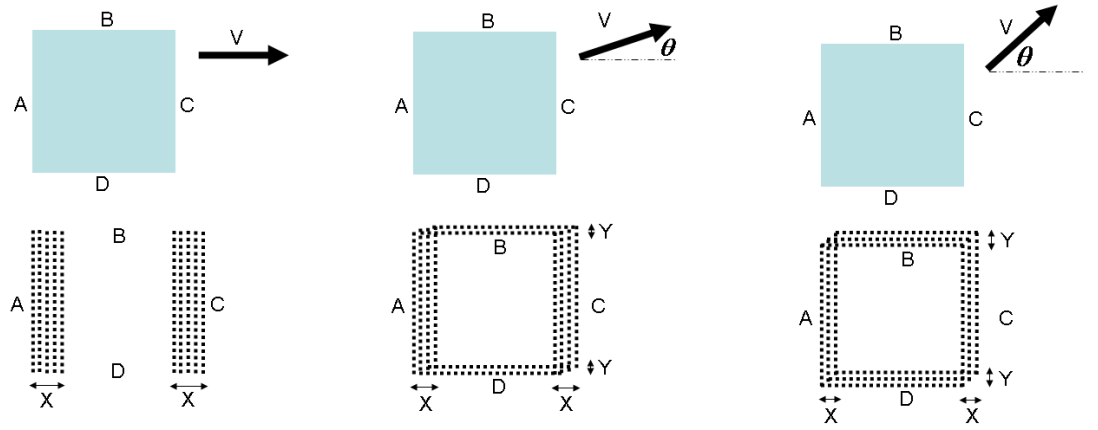
Chapter 4

Global Velocity and Edge Estimation in Event-Based Videos

In this chapter, the global velocity of objects and its relation with normal motion flow are investigated. Then we use the normal flow vectors calculated previously to obtain the objects' global velocity and extract the edge information which is an important feature in conventional machine vision.

4.1 Introduction

Normally in a video, there are many points moving at different velocities. In a DVS video, these velocities can be estimated only by analyzing the generated events around the points. If an object with a uniform texture is moving in a DVS video, for a point inside this object, there is no generated event and as a result no velocity is calculated although it is moving at the same velocity with the object. If a point is located on the edge, most likely we will have some events and the velocity can be estimated based on these events. Even for a point on the edge, the events rate depends on many factors; One of them is the edge velocity. If an edge is moving fast, a burst of events occurs in that area. For a slow moving edge, less events are



(a) For horizontal velocity (b) For a velocity at $\theta < 45°$ (c) For a velocity at $\theta = 45°$

Figure 4.1: The effect of velocity in different directions on a simple square

generated and for a static edge, there is no event except some noises.

Figure 4.1 shows a simple square moving at velocity **V** in different directions and all the events are captured within a time duration of T.

In figure 4.1(a), the object is moving horizontally. Although the edges B/D are moving at the same velocity as that of the edges A/C, they are not visible in the captured video by DVS. The width of shaded area around edges A/C is about $X = VT$. The difference between A/C and B/D is that A/C are normal to velocity V in contrast to B/D which are parallel.

In figure 4.1(b), the object is moving along the direction $\theta < 45°$ and all the edges are visible. However the width of the shaded areas on A/C and B/D are different ($X = VT\cos\theta > Y = VT\sin\theta$). As a result, the estimation is that edges A/D are moving at velocity $V\cos\theta$ and edges B/C are moving at velocity $V\sin\theta$ although their actual velocities are **V**.

In figure 4.1(c), the object is moving along the direction $\theta = 45°$. All the edges are visible and the width of shaded areas on A/C and B/D are the same ($X = VT\cos(45°)$ and $Y = VT\sin(45°)$). Therefore the estimation is that all the edges are moving at velocity $V\frac{\sqrt{2}}{2}$ although their actual velocities are **V**.

The problem highlighted above for DVS videos is equivalent to the well-known aperture problem in the conventional image processing. Here, this problem is more important since the DVS is sensitive only to the intensity variation and it does not capture any other detail. Let us assume an infinite line in a space with a pure tangential movement. A normal camera can capture this line, although it cannot capture the movement of this line according to the aperture problem. On the other hand, a DVS cannot capture neither this line nor its movement since the events are generated just in case that a normal displacement exists. In other words, this line is completely invisible in a DVS video. The proposed algorithms extract the velocity based on the normal displacement of an edge. As a result, the estimated velocity by e.g. PCA is only the normal component of the velocity, not the actual value. If there is a movement along an edge, it cannot be recognized based on only the events in that area, instead the whole object should be considered. Based on this idea, we propose another algorithm for calculating the global velocity using normal velocity components obtained by PCA algorithm. Compared to the conventional optical flow problem, PCA equation is equivalent to the brightness constancy assumption (BCA). The next section proposes second necessary equation for calculating the global velocity.

4.2 Global velocity

The normal flow vectors were calculated in previous chapter and we discussed the global velocity effect on the normal flow vectors. Here we calculate a solid object's global velocity based on the normal motion flow. Referring to figure 4.2, let us consider a solid object L moving at velocity **U** which is perpendicular to the optical axis of the DVS. In addition, object L does not have any rotational displacement at all. Let P_i be one point of the edge and x be the vertical axis. Under these assumptions, P_i will also have the same velocity **U**.

- $\mathbf{E}_i$ is the edge direction at P_i which is along the second eigenvector ($\mathbf{w}_{2i}$) in PCA.

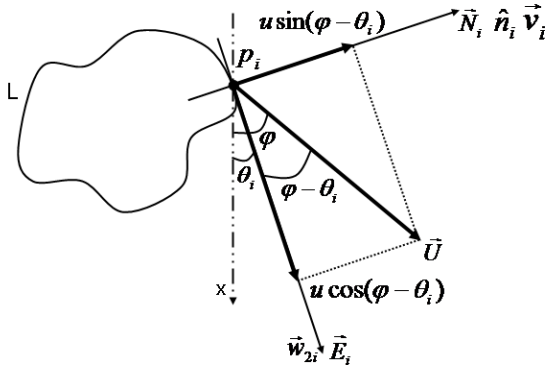


Figure 4.2: A solid object is moving at velocity **U** which is estimated based on the normal velocities of the edge.

- $\mathbf{N}_i$ is the normal direction to the edge at P_i. Its direction corresponds to the edge normal vector ($\hat{\mathbf{n}}_i$) at P_i.
- θ_i is the angle between edge direction $\mathbf{E}_i$ and x axis which is obtained by the PCA of the events.
- φ is the angle between global velocity **U** and x axis.
- $\mathbf{v}_i$ is the estimated velocity by PCA at P_i which is normal to edge ($\mathbf{v}_i = v_i\hat{\mathbf{n}}_i$).
- $\varphi - \theta_i$ is the angle between global velocity **U** and edge direction $\mathbf{E}_i$.

Global velocity **U** is mapped on two directions $\mathbf{E}_i$ and $\mathbf{N}_i$.

- $u\cos(\varphi - \theta_i)$ is the mapping of the global velocity on the edge direction.
- $\mathbf{U}^T\hat{\mathbf{n}}_i = u\sin(\varphi - \theta_i)$ is the mapping of the global velocity on the normal direction to the edge.

In this problem all the parameters are known except global velocity **U** including magnitude u and direction φ. v_i is obtained from PCA of the events as an estimation of the normal velocity to the edge. In fact v_i can be considered as an estimation of $\mathbf{U}^T\hat{\mathbf{n}}_i = u\sin(\varphi - \theta_i)$. As a result, the difference $|v_i - \mathbf{U}^T\hat{\mathbf{n}}_i|$ should be a small

value and the summation of squared differences over all the points on the object edge should be minimized as well. Let define $f(\mathbf{U})$ as a cost function for this problem.

$$f(U) = \sum_i (v_i - \mathbf{U}^T \hat{\mathbf{n}}_i)^2 = \sum_i (v_i - \mathbf{U}^T \hat{\mathbf{n}}_i)(v_i - \hat{\mathbf{n}}_i^T \mathbf{U}) =$$

$$= \sum_i (v_i^2 - 2v_i \hat{\mathbf{n}}_i^T \mathbf{U} + \mathbf{U}^T \hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^T \mathbf{U})$$

At the minimum point, the gradient vector of $f(U)$ should be zero.

$$\nabla f = \sum_i (-2v_i \hat{\mathbf{n}}_i + 2\hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^T \mathbf{U})$$

$$\nabla f = 0 \rightarrow \sum_i (\hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^T)\mathbf{U} = \sum_i v_i \hat{\mathbf{n}}_i = \sum_i \mathbf{v}_i$$

By defining the autocorrelation matrix $R_{\hat{\mathbf{n}}}$ of the edges normal vectors and the mean estimated velocity vector $\mathbf{V}$ for N points which are moving simultaneously at the same global velocity $\mathbf{U}$, the equation will be:

$$R_{\hat{\mathbf{n}}} = \frac{1}{N} \sum_i^N \hat{\mathbf{n}}_i \hat{\mathbf{n}}_i^T =$$

$$= \frac{1}{N} \begin{bmatrix} \sum_i^N \sin^2(\theta_i) & -\sum_i^N \sin(\theta_i)\cos(\theta_i) \\ -\sum_i^N \sin(\theta_i)\cos(\theta_i) & \sum_i^N \cos^2(\theta_i) \end{bmatrix}$$

$$\mathbf{V} = \frac{1}{N} \sum_{i=1}^N \mathbf{v}_i$$

$$R_{\hat{\mathbf{n}}}\mathbf{U} = \mathbf{V} \rightarrow \mathbf{U} = R_{\hat{\mathbf{n}}}^{-1}\mathbf{V}$$

If several moving objects exist with different global velocities, this algorithm can be applied on any object or any connected part of the edge separately. Even for an articulated object, this algorithm is locally applicable since the actual velocity is usually smooth.

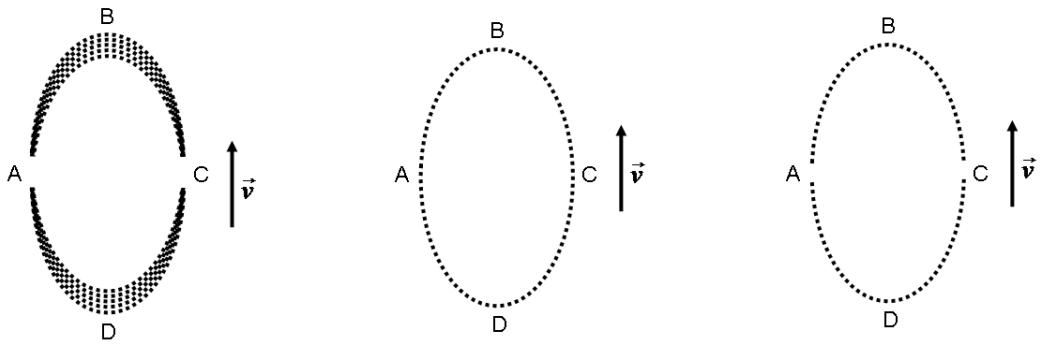
4.3 Edge detection

Captured videos by DVS have some intrinsic differences with the conventional videos. In a normal camera, the shutter speed is used to control the duration of the exposure and is determined by the speed of the object to be imaged. For faster moving objects, the shutter speed should be higher. Otherwise, the object will appear blurred. However, this issue is worse for a DVS since a DVS captures only the intensity variation. If one part of an object does not have enough movement, it will disappear from the captured video. If we wait for that part to have enough movement, the other faster parts will be blurred. In summary, capturing the events for a short time leads to narrow edges in some parts and disconnected edges in other parts. On the other hand, long capturing time results in connected edges in some parts and blurred edges in other parts. The objective of edge detection in DVS is to capture narrow connected edges everywhere rather than blurred or detached edges to reflect more accurately the edge of the object. In other words, edge is the mean of edge pixels collected during a time window.

Figure 4.3 shows a simple ellipse moving up at velocity $\mathbf{v}$. The events are captured for a specific time duration. As shown in figure 4.3(a), for this time duration, there are some blurred areas (B/D) and some disconnected areas (A/C) which are not desirable. Figure 4.3(b) shows the ideal case without any blurred or disconnected region which is the aim of this section.

4.3.1 Removing shaded area

A key advantage of a DVS compared to a conventional camera is that the "exposure time" of a DVS can be locally adjusted for different regions after data acquisition and not predetermined during image acquisition. After the pixel-wise normal velocity is obtained, the image is divided into sub-images and the mean value of pixel-wise velocity in each sub-image is assigned as the shutter speed for that region. The exposure time is calculated such that there is at least one-pixel movement based on



(a) Captured events in a specific time interval

(b) The ideal goal of edge detection

(c) The result of our proposed algorithm

Figure 4.3: A simple ellipse is moving up at velocity $\vec{v}$

the assigned shutter speed.

This procedure removes the blurred areas efficiently. However in practice we have to define an upper bound for the exposure time. The edges without enough displacement within the maximum exposure time will disappear in the captured frame. This can be solved by applying a filter to the captured frame from the DVS.

4.3.2 Edge-dependent Gaussian filtering

In order to reduce the disconnected edge area, a desired property of the filter is to extend or connect the gap along the orientation of the edge. An edge-dependent Gaussian filter meets such a requirement. The parameters of this Gaussian filter should be determined locally based on the edge properties in every pixel to extract distinct edges in the frame. The overall result of these procedures could be as shown in figure 4.3(c) where the blurred areas B/D are totally removed while the disconnected areas A/C are reduced.

The following equation describes a 2-D Gaussian filter in the general form.

$$g(x, y; \theta, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2})$$

$$x' = x\cos(\theta) + y\sin(\theta) \quad , \quad y' = -x\sin(\theta) + y\cos(\theta)$$

Figure 4.4 represents different Gaussian filters. Figure 4.4(a) is a radial Gaussian filter with standard deviation σ. It is scaled by factor $\gamma \geq 0$ along y axis and then rotated by angle θ. We need a narrow filter along the edge which is achieved by setting the scaling factor $\gamma \to \infty$ (figure 4.4(b)). If $\gamma \to \infty$, y' should be close to zero. Otherwise $\gamma^2 y'^2$ becomes infinity and the filter output will be zero. Moreover θ is the filter direction defined by the edge direction. For simplicity, this filter is applied at four major directions $\theta = 0°, 45°, 90°, 135°$ on the frame. The final result is chosen pixel-wise from the outputs of these four filters based on the edge directions.

Equation (4.1) describes the edge-dependent Gaussian filter (figure 4.4(c)) used in this section.

$$g(x,y) = \exp(-\frac{x'^2}{2\sigma^2}) \ , \ x' = x\cos(\theta) + y\sin(\theta) \tag{4.1}$$

$$y' = 0 \to -x\sin(\theta) + y\cos(\theta) = 0 \to y = x\tan(\theta)$$

Referring to the above equations, filter $g(x,y)$ has non-zero values only on a line making angle θ with x axis. It is similar to a 1-D Gaussian filter rotated by angle θ and placed in 2-D space. σ controls the blurring intensity along the edges. Larger σ extends the edges and fills the gaps more effectively. However it is limited by the edge curvature such that for edges with larger curvature, a Gaussian filter with smaller σ is needed. $g(x,y)$ is a special case of Gabor filters with no modulation. Gabor filters can be used for directional features extraction from the DVS output [58].

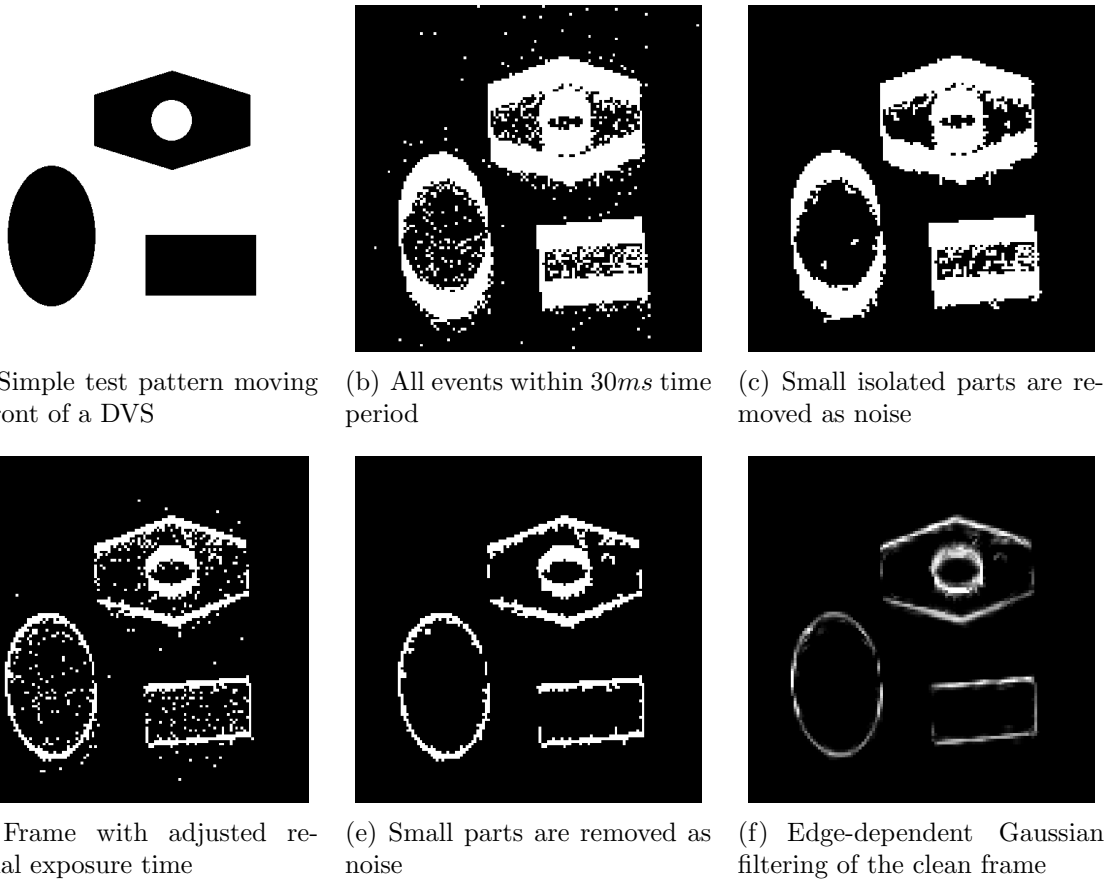
4.4 Implementation

In this section, the full procedure is described in details. Figure 4.5(a) shows a test pattern in black printed on a white paper. A DVS captures the movement of this paper and the output events are used to represent the result of every stage.

1. Let us consider that we want to analyze this video at instance T. All events

(a) Radial Gaussian (b) Narrow Gaussian (c) Edge-dependent Gaussian

Figure 4.4: Different kernels of Gaussian filter



(a) Simple test pattern moving in front of a DVS

(b) All events within $30ms$ time period

(c) Small isolated parts are removed as noise

(d) Frame with adjusted regional exposure time

(e) Small parts are removed as noise

(f) Edge-dependent Gaussian filtering of the clean frame

Figure 4.5: Results of algorithms on a test video

are captured within time interval $[T - \frac{\Delta T}{2}, T + \frac{\Delta T}{2}]$. ΔT equals roughly to the time of a frame in conventional videos (algorithm 2). Figure 4.5(b) represents all events in $30ms$ time period. There are some small isolated parts which are most likely noise. All of these parts smaller than a predefined number of pixels P are removed. Figure 4.5(c) represents the frame after removing the small isolated parts. The procedure of this stage is proposed in algorithm 3 which is run with $P = 4$ and $S = 1$.

Algorithm 2 Constructing a frame (preparation)

(a) **initialization:**
$T \leftarrow$ Instance of the video at which it is analyzed
$S \leftarrow$ Number of sub-images in the height or width of the image
$\Delta T \leftarrow S \times S$ matrix for time durations of all sub-images

(b) **for** each $sub - image(i, j)$ **do**
Save all events within interval $[T - \frac{\Delta T(i,j)}{2}, T + \frac{\Delta T(i,j)}{2}]$
end for

(c) Construct $frame$ by merging all sub-images

Algorithm 3 Removing small isolated parts (noise suppression)

(a) **initialization:**
$P \leftarrow$ Maximum size of noise blocks
$frame \leftarrow$ Result of algorithm 2

(b) **for** each $isolated - part$ of $frame$ **do**
if $area \leq P$ **then**
Remove $isolated - part$ from $frame$
end if
end for

(c) Return $frame$

2. PCA of the events is performed as described in algorithm 4 and the events plane is recognized. For noisy videos, we can ignore the events far from the plane and apply PCA again to obtain more accurate eigenvectors and edges velocities and

directions subsequently.

Algorithm 4 Performing PCA of the events (motion analysis-noise suppression)

(a) **initialization:**
$M \leftarrow$ Window size
$frame \leftarrow$ Result of algorithm 3

(b) **for** each non-zero $pixel(i, j)$ of $frame$ **do**
Get $M \times M$ window of events centered at $pixel(i, j)$
Perform PCA of the events and obtain eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$
for each *event* in *window* **do**
if $event.\mathbf{w}_1 > threshold$ **then**
Remove *event* from $frame$ and *window*
end if
end for
Perform PCA of the left events again and obtain eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$
Calculate velocity $\mathbf{v}(i, j)$ of $pixel(i, j)$ based on the eigenvectors
end for

(c) Return $frame, \mathbf{v}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$

3. In this stage, regional exposure time is calculated (algorithm 5) based on edge velocities obtained in the last stage. Then algorithm 2 is applied again to remove shaded areas. As shown in figure 4.5(d), e.g. the thickness of ellipse is the same everywhere compared to figure 4.5(b). Again algorithm 3 is used to remove small parts which are most likely noise (figure 4.5(e)).

4. Using the normal velocities of the boundary, algorithm 6 is applied to calculate the global velocity. Finally the edge-dependent Gaussian filter is exploited to extend disconnected edges and reduce the gaps as much as possible (algorithm 7). Moreover this filter has a noise suppression effect since any pixel not along the edges is faded by this filter while the edge points are magnified (figure 4.5(f)). In addition, the non-maximal suppression algorithm can be applied on the final result to guarantee one-pixel thickness of the edges [154].

Algorithm 5 Adjusting regional exposure times (edge detection)

(a) **initialization:**
$S \leftarrow$ Number of sub-images in the height or width of the image
$\mathbf{v} \leftarrow$ Result of algorithms 4

(b) **for** each $sub-image(i,j)$ **do**
Calculate shutter speed $V(i,j)$ which is the mean of non-zero velocities $|\mathbf{v}|$ within $sub-image(i,j)$
Calculate exposure time $\Delta T(i,j) = \frac{1}{V(i,j)}$ which is needed for one-pixel displacement
end for

(c) Return ΔT

Algorithm 6 Estimating the global velocity (motion analysis)

(a) **initialization:**
$\mathbf{v}, \mathbf{w}_3 \leftarrow$ Results of algorithm 4
$frame \leftarrow$ Result of algorithm 3

(b) **for** each $sub-edge(k)$ from $frame$ **do**
Calculate $R = \frac{1}{N}\sum_{i=1}^{N} w_3^T w_3$ and $\mathbf{V} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}$
Calculate global velocity $\mathbf{U} = R^{-1}\mathbf{V}$
end for

(c) Return $frame, \mathbf{U}$

Algorithm 7 Edge-dependent Gaussian filtering of frames (edge detection)

(a) **initialization:**
$\sigma \leftarrow$ Variance of the Gaussian filter
$\mathbf{w}_2 \leftarrow$ Result of algorithms 4
$frame \leftarrow$ Result of algorithm 6

(b) **for** each non-zero $pixel(i,j)$ in $frame$ **do**
Calculate the filter mask based on σ and $\mathbf{w}_2(i,j)$
Center the mask at $pixel(i,j)$ and calculate its new value
end for

(c) Return $frame$

4.5 Experiment II

In this experiment, an artificial DVS video of a moving circle is created by a computer as shown in figure 4.6. Using artificial video has several advantages for our purpose of testing the proposed algorithm. The main advantages are:

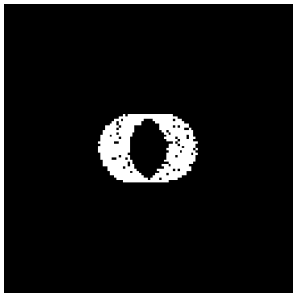
1. In this simulation of DVS output, we can obtain the exact ground truth, to compare with algorithm outputs.

2. The simulated output can be used to control the noise of the input stream, from a noise free baseline, to different levels of noise. This can be used to obtain a baseline for proposed algorithms.

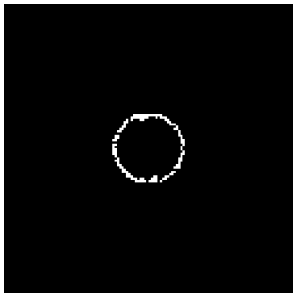
3. By changing parameters of the simulated output, one can easily simulate different hardware in terms of spatial and temporal resolutions.

Let us consider a simple circle with radius 15 pixels moving horizontally in the video. The spatial and temporal resolution of this virtual DVS is 128×128 pixels and $1\mu s$. At instance $t = 0$, the circle center is located at $(x, y) = (64.5, 0.5)$. It travels the frame width in $1s$. Therefore at instance $t = 1$, the circle center approaches $(x, y) = (64.5, 128.5)$. In this duration, 20000 events are generated randomly on the circle boundary. The occurrence rate of the events is proportional to the edge normal velocity. As a result, there is no event at the highest or lowest points of the circle similar to a real DVS video. The circle position and motion are analyzed at $t = 0.5s$ when the exact ground truths (GTs) of the estimated values are known. The proposed algorithm is applied with different circular window sizes $M = 4, 5, 6, \ldots, 15$ and some pixels are recognized as the edge pixels. For these edge pixels:

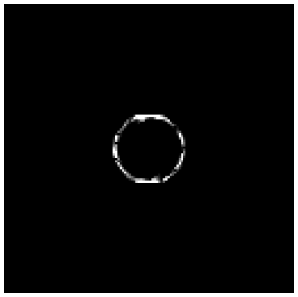
Figure 4.7 shows the estimated velocity magnitudes for the edge pixels. We explained before that the effect of quantization error is a biased estimation of the velocity. Also we showed that the estimated velocity is always less than the actual value. Knowing the quantization error, we are able to compensate its effect. This is a subject for our future work.



(a) All the events in time duration $50ms$ centered at $t = 0.5s$



(b) All the events in adjusted exposure time durations



(c) The result of the edge-dependent Gaussian filtering

Figure 4.6: A horizontally left to right moving circle with radius 15 pixels in an artificial 128×128 DVS video

Figure 4.8 shows the estimated velocity direction for the edge pixels. We have only about 0.02 radian or 1° error in this part.

Figure 4.9 shows the estimated radius for the edge pixels. This results is also accurate since we have only about 0.5 pixel error in our estimation.

Figures 4.10 and 4.11 show the estimated x-position and y-position of the circle center i.e. the mean value of the detected edge pixels.

- The mean value of the x-coordinates is calculated as the center x-position (GT=64.5).
- The mean value of the y-coordinates is calculated as the center y-position (GT=64.5).
- The mean and standard deviation of the speed values is calculated as the velocity magnitude (GT=1 frame width per second).
- The mean and standard deviation of the velocity angles is calculated as the velocity direction (GT=$\frac{\pi}{2} \approx 1.57$ radian).
- The mean and standard deviation of the distances from the center point is calculated as the circle radius (GT=15 pixels).

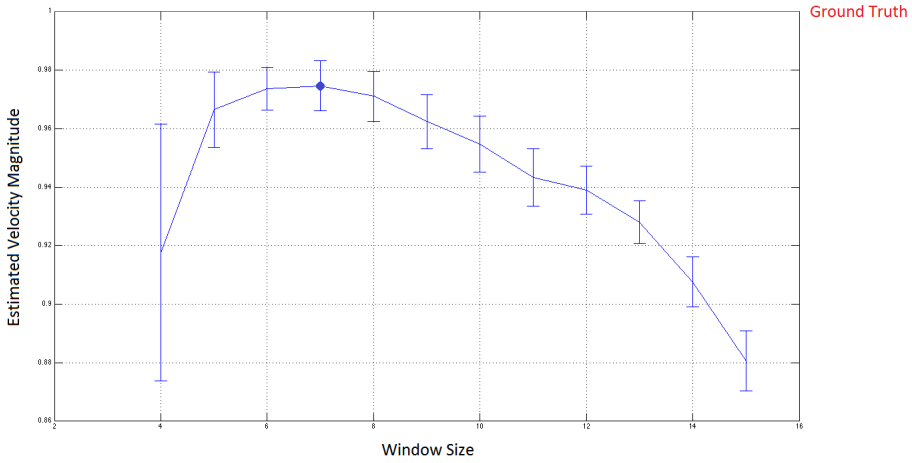


Figure 4.7: The estimated velocity magnitude for different window sizes (blue point represents the optimal window size)

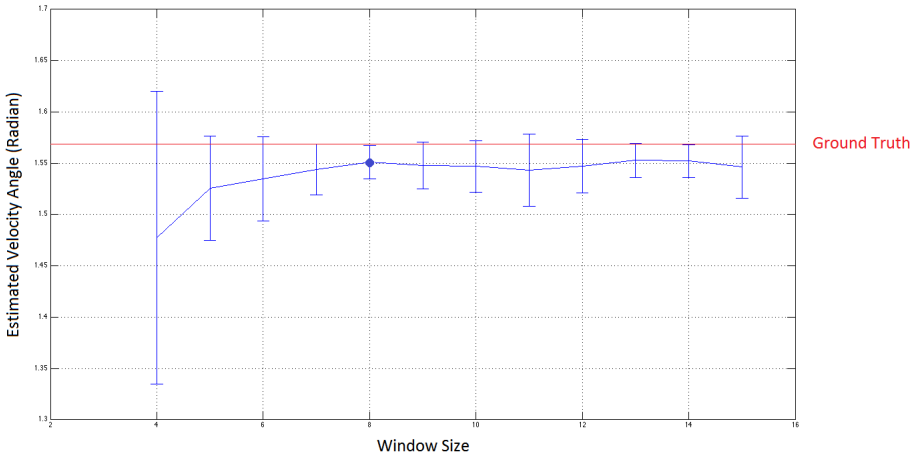


Figure 4.8: The estimated velocity direction for different window sizes (blue point represents the optimal window size)

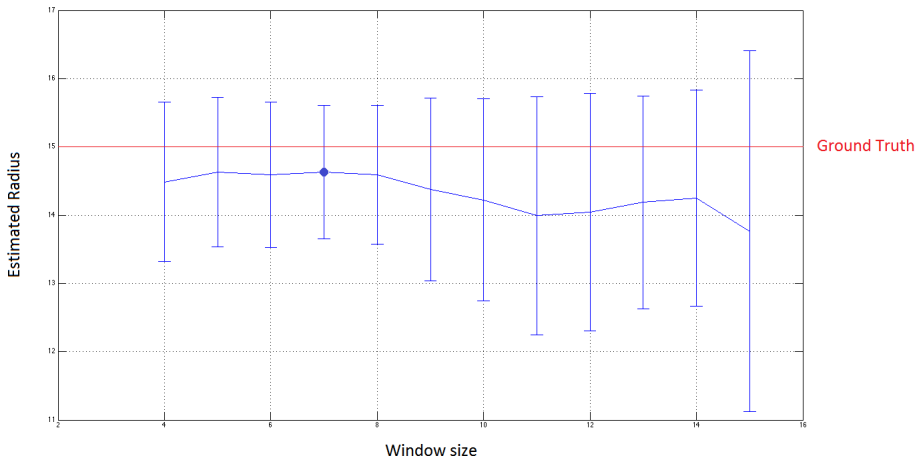


Figure 4.9: The estimated circle radius for different window sizes (blue point represents the optimal window size)

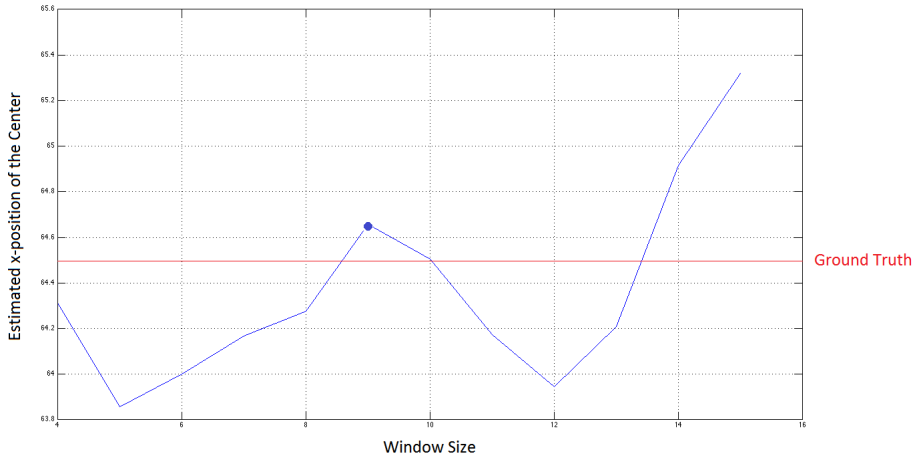


Figure 4.10: The estimated x-position of the circle center for different window sizes (blue point represents the optimal window size)

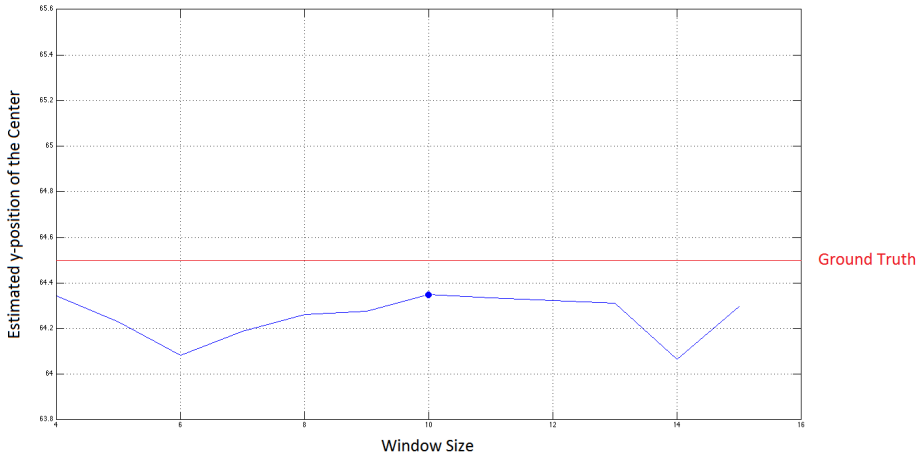


Figure 4.11: The estimated y-position of the circle center for different window sizes (blue point represents the optimal window size)

Table 4.1: The results of experiment II in (mean ± standard deviation) format. All rows including "velocity magnitude", "velocity direction", "circle radius", "center x", "center y" are normalized by their ground truth "1 frame width per second", "$\frac{\pi}{2}$ radian", "15 pixels", "64.5 pixels", "64.5 pixels" respectively. The standard deviation is not presented in last two rows since it might be misleading for these rows which represent the mean value of (x,y) coordinates of circle boundary pixels. The experiment shows that the optimal window size is 8×8 which is approximately half of the examined circle radius.

Window Size	4	5	6	7	8	9	10	11	12	13	14	15
Velocity Magnitude	0.918± 4.391%	0.967± 1.279%	0.974± 0.731%	0.975± 0.858%	0.971± 0.853%	0.962± 0.921%	0.955± 0.959%	0.943± 0.977%	0.939± 0.820%	0.928± 0.735%	0.908± 0.855%	0.881± 1.030%
Velocity Direction	0.941± 9.082%	0.971± 3.226%	0.977± 2.614%	0.983± 1.583%	0.987± 1.028%	0.985± 1.460%	0.985± 1.592%	0.982± 2.233%	0.985± 1.666%	0.989± 1.056%	0.988± 1.009%	0.985± 1.925%
Circle Radius	0.966± 7.786%	0.975± 7.295%	0.973± 7.116%	0.975± 6.518%	0.973± 6.790%	0.959± 8.945%	0.948± 9.870%	0.933± 11.65%	0.936± 11.60%	0.946± 10.39%	0.950± 10.55%	0.918± 17.62%
Center x	0.997	0.990	0.992	0.995	0.997	1.002	1.000	0.995	0.991	0.995	1.006	1.013
Center y	0.998	0.996	0.994	0.995	0.996	0.997	0.998	0.997	0.997	0.997	0.993	0.997

Table 4.2: The results of experiment III. All values are normalized by their ground truth.

Actual Velocity (cm/s)	2	4	6	8	10	12	14	16	18	20
Velocity Magnitude	0.967± 3.58%	0.973± 3.37%	0.993± 2.85%	0.997± 2.49%	0.982± 2.64%	1.019± 7.39%	0.997± 4.55%	0.979± 1.72%	0.961± 4.53%	0.974± 1.87%
Velocity Direction	1.006± 3.46%	0.985± 2.38%	1.000± 3.43%	0.993± 2.08%	1.014± 2.21%	1.038± 4.94%	1.000± 2.84%	0.997± 1.62%	0.979± 2.32%	0.984± 2.05%
Circle Radius	1.021± 1.56%	1.023± 3.62%	1.028± 4.28%	1.028± 7.80%	1.030± 2.05%	1.028± 7.54%	1.035± 2.37%	1.032± 3.19%	1.033± 2.67%	1.035± 2.50%

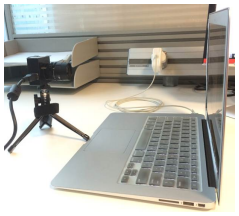
Table 4.1 shows the results of this experiment. All values are normalized by their corresponding ground truth. It is observed that the best result is obtained by the window sizes 7 or 8 which is about half of the examined circle radius. As explained before, PCA has better results on the larger windows which can regulate the quantization disturbance to some extent. In contrast, the curve edges can not be assumed as lines which is the main assumption of this study. This issue may be solved by considering the edge as a curve within a large window.

4.6 Experiment III

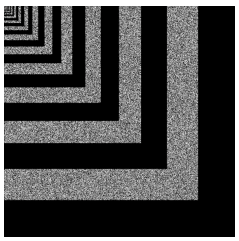
In this experiment, ten frame-based videos are generated showing horizontally moving circles at different velocities. These videos are displayed on a 13.3-inch MacBook Air with the resolution 1440×900 and aspect ratio 16:10 (figure 4.12(a)). Therefore the display area is 28.65×17.90 cm^2 and there are about 50 pixels in a centimeter. Here are the properties of these videos.

- All videos (with zero-one pixels) are showing a white circle moving horizontally left to right on a black background.
- All frame rates are 100 frames per second.
- The radius of circles is exactly 5cm.
- The velocities of circles in ten videos are 2, 4, 6, ..., 20 cm/s.
- The circles are moving on a horizontal line passing the center point of the display.

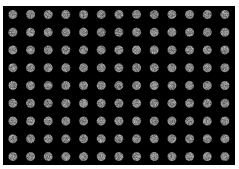
In addition, two more videos are generated for focusing (figure 4.12(b)) and calibration (figure 4.12(c)) purposes of the DVS [67]. The focusing pattern contains some squares placed in logarithmic distances while the calibration pattern includes 9×13 circles with radius 1cm. Their centers are located in equal distances (2cm) and the



(a) Experiment setup



(b) Focussing video



(c) Calibration video

Figure 4.12: The experiment setup and focussing/calibration videos

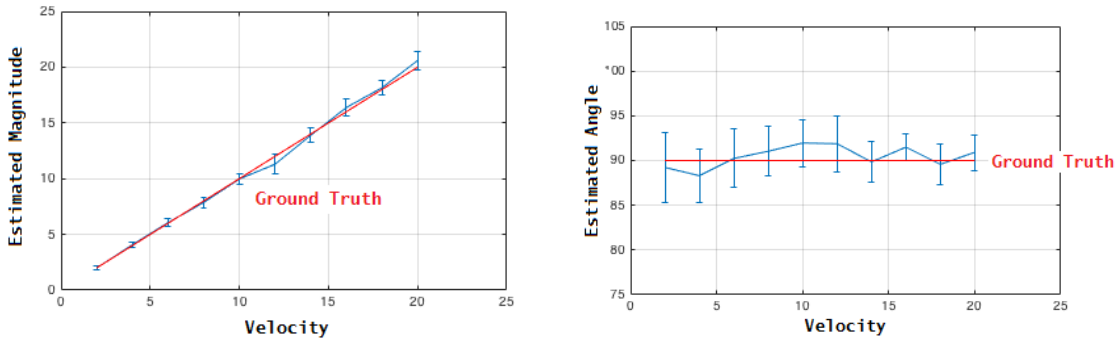
middle one is exactly on the center point of the display. In these videos, bright areas are zero-one pixels which are randomly assigned.

1. The focusing pattern is displayed and the focal length of the DVS is adjusted till the maximum number of squares are visible in the output. It is observed that an in focus DVS captures a sharp video and it has the maximum events rate for a special pattern.

2. After focusing the DVS, the calibration video is used in order to obtain matrices A and B by assuming the transformation equation as $x = AX + B$ from monitor coordinate X to video coordinate x. In the experiment, we achieved a transformation accuracy of 0.2 pixels for circles centers.

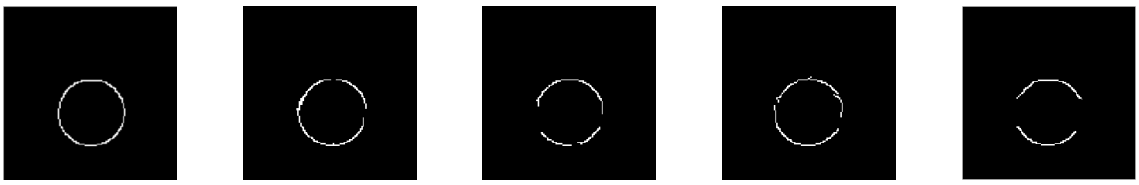
3. Ten test videos are displayed and the DVS captures their videos which will be analyzed at the instance the circles approach the center point of the display. Similarly to experiment I, the velocities magnitudes and directions of the edge pixels and their distances from the origin are measured as seen in table 4.2. The magnitude and direction of the velocity are shown in figure 4.13. All values are normalized by their ground truths which are 90° for velocity direction and $5cm$ for circle radius.

For the test video with velocity $20cm/s$, the edge direction and velocity are calculated as shown in figure 4.15. Figure 4.14 also shows the edge connectivity results for different velocities. We lose significant parts of the boundary for velocities 16 and 20



(a) Estimated magnitudes for different velocities (b) Estimated directions for different velocities

Figure 4.13: Estimated magnitudes and directions vs. different velocities. For magnitudes, the extracted values have a linear relation to the actual values. The values are normalized by a fixed coefficient to fit the ground truth.



(a) 6 cm/s (b) 12 cm/s (c) 16 cm/s (d) 18 cm/s (e) 20 cm/s

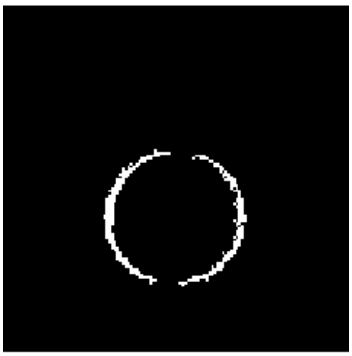
Figure 4.14: Edge connectivity results for different velocities

and insignificant parts for velocities 12 and 18. In all other cases, edge connectivity is preserved. It is noted that in the final stage of this experiment, Non-Maximal Suppression (NMS) algorithm is also applied [154].

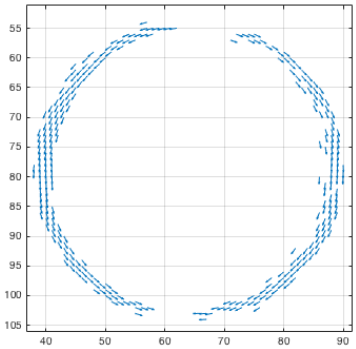
4.7 Experiment IV

We applied our algorithm on two samples of the N-MNIST dataset [155]. Figure 4.16 shows the algorithm results for '7' and '8' digits. The videos are analyzed at instance $0.5s$. To assess the accuracy of the velocity magnitude, we define a parameter as the ratio of the standard deviation to the mean value of all boundary pixels.

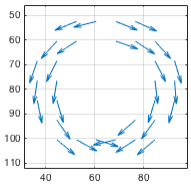
The experiment results illustrate that the proposed approach is able to estimate



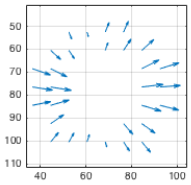
(a) All events within $10ms$ after cleaning small isolated parts



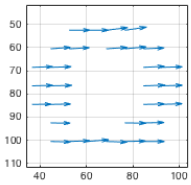
(b) Extracted pixelwise edge directions



(c) Edge direction



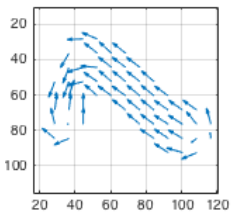
(d) Normal velocity



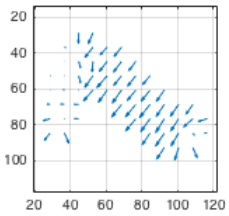
(e) Global velocity

Figure 4.15: Results for the circle moving at velocity $20cm/s$. The vectors are extracted pixelwise. However for better presentation in the second row, each vector corresponds to a 8×8 pixels block.

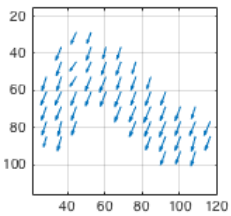
(a) All events within $50ms$



(b) Edge direction



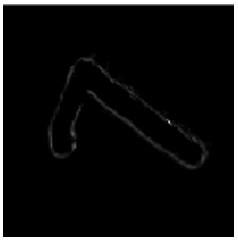
(c) Normal velocity



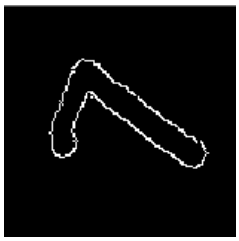
(d) Global velocity

(e) All events within $200ms$

(f) Frame with adjusted regional exposure time

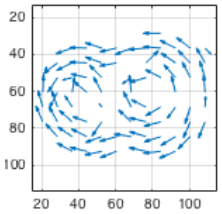


(g) After edge-dependent Gaussian filtering

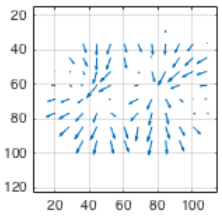


(h) After NMS and removing small isolated parts

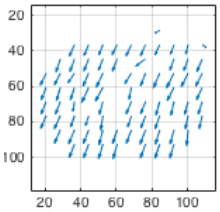
(i) All events within $50ms$



(j) Edge direction



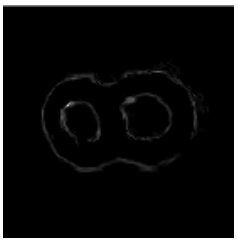
(k) Normal velocity



(l) Global velocity

(m) All events within $200ms$

(n) Frame with adjusted regional exposure time



(o) After edge-dependent Gaussian filtering

(p) After NMS and removing small isolated parts

Figure 4.16: Algorithm results for character '7' and digit '8' from N-MNIST dataset. Each vector corresponds to a 8×8 pixels block. The *std/mean* parameters for global velocity magnitude are 10.9% and 14.9% while the estimated directions are $(-20 \pm 4)^o$ and $(-21 \pm 11)^o$.

well not only the the normal motion vectors, but also the actual motion vectors of the objects from a DVS video.

4.8 Conclusion

We proposed a method to compute the motion and detect the edge of an object from the DVS video. We distinguished the normal velocity to the edge from the velocity along the edge and estimated them separately to achieve more accurate results. We used the orientation of the event plane to estimate the normal velocity locally and consider the whole object movement to extract the actual velocity globally. We also introduced an algorithm based on the regional exposure time followed by an edge-dependent Gaussian filtering for detecting the edges. In the next chapter, we propose a method for multiple lines detection and tracking in DVS videos which is suitable for direct implementation on Neuromorphic hardware e.g. TrueNorth as well as FPGA with minimum preprocessing.

Chapter 5

Event-Based Hough Transform for Lines Detection and Tracking

In this chapter, we develop an event-based Hough transform and implement in a spiking neural network to detect lines on DVS output. An event-based clustering algorithm is applied on the parameter space spikes to segment multiple lines and track them. We use a lateral inhibition strategy in our spiking neural network to suppress noise lines from being detected. Finally, the inhibitory window shape is optimized to suppress the lines which are close together in Cartesian space and are not necessarily close together in parameter space.

5.1 Introduction

Hough transform is a well-known method in conventional machine vision for extracting specific shapes e.g. lines as a useful feature for recognition tasks. Most of methods using Hough transform (HT), such as works in [156], [157], [158], are designed for frame-based systems, and re-defining them for event-based systems would require substantial and fundamental changes to their algorithms, with subsequent testing to show the correctness of the new definitions in the event-based systems. We here take

another approach, by proposing a framework to apply HT on event-based systems, by building Hough space (parameter space) using a Spiking Neural Network (SNN). While keeping the desired properties of HT, this technique allows relatively easy implementation and application of HT on event-based output streams, by traversing the output through the SNN structure. The other important benefit of coupling HT and SNN is that the new framework can be easily implemented in Neuromorphic and FPGA-based hardware. This second benefit of the proposed framework has significant impact in hardware implementation and industrialization of the framework. In contrast to frame-based approached discussed before, the proposed framework can be implemented as add-on hardware and work in real-time, with no CPU requirement. The main idea is using LIF spiking neurons to construct a two-dimensional Spiking Neural Network (SNN) which represents the parameter space of Hough transform for line detection. The main advantages of using a spiking neural network for building the parameter space are as follow:

- Each spiking neuron by itself can operate as a bin in parameter space and carry out the whole task of voting and thresholding which are needed in Hough transform.
- SNN can be directly implemented on a hardware without any processing power demand.
- The inhibitory connection within the network is a strong capability that improves the performance of conventional Hough transform by suppressing redundant lines.

As shown in Figure 5.1, the proposed SNN processes the events from a DVS sensor in an event-driven manner, and generates output spikes which represent the parameters of detected lines. Local lateral inhibition is adopted in our SNN to prevent neurons from firing in the wrong locations and thus suppress noise lines. At the last stage, an event-based clustering procedure is applied on the parameter space spikes to segment lines and track them.

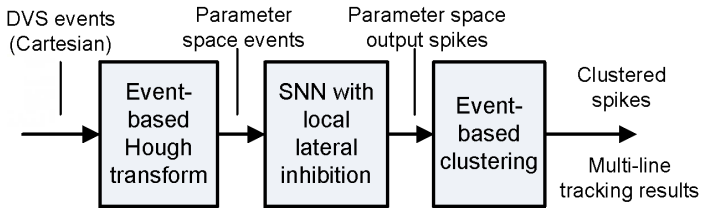
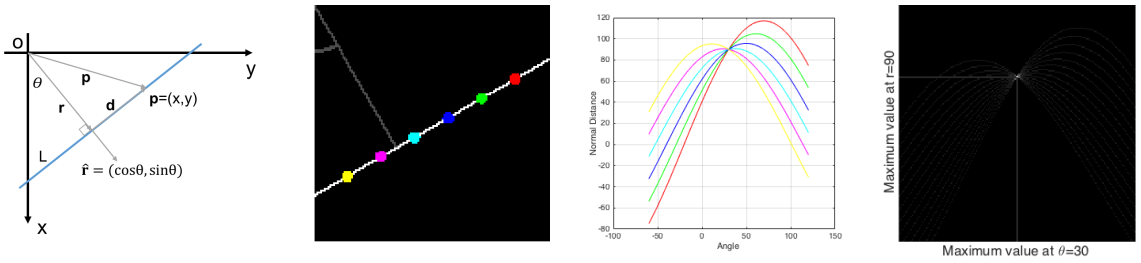


Figure 5.1: Block diagram of the proposed algorithm



(a) A simple line on a plane can be uniquely defined by two parameters r and θ

(b) Six points on a line with parameters $(r, \theta) = (90, 30°)$ in a 128×128 frame

(c) Six sinusoidal curves in parameter space are coincident at $(r, \theta) = (90, 30°)$

(d) Parameter space for 15 sinusoidal curves contains a bright point at $(r, \theta) = (90, 30°)$

Figure 5.2: Hough transform procedure for line detection

5.2 Hough transform for line detection

Let us consider the problem of Hough transform for line detection. Every line L can be uniquely defined by two parameters including the normal distance r from the origin and angle θ between the normal vector $\mathbf{r}$ and x-axis (see figure 5.2(a)). Assuming $\hat{\mathbf{r}} = (\cos\theta, \sin\theta)$ as the unit vector perpendicular to the line L, for every point $\mathbf{p} = (x, y)$ on the line:

$$\hat{\mathbf{r}}.\mathbf{p} = r \rightarrow x\cos\theta + y\sin\theta = r \tag{5.1}$$

Moreover the distance d from point $\mathbf{p} = (x, y)$ to the normal vector $\mathbf{r}$ on the line L can be calculated as follows.

$$d = |\hat{\mathbf{r}} \times \mathbf{p}| = y\cos\theta - x\sin\theta \tag{5.2}$$

Essentially the parameters chosen for determining lines are r and θ. Equation (5.1) transforms every point from Cartesian coordinate to parameter space (r, θ). In other words, every point in Cartesian space is transformed to a sinusoidal curve in parameter space defined by equation (5.1).

Referring to Figure 5.2(b), let us consider a line with parameters $r = 90$ and $\theta = 30^\circ$ in a frame with 128×128 resolution. There are six points highlighted with six different colors on the line. According to equation (5.1), every point is transformed to the parameter space as shown in Figure 5.2(c). Obviously, all six sinusoidal curves, are coincident at $(r, \theta) = (90, 30^\circ)$ since their corresponding points in Cartesian space are all on a straight line. If we perform this procedure for more points on the line (e.g. 15) and represent the parameter space as an image, there will be a bright point at $(r, \theta) = (90, 30^\circ)$ as shown in Figure 5.2(d). Therefore, lines in a frame can be extracted by finding local maximums in this parameter space. Sometimes the objective is to find small parts of whole lines in a frame (e.g. the edges of a triangle). In this case, the points distribution on the line is important and it can be considered

by calculating and comparing the distance defined by equation 5.6 for every point.

5.3 Event-Based Hough Transform in a Spiking Neural Network

This section illustrates the proposed event-based Hough transform algorithm implemented in an SNN for line detection.

5.3.1 Spiking Neuron Model

We use LIF spiking neurons to build an SNN that represents the parameter space of Hough transform for line detection. Referring to Figure 5.3, every Spiking Neuron (SN) has some inputs (one input is used here for simplicity) and an output. The input is a spike train that influences the neuron's Membrane Potential (MP). Every input spike causes an increase of the MP which is always decaying by a fixed rate. Whenever the MP exceeds a certain threshold, a spike is generated in the output, the MP is then reset and the neuron enters a refractory period, during which the neuron's MP remains zero and input spikes are ignored.

By ignoring the polarity of DVS output events, we can use the model of figure 5.3 to build an SNN for line detection. However, experiments show that considering the polarity of events can produce more robust results especially when the one side of a line is very dark and as a result many noisy events are generated in that area by DVS.

Let us investigate the pattern of generated events for a moving circle at the velocity of $10cm/s$. A video of a white circle with the radius of $5cm$ is generated on a black background which is moving from the left to the right side on the horizontal mid line of the screen. This video is played in $100FPS$ which is the maximum rate possible in a mac book air. This moving circle is captured by a DVS while the transformation matrix is known from real coordinates to camera coordinates based on the camera

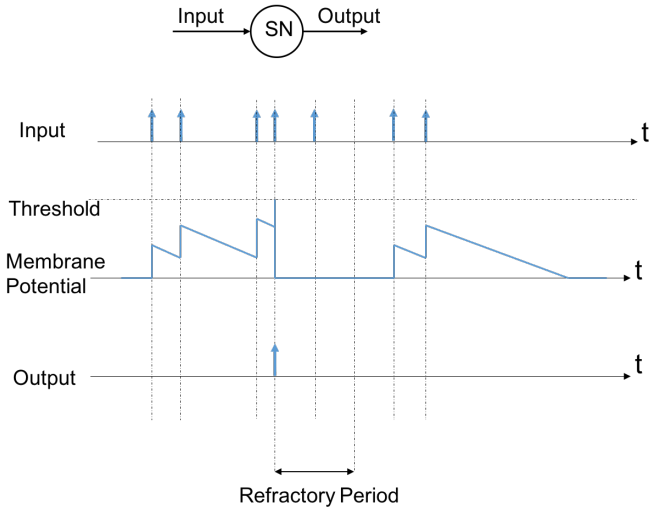


Figure 5.3: Input and output of a simple Integrate and Fire neuron. In this model, all output spikes are positive

calibration procedure. If the origin is fixed on the circle center which is moving horizontally from left to right, we expect positive/negative events at the right/left half of the circle ($[0^o, 180^o]/[-180^o, 0^o]$) as seen in figure 5.4(A). By receiving each event, its distance from the circle center, as well as its angle with respect to vertical axis of real coordinates, are calculated and stored in the event vector.

Figure 5.4(B) shows the distribution of events distances from the circle center for all events. Since the circle radius is $5cm$, the events are mostly located at the distance of $5cm$ from the circle center. They have exponential-like distribution ($\lambda e^{-\lambda r}$) around the boundary both inside and outside of the circle. It is observed that the exponential distribution inside the circle (brighter side) has a larger parameter λ than the exponential distribution outside the circle (darker side).

Figure 5.4(D) shows the distribution of events angles with respect to vertical axis only for negative events. Negative events are mostly located at the left half of the circle as we expected.

Figure 5.4(C) shows the distribution of events angles with respect to vertical axis only for positive events. Positive events are located at both sides of the circle although

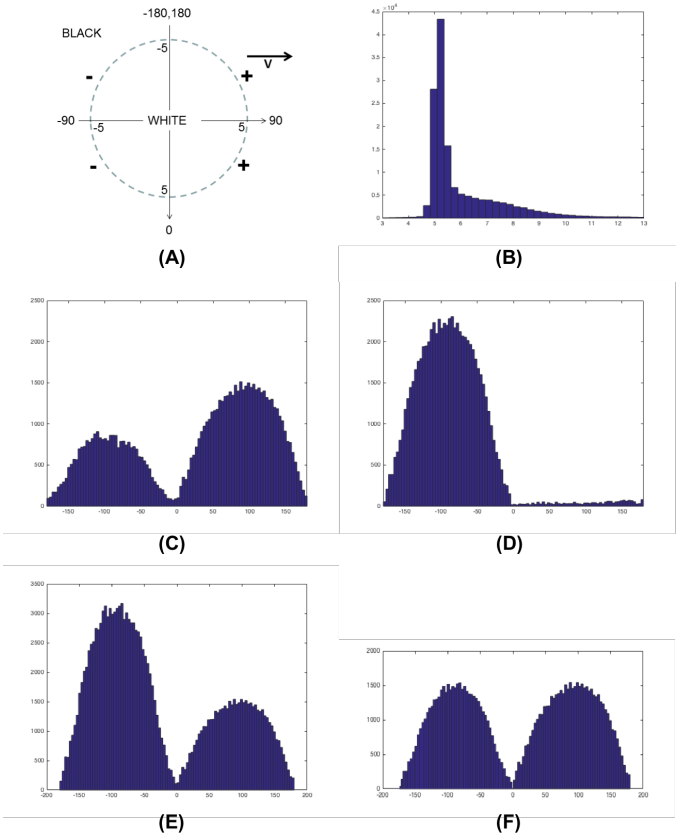


Figure 5.4: (A) A moving circle and expected events, (B) Distribution of all events over different distances from the origin, (C) Distribution of positive events over different angles, (D) Distribution of negative events over different angles, (E) Distribution of all events over different angles not considering their polarization, (F) Distribution of all events over different angles considering their polarization

their number at the right side is larger than the left side. This is the behavior we might not be able to predict at the beginning.

Figure 5.4(E) shows the distribution of events angles with respect to the vertical axis for all events including negative and positive events. Ignoring the polarization of events, the left edge of the circle generates more events than the right edge. Other experiments also verify that high to low transition of intensity generates more events than low to high transition. It is observed that low to high transition generates only positive events while the high to low transition generates both negative and positive events although the negative events are dominant. If the polarization of events is considered such that each positive event cancels a negative event at the left side of the circle, the remaining negative events at the left side will be as many as positive events at the right side of the circle as seen in figure 5.4(F). This figure also verifies that the events have a cosine-like distribution over the circle boundary which is proportional to the normal velocity of the circle.

According to the observations above, if we ignore the events polarization and use the spiking neuron in figure 5.3, the neurons correspond to the circle left side receive more excitation than the neurons correspond to the right side; i.e. the excitation for different neurons is imbalanced in different sides and it should be compensated by setting different thresholds for membrane potentials. A more simple solution which is used in this study is that we utilize events polarization in neurons excitation as proposed in figure 5.5. Although it is different from the normal model of a spiking neuron, it is more robust for neurons in different situations. Two models do not have substantial differences in behavior, however the second model is customized for this particular application and it is still possible to be implemented on the hardware.

As shown in Figure 5.5, the Spiking Neuron (SN) we use in this study, has some inputs (one input is used here for simplicity) and an output. The input is a spike train that influences the neuron's Membrane Potential (MP). Each input spike causes an increase (for positive event) or decrease (for negative event) of the MP. Note that MP is always decaying by a fixed rate. Whenever the MP exceeds the + or - threshold,

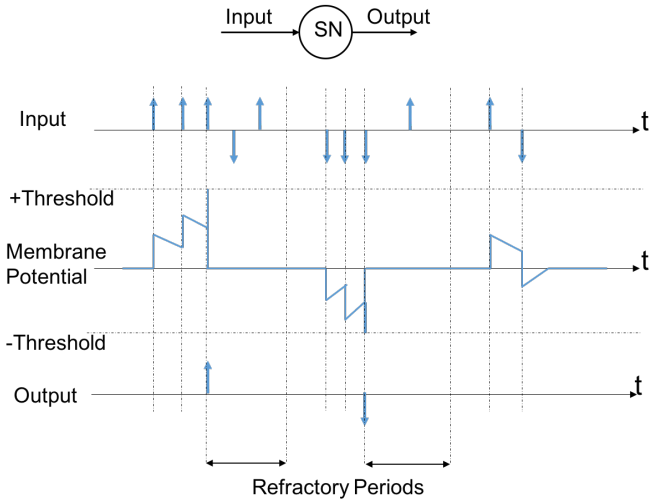


Figure 5.5: Input and output of the LIF neuron used in this paper. In this model, output spikes can be positive or negative

a spike with corresponding polarity is generated in the output. Then MP is reset to zero and the neuron enters a refractory period, during which MP remains zero and input spikes are ignored. In this work, we apply a zero refractory period for simplicity; this can also lead to more output spikes. Note that the firing of other SNs can also force an SN to reset through lateral inhibition in a network (which will be explained in details in Section 5.3.2).

We update the MPs and neuron states only when an input spike arrives. Let us define s_i as the i^{th} input spike with a time stamp t_i and polarity ($s_i = \pm 1$). Algorithm 8 is performed for each input spike s_i. In this algorithm, v_i is the neuron membrane potential, λ the rate of linear decay and v_{th} the threshold for membrane potential.

5.3.2 The Proposed SNN for Event-Based Hough Transform

Let us consider a line moving from normal distance A to B with a fixed slope as shown in figure 5.6. The parameter space is built up by a two dimensional SNN with one dimension for angle θ and the other for normal distance r. To cover all possible lines in a 128×128 frame, θ and r are limited to $(-90°, 180°)$ and $(1, 128\sqrt{2} \approx$

Algorithm 8 Updating procedure of a spiking neuron when receiving an input spike ($\lambda = 3mv/ms, v_{th} = 15mv$)

for every input spike s_i at t_i

- $v_i \leftarrow sign(v_{i-1})min(|v_{i-1}| - \lambda(t_i - t_{i-1}), 0)$
- $v_i \leftarrow v_i + s_i$
- **if** $|v_i| \geq v_{th}$ **then**

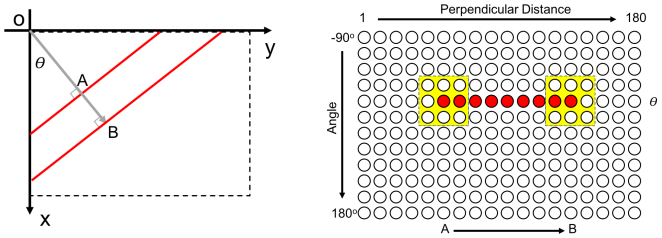
  Generate output spike $f = sign(v_i)$ at t_i

  Reset all laterally connected neurons

  $v_i \leftarrow 0$

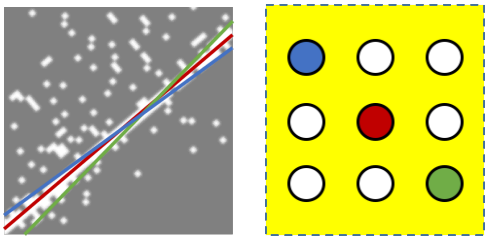
  end if

end for



(a) Cartesian space (b) Parameter space

Figure 5.6: A 128×128 frame and its corresponding parameter space on a 200×300 SNN. θ is limited between $-90°$ and $180°$ while r between 1 and $128\sqrt{2} \approx 180$. Red color shows the firing neurons during line movement from A to B. Yellow window indicates local lateral connection (for inhibition). Each neuron is laterally connected to all neurons that are within a window around that neuron



(a) Cartesian space (b) Parameter space

Figure 5.7: Local lateral inhibition to suppress noise lines. Without lateral inhibition, SNN detects three lines (red, blue and green). The best fitted one is the red line detected by the red neuron. The red neuron is expected to fire before blue or green ones. With local lateral inhibition, when the red neuron (the correct one) fires, it inhibits all laterally connected neurons and thus blue and green lines can be suppressed.

180). A local lateral inhibition strategy is adopted in our SNN. The output of every neuron is laterally connected to all the neighboring neurons that are located within a window centred at that neuron. Whenever a neuron fires, it forces itself as well as all neighboring neurons within a window to be reset. This window is represented by the yellow rectangles in figure 5.6. During line movement from point A to B, we will have many spikes in the parameter space highlighted by red color.

Local lateral inhibition allows the SNN to suppress noise lines (or redundant lines) from being detected. This is illustrated in Figure 5.7 which shows the events generated by moving an edge in front of the DVS sensor. The detected lines using the SNN without local lateral inhibition are superimposed onto the DVS events. The best fitted line is the red line which is detected by the red neuron in Figure 5.7. However, the blue and green lines also cover many events in the frame and they are detected by the firing of the blue and green neurons in the parameter space, respectively. These two lines can be suppressed by the local lateral inhibition. Each neuron is laterally connected to all its neighboring neurons that are within a window around that neuron. These connections cause a local competition between the neurons inside the yellow window. Whenever the first neuron (red one in this case) fires, it resets all neighboring neurons, prevents them from firing, and thus suppresses the noise lines.

The parameter space is built up by $N \times M$ SNs; i.e. N rows for θ quantization and

Algorithm 9 Event-based Hough transform and neuron excitation in the parameter space SNN

for every received event $e_i = (t_i, x_i, y_i, s_i)$

- Calculate $r_{\theta_j} = \arg\min_{r_k} |r_k - x_i \cos\theta_j - y_i \sin\theta_j|$ for $1 \leq j \leq N$ and $1 \leq k \leq M$
- Excite all neurons (θ_j, r_{θ_j}) at t_i for $1 \leq j \leq N$ with s_i (algorithm 8)
- Reset all neurons in first column ($k = 1$)

end for

M columns for r quantization. Therefore every neuron located at (j, k) is identified by (θ_j, r_k) values. Algorithm 9 shows the proposed event-based Hough transform and neuron excitation in the parameter space SNN. Assume that an event $e_i = (t_i, x_i, y_i, s_i)$ is received from DVS, meaning that there is an intensity change of pixel (x_i, y_i) at time t_i. The term s_i is either 1 for dark-to-bright change or -1 for bright-to-dark change. For every $\theta_j (1 \leq j \leq N)$, the corresponding r_{θ_j} is calculated using equation (5.1) and rounded to the nearest possible $r_k (1 \leq k \leq M)$. Then all neurons $(\theta_j, r_{\theta_j})(1 \leq j \leq N)$ inside the parameter space are excited by s_i based on algorithm 8. We need to ignore all neurons in the first column ($k = 1$) because they can be wrongly excited by the cases that Hough transform of an event is partly outside of the boundary of the parameter space.

Algorithm 10 Event-based clustering for segmentation and tracking of multiple detected lines

for every received spike $f_i = (t_i, r_i, \theta_i)$ from parameter space

- Find all active clusters $C_l = (T_l, R_l, \Theta_l)$ that $t_i - T_l \leq T_{threshold}$
- Find the minimum Euclidean distance D_i between (r_i, θ_i) and all active clusters (R_l, Θ_l). Suppose $C_{l=m}$ is the winner; i.e. $D_i = ||(r_i, \theta_i) - (R_m, \Theta_m)||$
- **if** $D_i \leq D_{threshold}$ **then**

  f_i belongs to cluster $C_{l=m}$

  $(R_m, \Theta_m) = (1 - \alpha)(R_m, \Theta_m) + \alpha(r_i, \theta_i) \qquad (\alpha = 0.1)$

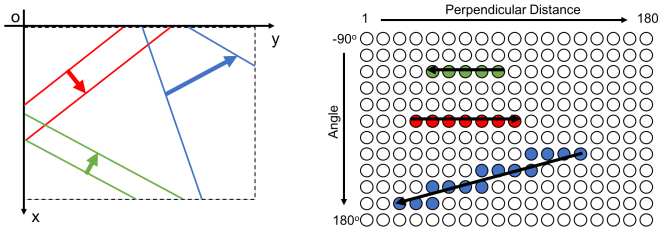
  else

  Generate a new cluster $C_{L+1} = (t_i, r_i, \theta_i)$

  end if

end for

In cases that there are more than one moving line in the frame, we need a segmentation procedure to distinguish between them as shown in Figure 5.8. Many spikes



(a) Cartesian space (b) Parameter space

Figure 5.8: Three moving lines in Cartesian space defined by three colors and their transformations in the parameter space. Spikes are segmented in three different series correspond to three lines.

are generated from the parameter space SNN in different locations. Since every line is moving smoothly in Cartesian space, the corresponding spikes in parameter space are "moving" smoothly as well and they produce a segment (cluster). The l^{th} segment is defined by a cluster $C_l = (T_l, R_l, \Theta_l)$ which shows the time and location of the segment. We use an event-based clustering method [3, 4] to do the segmentation and tracking of different lines. For every spike $f_i = (t_i, r_i, \theta_i)$ from the parameter space SNN, we calculate the Euclidean distances from this spike location to all active clusters, and find the smallest distance and corresponding cluster. If this distance is less than a threshold, the spike f_i is considered as belonging to that corresponding cluster, and the corresponding cluster is then updated; otherwise, a new cluster is generated by this spike. The cluster activity (active or inactive) is defined by comparing current time stamp (of the SNN spike) with the last update time of this cluster. If the cluster is old enough (i.e. no update for a long time), it is considered inactive and can not win any spike in the future.

5.4 More Efficient Parameter Space

To cover all possible lines within a frame, θ is limited between -90^o and 180^o while r between 0 and $180\sqrt{2}$ in the parameter space. However, the possible area is not a simple rectangle according to the transformation Equation 5.1. Depending on r and

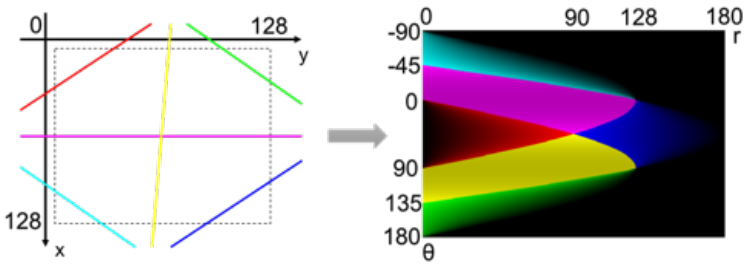


Figure 5.9: Different line positions in Cartesian space and their corresponding areas in the parameter space. Each color intensity in the parameter space shows the line length inside the video frame.

θ values, the corresponding lines in the Cartesian space have variable length and the different situation inside the video frame. Figure 5.9 shows different line positions in the video frame and their corresponding areas in the parameter space. This area covers about 60% of the whole rectangle based on the simulation. As a result, the number of spiking neurons can be reduced by 40% in hardware implementation.

Each color intensity in the parameter space represents the corresponding line length within the video frame as seen in figure 5.9. Smaller lines generate fewer events in DVS and the corresponding spiking neuron in parameter space receives less excitation subsequently and therefore is not likely to fire. To solve this issue, the internal threshold for neurons membrane potential should be proportional to the line length or the colors intensity in the parameter space. However, these values should be larger than a minimum threshold not to detect extremely short lines at the frame corners.

5.5 More Efficient Inhibitory Window

The issue with the small lines was discussed in the previous section. Another issue stems from the fact that proximity in the parameter space does not necessarily indicate proximity in Cartesian space. Therefore, it is possible that some lines which are placed too close together in the parameter space be regarded as one line, while they are placed far apart in the Cartesian space.

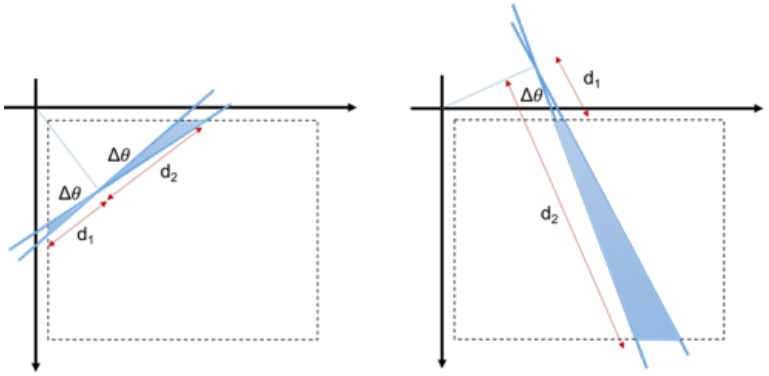


Figure 5.10: Left figure shows two lines which are near in both Cartesian space and parameter space. Right figure shows two lines which are near in parameter space, not in Cartesian space.

To make it clear, Figure 5.10 is considered. This figure shows two sets of couple lines. First couple lines are (θ_1, r) and $(\theta_1 + \Delta\theta, r)$ and second couple lines are (θ_2, r) and $(\theta_2 + \Delta\theta, r)$ at the left and right side of the figure respectively. The distance between the corresponding points of every couple lines in parameter space are the same despite their distances in Cartesian space which are clearly different. This means that $\Delta\theta$ is the same in Cartesian space (Figure 5.10) for two lines couples and thus the distance between the corresponding points of the couple lines is the same in parameter space.

This specific case can happen in our inhibitory window. As detailed in section 5.3.2, we inhibit neurons within a small window around the correct neuron, in the parameter space, to reduce the noise. Applying the inhibitory window directly in the parameter space, can cause inhibition in detection of a line in the Cartesian space, far from and unrelated to the main detected line. To address this problem, we improve the inhibitory window, by applying the window in the Cartesian space, and optimizing its shape to suppress the lines which are close together in the Cartesian space, and not necessarily in parameter space.

5.5.1 Distance Metric in Cartesian Space

To determine which neurons to be reset, a distance metric D is defined. This metric is consistent with human perception of the distance between two lines. It is the average distance obtained by dividing the area between two lines by the length of the lines. Assuming the lines are close together, two situations can be considered.

- If the intersection point is inside the frame:

$$D = \frac{S}{d} = \frac{S_1 + S_2}{d_1 + d_2} = \frac{1}{2}\pi\Delta\theta\frac{d_1^2 + d_2^2}{d_1 + d_2} \tag{5.3}$$

- If the intersection point is outside the frame:

$$D = \frac{S}{d} = \frac{S_2 - S_1}{d_2 - d_1} = \frac{1}{2}\pi\Delta\theta\frac{d_2^2 - d_1^2}{d_2 - d_1}$$

$$D = \frac{1}{2}\pi\Delta\theta(d_1 + d_2) \tag{5.4}$$

The intersection point of two lines (θ, r) and $(\theta + \Delta\theta, r + \Delta r)$ can be calculated as follows.

$$\begin{cases} xcos(\theta) + ysin(\theta) = r \\ xcos(\theta + \Delta\theta) + ysin(\theta + \Delta\theta) = r + \Delta r \end{cases}$$

$$\begin{bmatrix} cos(\theta) & sin(\theta) \\ cos(\theta + \Delta\theta) & sin(\theta + \Delta\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \\ r + \Delta r \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{sin(\Delta\theta)} \begin{bmatrix} sin(\theta + \Delta\theta) & -sin(\theta) \\ -cos(\theta + \Delta\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} r \\ r + \Delta r \end{bmatrix}$$

Assuming $\Delta\theta$ is very small:

$$\Delta\theta \ll 1 \quad \rightarrow \quad \Delta\theta^k \approx 0 \quad for \quad k > 1$$

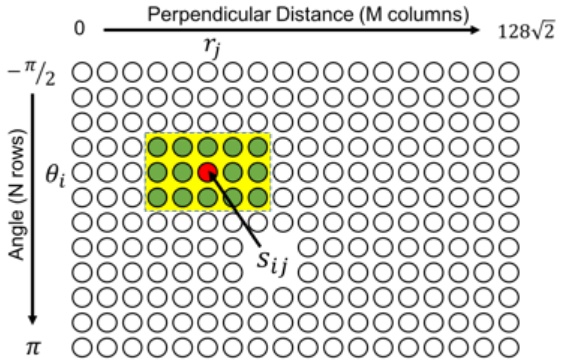


Figure 5.11: The parameter space is implemented in a spiking neural network. Inside the yellow window, the inhibitory inputs of some green neurons are laterally connected to the output of central neuron (red neuron)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\Delta\theta} \begin{bmatrix} \Delta\theta cos(\theta) + sin(\theta) & -sin(\theta) \\ \Delta\theta sin(\theta) - cos(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} r \\ r + \Delta r \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\Delta\theta} \begin{bmatrix} r\Delta\theta cos(\theta) - \Delta r sin(\theta) \\ r\Delta\theta sin(\theta) + \Delta r cos(\theta) \end{bmatrix}$$

$$\begin{cases} x = rcos(\theta) - \frac{\Delta r}{\Delta\theta} sin(\theta) \\ y = rsin(\theta) + \frac{\Delta r}{\Delta\theta} cos(\theta) \end{cases} \tag{5.5}$$

5.5.2 Lateral Inhibitory Connections

The inhibitory connections are set just one time at the beginning of the simulation. The spiking neural network consists of $N \times M$ neurons as shown in Figure 5.11 (N rows for θ quantization while M columns for r quantization). Any neuron s_{ij} represents a line (θ_i, r_j) in the parameter space. Steps of setting the inhibitory connections of s_{ij} are as follows:

- For any spiking neuron s_{ij} (red neuron), find all neurons s_{kl} (green neurons) inside a window centered at s_{ij} (yellow window).
- Considering any green neuron s_{kl} and the red neuron s_{ij}, find the intersection

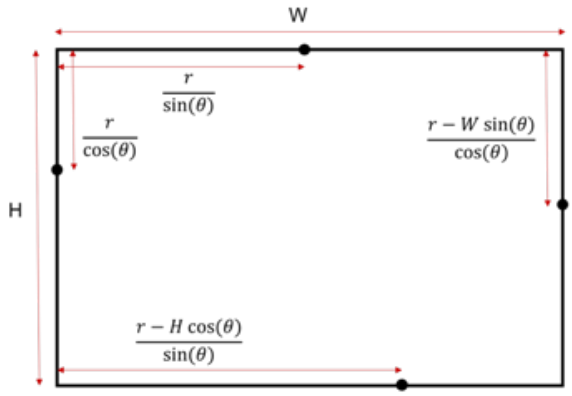


Figure 5.12: Four possible points of the frame edges that an arbitrary line (θ, r) can pass through. Any line passes through only two points among four possible points.

point of lines (θ_k, r_l) and (θ_i, r_j) using Equation 5.5.

- Depending on the position of line (θ_i, r_j) in Figure 5.9, find two points of the frame edge that the line is passing through (Figure 5.12). Calculate the distances d_1 and d_2 from these points (Figure 5.10) to the intersection point obtained in the previous step.

- Depending on the position of the intersection point, whether it is inside or outside the frame, obtain the distance metric, D based on Equation 5.3 or Equation 5.4.

- If $D < D_{threshold}$, set an inhibitory connection between s_{kl} and s_{ij}.

5.6 Extended event-based Hough Transform

In section 5.2, it was discussed that a simple method for detecting a line is setting a threshold on the number of votes in Hough parameter space. If this threshold is large enough, then the full-length lines in a frame can be detected. If the objective is to recognize small lines as well, the threshold should be small enough since such lines contain a small number of events in a frame and as a result, the corresponding

point in parameter space receives fewer votes compared to longer lines. There are two major drawbacks in this procedure:

- If there are two or more small separated lines exactly on a same direction as seen in figure 5.13(A), even if enough votes are received from those lines for the corresponding direction in the parameter space, it is not possible to recognize which vote comes from which line in Cartesian space. Therefore, we can not recognize the exact position of small lines on the main direction; i.e. the main direction can be estimated, although it is not possible to find out which part and what ratio of the detected direction is really covered by small lines.

- If the threshold is set small enough to detect small lines, the procedure can not detect the concentration of supporting events in Cartesian space. To illustrate the problem better, two scenarios are shown in figures 5.13(B) and 5.13(C). A few points are concentrated on a direction and form a small line in figure 5.13(B). On the other hand, the same number of points are distributed randomly on the same direction as shown in figure 5.13(C). This scenario is most likely caused by noise. If the threshold is fixed for both cases, the procedure detects a small line in both cases although it is wrong for the second scenario.

- If an arbitrary shape (not necessarily a straight line) exists in the frame, the standard event-based Hough transform fails to detect the shape.

To address the above mentioned problems, we should consider the position of points on the detected direction. This position can be described by the distance of each point on the direction from the perpendicular line which is shown by d in figure 5.16(A). The distance d can be considered as the cross product of vector $\mathbf{p} = (x, y)$ and unit normal direction ($\hat{\mathbf{r}} = (cos\theta, sin\theta)$) as follows:

$$\mathbf{p} \times \hat{\mathbf{r}} = \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ x & y & 0 \\ \cos\theta & \sin\theta & 0 \end{vmatrix} = (x\sin\theta - y\cos\theta)\hat{\mathbf{k}} = d\hat{\mathbf{k}}$$

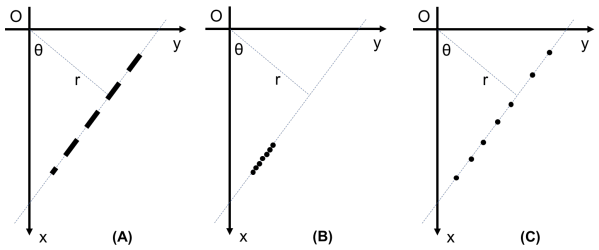


Figure 5.13: (A) Multiple small lines on a direction. The standard event-based Hough transform can not detect the position of lines on the detected direction; (B) 7 events from a small line on a direction; (C) 7 events on the same direction caused by noise. The standard event-based Hough transform can not distinguish between these two cases.

$$d = x \sin\theta - y \cos\theta \tag{5.6}$$

Based on the above definition, the right/left side of the perpendicular line is considered as positive/negative d on the direction; i.e. d is a negative value in figure 5.2(A)).

d value can be considered as the negative derivative of r with respect to θ based on equation 5.1. As a result, the slope of curves passing through the common point in figure 5.2(C), shows the distance of corresponding point in Cartesian space from the perpendicular line. As a result, to check whether a local maximum in parameter space shows a real line in Cartesian space, we should check the distribution of curves around the maximum point. This distribution directly affects the shape of contour lines (equipotential contours) around the maximum point and the sharpness of that. The investigation of this issue is beyond the scope of this thesis. Alternatively we can use distance d as the third parameter beside r and θ to build a three-dimensional Hough space as seen in figure 5.14. In a conventional 2D parameter space corresponds to a frame containing a line, there is a bright point at particular r and θ while in new 3D parameter space, there are many bright points in different d at the same r and θ as seen in figure 5.14(B).

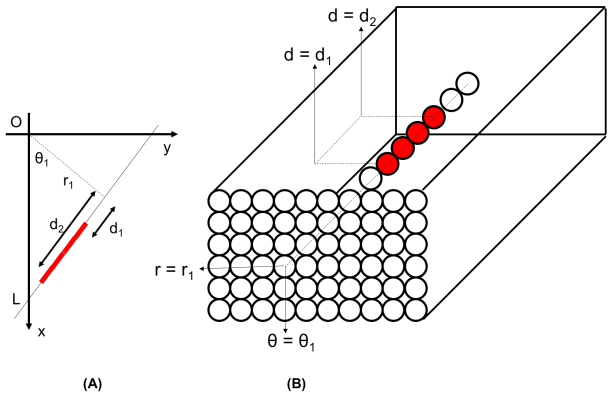


Figure 5.14: (A) A line (marked by red) in Cartesian space, (B) corresponding three-dimensional Hough area consists of different bins. Red bins represent bright points those receive more votes from Cartesian space. These bins are substituted with spiking neurons

5.6.1 Parameter space quantization

In real application, the parameter space should be quantized in all dimensions. In [159], a uniform quantization was used for r and θ. The quantization method of parameter space can highly affect the quality and accuracy of the implementation. Using polar coordinates in parameter space, uniform quantization of θ is not the best choice as seen in figure 5.15. If a line element is located near to the perpendicular line, a small change in θ results in a small displacement of the line element. On the other hand, the same change of θ for the elements which are far from the perpendicular line causes a large displacement of the elements. In other words, the Cartesian space is not transformed uniformly to different bins in parameter space.

Let us focus on quantization issue. Δr is roughly defined by the thickness of lines we want to detect. On the other hand, Δd should be considered at least a few times of Δr since for a line it is obvious that the width should be several times smaller than the length. Herein, we use a uniform quantization for r and d while the quantization of θ is optimized for better performance.

Let us consider a small linear element $E1(\theta, r, d)$ as seen in figure 5.16(B). By fixing r and d of this element, a small change $\Delta\theta$ of the angle results in a new position at

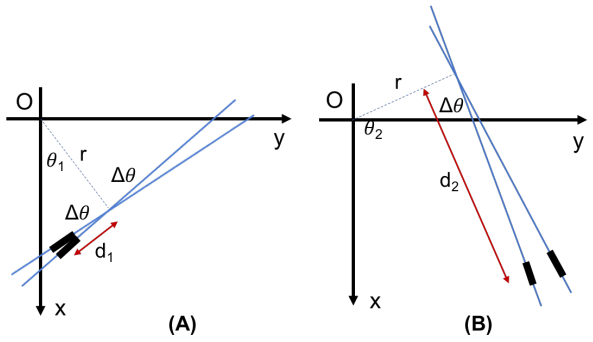


Figure 5.15: The effect of uniform quantization of parameter space. Two line elements with the same distance of $\Delta\theta$ in Hough area, although (A) the elements can be near or (B) the elements can be far in Cartesian space

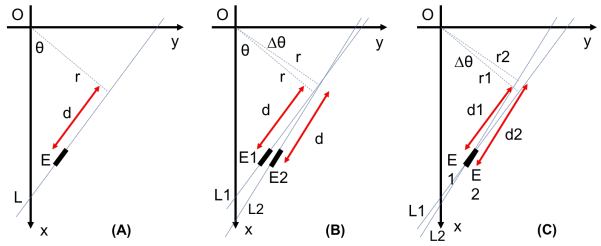


Figure 5.16: (A) Each line element is uniquely defined by three parameters θ, r, d. (B) For fixed values of r and d, a small change in θ cause a radial displacement of $d\Delta\theta$ and a tangential displacement of $r\Delta\theta$ in the element position. (C) Redundant lines are located at the same place with a small difference in their orientations. These lines are suppressed by lateral inhibitory connections within the network.

$E2(\theta+\Delta\theta, r, d)$. The displacement from $E1$ to $E2$ consists of two radial and tangential displacement. Approximately, the radial part is $d\Delta\theta$ while the tangential part is $r\Delta\theta$. To cover all over the frame in Cartesian space and not to have redundant elements at the same time, $E1$ and $E2$ should not intersect each other in the frame and there should be no gap between them. As a result, the tangential displacement should be smaller than the element length and the radial displacement should be smaller than the element thickness as follows:

$$r\Delta\theta < \Delta d \Rightarrow \Delta\theta < \frac{\Delta d}{r} \tag{5.7}$$

$$d\Delta\theta < \Delta r \Rightarrow \Delta\theta < \frac{\Delta r}{d} \tag{5.8}$$

To find out which one of above conditions is more restrictive, we should compare $\frac{\Delta d}{r}$ and $\frac{\Delta r}{d}$ to see which one is smaller. By defining stretch ratio as $SR = \frac{\Delta d}{\Delta r}$ for line elements, if d is smaller than $\frac{r}{SR}$, the condition 5.7 is more restrictive than condition 5.8 and vice versa. The stretch ratio is a value greater than one (two, three or even more) defined by r and d quantization which is fixed during the whole procedure. For a specific value of r and d, if $d < \frac{r}{SR}$ the angle θ is quantized uniformly while for greater values of d, the quantization of θ should be finer by increasing d.

5.6.2 Spiking Neural Network

In this study we build the parameter space using a spiking neural network the same as what is done in section 5.3.2. The key difference is that a three-dimensional network is used here (figure 5.14(B)) to extract not only the whole lines in the frame, but also any other small lines which can potentially make any arbitrary shapes in the frame. Then based on the direction of these small elements, important parameters like curvature can be obtained and the whole shape can be extracted or encoded subsequently. The main limitation of 3D networks is the number of neurons they need. However in this application, the number of neurons can be reduced 10 times by removing those neurons do not receive any vote from the Cartesian space. Our experiment shows that 40K neurons are needed for building a 3D parameter space for 128×128 frame in Cartesian space with $\Delta r = 3pixels$ and $\Delta d = 9pixels$. SNNs in this scale can be easily implemented with current hardware technology.

5.7 Inhibitory Connections

Inhibitory connections between neurons are used for suppressing lines which are close together. As seen in figure 5.16(C), let us suppose two different lines $E1 = (\theta_1, r_1, d_1)$ and $E2 = (\theta_2, r_2, d_2)$ be the best matching line elements to some existing

Table 5.1: Quantitative analysis of line detection results on artificially generated line events

Line	Input Events #	Output Spikes #	Spike Time (ms) First/Last		θ error (degree) Mean/SD		r error (pixel) Mean/SD	
Cyan	9100	158	2630	4994	0.04	0.18	0.01	0.22
Blue	14116	278	6	4953	0.07	0.21	0.12	0.24
Green	18022	418	20	4993	-0.02	0.23	-0.03	0.21
Red	31654	923	0	4970	0.04	0.31	0.06	0.31

events in Cartesian space. The corresponding neuron to the best matching line fires and the line is detected. We want to suppress all other lines which are most probably capable of being wrongly detected as redundant lines. Referring to equations 5.1 and 5.6:

$$r = x\cos\theta + y\sin\theta \Rightarrow \Delta r = (-x\sin\theta + y\cos\theta)\Delta\theta = -d\Delta\theta$$

$$d = x\sin\theta - y\cos\theta \Rightarrow \Delta d = (x\cos\theta + y\sin\theta)\Delta\theta = r\Delta\theta$$

Above equations show the mathematical relation between $\Delta\theta$ and $(\Delta r, \Delta d)$. If the objective is to suppress all redundant lines within $\pm\Delta\theta$ range in angle, the corresponding $(\Delta r, \Delta d)$ are calculated based on above equations. In this study, we suppress all redundant lines in all directions. It is noted that all crossing points are removed if the lines are suppressed in all directions. When a neuron fires, we find its corresponding point in Cartesian space. On this point, a resetting event is supposed. This event is transformed to parameter space and reset all affected neurons in the network.

Each output spike from SNN represents a small line element in Cartesian space. These spikes can be further investigated to find their relation in Cartesian space. Ordering spikes of SNN enables us to span and track any arbitrary shape in the frame. This is a part of our future work.

5.8 Experiment V

We evaluated our event-based Hough transform on some artificially generated line events. As shown in figure 5.17, we generated artificial events by moving 4 different

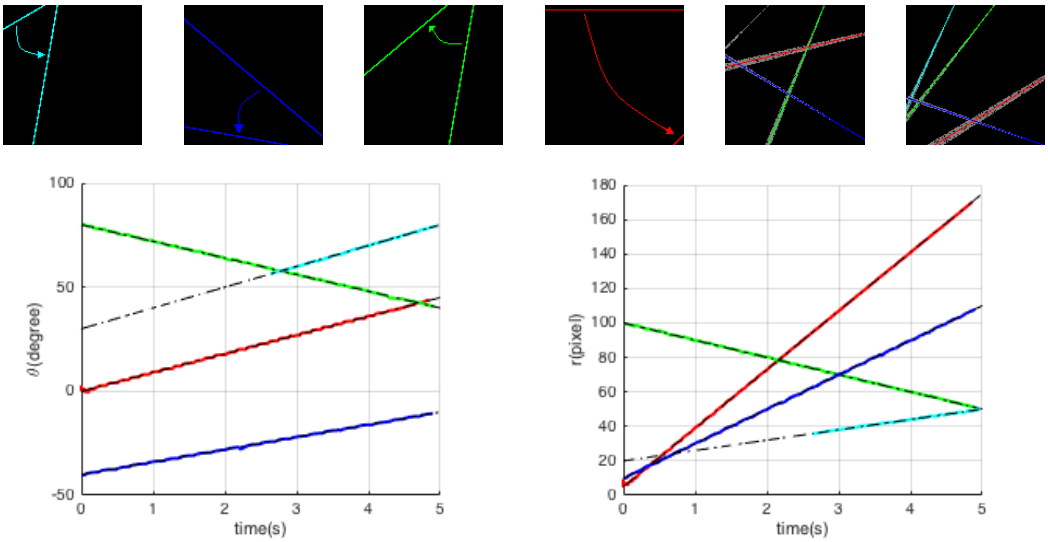


Figure 5.17: Line detection results on artificially generated line events. 1st row: Traces of 4 different lines and results (in colors) overlaid on input events (grey pixels) at certain time instances (1.5s and 3.5s). 2nd row: Results (in colors) superimposed on ground truth (dashed lines) during the whole time course.

lines in a single scene. The first row shows the moving traces of these lines and the line detection results at two different time instances. Detected lines are shown in colors, and the input events (within 200 ms around the time instance) are shown in grey. The second row shows the detection results (in colors) superimposed on ground truth (dashed lines) during the whole time course (0-5 s) for angle θ and normal distance r, respectively. We can see that the propose algorithm can accurately detect and track the four moving lines. Table 5.1 reports the statistics of the line detection results. As we can see, the errors are very trivial, which quantitatively demonstrates the accuracy of the proposed algorithm. Note that the first line (in color cyan) is not detected until t = 2630 ms. This is because the initial length of this line is quite short, leading to a small number of input events and thus few output spikes in the corresponding cluster. A cluster with few spikes at the beginning will remain hidden in order to prevent trivial lines from being detected/reported.

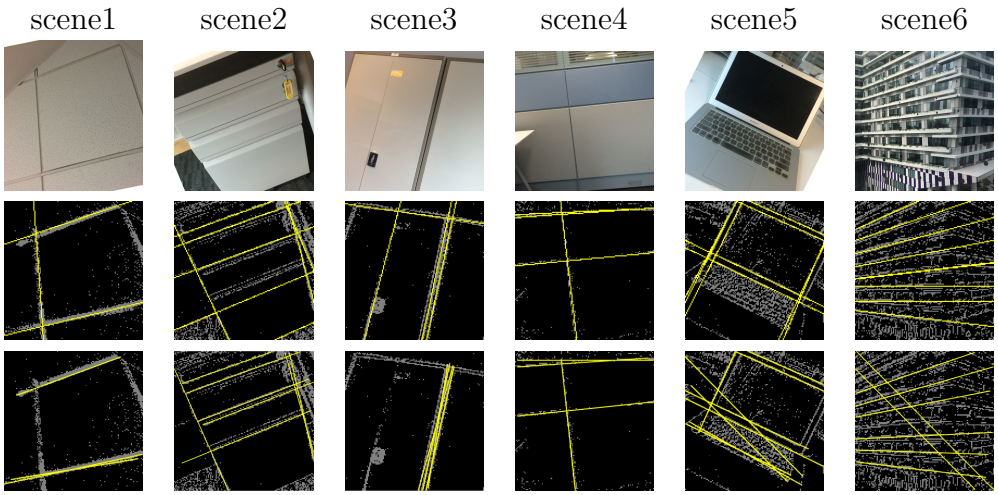
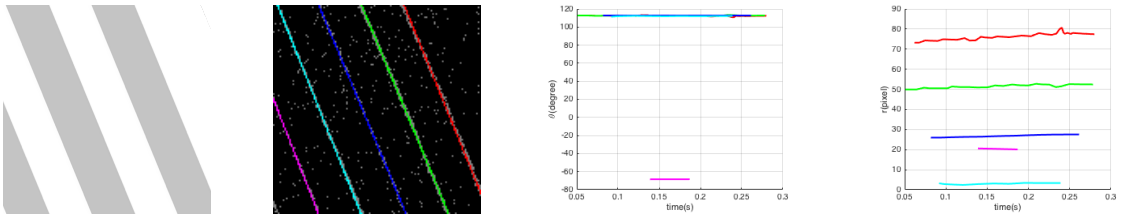


Figure 5.18: Line detection results on various scenarios; 1st row: images captured by a conventional camera, depicting various scenes that the DVS sensor was recording; 2nd row: The proposed event-based algorithm's line detection results (yellow) superimposed onto DVS events (grey); 3rd row: Conventional frame-based hough transform's results using MATLAB standard functions for line detection with the same number of the lines.

5.9 Experiment VI

We also evaluated the proposed algorithm for event-based Hough transform on various real DVS event streams, and compared the results with those of conventional frame-based hough transform. The results are illustrated in Figure 5.18. The first row shows the images captured by a conventional camera; they depict the various scenes that the DVS sensor was recording. Various indoor and outdoor scenes have been recorded, including room ceiling, storage cabinets, cubicle panel, laptop, and buildings. The second row illustrates the proposed algorithm's line detection results (yellow) superimposed onto DVS events (gray). The third row shows the results of conventional frame-based hough transform algorithm applied on frames reconstructed from DVS events. For each scenario, we let the frame-based hough transform detect the same number of lines as our event-based algorithm does. As we can see, the proposed event-based algorithm provides better detection results, especially in scenes 3, 5, and 6. In addition, tracking of the detected multiple lines is very easy using



(a) Printed pattern (b) Detected lines (c) θ diagrams in time (d) r diagrams in time

Figure 5.19: A pattern of five equidistant parallel edges, five detected lines in a frame and their θ and r diagrams

our algorithm, whereas it is quite hard to achieve using conventional frame-based algorithm.

5.10 Experiment VII

In this experiment, a pattern is printed on a paper as shown in Figure 5.19(a) with five equidistant parallel edges. Then we shake this paper in front of a DVS while it is capturing the video. We apply our Hough transform algorithm on the generated events and extract five lines as seen in Figure 5.19(b) in different colors including magenta, cyan, blue, yellow and red. Figures 5.19(c) and 5.19(d) represent θ and r values of detected lines in time. θ values are almost overlapped since the lines are parallel and r diagrams are equidistant as we expected (For magenta line, we need to add 180^o to θ values and change the sign of r values to negative instead).

To show the accuracy of results, the differences between θ and r values of any two neighboring lines are calculated. Table 5.2 represents the mean, standard deviation (SD), minimum and maximum of these values for $\Delta\theta$ and Δr. This table shows that five detected lines are parallel and equidistant as they are expected.

Table 5.2: Mean, standard deviation, minimum and maximum values of θ and r differences

	Mean	SD	Min	Max
$\Delta\theta$	0.1	0.6	-2.5	1.3
Δr	24.3	0.89	22.4	29.2

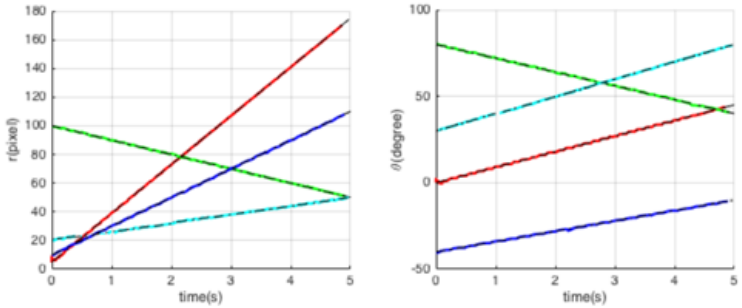


Figure 5.20: The perpendicular distance and the angle of detected lines during the whole video time span. The results (in colors) are well matched with the ground truth (dashed lines).

5.11 Experiment VIII

The revised version of event-based Hough transform is applied on the same artificially generated data used previously in section 5.8. As shown in Figure 5.17, this synthetic video includes four different lines overlaid in a single frame travelling from their initial states to the final states. The events are randomly generated on these lines during the whole video time span which is 5s.

The parameter space is built up by 200×300 spiking neurons with the decaying rate of $3mv/ms$. As explained in section 5.4, the thresholds of the membrane potentials are chosen proportional to the line length ranging from $6mv$ to $18mv$. The inhibitory window is a simple rectangle of $10^o \times 5(pixels)$.

Figure 5.20 shows the results (in colored lines) of the normal distance and angle of the detected lines superimposed on the ground truth (in dashed lines). Table 5.3 reports some quantitative results of the experiment. The trivial errors in this table show the accuracy of the proposed algorithm. The most significant outcome of this

Table 5.3: Quantitative analysis of line detection results on the synthetic video of four moving lines

Line	Input Events#	Output Spikes#	Spike Time(ms) First/Last		θ error(degree) Mean/SD		r error(pixel) Mean/SD	
Cyan	9100	211	23	4995	0.04	0.25	0.03	0.27
Blue	14116	263	14	4974	0.06	0.19	0.15	0.23
Green	18022	412	21	4994	-0.02	0.23	-0.01	0.21
Red	31654	1037	4	4995	0.05	0.32	0.09	0.35

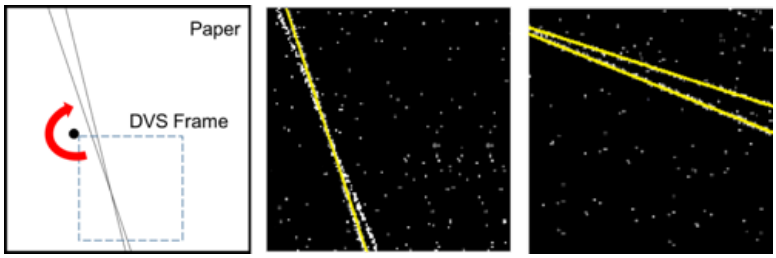


Figure 5.21: A point and two lines with the angle of five degrees were printed on a white paper. The point was pinned on a wall and DVS captured the clockwise rotation of the paper around the point. The algorithm results are shown for two instances of the video.

experiment is that small lines are detected using the proposed algorithm. This is shown when the cyan line is detected from almost beginning of the video whilst in section 5.8 cyan line was not detected until $t = 2630ms$.

5.12 Experiment IX

The effect of inhibitory window optimization is evaluated on real captured data using DVS. A point and two lines with the angle of five degrees are printed on a white paper as shown in 5.21. The page is pinned at the printed point on a wall and in front of a DVS. DVS position is adjusted to capture the pinned point at the upper left corner of the frame. Then the paper is rotated clockwise around the pin while DVS is capturing the video. All implementation details are the same with the previous experiment except the lateral inhibitory connection which are set by $D_{threshold} = 5\, pixels$ according to section 5.5.2.

The results are shown for two instances of the video. In early instances of rotation, the algorithm detected only one line in the video. After a while and by increasing the area between two lines, both lines were detected as illustrated in Figure 5.21. The ability of detecting lines which are in a close proximity in parameter space has been significantly improved in the revised algorithm compared to the initial version with a simple rectangular inhibitory window which detects only one line from the beginning to the end of this video stream.

5.13 Experiment X

In this experiment, we apply our extended event-based Hough transform on some samples from previous experiments. The objective of this experiment is to show how the extended event-based Hough transform can remove the noise and clean videos. This is done by detecting all small lines in the frame and removing remaining events from the video. Moreover, this experiment shows that the algorithm can be applied on any arbitrary shape e.g. circle to recognize its boundary as a series of small lines. We have chosen two noisy scene from experiment VI. Extended event-based Hough transform results are seen in figure 5.22. This experiment shows the capability of the algorithm in cleaning noisy videos. Another video is chosen from Experiment III. This video shows a moving circle from left to right. As we have extracted the transformation matrix from real coordinates to camera coordinates, we know the exact position of the circle center and we can calculate events distance from the circle center. Figure 5.23 shows the circle before and after applying the extended event-based Hough transform algorithm. We plot the events distance distribution from the circle center. As seen in figure 5.23, the distribution is more narrow around $5cm$ (circle radius) after cleaning the video. The circle boundary is recognized as a series of small lines. The lines distance from the circle center is $5.10 \pm 0.16cm$. Moreover, the error of these lines orientation with respect to the ground truth is 3^o. In future, we use extracted line elements to detect any arbitrary shapes and their parameters

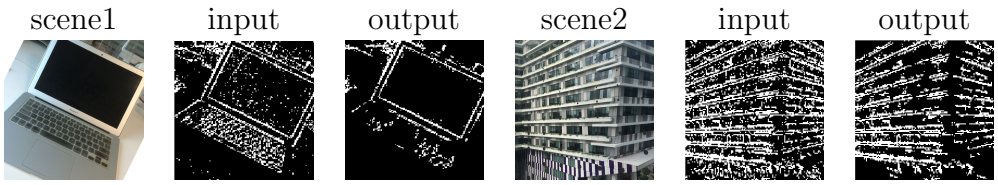


Figure 5.22: Extended event-based Hough transform results on two noisy scenes. The algorithm output contains less noise compared to the algorithm input.

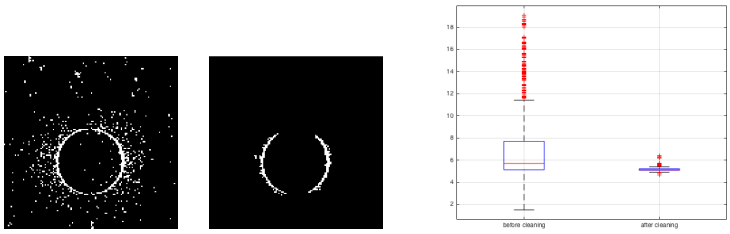


Figure 5.23: A video of a moving circle with the radius of 5cm. The extended event-based Hough transform is applied to detect the circle as a series of small lines. 40 spikes correspond to 40 small lines are received from the spiking neural network. The result shows more narrow distribution of events around the circle boundary

e.g. curvature.

5.14 Conclusion

An event-based Hough transform approach based on a spiking neural network was proposed to perform multiple lines detection and tracking on Dynamic Vision Sensors. Both parameter space and inhibitory connections were optimized for efficient implementation on Neuromorphic specific hardware (e.g. TrueNorth or FPGA) without any further preprocessing. This chapter is different compared to the proposed approach in Chapters 3 and 4 for investigating objects' motion and edge which requires buffering of the input over a longer duration.

In this thesis, we describe how our framework is moving from frame-based paradigm to event-based paradigm. Therefore, the further we go from Chapter 3 to Chapter 5, the lesser applicable comparison between the two families of algorithms (frame-based

and event-based) becomes. To compare with the state-of-the-art, in Chapter 3, we compare our algorithm with 9 other techniques from the literature to estimate the normal flow. In the next chapter, Chapter 4, we build upon our results from Chapter 3 to propose our algorithm for global flow estimation in event-based systems, and compare our estimated global flow with the ground truth. Next, in this chapter, we proposed re-constructing conventional Hough transform with Spiking Neural Networks (SNN), to allow its application on event-based systems, without losing its desired properties. While the proposed event-based HT in this chapter cannot be directly compared with frame-based techniques, for the sake of comparison, we conduct Experiment VI. In this experiment, we create a frame by collapsing all the events of a selected time-span, ignoring their temporal information. Then we apply frame-based HT on this frame, and compare the results against application of our event-based HT on the same time-span. Although the results of this experiment shows better performance from our algorithm, this is not to claim that our event-based technique performs better than state-of-the-art frame-based methods, as these two categories have different and non-comparable operating conditions. Therefore, an accurate frame-based algorithm cannot be applied to event-based systems, without undergoing fundamental changes, which does not guarantee the same level of performance of the converted algorithm. We can extend this event-based method to more complicated shapes e.g. curves as a part of our future work.

Chapter 6

Conclusion and Future Work

This chapter concludes the thesis and presents some areas as future works.

6.1 Conclusion

Dynamic Vision Sensor (DVS) is a relatively new event-based video camera. Comparing to a conventional frame-based camera, DVS offers great advantages in terms of power consumption, data rate, speed, and dynamic range. Simulating biological retinas, DVS sensors are sensitive to the intensity change rather than the absolute intensity value. Conventional cameras synchronously capture frames (e.g. 30 frames per second) with each frame containing the intensity values of all the pixels. In contrast, DVS sensors only report pixels with intensity changes and output them autonomously as an asynchronous stream of binary events, thus creating significantly less redundancy, and allowing high temporal resolution with low bit rate. These interesting advantages of DVS is applicable to many applications where conventional video capturing devices fail to perform, such as ultra-high-frame rate video-based control systems for fast moving objects, robotic actuators that should respond to rapid changes in the environment, or tracking micro particles.

Despite many desirable properties of DVS, one main difference between these cam-

eras and conventional cameras have prevented this field from rapid advancement. While conventional cameras operate in a frame-based output stream, DVS cameras provide event-based output stream, and therefore, conventional image processing algorithms cannot be directly applied to DVS output. The difference between frame-based paradigm and event-based paradigm have resulted in a significant gap between the image processing algorithms applicable to conventional camera output and DVS camera output. In this work, we intend to reduce this gap, by proposing several algorithms for low level processing of DVS videos.

We proposed a procedure for analysing objects motion in a DVS video and localizing their edges. For motion analysis, we exploited the principal component analysis of the events to extract the properties of the events plane in 3-D spatio-temporal space. We distinguished the normal velocity to the edge from the velocity along the edge and tackled them separately for more accurate results. We used the events plane orientation to estimate the normal velocity locally and considered the whole object movement to extract the actual velocity globally. We investigated the time and location discretization effects on the final results and showed that the spatial quantization error is more important than the temporal quantization error. In addition, we introduced an algorithm based on the regional exposure time followed by an edge-dependent Gaussian filtering for detecting the edges.

As another approach to edge and motion analysis of objects in event-based videos, an event-based Hough transform using a spiking neural network (SNN) was proposed to perform multiple line detection and tracking for Dynamic Vision Sensors. SNN with local lateral inhibition was efficient in detecting correct lines as well as suppressing incorrect ones, while event-based clustering on the SNN output spikes allowed efficient tracking of the detected multiple lines. Subsequently we proposed some improvements to detect very small lines in frame corners. Moreover, we could reduce the detection noise by suppressing the lines which were visually close together in the Cartesian space. Our method is able to perform this without interfering with lines that are in close proximity in the parameter space, but not in the Cartesian space. Extensive

experiments on both artificial and real DVS events streams demonstrated the efficacy of algorithms presented in this thesis.

Our framework can serve as a methodology for analytical assessment of previously proposed event-based optical flow algorithms, that can precisely describe behaviour of these algorithms, and their reported result. Until now, this has been a missing component in the literature, where analytical assessment of algorithms was substituted by estimation techniques. In contrast, our framework allows such analytical assessment to obtain insights from system behaviour and subsequently implementing new algorithms for that system. The framework proposed in this work is however far from complete. There are several possible avenues of improvement that are discussed in the next section.

6.2 Recommendations for Future Research

Although we have good results on simple shapes with uniform textures currently, we need to extend these issues in future for complex objects.

- We explained the effect of quantization error in our results and stated that it is one of the intrinsic properties of the DVS. If we know this error, we can improve our estimations. In future we should investigate this error more to know what parameters are effective on that. Currently we can mention some of them e.g. edge velocity and direction, focusing the DVS, lighting of the scene, etc.

- We know that for some purposes e.g. human action recognition we need the whole body in the captured video. There are many details inside a body shape e.g. different colors, textures, etc and an object with non-uniform texture has many events inside. Figure 6.1 shows a walking person in a DVS video. We focus on a small part of his head. As seen in Figure 6.2(a), there is no clear edge inside the window. Alternatively we have a boundary full of events. We can consider a line or a curve to determine its boundary as shown in figure

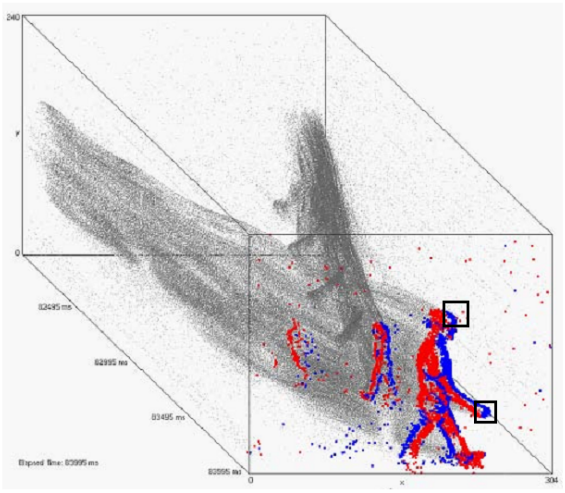
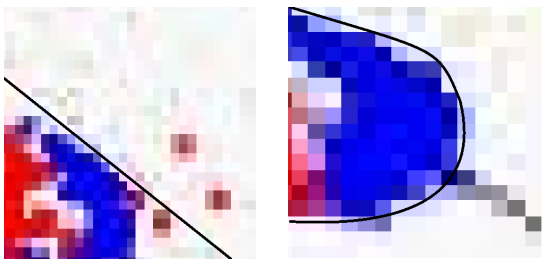


Figure 6.1: A walking man in a DVS video. There is no clear edge in a small window on the head while there is a curve edge in a small window on the hand

6.2(a). One side of the line is inside the body full of events and the other side is outside of the body with a few events. We can develop an algorithm sensitive to the boundary events rather than all the events. Based on the analysis of edge velocity and edge boundary, we can also extend the approach for connected edge solution . This could be used to filter out those non-boundary events. SVM can be an appropriate solution for this issue.

- As stated before, for human action recognition in video, we usually need to have



(a) A small window of head area. One-sided tangent line

(b) A small window of hand area. Curve fitting on the boundary

Figure 6.2: Two small windows of head and hand areas of the body

the whole human body in the video frame. Since the spatial resolution of the DVS is relatively low (128×128), the body parts e.g. head or hands are very small in the captured video about a few pixels. The mathematical explanation of this issue is that the boundary curvature of these areas is high. Figure 6.2(b) Shows a small window on the hand area of a walking man in Figure 6.1.

The edges with higher curvature can not be assumed as lines unless we decrease the window size and bear more error in our final results. We can use higher order of approximation to solve this problem. i.e. we assume the edge is a curve of order two or higher rather than a line within a window.

The temporal resolution of a DVS is high enough that we can assume the edge has a constant velocity and a fixed topology in a small spatio-temporal window. But its spatial resolution is relatively low. Accordingly we may have a curve edge in a spatio-temporal window rather than a line. We can consider the edge equation in a window as follows:

$$ax^2 + by^2 + cxy + dx + ey = f$$

In the above equation a, b, c, d, e define the topology of a quadratic curve and f defines its position. For a moving curve we can make parameter f dependent on the time and velocity. We should adapt the parameters based on the events within the spatio-temporal window. Since we have more parameters here compared to linear PCA, we should use more events to estimate them. By considering the edges as curve, we can choose a larger spatio-temporal window with more events and obtain more accurate results.

- Our event-based Hough transform was extracting linear properties of the event-based videos including lines parameters, tracking and velocities. As an enhancement for this work, the next step is extracting the curve properties of objects boundaries in event-based videos. For this purpose, we need to work on higher orders of Hough transform and subsequently higher dimension of spiking neural

network. This stage is important for obtaining the properties of whole objects boundaries in the video and using them for detection/recognition purpose in the future.

- The most important advantage of event-based cameras is their high temporal resolution ($1\mu s$). This characteristic allows us to track very fast moving objects which is not possible in conventional frame-based cameras. Assuming a mark on the blade of a rotating fan, it has a significant displacement between 2 frames of a conventional video. As a result a conventional tracking algorithm will possibly miss the mark. Even if the mark is not missed, we will not know its position at any arbitrary instance. This problem can be solved by a DVS which has a continuous stream of events and reports all dynamics with a great temporal resolution. There are some event-based tracking algorithms in the literature. However we can improve these algorithms significantly using the results we currently have.

- Despite of a high temporal resolution, we miss a lot of details e.g. color, texture, fixed objects, etc. in a DVS video. An interesting idea for potential future work is co-capturing of a normal camera and DVS and correlating frame-based and event-based data streams, that simulate high-frame-rate normal camera.

- We have already extracted some features from event-based videos which can be used for detection/recognition purpose. We can use these features for a more complicated task e.g. object recognition in future.

- The image contrast around the edge can affect our event-based Hough transform since the events rate is highly dependent to objects color/brightness. In future, the algorithm can be revised again to detect the linear edges with different contrasts or the partial lines in the frame.

List of Publications

[1] **Sajjad Seifozzakerini**, Wei-Yun Yau, Bo Zhao, Kezhi Mao, "Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor" *The British Machine Vision Conference (BMVC 2016)*

[2] **Sajjad Seifozzakerini**, Wei-Yun Yau, Kezhi Mao, "Effect of Inhibitory Window on Event-Based Hough Transform for Multiple Lines Detection" *International Conference on Advances in Image Processing (ICAIP 2017)*

[3] **Sajjad Seifozzakerini**, Wei-Yun Yau, Kezhi Mao, "Motion analysis and edge detection of the event-based videos" Submitted to the *Journal of Neurocomputing*, Under Review

[4] **Sajjad Seifozzakerini**, Wei-Yun Yau, Kezhi Mao, Hossein Nejati, "Extended Event-Based Hough Transform in a Spiking Neural Network" Invited paper to "Neural Living Rooms", a research topic in *Frontiers in Computational Neuroscience*, Under Review

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 X 128 120 dB 15 us latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, feb 2008. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=4444573

[2] G. Orchard and R. Etienne-Cummings, "Bioinspired Visual Motion Estimation," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1520–1536, oct 2014. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6891162

[3] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," *2007 IEEE International Symposium on Circuits and Systems, New Orleans, LA*, no. 80 cm, pp. 845–848, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4252767

[4] B. Zhao, X. Zhang, S. Chen, K.-S. Low, and H. Zhuang, "A 64 X 64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 4, pp. 581–588, apr 2012. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6032080

[5] D. R. Valeiras, X. Lagorce, X. Clady, C. Bartolozzi, S.-H. Ieng, and R. Benosman, "An Asynchronous Neuromorphic Event-Driven Visual Part-Based Shape Tracking." *IEEE transactions on neural networks and learning systems*, mar 2015. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=7063246

[6] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems." *Frontiers in neuroscience*, vol. 7, p. 272, jan 2013. [Online]. Available: /pmc/articles/PMC3922083/?report=abstract

[7] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets." *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2706–19, nov 2013. [Online]. Available: http://www.computer.org/csdl/trans/tp/2013/11/ttp2013112706.html

[8] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network." *IEEE Trans. on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1963–1978, oct 2015. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6933869

[9] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91–106, 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0893608015000477

[10] L. Lapique., "Sur l'excitation electrique des nerfs." *Journal of Physiology*, pp. 620–635, 1907.

[11] A. L. Hodgkin and A. F. Huxley, "Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo," *The Journal of Physiology*, vol. 116, no. 4, pp. 449–472, apr 1952. [Online]. Available: http://doi.wiley.com/10.1113/jphysiol.1952.sp004717

[12] E. R. Lewis, "An electronic model of neuroelectric point processes," *Kybernetik*, vol. 5, no. 1, pp. 30–46, jul 1968. [Online]. Available: http://link.springer.com/10.1007/BF00288896

[13] R. Johnson and G. Hanna, "Membrane model: A single transistor analog of excitable membrane," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 401–411, mar 1969. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0022519369900125

[14] G. Roy, "A Simple Electronic Analog of the Squid Axon Membrane: The NEUROFET," *IEEE Transactions on Biomedical Engineering*, vol. BME-19, no. 1, pp. 60–63, jan 1972. [Online]. Available: http://ieeexplore.ieee.org/document/4120472/

[15] W. H. Brockman, "A Simple Electronic Neuron Model Incorporating Both Active and Passive Responses," *IEEE Transactions on Biomedical Engineering*, vol. BME-26, no. 11, pp. 635–639, nov 1979. [Online]. Available: http://ieeexplore.ieee.org/document/4122960/

[16] R. G. Runge, M. Uemura, and S. S. Viglione, "Electronic Synthesis of the Avian Retina," *IEEE Transactions on Biomedical Engineering*, vol. BME-15, no. 3, pp. 138–151, jul 1968. [Online]. Available: http://ieeexplore.ieee.org/document/4502558/

[17] W. Karplus and W. Soroka, "Analog methods: Computation and simulation," 1959. [Online]. Available: https://scholar.google.com/scholar?q=W.J.KarplusandW.W.Soroka.AnalogMethods{%}3AComputationandSimulation.McGraw-Hill{%}2C1959.

[18] C. Mead and L. Conway, *Introduction to VLSI systems*. Addison-Wesley Reading, MA, 1980, vol. 1080.

[19] C. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley Reading, MA, 1989.

[20] S.-C. Liu, T. Delbruck, J. Kramer, G. Indiveri, R. Douglas, and A. VLSI, "Circuits and Principles," 2002.

[21] C. M. Markan and P. Gupta, "Neuromorphic building blocks for adaptable cortical feature maps," in *2007 IFIP International Conference on Very Large Scale Integration*. IEEE, oct 2007, pp. 7–12. [Online]. Available: http://ieeexplore.ieee.org/document/4402464/

[22] Z. Yang, A. Murray, F. Worgotter, K. Cameron, and V. Boonsobhak, "A Neuromorphic Depth-From-Motion Vision Model With STDP Adaptation," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 482–495, mar 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1603632/

[23] R. Harrison and C. Koch, "An analog VLSI model of the fly elementary motion detector," *Advances in neural information processing*, 1998. [Online]. Available: http://papers.nips.cc/paper/1367-an-analog-vlsi-model-of-the-fly-elementary-motion-detector.pdf

[24] G. Indiveri, "Analog vlsi model of locust dcmd neuron response for computation of object approach," *PROGRESS IN NEURAL PROCESSING*, 1998. [Online]. Available: https://books.google.com/books?hl=en{&}lr={&}id=KJvVCgAAQBAJ{&}oi=fnd{&}pg=PA47{&}dq=G.+Indiveri.+Analogue+VLSI+model+of+locust+DCMD+neuron+response+for+computation+of+object+aproach.+In+L.S.+Smith+and+A.+Hamilton,+editors,+Neuromorphic+Systems:+engineering+silicon

[25] A. Andreou, R. Meitzler, K. Strohbehn, and K. Boahen, "Analog VLSI neuromorphic image acquisition and pre-processing systems," *Neural*

Networks, vol. 8, no. 7-8, pp. 1323–1347, jan 1995. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0893608095000984

[26] T. Delbruck and C. Mead, "Analog VLSI transduction," 1996. [Online]. Available: https://scholar.google.com/scholar?q=T.DelbruckandC.A.Mead.AnalogVLSItransduction.TechnicalReportCNSMemo30{%}2CCaliforniaInstituteofTechnologyComputationandneuralSystemsProgram{%}2CPasadenaCalifornia{%}2CUSA{%}2CApril1996.

[27] C. A. Mead and M. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, no. 1, pp. 91–97, jan 1988. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/089360808890024X

[28] H. V. Shurmer and J. W. Gardner, "Odour discrimination with an electronic nose," *Sensors and Actuators B: Chemical*, vol. 8, no. 1, pp. 1–11, apr 1992. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/092540059285001D

[29] T. Pearce, "Computational parallels between the biological olfactory pathway and its analogue 'The Electronic Nose': Part II. Sensor-based machine olfaction," *Biosystems*, vol. 41, no. 2, pp. 69–90, jan 1997. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0303264796016607

[30] T. J. Koickal, A. Hamilton, S. L. Tan, J. A. Covington, J. W. Gardner, and T. C. Pearce, "Analog VLSI Circuit Implementation of an Adaptive Neuromorphic Olfaction Chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 60–73, jan 2007. [Online]. Available: http://ieeexplore.ieee.org/document/4061019/

[31] M. Glover, A. Hamilton, and L. S. Smith, "Analogue VLSI Leaky Integrate-and-Fire Neurons and Their Use in a Sound Analysis System," *Analog Integrated Circuits and Signal Processing*, vol. 30, no. 2, pp. 91–100, 2002. [Online]. Available: http://link.springer.com/10.1023/A:1013747426448

[32] A. V. Schaik, E. Fragnière, and E. Vittoz, "A silicon model of amplitude modulation detection in the auditory brainstem," *Advances in Neural*, 1997. [Online]. Available: http://papers.nips.cc/paper/1265-a-silicon-model-of-amplitude-modulation-detection-in-the-auditory-brainstem.pdf

[33] A. van Schaik, E. Fragniere, and E. Vittoz, "An analogue electronic model of Ventral Cochlear Nucleus neurons," in *Proceedings of Fifth International Conference on Microelectronics for Neural Networks.* IEEE Comput. Soc. Press, pp. 52–59. [Online]. Available: http://ieeexplore.ieee.org/document/493772/

[34] T. J. Hamilton, C. Jin, A. van Schaik, and J. Tapson, "An Active 2-D Silicon Cochlea," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 1, pp. 30–43, mar 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4489970/

[35] E. Fragnière, A. van Schaik, and E. A. Vittoz, "Design of an Analogue VLSI Model of an Active Cochlea," *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1/2, pp. 19–35, 1997. [Online]. Available: http://link.springer.com/10.1023/A:1008234622348

[36] J. Lazzaro and J. Wawrzynek, "Speech Recognition Experiments with Silicon Auditory Models," *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1/2, pp. 37–51, 1997. [Online]. Available: http://link.springer.com/10.1023/A:1008259307326

[37] W. Liu, A. Andreou, and M. G. Jr, "Analog cochlear model for multiresolution speech analysis," *Advances in Neural*, 1993. [Online]. Available: http://papers.nips.cc/paper/697-analog-cochlear-model-for-multiresolution-speech-analysis.pdf

[38] W. Liu, A. Andreou, and M. Goldstein, "Voiced-speech representation by an analog silicon model of the auditory periphery," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 477–487, may 1992. [Online]. Available: http://ieeexplore.ieee.org/document/129420/

[39] J. Lazzaro and C. A. Mead, "A Silicon Model Of Auditory Localization," *Neural Computation*, vol. 1, no. 1, pp. 47–57, mar 1989. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.1.47

[40] C. Mead and M. Ismail, Eds., *Analog VLSI Implementation of Neural Systems*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, MA: Springer US, 1989, vol. 80. [Online]. Available: http://link.springer.com/10.1007/978-1-4613-1639-8

[41] P. Furth and A. Andreou, "A design framework for low power analog filter banks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, no. 11, pp. 966–971, 1995. [Online]. Available: http://ieeexplore.ieee.org/document/477209/

[42] R. Lyon and C. Mead, "An analog electronic cochlea," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1119–1134, jul 1988. [Online]. Available: http://ieeexplore.ieee.org/document/1639/

[43] M. Pearson, M. Nibouche, I. Gilhespy, K. Gurney, C. Melhuish, B. Mitchinson, and A. Pipe, "A Hardware Based Implementation of a Tactile

Sensory System For Neuromorphic Signal Processing Applications," in *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, vol. 4. IEEE, pp. IV–1153–IV–1156. [Online]. Available: http://ieeexplore.ieee.org/document/1661178/

[44] P. Argyrakis, A. Hamilton, B. Webb, Y. Zhang, T. Gonos, and R. Cheung, "Fabrication and characterization of a wind sensor for integration with a neuron circuit," *Microelectronic Engineering*, vol. 84, no. 5-8, pp. 1749–1753, may 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167931707002729

[45] D. Roy, "Design and Developmental Metrics of a Skin-Like' Multi-Input Quasi-Compliant Robotic Gripper Sensor Using Tactile Matrix," *Journal of Intelligent and Robotic Systems*, vol. 46, no. 4, pp. 305–337, aug 2006. [Online]. Available: http://link.springer.com/10.1007/s10846-006-9062-4

[46] G. Vasarhelyi, M. Adam, E. Vazsonyi, Z. Vizvary, A. Kis, I. Barsony, and C. Ducso, "Characterization of an Integrable Single-Crystalline 3-D Tactile Sensor," *IEEE Sensors Journal*, vol. 6, no. 4, pp. 928–934, aug 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1661573/

[47] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input." *Biological cybernetics*, vol. 95, no. 1, pp. 1–19, jul 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16622699

[48] C. R. Laing and A. Longtin, "Dynamics of Deterministic and Stochastic Paired ExcitatoryInhibitory Delayed Feedback," *Neural Computation*, vol. 15, no. 12, pp. 2779–2822, dec 2003. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/089976603322518740

[49] A. N. Burkitt, "A review of the integrate-and-fire neuron model: II. Inhomogeneous synaptic input and network properties," *Biological Cybernetics*, vol. 95, no. 2, pp. 97–112, aug 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16821035http://link.springer.com/10.1007/s00422-006-0082-8

[50] E. M. Izhikevich, "Simple model of spiking neurons." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 14, no. 6, pp. 1569–72, jan 2003. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18244602

[51] E. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–70, sep 2004. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15484883http://eaton.math.rpi.edu/CSUMS/Papers/Neuro/Izhikevich04.pdf

[52] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, aug 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6515159

[53] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "Artificial brains. A million spiking-neuron integrated circuit with a scalable communication network and interface." *Science (New York, N.Y.)*, vol. 345, no. 6197, pp. 668–73, aug 2014. [Online]. Available: http://science.sciencemag.org.ezlibproxy1.ntu.edu.sg/content/345/6197/668.abstract

[54] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons." *IEEE transactions on neural networks and learning systems*, vol. 24, no. 10, pp. 1539–52, oct 2013. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6469239

[55] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting." *Neural computation*, vol. 22, no. 2, pp. 467–510, feb 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19842989

[56] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Precise-spike-driven synaptic plasticity: learning hetero-association of spatiotemporal spike patterns." *PloS one*, vol. 8, no. 11, p. e78318, jan 2013. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3818323{&}tool=pmcentrez{&}rendertype=abstract

[57] J. Hu, H. Tang, K. C. Tan, H. Li, and L. Shi, "A spike-timing-based integrated model for pattern recognition." *Neural computation*, vol. 25, no. 2, pp. 450–72, feb 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23148414

[58] S. Chen, P. Akselrod, B. Zhao, J. A. P. Carrasco, B. Linares-Barranco, and E. Culurciello, "Efficient feedforward categorization of objects and human postures with address-event image sensors." *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 2, pp. 302–14, feb 2012. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=5871650

[59] R. Gutig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing-based decisions." *Nature neuroscience*, vol. 9, no. 3, pp. 420–8, mar 2006. [Online]. Available: http://www.nature.com.ezlibproxy1.ntu.edu.sg/neuro/journal/v9/n3/full/nn1643.htmlhttp://www.ncbi.nlm.nih.gov/pubmed/16474393

[60] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, may 2010, pp. 2426–2429. [Online]. Available: http://ieeexplore.ieee.org/document/5537149/

[61] J. A. Lenero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6 us Latency Asynchronous Frame-Free Event-Driven Dynamic-Vision-Sensor," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1443–1455, jun 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5746543/

[62] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128 X 128 1.5% Contrast Sensitivity 0.9% FPN 3 us Latency 4 mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, mar 2013. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6407468

[63] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 X 128 120db 30mw asynchronous vision sensor that responds to relative intensity change," in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*. IEEE, 2006, pp. 2060–2069. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=1696265

[64] R. Berner, C. Brandli, M. Yang, and S. Liu, "A 240X180 10mW 12us latency sparse-output vision sensor for mobile applications," *VLSI Circuits (VLSIC),*, 2013. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6578656/

[65] C. Brandli, R. Berner, Minhao Yang, Shih-Chii Liu, and T. Delbruck, "A 240X180 130 dB 3 us Latency Global Shutter Spatiotemporal Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, oct 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6889103

[66] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, 2014.

[67] R. D. Scaramuzza and U. Floreano, "High-Speed Pose Estimation using a Dynamic Vision Sensor," Master Thesis, University of Zurich, 2014. [Online]. Available: http://www.kutter-fonds.ethz.ch/App{_}Themes/default/datalinks/BasilHuber{_}UniZ{_}MT2014.pdf

[68] D. Bauer, A. N. Belbachir, N. Donath, G. Gritsch, B. Kohn, M. Litzenberger, C. Posch, P. Schön, and S. Schraml, "Embedded Vehicle Speed Estimation System Using an Asynchronous Temporal Contrast Vision Sensor," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1, pp. 1–12, jan 2007. [Online]. Available: http://dl.acm.org.ezlibproxy1.ntu.edu.sg/citation.cfm?id=1287505.1287539

[69] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor." *Frontiers in neuroscience*, vol. 7, p. 223, jan 2013. [Online]. Available: http://journal.frontiersin.org/Journal/10.3389/fnins.2013.00223/abstract

[70] Z. NI, C. PACORET, R. BENOSMAN, S. IENG, and S. RÉGNIER, "Asynchronous event-based high speed vision for microparticle tracking," *Journal of Microscopy*, vol. 245, no. 3, pp. 236–244, mar 2012. [Online]. Available: http://doi.wiley.com/10.1111/j.1365-2818.2011.03565.x

[71] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking." *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1710–20, aug 2015. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6899691

[72] B. Kohn, A. N. Belbachir, T. Hahn, and H. Kaufmann, "Event-driven body motion analysis for real-time gesture recognition," in *2012 IEEE International Symposium on Circuits and Systems*. IEEE, may 2012, pp. 703–706. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6272132

[73] D. Weikersdorfer and J. Conradt, "Event-based particle filtering for robot self-localization," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, dec 2012, pp. 866–870. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6491077

[74] M. Chen, B. Leibe, and B. Neumann, Eds., *Computer Vision Systems*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7963. [Online]. Available: http://link.springer.com/10.1007/978-3-642-39402-7

[75] X. Clady, C. Clercq, S.-H. Ieng, F. Houseini, M. Randazzo, L. Natale, C. Bartolozzi, and R. Benosman, "Asynchronous visual event-based time-to-contact." *Frontiers in neuroscience*, vol. 8, p. 9, jan 2014. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3916774{&}tool=pmcentrez{&}rendertype=abstract

[76] C. Brandli, T. A. Mantel, M. Hutter, M. A. Höpflinger, R. Berner, R. Siegwart, and T. Delbruck, "Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor." *Frontiers in neuroscience*, vol. 7, p. 275, jan 2013. [Online]. Available: http://journal.frontiersin.org/Journal/10.3389/fnins.2013.00275/abstract

[77] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow." *Neural networks : the official journal of the International Neural Network Society*, vol. 27, pp. 32–7, mar 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608011002930

[78] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow." *IEEE transactions on neural networks and learning systems*, vol. 25, no. 2, pp. 407–17, feb 2014. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=6589170

[79] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, vol. 2, pp. 674–679, aug 1981. [Online]. Available: http://dl.acm.org.ezlibproxy1.ntu.edu.sg/citation.cfm?id=1623264.1623280

[80] H.-H. Nagel, "Displacement vectors derived from second-order intensity variations in image sequences," *Computer Vision, Graphics, and Image Processing*, vol. 21, no. 1, pp. 85–117, jan 1983. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0734189X83800309

[81] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, aug 1981. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0004370281900242

[82] A. YAMAMOTO, K. YAGI, K. SEKIOKA, M. TOKUDA, Y. SAWAKI, and T. INABA, "Analysis of Left Ventricular Wall Locus Tracing and Motion by optical-flow using MRI," *Journal of the Visualization Society of Japan*, vol. 18, no. Supplement2, pp. 99–100, 1998. [Online]. Available: http://joi.jlc.jst.go.jp/JST.Journalarchive/jvs1990/18.Supplement2{_}99?from=CrossRef

[83] S. Uras, F. Girosi, A. Verri, and V. Torre, "A computational approach to motion perception," *Biological Cybernetics*, vol. 60, no. 2, pp. 79–87, dec 1988. [Online]. Available: http://link.springer.com/10.1007/BF00202895

[84] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, apr 1983. [Online]. Available: http://ieeexplore.ieee.org/document/1095851/

[85] F. Glazer, G. Reynolds, and P. Anandan, "Scene Matching by Hierarchical Correlation,," 1983. [Online]. Available: http://www.dtic.mil/docs/citations/ADP001212

[86] J. Little and A. Verri, "Analysis of differential and matching methods for optical flow," in *[1989] Proceedings. Workshop on Visual Motion.* IEEE Comput. Soc. Press, pp. 173–180. [Online]. Available: http://ieeexplore.ieee.org/document/47107/

[87] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 283–310, jan 1989. [Online]. Available: http://link.springer.com/10.1007/BF00158167

[88] E. Adelson and J. Bergen, "The extraction of spatiotemporal energy in human and machine vision," *Proc. IEEE Workshop on Visual Motion*, 1986. [Online]. Available: https://scholar.google.com/scholar?q=Adelson{%}2CE.H.{%}2CandBergen{%}2CJ.R.1986.Theextractionofspatiotemporalenergyinhumanandmachinevision{%}2CProc.IEEEWorkshoponVisualMotion{%}2CCharleston{%}2Cpp.151156.

[89] H. Barman, L. Haglund, H. Knutsson, and G. Granlund, "Estimation of velocity, acceleration and disparity in time sequences," in *Proceedings of the IEEE Workshop on Visual Motion.* IEEE Comput. Soc. Press, 1991, pp. 44–51. [Online]. Available: http://ieeexplore.ieee.org/document/212789/

[90] J. Bigun, G. Granlund, and J. Wiklund, "Multidimensional orientation estimation with applications to texture analysis and optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 775–790, 1991. [Online]. Available: http://ieeexplore.ieee.org/document/85668/

[91] B. Jahne, "Image sequence analysis of complex physical objects: nonlinear small scale water surface waves," *Proc. IEEE ICCV London*, 1987. [Online]. Available: https://scholar.google.com/scholar?q=Jahne{%}2CB.1987.Imagesequenceanalysisofcomplexphysicalobjects{%}

3Anonlinearsmallscalewatersurfacewaves{%}2CProc.1stIntern.Conf.Comput. Vis.{%}2CLondon{%}2Cpp.191200.

[92] D. J. Heeger, "Optical flow using spatiotemporal filters," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 279–302, jan 1988. [Online]. Available: http://link.springer.com/10.1007/BF00133568

[93] D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 77–104, aug 1990. [Online]. Available: http://link.springer.com/10.1007/BF00056772

[94] A. Waxman, J. Wu, and F. Bergholm, "Convected activation profiles and the measurement of visual motion," in *Proceedings CVPR '88: The Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc. Press, pp. 717–723. [Online]. Available: http://ieeexplore.ieee.org/document/196313/

[95] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, feb 1994. [Online]. Available: http://link.springer.com/10.1007/BF01420984

[96] D. Sun, S. Roth, and M. J. Black, "A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them," *International Journal of Computer Vision*, vol. 106, no. 2, pp. 115–137, jan 2014. [Online]. Available: http://link.springer.com/10.1007/s11263-013-0644-x

[97] B. Rueckauer and T. Delbruck, "Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor," *Frontiers in Neuroscience*, vol. 10, p. 176, apr 2016. [Online]. Available: http://journal.frontiersin.org/Article/10.3389/fnins.2016.00176/abstract

[98] T. Delbrück, "Frame-free dynamic digital vision," 2008.

[99] T. Delbruck, "Fun with Asynchronous Vision Sensors and Processing." Springer, Berlin, Heidelberg, 2012, pp. 506–515. [Online]. Available: http://link.springer.com/10.1007/978-3-642-33863-2{_}52

[100] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Frontiers in Neuroscience*, vol. 9, p. 137, apr 2015. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2015.00137/abstract

[101] D. J Thornley, *Anisotropic Multidimensional Savitzky Golay kernels for Smoothing, Differentiation and Reconstruction*, jan 2006.

[102] P. V. C. Hough, "METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS," 1962.

[103] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, jan 1972. [Online]. Available: http://dl.acm.org.ezlibproxy1.ntu.edu.sg/citation.cfm?id=361237.361242

[104] D. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, jan 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0031320381900091

[105] R. K. Yip, P. K. Tam, and D. N. Leung, "Modification of hough transform for circles and ellipses detection using a 2-dimensional array," *Pattern Recognition*, vol. 25, no. 9, pp. 1007–1022, sep 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/003132039290064P

[106] H. Li, M. A. Lavin, and R. J. Le Master, "Fast Hough transform: A hierarchical approach," *Computer Vision, Graphics, and Image Processing*, vol. 36, no. 2-3, pp. 139–161, nov 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0734189X86900733

[107] P. Mukhopadhyay and B. B. Chaudhuri, "A Survey of Hough Transform," *Pattern Recogn.*, vol. 48, no. 3, pp. 993–1010, mar 2015. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2014.08.027

[108] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, jan 1972. [Online]. Available: http://doi.acm.org/10.1145/361237.361242

[109] D. J. Kerbyson and T. J. Atherton, "Circle detection using Hough transform filters," in *Fifth International Conference on Image Processing and its Applications, 1995.*, jul 1995, pp. 370–374.

[110] G. C. Stockman and A. K. Agrawala, "Equivalence of Hough Curve Detection to Template Matching," *Commun. ACM*, vol. 20, no. 11, pp. 820–822, nov 1977. [Online]. Available: http://doi.acm.org/10.1145/359863.359882

[111] J. Illingworth and J. Kittler, "A Survey of the Hough Transform," *Comput. Vision Graph. Image Process.*, vol. 44, no. 1, pp. 87–116, aug 1988. [Online]. Available: http://dx.doi.org/10.1016/S0734-189X(88)80033-1

[112] D. C. W. Pao, H. F. Li, and R. Jayakumar, "Shapes Recognition Using the Straight Line Hough Transform: Theory and Generalization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 11, pp. 1076–1089, nov 1992. [Online]. Available: https://doi.org/10.1109/34.166622

[113] C. Galamhos, J. Matas, and J. Kittler, "Progressive probabilistic Hough transform for line detection," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, 1999, p. 560 Vol. 1.

[114] L. Xu, E. Oja, and P. Kultanen, "A New Curve Detection Method: Randomized Hough Transform (RHT)," *Pattern Recogn. Lett.*, vol. 11, no. 5, pp. 331–338, may 1990. [Online]. Available: http://dx.doi.org/10.1016/0167-8655(90)90042-Z

[115] J. R. Bergen and H. Shvaytser(Schweitzer), "A Probabilistic Algorithm for Computing Hough Transforms," *J. Algorithms*, vol. 12, no. 4, pp. 639–656, dec 1991. [Online]. Available: http://dx.doi.org/10.1016/0196-6774(91)90037-Y

[116] P. M. Merlin and D. J. Farber, "A Parallel Mechanism for Detecting Curves in Pictures," *IEEE Trans. Comput.*, vol. 24, no. 1, pp. 96–98, jan 1975. [Online]. Available: https://doi.org/10.1109/T-C.1975.224087

[117] H. Hakalahti, D. Harwood, and L. S. Davis, "Two-dimensional Object Recognition by Matching Local Properties of Contour Points," *Pattern Recogn. Lett.*, vol. 2, no. 4, pp. 227–234, jun 1984. [Online]. Available: http://dx.doi.org/10.1016/0167-8655(84)90029-1

[118] V. F. Leavers, "Active intelligent vision using the dynamic generalized Hough Transform," in *Proc. BMVC*, 1990, pp. 11.1–11.6.

[119] D.-M. Tsai, "An improved generalized Hough transform for the recognition of overlapping objects," *Image and Vision Computing*, vol. 15, no. 12, pp. 877–888, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885697000334

[120] A. Kimura and T. Watanabe, "An Extension of the Generalized Hough Transform to Realize Affine-Invariant Two-dimensional (2D) Shape Detection," in *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 1 - Volume 1*, ser. ICPR '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 10 065—-. [Online]. Available: http://dl.acm.org/citation.cfm?id=839290.842616

[121] J. Illingworth and J. Kittler, "The Adaptive Hough Transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 690–698, may 1987. [Online]. Available: https://doi.org/10.1109/TPAMI.1987.4767964

[122] S.-C. Jeng and W.-H. Tsai, "Scale- and Orientation-invariant Generalized Hough Transform&Mdash;a New Approach," *Pattern Recogn.*, vol. 24, no. 11,

pp. 1037–1051, nov 1991. [Online]. Available: http://dx.doi.org/10.1016/0031-3203(91)90120-T

[123] A. Beinglass and H. J. Wolfson, "Articulated object recognition, or: how to generalize the generalized Hough transform," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, jun 1991, pp. 461–466.

[124] A. Samal and J. Edwards, "Generalized Hough Transform for Natural Shapes," *Pattern Recogn. Lett.*, vol. 18, no. 5, pp. 473–480, may 1997. [Online]. Available: http://dx.doi.org/10.1016/S0167-8655(97)00023-8

[125] N. Bonnet, "An unsupervised generalized Hough transform for natural shapes," *Pattern Recognition*, vol. 35, no. 5, pp. 1193–1196, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320301002199

[126] R.-C. Lo and W.-H. Tsai, "Perspective-transformation-invariant generalized hough transform for perspective planar shape detection and matching," *Pattern Recognition*, vol. 30, no. 3, pp. 383–396, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320396000945

[127] N. Kiryati, M. Lindenbaum, and A. M. Bruckstein, "Digital or Analog Hough Transform?" *Pattern Recogn. Lett.*, vol. 12, no. 5, pp. 291–297, may 1991. [Online]. Available: http://dx.doi.org/10.1016/0167-8655(91)90412-F

[128] J. A. O. David Cyganski William F. Noel, "Analytic Hough transform," *Proc.SPIE*, vol. 1260, pp. 1260 – 1260 – 12, 1990. [Online]. Available: https://doi.org/10.1117/12.20013

[129] R. Klette and A. Rosenfeld, "Digital Straightness: A Review," *Discrete Appl. Math.*, vol. 139, no. 1-3, pp. 197–230, apr 2004. [Online]. Available: http://dx.doi.org/10.1016/j.dam.2002.12.001

[130] L. Dorst and A. W. M. Smeulders, "Discrete Representation of Straight Lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 4, pp. 450–463, apr 1984. [Online]. Available: https://doi.org/10.1109/TPAMI.1984.4767550

[131] I. D. Svalbe, "Natural Representations for Straight Lines and the Hough Transform on Discrete Arrays," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 9, pp. 941–950, sep 1989. [Online]. Available: https://doi.org/10.1109/34.35497

[132] E. R. Davies, "Image Space Transforms for Detecting Straight Edges in Industrial Images," *Pattern Recogn. Lett.*, vol. 4, no. 3, pp. 185–192, jul 1986. [Online]. Available: http://dx.doi.org/10.1016/0167-8655(86)90018-8

[133] P. K. Ser and W. C. Siu, "Sampling Hough algorithm for the detection of lines and curves," in *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, vol. 5, may 1992, pp. 2497–2500 vol.5.

[134] P.-K. Ser and W.-C. Siu, "A New Generalized Hough Transform for the Detection of Irregular Objects," *Journal of Visual Communication and Image Representation*, vol. 6, no. 3, pp. 256–264, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S104732038571022X

[135] B. Gatos, S. J. Perantonis, and N. Papamarkos, "Accelerated Hough transform using rectangular image decomposition," *Electronics Letters*, vol. 32, no. 8, pp. 730–732(2), apr 1996. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/el{_}19960510

[136] C.-P. Chau and W.-C. Siu, "Generalized dual-point Hough transform for object recognition," in *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, vol. 1, 1999, pp. 560–564 vol.1.

[137] C. F. Olson, "Constrained Hough Transforms for Curve Detection," *Comput. Vis. Image Underst.*, vol. 73, no. 3, pp. 329–345, mar 1999. [Online]. Available: http://dx.doi.org/10.1006/cviu.1998.0728

[138] T. Achalakul and S. Madarasmi, "A concurrent modified algorithm for Generalized Hough Transform," in *Industrial Technology, 2002. IEEE ICIT '02. 2002 IEEE International Conference on*, vol. 2, dec 2002, pp. 965–969 vol.2.

[139] K. Hiroyasu and N. Munetoshi, "On a Fast Hough Transform Method PLHT Based on PiecewiseLinear Hough Function," *Systems and Computers in Japan*, vol. 21, no. 5, pp. 62–73, 1990. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/scj.4690210506

[140] H. Li, M. A. Lavin, and R. J. Le Master, "Fast Hough Transform: A Hierarchical Approach," *Comput. Vision Graph. Image Process.*, vol. 36, no. 2-3, pp. 139–161, nov 1986. [Online]. Available: http://dx.doi.org/10.1016/0734-189X(86)90073-3

[141] C. R. Dyer, "Gauge Inspection Using Hough Transforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 6, pp. 621–623, jun 1983. [Online]. Available: https://doi.org/10.1109/TPAMI.1983.4767452

[142] J. L. C. Sanz, I. Dinstein, and D. Petkovic, "Computing Multi-colored Polygonal Masks in Pipeline Architecture and Its Application to Automated Visual Inspection," *Commun. ACM*, vol. 30, no. 4, pp. 318–329, apr 1987. [Online]. Available: http://doi.acm.org/10.1145/32232.32235

[143] A. M. Wallace, "Greyscale image processing for industrial applications," *Image and Vision Computing*, vol. 1, no. 4, pp. 178–188, 1983. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0262885683900173

[144] H. K. Yuen, J. Illingworth, and J. Kittler, "Detecting Partially Occluded Ellipses Using the Hough Transform," *Image Vision Comput.*, vol. 7, no. 1, pp. 31–37, feb 1989. [Online]. Available: http://dx.doi.org/10.1016/0262-8856(89)90017-6

[145] H. Muammar and M. Nixon, "Approaches to extending the Hough transform," in *International Conference on Acoustics, Speech, and Signal Processing,*, may 1989, pp. 1556–1559 vol.3.

[146] V. Chatzis and I. Pitas, "Randomized fuzzy cell Hough transform," in *Proceedings of 6th International Fuzzy Systems Conference*, vol. 2, jul 1997, pp. 1185–1190 vol.2.

[147] R. Chan, "New parallel Hough transform for circles," *IEE Proceedings E - Computers and Digital Techniques*, vol. 138, no. 5, pp. 335–344, sep 1991.

[148] A. Ajdari Rad, K. Faez, and N. Qaragozlou, "Fast Circle Detection Using Gradient Pair Vectors." *DICTA*, pp. 879–888, 2003.

[149] D. Ioannou, W. Huda, and A. F. Laine, "Circle recognition through a 2D Hough Transform and radius histogramming," *Image and Vision Computing*, vol. 17, no. 1, pp. 15–26, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885698000900

[150] S. Tsuji and F. Matsumoto, "Detection of Ellipses by a Modified Hough Transformation," *IEEE Trans. Comput.*, vol. 27, no. 8, pp. 777–781, aug 1978. [Online]. Available: https://doi.org/10.1109/TC.1978.1675191

[151] H. Fei, G. Yanling, and W. Lili, "A New Ellipse Detector Based on Hough Transform," in *Proceedings of the 2009 Second International Conference on Information and Computing Science - Volume 02*, ser. ICIC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 301–305. [Online]. Available: https://doi.org/10.1109/ICIC.2009.187

[152] V. Chatzis and I. Pitas, "Select and split fuzzy cell Hough transform-a fast and efficient method to detect contours in images," in *Proceedings of IEEE 5th International Fuzzy Systems*, vol. 3, sep 1996, pp. 1892–1898 vol.3.

[153] S. Haykin, "Adaptive filter theory (3rd ed.)," jan 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=230061

[154] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, nov 1986. [Online]. Available: http://ieeexplore.ieee.org.ezlibproxy1.ntu.edu.sg/articleDetails.jsp?arnumber=4767851

[155] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades." *Frontiers in neuroscience*, vol. 9, p. 437, jan 2015. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2015.00437/abstract

[156] K.-L. Chung, Y.-H. Huang, S.-M. Shen, A. S. Krylov, D. V. Yurin, and E. V. Semeikina, "Efficient sampling strategy and refinement strategy for randomized circle detection," *Pattern Recognition*, vol. 45, no. 1, pp. 252–263, jan 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320311002792

[157] L. Jiang, "Efficient randomized Hough transform for circle detection using novel probability sampling and feature points," *Optik - International Journal for Light and Electron Optics*, vol. 123, no. 20, pp. 1834–1840, oct 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402612002550

[158] J. Ni, Z. Khan, S. Wang, K. Wang, and S. K. Haider, "Automatic detection and counting of circular shaped overlapped objects using circular hough transform and contour detection," in *2016 12th World Congress on Intelligent Control and Automation (WCICA)*. IEEE, jun 2016, pp. 2902–2906. [Online]. Available: http://ieeexplore.ieee.org/document/7578268/

[159] S. Seifozzakerini, W.-Y. Yau, B. Zhao, and M. Kezhi, "Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor," in *BMVC*, 2016, pp. 1–12.

END