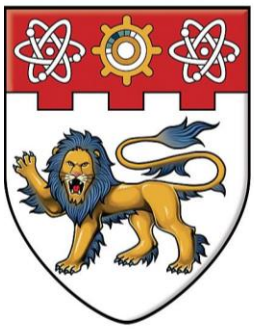
ASYNCHRONOUS HIGH-SPEED FEATURE EXTRACTION IMAGE SENSOR

HUANG JING

2018



NANYANG TECHNOLOGICAL UNIVERSITY

ASYNCHRONOUS HIGH-SPEED FEATURE EXTRACTION IMAGE SENSOR

HUANG JING

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2018

ASYNCHRONOUS HIGH SPEED FEATURE EXTRACTION IMAGE SENSOR

Huang Jing

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2018

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Chen Shoushun, for the continuous support of my PhD study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. It's my honour to be his PhD student and studying in his smart sensor group has always been a pleasure. The joy and enthusiasm he has for his research was contagious and motivational for me, especially during tough times in the PhD pursuit.

My sincere thanks also goes to Guo Menghan and Dr Wang Shizheng for their significant guidance and inspiration in my study of both hardware and software design. They have not only broadened my horizon and enrich my knowledge, but also impressed me by their conscientiousness and rigorousness in research.

The members of the smart sensor group have contributed immensely to my personal and professional time at NTU. The group has been a source of friendships as well as good advice and collaboration. I am especially grateful for the group members: Dr Qian Xinyuan, Dr Yu Hang, Guo Heng, Liu Lifen and Ding Ruoxi for their sharing of valuable experiences and advices in my research.

Last but not the least, I would like to extend my thanks to my family and my boyfriend for all their encouragement and supporting me spiritually throughout my PhD pursuit and life in general.

Contents

Acknowledgement ii

Contents iii

List of Figures vi

List of Tables xi

List of Abbreviations xii

Abstract xiv

Chapter 1 Introduction 1

1.1 Background and Motivation 1

1.2 Thesis Contributions 6

1.3 Thesis Organization 8

Chapter 2 Literature Review 10

2.1 Data Acquisition – Vision Sensors 10

2.1.1 Frame-based Image Sensors 10

2.1.2 Event-based CMOS Image Sensor 16

2.2 Object Visual Tracking 31

2.2.1 Fundamentals 32

2.2.2 Frame-based Tracking-by-Detection Algorithms 49

2.2.3 Event-based Object Tracking Methods 53

2.2.4 Advantages and Limitations of Event-based Tracking Methods 56

2.3 Optical Flow Algorithms 58

2.3.1 Fundamentals 58

2.3.2 Frame-based Optical Flow Algorithms....64
2.3.3 Event-based Optical Flow Algorithms....75
2.3.4 Advantages and Limitations of Event-based Optical Flow Estimation....78
Chapter 3 Motion Sensor with Asynchronous Grayscale Events for Event-guided High-Speed Object Tracking....81
3.1 Motivation....81
3.2 Image Sensor Design....82
3.2.1 Sensor Architecture....82
3.2.2 Intensity Readout Path....84
3.2.3 System Control and Handshaking Protocol....95
3.3 Sensor Implementation and Measurement....96
3.4 Proposed Event-guided Structured Output Tracking Algorithm....101
3.4.1 About *Struck* Tracking Algorithm....104
3.4.2 Event-guided Structured output tracking (ESSVM)....106
3.5 Experiments and Results....113
3.5.1 Experiment Setup....113
3.5.2 Experiments on benchmark sequences....114
3.6 Summary....120
Chapter 4 Motion Sensor with Pixel-Rendering Module for Optical Flow Extraction....121
4.1 Motivation....121
4.2 Design Concepts....123
4.2.1 Lucas Kanade Method (LK)....123
4.2.2 Markov Random Field (MRF)....125
4.2.3 Pixel Rendering Module (PRM)....127

4.2.4 Asynchronous Grayscale Events....128
4.3 Image Sensor Design....129
4.3.1 Sensor Architecture....129
4.3.2 Pixel Design....132
4.4 Sensor Implementation and Measurement....136
4.5 Experiments and Results....137
4.5.1 Methodology....138
4.5.2 Flow Estimation using Simulated Data....139
4.5.3 Flow estimation using real data....146
4.6 Summary....148
Chapter 5 Conclusions and Future Work....151
5.1 Conclusion....151
5.2 Future Research....153
Author's Publications....155
Bibliography....156

List of Figures

Figure 1.1 Example of advanced driver assistance system with road sign recognition, pedestrian detection, vehicle detection and tracking, and speed estimation. [1] 2
Figure 1.2 The output data of different types of image sensors. (a) Fast rotating hard-disk with a speed of 1200rpm is captured at 30fps and 240fps, (b) high speed camera [240fps] captures a sequence of images. (c) event-based motion sensor outputs asynchronous dynamic event. 5
Figure 2.1 Block level architecture of frame-based image sensor, taking a 4 × 4 pixel array as an example. 11
Figure 2.2 Photodiode charge accumulation vs. time with two static photocurrents. 12
Figure 2.3 High-resolution CMOS image sensors. (a) Canon prototype 250MP full frame sensor [33], (b) Sony IMX324 CMOS image sensor for automotive cameras [34]. 14
Figure 2.4 Illustration of the average illumination effect of APS imagers. 15
Figure 2.5 Cellular organization of the retina [38]. 16
Figure 2.6 Typical architecture of a dynamic vision sensor. 18
Figure 2.7 Schematic of a DVS pixel in block level. 19
Figure 2.8 Operation principle of DVS [23]. 20
Figure 2.9 Timing diagram of four-phase handshaking protocol: RR-row request, RA-row acknowledgement, CR-column request, CA-column acknowledgement. 21
Figure 2.10 Communication interface between pixels of different rows and the arbiter tree of two-input 22
Figure 2.11 Examples of DVS result provided by IniLabs. (a) DVS result of a rotating circle and bars, (b) DVS result of a moving hand [44]. 24
Figure 2.12 Architecture of ATIS pixel and timing diagram of operation principle. (a) Structure of ATIS pixel, (b) Timing diagram of exposure measurement unit. 26
Figure 2.13 Architecture of DAVIS and an example. (a) Structure of DAVIS pixel, (b) an example of DAVIS result from the 240×180 sensor, including a full frame image with pixel intensities and binary on/off event marked as red/green events, (c) binary events and intensity of the region of interest (ROI) with an obvious lag. 28
Figure 2.14 Basic steps for object tracking-by-detection [61]. 32
Figure 2.15 Illustration of object detection algorithms. (a) Temporal differencing conducted on two consecutive frames [64], (b) background subtraction method by subtracting the referenced background model from the current frame [65]. 34
Figure 2.16 (a) Original Lena image, (b) 3D colour distribution of the original Lena image, (c) colour histogram of the original Lena image [81]. 39

Figure 2.17 An overview of static HOG feature extraction. The detector window is tiled with a grid of overlapping clocks. Each block contains a grid of spatial cells. For each cell, the weighted vote of image gradients in orientation histograms is performed. These are locally normalized and collected in one big feature vector [92]. 42
Figure 2.18 Example templates of Haar-like features. 43
Figure 2.19 An example of the Haar-like feature calculation using integral image. 44
Figure 2.20 Decision boundaries with k=1,3,5,7 respectively. The decision boundaries become smoother with increasing value of k. [99] 46
Figure 2.21 (a) Hyper planes for linearly separable data, (b) optimum hyperplane with maximum margin for an SVM trained with sampled from two classes. Samples on the margin are called support vectors. 47
Figure 2.22 Functionality of kernel functions. Nonlinearly separable data can be mapped into linearly separable vectors in higher dimensional space by applying kernel functions. 48
Figure 2.23 Indoor experiments of tracking the patterns on a rotating white plate. Affine transformations including the rotation, translation, and scaling in the x, y-dimensions are applied and estimated simultaneously as shown in the second, third and fourth columns.[25] 54
Figure 2.24 Feature detection and tracking on Lucky Luke dataset. (a) DAVIS frame for initialization, (b) frame with centres of detected features (green crosses), (c) space-time view in the image plane of the tracked features' trajectories. [26] 55
Figure 2.25 Example applications based on optical flow estimation technology. 58
Figure 2.26 An example optical flow field in vector expression of a rotating object. 59
Figure 2.27 Examples of optical flow sensors. (a) APM optical flow sensor v10 [125], the image sensor and processing unit are separable, (b) PX4FLOW smart optical flow camera [126] which integrate the image sensor and programming processor onto a single vision chip. 60
Figure 2.28 Motion field and optical flow of a barber's pole [127]. 61
Figure 2.29 Illustration of the aperture problem. 61
Figure 2.30 Illustration of brightness constancy constraint. 63
Figure 2.31 Smoothness constraint, based on the fact that neighbouring two points in the visual scene are projected to the nearby two pixels in the image. 64
Figure 2.32 Illustration of image aliasing. (a) Nearest match is correct (no aliasing), (b) nearest match is incorrect (aliasing). 68
Figure 2.33 A simple example of coarse-to-fine optical flow estimation steps. 69
Figure 2.34 Two types of visualization of the motion field transforming from frame1 to frame2 [138]. 72
Figure 2.35 Examples of frames and ground truth from the main optical flow evaluation benchmarks. 73
Figure 2.36 Example of optical flow estimation using event-based algorithm. (a) DVS result of moving hands, (b) optical flow estimation in arrow visualization. 76

Figure 2.37 (a) Accumulated events of a rotating dot in spatial-temporal domain, (b) general principle of event-based visual flow using plane fitting method. 77
Figure 3.1 Design ideas contain three main parts, (a) directly take the output of logarithmic detector as intensity output by introducing a buffer and readout switch; (b) introduce a global control signal to generate full-frame images; (c) the buffered analog intensity value is converted and outputted through a specifically designed column analog readout path. 83
Figure 3.2 Sensor architecture with a 4 × 4 pixel array taken as an example. Based on traditional DVS (Fig.2.6), modifications are conducted in pixel, column readout processing and signal control on system level, which are labelled in blue. 84
Figure 3.3 Schematic of a complete intensity readout path. 85
Figure 3.4 logIph against VGSfb of transistor Mfb under subthreshold conduction. 87
Figure 3.5 (a) Schematic of VCA (voltage calibration and amplification), (b) timing diagram of control signals generated by the Local Timing controller. 88
Figure 3.6 Illustration of the operation principles of VCA. V_(IN_CM) is set to the middle level of VCA input voltage range, which is adjustable according to the test situation. V_CM is set at the centre of ADC input range, which is also the reference voltage of OTA. 90
Figure 3.7 Block diagram of the 9-bit SAR ADC. 92
Figure 3.8 (a) Schematic of the 9-bit ADC, (b) reference voltages MUX. 93
Figure 3.9 An example of ADC conversion. 94
Figure 3.10 Handshaking protocol between blocks for the new DVS sensor. 95
Figure 3.11 Layout of the motion sensor with asynchronous grayscale events. 97
Figure 3.12 Sample images of the image sensor. (a) Full-frame images captured by forcefire signal, (b) reconstructed images using binary events within a 20ms period. 98
Figure 3.13 Illustration of FPN extraction. (a) Full-frame reference image capture under uniform light condition, (b) extracted FPN pattern by subtracting the mean pixel value from each pixel. 99
Figure 3.14 Overview of the designed motion sensor. (a) Designed motion sensor with 384×320 pixels. (b) Scenario of a rotating plate with four different geometrical patterns. (c) The spatial-temporal spacing of events generated in response to the rotating plate: pixels independently generate asynchronous event packages. Static background can be captured at first and small patches of region-of-interest (ROI) can be reconstructed by updating the intensities of dynamic events on the pre-captured background. 102
Figure 3.15 Structure of the tracking algorithm based on structured support vector machine (SSVM). 105
Figure 3.16 Demonstration of the proposed tracking method using ESSVM. (a) Sample image from biker_head sequence with the bonding box of ROI, (b) illustration of a fast-moving object tracking process, including demonstrations of EPSI and EISS. 107
Figure 3.17 (a) Workstation for event generation, (b) a monitor is used for displaying images that are observed by the designed motion sensor, (c) sample observed frame on the display monitor, (d) events

are collected and reconstructed to frames, left image uses raw events while the right image shows the reconstructed frame after temporal noise filtering. 114
Figure 3.18 Tracking performance of car_rc_rolling sequence. (a) Tracking errors for both ESSVM and state-of-the-art algorithms on car_rc_rolling sequence at 30fps. (b) Tracking errors of ESSVM for the first 1.2s and (b1-b8) are sample frames showing tracking performance of all listed trackers, the colors of bonding boxes are consistent with the legend in (a). 117
Figure 3.19 Evaluation of event-guiding methods by comparing the ALE for SSVM, ESSVM- and ESSVM on the testing sequences. 118
Figure 3.20 Plot of average location error over 30fps sequences and real-time performance. 119
Figure 4.1 Flow vectors in a neighbourhood window, usually 3x3 or 5x5, are assumed to be the same. 124
Figure 4.2 Pixel connection of N4 neighbourhood system (left) and N8 neighbourhood system (right). 125
Figure 4.3 MRF neighbourhood system in spatial-temporal domain. If each point represents a pixel with absolute intensity, the blue pixels construct MRF in spatial domain for spatial gradient calculation while green pixels construct MRF in temporal domain for temporal gradient calculation. 126
Figure 4.4 Functionality comparison between a traditional DVS and the proposed optical flow motion sensor with PRM. (a) Traditional DVSs only output the position Xp, Yp of pixels labeled in black, (b) novel motion sensor with PRM outputs the event packages from both active pixels and their neighbor pixels as labeled in both black and yellow. Each event package consists of pixel position Xp, Yp, captured intensity Ip and event-triggered time-stamp Tp. 127
Figure 4.5 Design idea of the pixel rendering module (PRM). A single activated pixel Pc triggered by intensity variation will simultaneously activates its neighbour pixels ($P22, P31, P42, P33$) to generate grayscale events. 130
Figure 4.6 Architecture of the optical flow oriented motion sensor. 131
Figure 4.7 (a) Pixel architecture of the novel designed vision sensor. (b) Architecture of the intra-pixel handshaking logic of Pixel Rendering Module. L, R, U, D represents the EP signal received from its left, right, up and down neighbour pixels. 133
Figure 4.8 Inter-pixel communication due to PRM. (a) Illustration of communications between pixels under pixel rendering mechanism. RR, RA, CR and CA are signal for AER handshaking, while pixels in the same column share the IANA buses for intensity analog value readout, (b) four examples of pixel arrays with different pixel status pattern, red pixels represent the pixels activated by illumination detector and green pixels represent pixels activated by self-activated (red) pixels. 134
Figure 4.9 An example of main signal timing diagram according to Fig.4.8 (a). 135
Figure 4.10 (left) Top level layout of the fabricated motion sensor with pixel rendering module, (right) pixel layout. 136
Figure 4.11 Four sets of experiment data and results. The test sequences from top to bottom are Grove2, Grove3 (from Middlebury database [138], bamboo-1 and ambush 7 (from MPI database [139]). For each

data set, the first row from left to right: original image in gray-level events under PRM and a binary map indicating the active events without pixel rendering; the second row from left to right: original color image, flow using event with intensity variation, flow using event with absolute intensity, flow using events from our sensor and ground truth. 140
Figure 4.12 The difference of output events with or without pixel rendering. 142
Figure 4.13 Evaluation results of the simulated event sensors in terms of average angle error (AAE) and average endpoint error (AEE)..... 143
Figure 4.14 Testing platform including the optical flow motion sensor, a static board with testing patterns and a linear motor controlled by a PC. The whole testing platform is placed in a dark room. 146
Figure 4.15 Flow estimation results of four examples using real data. Col-1 shows the full frame images of each testing scenario captured by the optical flow motion sensor controlled by the forcefire signal. Col-2 images are example event slices containing the binary events triggered due to moving objects. Col-3 images show the corresponding grayscale events. Col-4 shows the optical flow results in colour coding and Col-5 are the optical flow results in motion vectors. 147
Figure 5.1 An example 3x3 array pixels with smart design 154

List of Tables

TABLE I SPECIFICATION SUMMARY OF DVS IN DIFFERENT VERSIONS 25
TABLE II COMPARISON OF OBJECT DETECTION METHODS 36
TABLE III SRER EVALUATION RESULTS ON THE TB-50 DATASET [59]. 51
TABLE IV OVERVIEW CHARACTERISTICS OF THE EVALUATION METRICS [116] 52
TABLE V SPECIFICATIONS COMPARED WITH PRIOR DVS IMPLEMENTATIONS 100
TABLE VI FLOW ESTIMATION RESULTS 115
TABLE VII SPECIFICATIONS COMPARED WITH PRIOR DVS IMPLEMENTATIONS 137
TABLE VIII ACCURACY AND EVENT-DENSITY FOR SAMPLE SEQUENCES 145

List of Abbreviations

AAE	Average Angle Error
ADAS	Advanced Driver Assistance System
ADC	Analog-Digital Conversion
AE	Angular Error
AEE	Average Endpoint Error
AER	Address-Event Representation
APS	Active Pixel Sensor
ATIS	Asynchronous Time-based Image Sensor
CA	Column Acknowledgement
CBIR	Content-based Image Retrieval
CCD	Charge-Coupled Device
CDS	Correlated Double Sampling
CMOS	Complementary Metal–Oxide–Semiconductor
CNN	Convolutional Neural Network
CR	Column Request
DAVIS	Dynamic and Active-pixel Vision Sensor
DPS	Digital-Pixel Sensor
DT	Decision Tree
DVS	Dynamic Vision Sensor
EISS	Event Intensity-guided Sample Supplement
EPE	Endpoint Error
EPSL	Event Position-guided Search Localization
ESSVM	Event-guided SSVM
FFT	Fast Fourier Transform
FPN	Fixed Pattern Noise
GLCM	Grey-Level Co-occurrence Matrix
GMS	Grayscale Motion Sensor
HOG	Histogram of Oriented Gradients
k-NN	k-Nearest Neighbour

LBP	Local Binary Pattern
LDP	Local Directional Pattern
LK	Lucas Kanade
MEI	Motion Energy Image
MHI	Motion History Image
MOSSE	Minimum Output Sum of Squared Error
MRF	Markov Random Field
OTA	Operational Transconductance Amplifier
PLL	Phase-Locked Loop
PPS	Passive-Pixel Sensor
PRM	Pixel Rendering Module
PRM-GMS	Grayscale Motion Sensor with Pixel Rendering Module
PWM	Pulse Width Modulation
RA	Row Acknowledgement
RAAE	Relative Average Angle Error
RAEE	Relative Average Endpoint Error
RFC	Recycling Folded Cascode
RGB	Red-Green-Blue
ROI	Region-Of-Interest
RR	Row Request
SDRAM	Synchronous Dynamic Random-Access Memory
SNR	Signal-to-Noise Ratio
SRE	Spatial Robustness Evaluation
SSRAM	Synchronous Static Random-Access Memory
SSVM	Structured Support Vector Machine
SVM	Support Vector Machine
TRE	Temporal Robustness Evaluation
UAV	Unmanned Aerial Vehicle
VCA	Voltage Calibration and Amplification
VOT	Visual Object Tracking

Abstract

In recent years, motion detection and analysis in computer vision has drawn an increasing attention in a wide range of applications including high speed surveillance, traffic enforcement, automotive crash safety, and navigation of autonomous vehicles, etc. These applications require continuous image acquisition and real-time processing techniques, including optical flow, image segmentation, object recognition and object tracking, etc. Traditionally, optical flow and object tracking are computed on a series of consecutive image frames captured by standard cameras. Such imaging systems employ charge-coupled device (CCD) or complementary metal-oxide-semiconductor (CMOS) active pixel sensors (APS) and capture images at a certain frame rate. To detect high speed motion activities, the sensors need to run at a very high frame rate, i.e. more than 500 frames/second. Massive quantities of raw, redundant image data must be transmitted and processed before the features of interest are obtained. These data exert heavy pressure on the transmission bandwidth and the following processing stage, thus, such frame-based data acquisition and processing systems are difficult to satisfy real-time requirement. Moreover, capturing fast motions using standard cameras leads to motion blur and loss of the motion trajectory during the blind time between two consecutive frames, which restrict the accuracy of object tracking. The tracking accuracy issue can be alleviated by using high-speed cameras. However, it introduces much more redundant data and tremendous computational cost.

The development of dynamic vision sensors (DVSs) has provided a solution to the afore-mentioned issues. Unlike frame-based imagers, DVSs have no concept of 'frame' but output asynchronous pixel events. Each pixel continuously detects and logarithmically responds to the illumination changes. A pixel is triggered to output an event only when its detected illumination change goes beyond the defined threshold. Such mechanism enables DVSs to reduce data redundancy by filtering out the redundant data from static background. Considering that illumination variation is related to actual movement in most common scenarios, the DVSs are capable of

extracting motion related information directly. Several event-based algorithms for object tracking and optical flow have been developed taking advantage of the spatial-temporal characteristics of binary events. It has been proven that the employment of DVS in object tracking and optical flow can reduce the computational cost significantly. However, the accuracy of the event-based algorithms is restricted due to the lack of absolute light intensity of DVS events. Moreover, the pixels in a traditional DVS work individually, resulting in high sparseness of pixel events and poor accuracy in flow estimation with respect to fast moving objects.

This thesis presents the developments of two smart event-based motion sensors with feature extraction, such that the designed motion sensors can be used to develop more algorithms in the field of motion detection and analysis. In addition, an event-driven tracking algorithm tailored for the designed motion sensor is also proposed. With the novel designed motion sensors and proposed algorithm, the event-based optical flow and object tracking algorithms can achieve much higher accuracy compared to those using traditional DVSs while maintaining real-time performance. The main contributions of this thesis are summarized in the following four aspects: (1) The recently developed event-based DVSs are exhaustively investigated. Compared with standard cameras, event imagers have high temporal resolution and low latency, both in the order of micro-seconds, and a large dynamic range. Furthermore, event-driven signal processing algorithms in the context of object tracking and optical flow are reviewed, showing possibilities in real-time applications. (2) The first hybrid motion sensor that can generate both frames and asynchronous events with intensity (named grayscale events) is designed and implemented. The designed hybrid sensor has independent pixels that report asynchronous events, including pixel locations, with their corresponding illumination. It significantly reduces the redundant data from static background and extract essential information for motion analysis. In contrast to existing event cameras with intensity readout (DAVIS and ATIS cameras), the designed sensor is free of exposure time, contributing to fast-response imaging of high speed objects. In addition, pixels in this hybrid sensor are globally controlled to capture full frames on demand, thus benefits the environmental perceptions in computer vision tasks. Accordingly, a 384×320-pixel prototype sensor was designed and fabricated. (3) An

event-guided discriminative tracking method tailored for the designed hybrid motion sensor is proposed for detection and tracking of high speed moving objects. The proposed tracking algorithm incorporates a traditional frame-based tracking-by-detection algorithm with two event-guiding methods. Compared to the classical trackers, the proposed tracking algorithm exploits an online adaptive searching area to achieve a more accurate localization. Moreover, the gray-level intensity of event packages generated by the designed hybrid motion sensor is utilized to reconstruct supplement samples and update the tracker model over time. The experimental results indicate that the proposed algorithm presents high computational efficiency and outperforms the state-of-the-art trackers in terms of accuracy and real-time performance.

(4) A novel motion sensor with pixel rendering mechanism is designed and implemented for optical flow estimation. Besides the detection of illumination changes, pixels in the proposed motion sensor are interconnected and communicate event status among each other through a pixel rendering module (PRM). Each active pixel together with its four-neighbor pixels report grayscale events to provide sufficient data in gradient extraction which is essential in flow estimation. Thus, the proposed sensor fundamentally solves the accuracy problems of existing event-driven optical flow estimation, which are caused by event sparseness and lack of event intensity. A 64×64 prototype was fabricated in 0.35μm 2P4M Opto process. Experiments were conducted on both synthesized data based on Middlebury and MPI database and real data collected from the designed sensor, showing an improvement (by 17%) in accuracy performance of event-flow estimation.

Chapter 1 Introduction

1.1 Background and Motivation

During the last decades, with the development of automotive industry, the world is getting smaller with respect to reachability. In the meanwhile, an increasing number of road accidents has been reported, threatening life and safety of human beings. There have been continuous efforts involved to ensure road safety. Driving infrastructures such as roads, signals and sign boards are improved and driving vehicles are equipped with most advanced technologies such as power steering system and brake system. However, a full 94 percent of the road accidents are caused by human errors [1] and mistakes made by drivers directly led to accidents. Although human beings are equipped with the best performed vision system, accidents can still happen due to driver fatigue, distraction and inattention. In recent years, advanced driver assistance systems (ADASs) have been designed to prevent accidents by helping people drive safely or providing autonomous driving cars. Nevertheless, it is always a challenge to make autonomous cars self-sense the environment and drive like a human. In many countries like U.S.A, China and Germany, with the gradual emergence of autonomous driving research, efforts are made to build smart driving systems that can be programmed to follow traffic rules and drive more safely without any fatigue. Main challenges involve understanding complex traffic patterns and taking real time decisions using visual data, etc. Techniques of motion detection and analysis are necessary in most of the computer vision processing in ADASs.

In recent years, motion detection and analysis have become of great importance in a broad range of applications including high speed surveillance, medicine, film industry, traffic enforcement, automotive crash safety, and navigation of autonomous vehicles,

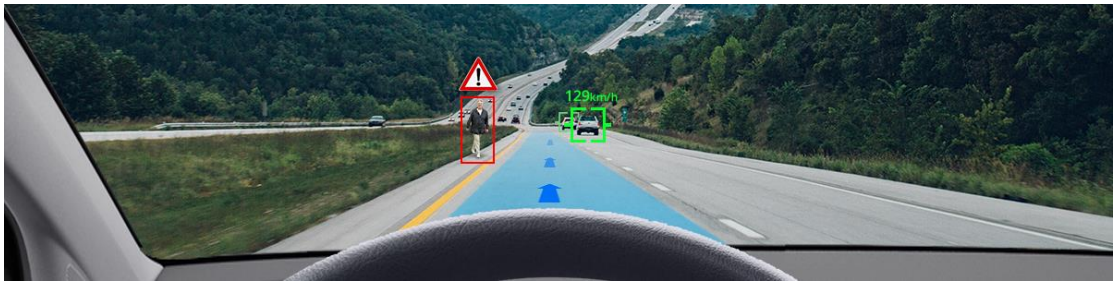


Figure 1.1 Example of advanced driver assistance system with road sign recognition, pedestrian detection, vehicle detection and tracking, and speed estimation. [1]

just to name a few. These applications require continuous image acquisition and real-time processing. The main techniques in this scope include optical flow, image segmentation, object recognition and object tracking, etc. Among these techniques, object tracking is widely used in surveillance and vehicle navigation for the detection and tracking of pedestrian, vehicles and obstacles. In addition, optical flow is a significant methodology for motion analysis, since it can provide complete information about motion and has become fundamental for more complicated tasks like segmentation and tracking.

Motions, which are traditionally captured in image sequences acquired by a video camera, can be induced both by movements of objects in a three-dimensional scene and by camera motion. From an applicative viewpoint, the motion information brought by the dynamic behavior of observed objects or by the movement of the camera itself is a decisive element for the interpretation of observed phenomena. The goal of visual tracking is to locate a moving object over time. Object tracking algorithms have acquired priority due to the availability of highly sophisticated computers, high quality and inexpensive cameras. However, it is an enormous challenge to design a fast and robust tracker according to various critical conditions in visual tracking, such as deformation, illumination variation, occlusion, blur and fast motion, background clutter and so on. Current object tracking methods are conducted on videos, i.e. a series of image frames, which are captured using conventional frame-based image sensors. Over the past decades, various frame-based visual tracking algorithms have been proposed to cope with these issues. Some of the classical tracking algorithms use generative models [2-6], which perform tracking by searching the best-matching windows, and some algorithms use discriminative models [7-10], which learn to distinguish the moving target from background typically through a trained classifier. Discriminative

[1] Picture retrieved from http://eu.automotive.panasonic.com/solutions/adas

methods, also known as tracking-by-detection methods, are preferred especially when the input data size is tremendous, as observed in [11, 12] that discriminative methods are more competing. In particular, trackers based on correlation filtering have been developed and made significant contributions in visual tracking since the proposal of Minimum Output Sum of Squared Error (MOSSE) filter [13]. The main achievement of correlation filter-based tracker is to speed up the post-processing by reducing the computational complexity using Fast Fourier Transform (FFT). In past two years, deep learning methods have also been applied in object tracking, and have shown great performance in benchmarks [14]. However, current methods can rarely handle tracking of fast moving objects especially in terms of the accuracy and real-time performance.

Optical flow is another important research topic in the field of motion analysis in computer vision. It is the lowest-level characterization of the estimation of a dense motion field, corresponding to the displacement of each pixel. Most high-level motion analysis tasks including image segmentation and classification employ optical flow as a fundamental basis. Accurate and fast measurement of optical flow is a necessary requirement for using this flow in vision tasks such as detecting moving obstacles crossing the path of a vehicle, visually guiding aircraft or space vehicle landing, or acquiring structure from motion information about the environment. Optical flow estimation has given rise to a tremendous quantity of work for 35 years. If a certain continuity can be found since the seminal works of Horn Schunck [15] and Lucas Kanade [16], a number of methodological innovations have progressively changed the field and improved performances. The progress of optical flow estimation techniques is marked by two major stepping stones: the quantitative evaluation of optical flow algorithms by Barron et al. [17] provided a dataset and error measures that became standard until succeeded by the more challenging Middlebury benchmark. Both helped identify difficult areas in flow estimation and opened the door for the development of dozens of new algorithms. Nowadays, state-of-the-art techniques provide a remarkable degree of accuracy. Barron et al. [17] and Fleet and Weiss [18] provide a basic introduction to gradient-based optical flow estimation; Sun et al. [19] give an extensive review of current flow models. However, the raw images captured by the conventional frame-based image sensors contain massive quantities of primitive and redundant

information. Their high accuracy requires massive computation and diminishes their usability in real-time applications (for instance, the highest-ranking algorithm on the Middlebury benchmark [20] takes $11\ min$ for two frames; the fastest has a runtime of $0.1s$.)

Traditionally, the computation of optical flow and object tracking is conducted on a series of consecutive image frames, which are captured by conventional frame-based image sensors. Such motion detection system employs charge-coupled devices (CCD) or complementary metal-oxide-semiconductor (CMOS) Active Pixel Sensors (APS) and captures images at a certain frame rate. To detect high speed motion activities, the sensors need to run at a very high frame rate, i.e. more than 500fps. Massive quantities of data are produced, often in the range of GB/s. The image flow contains a high level of redundancy and large amount of motion unrelated data, which have to be read out and processed before the features of interest are extracted. The optical flow computation time is thus far longer than the image acquisition time of objects in high speed motion and therefore, the optical flow based higher level algorithms cannot perform in real-time. Moreover, when capturing fast moving objects, frame-based cameras result in motion blur and loss of the motion trajectory between two consecutive frames, which restrict the accuracy of object tracking. The accuracy issue can be alleviated by employing high-speed cameras, however, it introduces much more redundant data and tremendous computational cost.

The development of asynchronous event-based artificial retinas [21, 22], a promising approach to visual signal processing, provides a potential solution to the above-mentioned issues. In contrast to conventional frame-based image sensors, the dynamic vision sensor (DVS) cameras are free of the concept of 'frame' and output asynchronous address-events [23, 24]. The events are classified into ON/OFF events respectively indicating positive and negative changes in log intensity at each pixel address. This approach has the following advantages, encoding light with log intensity implements a form of local gain control so that the DVS can handle scene illuminations from a few luces to more than $100k\ lux$. In particular, the local gain control enables reliable operation in scenes with high intra scene dynamic range. Events are conveyed

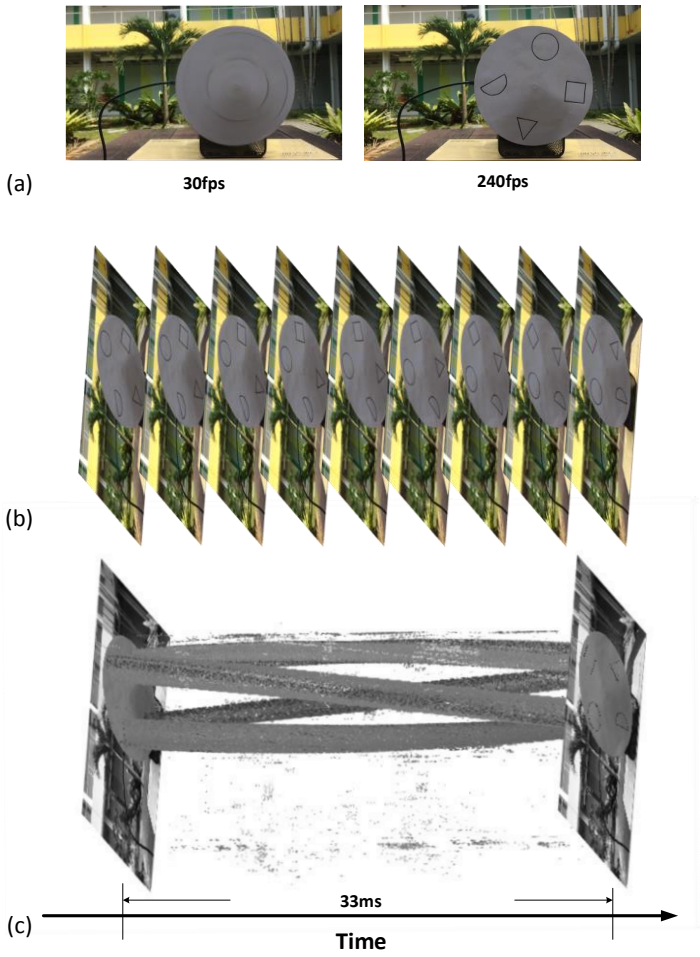


Figure 1.2 The output data of different types of image sensors. (a) Fast rotating hard-disk with a speed of 1200rpm is captured at 30fps and 240fps, (b) high speed camera [240fps] captures a sequence of images. (c) event-based motion sensor outputs asynchronous dynamic event.

to a processor over a standard USB interface at a temporal resolution of $1\mu s$ with low latency ($<1ms$). The temporal resolution of the DVS is thus equivalent to that of a frame-based camera running at several thousand frames per second. This high sample rate is achieved since redundant information originating from unchanged parts of the image are not transmitted, which is different from conventional cameras. This sparse address-event representation (AER) protocol reduces memory requirements and computational cost that makes the DVS potentially useful for high-speed and low-power computer vision tasks.

Object tracking and optical flow algorithm operating on the DVS output, i.e. event-based algorithms, provides a solution to the challenge that frame-based techniques are facing, namely large inter-frame displacements occurring in fast motion. For event-

based object tracking, Benosman et al. proposed a computationally efficient and robust pattern tracking using an asynchronous time-based image sensor developed from DVS. The method provides a continuous and iterative estimation of the geometric transformation between the model and the events representing the tracked target [25]. Tedaldi et al. presents an algorithm based on the Dynamic and Active-pixel Vision Sensor (DAVIS). The algorithm detects and tracks features from the DAVIS frames [26]. For event-based flow estimation, Benosman et al. [27] make use of the high temporal precision of DVS data by computing gradients on the surface consisting of most recent events. The same group also developed phased-based method to improve estimations in highly textured areas [28]. Brosch et al. [29] suggest an event-based flow computation method using biologically inspired filter-banks that detect the orientation of an edge. All these developments are indicative of the potential of event-based techniques to resolve some of the major problems of conventional, frame-based flow estimation. However, the lack of absolute light intensity value in each event package is the primary limitations of accurate tracking and flow estimation. In addition, the sparseness of the asynchronous events when observing fast motions further limits the accuracy of optical flow computation.

1.2 Thesis Contributions

The main objective of this research is to design functional and smart motion sensors based on the traditional DVSs, such that the designed motion sensors can be used to develop more algorithms in the field of motion detection and analysis. With the novel designed motion sensors, it is expected that the event-based computer vision algorithms can achieve much higher accuracy compared to those using traditional DVSs while maintain real-time performance. The main contributions of this thesis are summarized in the following four aspects:

(1) Investigations on the recently developed event-based DVSs and event-driven signal processing algorithms. In contrast to standard cameras that capture images at a fixed rate, the bio-inspired event-based dynamic vision sensors have independent, asynchronous pixels that report local illumination variation (namely ‘events’) once

occur, thus mimicking human retina. Compared with standard cameras, event cameras have high temporal resolution and low latency, both in the order of micro-seconds, and a large dynamic range. Furthermore, event-driven signal processing algorithms in the context of computer vision were reviewed, showing possibilities in real-time applications.

(2) Design and implementation of the first hybrid motion sensor generating both frames and asynchronous grayscale events. With a low latency and low power intensity readout scheme, the designed sensor has independent pixels that simultaneously report asynchronous events and corresponding illumination. It significantly reduces redundant data from static background and extract essential information for motion analysis. In contrast to existing event cameras with intensity readout (DAVIS and ATIS cameras), the designed sensor is free of exposure time, contributing to fast-response imaging of high speed objects. In addition, pixels in this sensor can be globally controlled to capture full frames on demand, thus benefits the environmental perceptions in computer vision tasks. The sensor was implemented using AMS 0.35μm 2P4M Opto process with an array of 384×320 pixels.

(3) Proposal of the first event-guided discriminative tracking method for detection and tracking of high speed moving objects. The proposed tracking algorithm is tailored for the designed hybrid motion sensor and incorporates a traditional frame-based tracking-by-detection algorithm with two event-guiding methods, taking use of the pixel location and intensity in each event package. Compared to the classical trackers, the proposed tracking algorithm exploits an online adaptive searching area to achieve a more accurate localization. Moreover, the gray-level intensity of event packages generated by the designed hybrid motion sensor is utilized to reconstruct supplement samples and update the tracker model over time. The proposed algorithm presents high computational efficiency and experiments over sequences from multiple tracking benchmarks demonstrate the superior accuracy and real-time performance, compared to the state-of-the-art trackers.

(4) Design and implementation of the first optical flow motion sensor with pixel rendering mechanism. Besides detection of illumination changes, pixels in the proposed motion sensor are interconnected and communicate event status with each

other through a pixel rendering module. Each active pixel together with its four-neighbour pixels report grayscale events to provide sufficient data in gradient extraction, which is essential in flow estimation. Thus, the proposed sensor fundamentally solves the accuracy problems, which are caused by event sparseness and lack of event intensity, of existing event-driven optical flow estimation. A 64×64 prototype was fabricated in $0.35\mu m$ 2P4M Opto process. Experiments were conducted on both synthesized data based on Middlebury and MPI database and real data collected from the designed sensor, showing an improvement (by 17%) in accuracy performance of event-flow estimation.

1.3 Thesis Organization

The organization of this thesis is as follows.

Chapter 2 gives a brief review of the related work in the literature. Based on the object tracking system, the review is divided into three main sections: data acquisition device, object tracking algorithms and optical flow algorithms. In data acquisition devices review, conventional frame-based cameras using APSs and current event-based vision sensors including the hybrid vision sensors are covered and compared. For review of object tracking algorithms, steps contained in an object tracking system, including object detection, feature extraction, classification and tracking methods, are reviewed. Among various methods of object detection, optical flow is intensively studied since it is significant in motion extraction and analysis. In the review of object tracking and optical flow algorithms, both frame-based and event-based algorithms are reviewed and discussed.

Chapter 3 illustrates the designed motion sensor with asynchronous pixel intensity extraction and a novel tracking algorithm, which combines the output events from the designed motion sensor and a classical frame-based tracking-by-detection method. The motion sensor is designed to output gray events instead of binary events. The intensity of each event is simultaneously outputted with the encoded pixel address to generate an

event package. The proposed event-guided discriminative tracking algorithm is evaluated on public benchmarks.

Chapter 4 illustrates the designed motion sensor with pixel rendering module to improve the accuracy of event-based optical flow estimation while maintaining low computational cost. The operating principle, architecture design and circuit details are discussed in detail. Moreover, the experimental results on both simulation data and sensor collected data are intensively analyzed in this chapter.

Chapter 5 draws a conclusion of the presented work and provides some recommendations for future work.

Chapter 2 Literature Review

This chapter provides exhaustive reviews on the related work and reported studies in literature. Based on the object tracking system, the review is divided into three main sections: data acquisition device, object tracking algorithms and optical flow algorithms since optical flow is one of the main detection methods in object tracking system. For data acquisition devices, conventional frame-based cameras using APSs and current event-based vision sensors including the hybrid vision sensors are covered and compared. For review of object tracking, steps contained in an object tracking system, including object detection, feature extraction, classification and tracking methods, are reviewed. Optical flow, one of the main object detection and a significant motion extraction method, in literature is reviewed in detail. In the review of object tracking and optical flow algorithms, both frame-based and event-based algorithms are reviewed and discussed.

2.1 Data Acquisition – Vision Sensors

2.1.1 Frame-based Image Sensors

In computer vision, image sensors are used for data acquisition. Semiconductor charge-coupled devices (CCD) were the mainstream solid-state image sensor technology for a long time since 1969 [30]. Currently complementary metal-oxide-semiconductor (CMOS) image sensors have joined CCDs as mainstream, mature technology. CCDs and CMOS imagers were both invented in the late 1960s and 1970s [31, 32]. CCD became dominant first, primarily because of the superior image quality with the

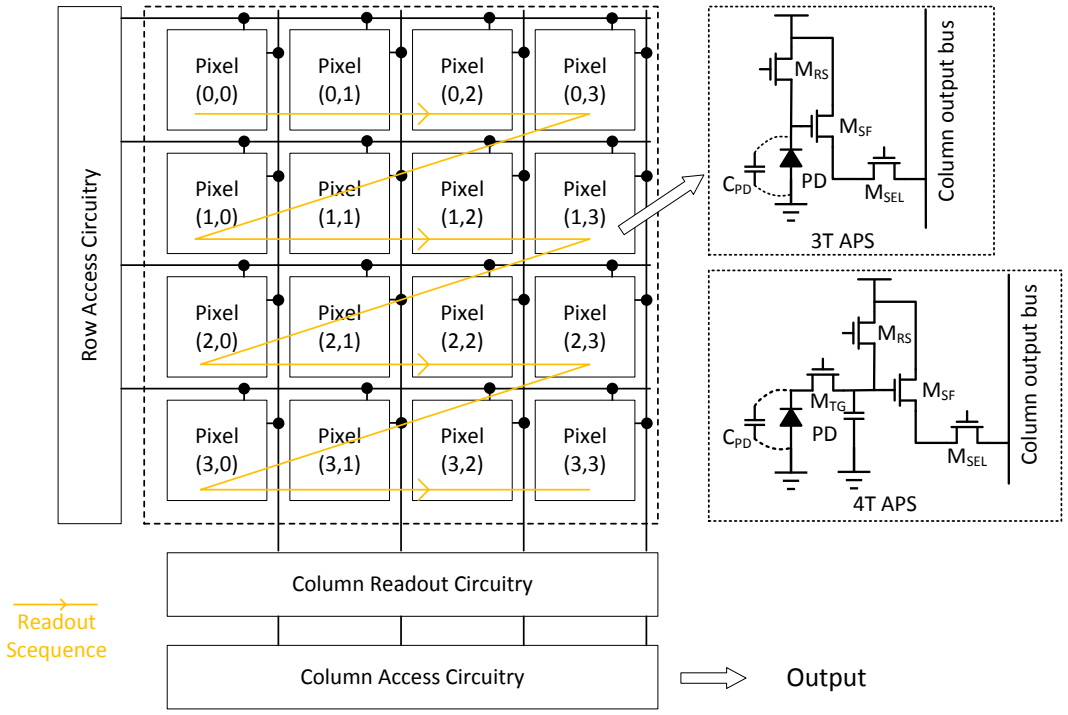


Figure 2.1 Block level architecture of frame-based image sensor, taking a 4 × 4 pixel array as an example.

fabrication technology available at the time. With the continuous advancement of the fabrication process, renewed interest in CMOS was based on expectations of lower power consumption, camera-on-a-chip integration, and a lower unit cost for high volume fabrication. Along with the improvements in both circuit design and fabrication process, the early deficiencies of the CMOS imagers over the CCD device, such as large pixel area, large temporal noise and fixed-pattern noise have been resolved. Currently, high speed CMOS imagers can be designed to have much lower noise than high speed CCDs. In addition, CMOS imagers continuously exploit new application prospects of the functional image sensors by integrating part of the back-end analog and digital processing units into the sensor level. With respect to different structures of CMOS imagers, the APSs performs a better tradeoff among some key specifications of image sensor, such as pixel size, readout speed, temporal and fixed pattern noise, compared with passive-pixel sensor (PPS) and digital-pixel sensor (DPS). Therefore, APS imagers are widely used among academic research and industrial community. Since the output of the aforementioned cameras are usually in a format of frames, such kinds of imagers are also called frame-based image sensors. The structures and operation

principles of the frame-based APS imagers will be elaborated in the following paragraphs.

The simplified block diagram of a frame-based image sensor is shown in Fig.2.1 [30]. The function blocks include a two-dimensional pixel array, readout circuitry, row access circuitry, column access circuitry and a system control logic. The pixel array is the major part of an image sensor and commonly consists of a large number of pixels which are uniformly distributed in vertical and horizontal dimensions. Each pixel has a photodetector which transfers the light intensity information into electrical voltage or current signals. The row and column access circuitries are incorporated to control the access to a single pixel and output its information through readout circuitry. In order to avoid any readout data collision, information stored in all pixels are read out one by one in serial. The control logic contains a communication interface between row and column access circuitry as well as all the necessary circuit control signal.

Commonly used architectures for pixel cells are 3-transistor Active Pixel Sensor (3T-APS) and 4T-APS. Here, 3T-APS is used to illustrate the operation of the frame-based APS image sensor. As shown in Fig.2.1, the pixel comprises of three transistors which are reset transistor M_{RS}, buffer transistor M_{SF} and selection transistor M_{SEL}. The operation of the 3T-APS comprises three modes: reset mode, integration mode and readout mode. The pixel is first being reset such that the cathode of the photodiode is reset to V_{dd}. After being reset, the pixel is switched into integration mode in which the voltage level of cathode of the photodiode begins to drop as the light injection generated

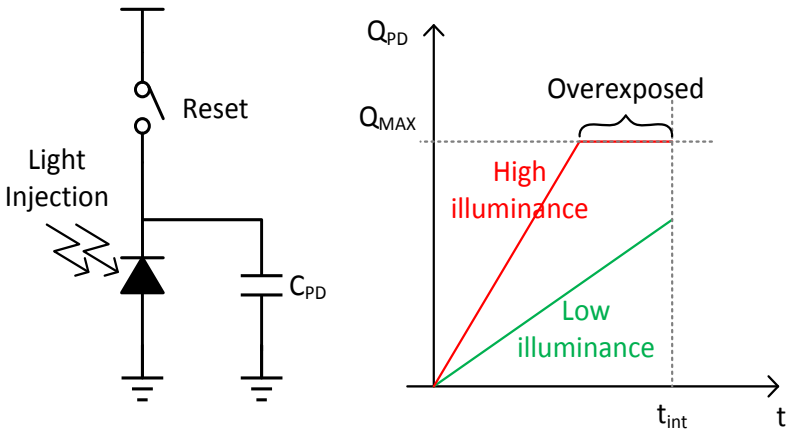


Figure 2.2 Photodiode charge accumulation vs. time with two static photocurrents.

photocurrent discharges the parasitic capacitance of this node. After certain exposure time, the pixel starts the readout mode with the selection transistor M_{SEL} turned on and the photodiode cathode voltage is buffered through M_{SF} to the column readout bus. All these operations are controlled by the external row and column access circuitries. Fig.2.2 shows the charges accumulated in the photodiode parasitic capacitance versus time with two static photocurrents, i.e. static light illuminance. It is observed that higher illuminance results in faster charge accumulation and thus voltage drop. If the accumulated charges reach the maximum before the end of exposure, the voltage remains at the minimum level and the pixel is termed as overexposed. Since the output voltage linearly responds to the illuminance, the capacity of C_{PD} limits the dynamic range of such APS sensors.

In order to capture a full image, the image sensor scans the whole pixel array once according to a particular order. Progressive scanning is one of the most commonly used method used to scan the pixel array row by row. Within the selected row, the pixel information is shifted out column by column. For example, a frame-based image sensor with a 4x4 pixel array, as shown in Fig.2.1, the whole pixel array is initially reset and then all pixels are exposed and start integration mode at the same time. After a certain exposure time, the row access circuitry selects the first row ($Row0$) at the beginning. Meanwhile, the column access circuitry selects the first column ($Column0$) and thus, the integration voltage of $Pixel\ (0,0)$ is transferred out by means of the readout circuitry. After that, the pixel in the next column, $Pixel\ (0,1)$, is selected and its integration voltage is scanned out as well. The same operation is executed repeatedly in this row until the information of the pixel in the last column is readout, the $Pixel\ (0,3)$ in this case. Once the row access circuitry completes processing an entire row, it switches to the next row ($Row1$) and the integration voltages of pixels from $Pixel\ (1,0)$ to $Pixel\ (1,3)$ are scanned out in sequence. Based on such progressive scanning mechanism, a full image (a frame) could be achieved until the information of $Pixel\ (3,3)$ (the pixel in the last row and column) is scanned out and the pixel array can be reset immediately to generate another frame. Framerate is defined as the frequency of which image frames are generated. It is a crucial specification in evaluating sensor performance of capturing fast moving objects.

The simplicity of the pixel structure is one of the main advantages for frame-based image sensors, since it enables compact sensor design. Besides, the high-speed readout, benefiting from the well-understanding progressive scanning mechanism (usually implemented using shift-register based scanners), also allows high-resolution implementation of CMOS image sensors. With the fast development of integrate circuit fabrication technologies, frame-based image sensors can achieve ultra-high resolution. As an example, Canon EOS 5DR, announced in Feb 2015, is a full-frame camera with a 50.6 Megapixel resolution. Canon has also developed a 250MP full frame sensor shown in Fig.2.3. The pixel array contains $19{,}580 \times 12{,}600$ pixels on roughly a $29.2 \times 20.2\ mm$ sensor. Recently, Sony Corporation has announced the release of the IMX324, a new $1/1.7$-type stacked CMOS image sensor equipped with the industry's highest resolution 7.42 effective megapixel on roughly a $13 \times 9\ mm$ sensor for forward-sensing cameras in ADAS.

With respect to motion detection, there are some deficiencies of frame-based image sensors. One of the main deficiencies is caused by the pixel exposure mechanism. From

Figure 2.3 High-resolution CMOS image sensors. (a) Canon prototype 250MP full frame sensor [33], (b) Sony IMX324 CMOS image sensor for automotive cameras [34].

the operation described above, it is clearly illustrated that pixels are first reset to a high voltage and then exposed over a time interval to accumulate charges. Their respective light information is determined by the cathode voltage of photodiode at the time point of readout. Since the final readout voltage is determined by the total charges generated by the photodiode, any transient change of light illuminance cannot be captured by the APS pixels and therefore, the output voltage only represents the average level of the

sensed light illuminance during the exposure period. This issue is illustrated in Fig.2.4, it shows that the APS imagers cannot capture the details of the time-varying photocurrent of a pixel, but report the averaged value instead. This problem is the main cause of motion blur when observing fast motions and could be mitigated by increasing the image sensor framerate. Some high-speed image sensors could enhance the framerate as high as thousands of frames per second. However, it dramatically increases the total power consumption as well as the design complexity of the whole system.

Data redundancy is another important issue of frame-based image sensors, especially in the applications concerning motion analysis. According to the sensor

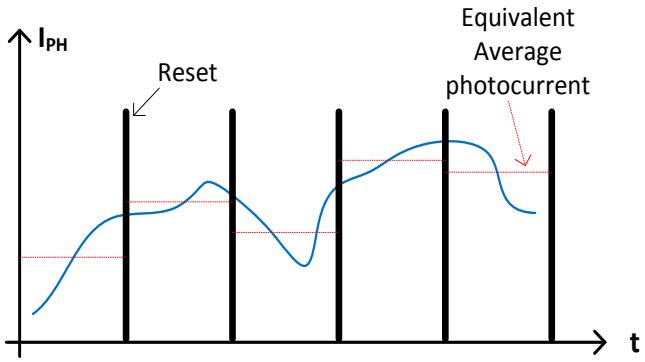


Figure 2.4 Illustration of the average illumination effect of APS imagers.

operation described above, each acquired image frame contains information of all pixels by means of the progressive scanning mechanism. The pixels are scanned periodically and mechanically to generate new frames, regardless on concerning whether the pixel information is effective and related to motion. The issue becomes more critical in the situation when high-speed cameras ($\geq$ 250fps) are used to detect fast motion. The information contained in pixels on stationary background in the visual scene is not useful for motion detection. However, it is still read out and processed along with each frame. As a result, tremendous redundant data is transmitted to be post processed, which in the increased readout channel bandwidth and power dissipation. Moreover, the cost of processing the system including large storage device and powerful post-processor is dramatically increased.

2.1.2 Event-based CMOS Image Sensor

2.1.2.1 Neuromorphic silicon retina

Despite the impressive development of frame-based imaging technologies, conventional frame-based image acquisition and corresponding processing system are still much less effective compared to their biological counterparts. Inspired by natural bio-vision systems, the "neuromorphic" concept was first introduced by Mead in the late 1980s. This concept is used to describe the circuits implemented using analog and asynchronous digital electronic circuitries that mimic the retina in biological vision system [35-37]. Studying and understanding the working principle of such retinas could stimulate the inspirations in terms of the development of "smart" image sensors which might be free from the aforementioned drawbacks of frame-based imagers in detecting high-speed motions.

In biological vision systems, the retina is an indispensable structure that is responsible for the acquisition and the first stage of processing of the visual information. As shown in Fig.2.5, the structure of the retina is complex with cells performing various functions. Rods and cones are the photoreceptors that encode the light information into electrical signals, which are transmitted to the driven horizontal cells and bipolar cells by means of the neurotransmitter. Horizontal cells are connected to photoreceptors and

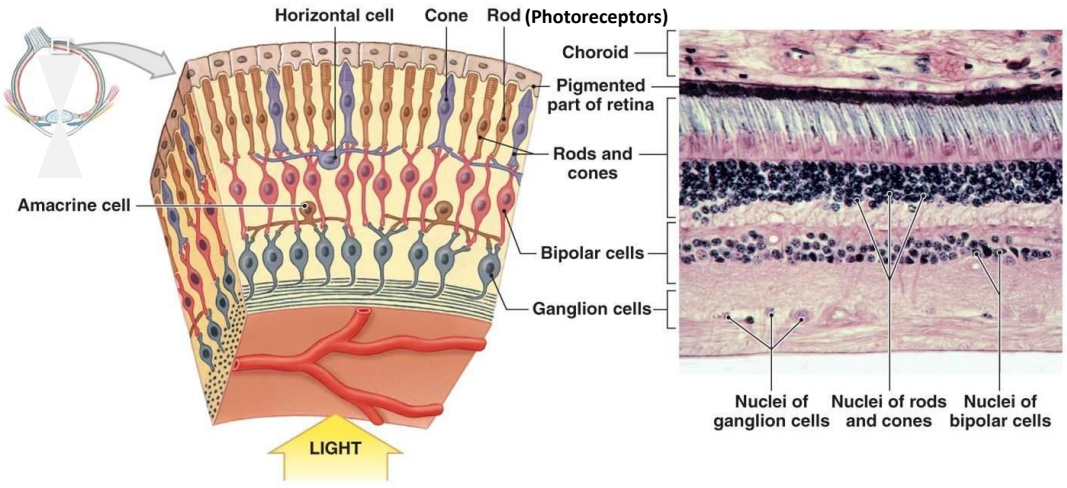


Figure 2.5 Cellular organization of the retina [38].

bipolar cells through synapses and help integrate and regulate the input current produced by photoreceptors. The bipolar cells can be classified into two types for coding bright and dark spatial-temporal contrast separately. Ganglion cells together with bipolar cells could be divided into two types, namely magno-cells and parvo-cells. In terms of cellular sensitivity, magno-cells are sensitive to transient change and parvo-cells concentrate on the sustained information in the scene [21].

Compared to frame-based image sensors, the operation principle of the biological retina is free of frames and asynchronously codes the visual information by detecting light intensity changes dynamically. The differentiation and collaboration of both types of ganglion cells (magno-cells and parvo-cells) enable the retina to distinguish between the static and dynamic light illuminance and help narrow down the interested area to be observed in the first stage of processing on the visual information. Light intensity changes are detected as spikes to trigger the biological vision system response, which is asynchronous and immediate. As a result, it gets rid of the average photocurrent due to photocurrent integration procedure in frame-based image sensors. Moreover, since only the dynamic area observed by the magno-cells needs to be further recognized by the parvo-cells, the data volume delivered for post-processing is smaller for the biological retina, which in turn allows higher information processing speed of the whole vision system. Based on the neuromorphic concept, the first silicon retina in domain of VLSI was proposed by Mahowald and Mead, which consists of circuits that mimic the photoreceptor, horizontal cells and bipolar cells [39]. However, there was no practically usable silicon retina until 2008 when the dynamic vision sensor (DVS) was developed [23].

2.1.2.2 Dynamic Vision Sensor (DVS)

Based on the understanding of the operation principle of biological retina, it was revealed that the active pixel in the frame-based image sensor is an emulation of parvo-cells, which can only report visual information in the scene. In order to enhance the functionality of the vision system as the biological retina, circuitries modeling the magno-cells were introduced such that the new vision system termed as silicon retina is able to directly detect light intensity changes. Although developed over decades, most

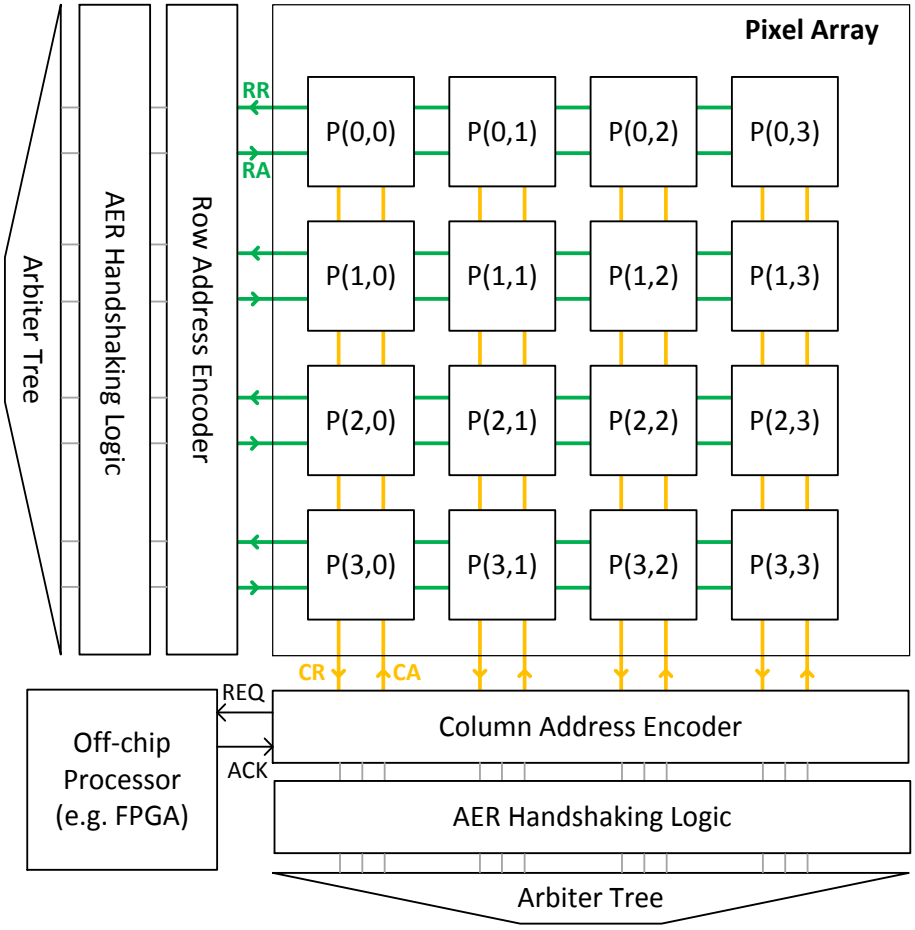


Figure 2.6 Typical architecture of a dynamic vision sensor.

of the early works on such silicon retinas were only for demonstration or verification of the neuromorphic theories [39-42]. At the early stage, the silicon retinas show deficiencies including large silicon area requirement and large pixel mismatches that make them unsuitable for real applications. The first usable practical neuromorphic device is the so-called dynamic vision sensor (DVS) proposed by Tobi's group in 2008 [23]. It is regarded as the first generation of dynamic vision sensor or event-based image sensor that asynchronously reports the light intensity changes detected by the pixels. In stark contrast to frame-based image sensors, the DVS has improved performance with wider dynamic range, lower latency, lower power consumption and acceptable pixel mismatch. The circuit topology, especially the pixel design, almost became the standard implementation in such kind of sensors.

A system level basic architecture of a DVS sensor with a simplified 4 × 4 pixel array is shown in Fig.2.6. The main functional blocks of DVS contain a pixel array, row and column address encoders and arbiter tree. The pixel array consists of multiple pixels in two dimensions, where the pixels have the same structure and are connected with each other through handshaking buses (RR, RA, CR and CA). Different from using shift registers in the progressive scanning, address encoders and arbiter tree are designed to implement the asynchronous four-phase handshaking protocol. Pixels detecting temporal contrast send out requests, which can be used to emulate the neuron spikes. Such pixels are regarded as active as they trigger events. With the help of address encoders and arbiter trees in both row and column directions, it is easy to locate the 2-D address of the active pixels reporting events. The details of the operation principle and typical pixel design is illustrated in the following parts.

DVS Pixel

Compared to the simple active pixels in frame-based image sensors, the pixel structure of DVS is more complex and exhibits four parts, as shown in Fig.2.7 [43]. The photoreceptor is a logarithmic receptor to convert light intensity information into electrical logarithmic output voltage (V_{PH}). The photocurrent produced by the photodiode is sourced by the feedback transistor M_{fb}. The gate of M_{fb} is connected to the output of an inverting amplifier ($A1$) whose input is connected to the photodiode. Since the output voltage V_{PH} dynamically responds to the input intensity, light intensity

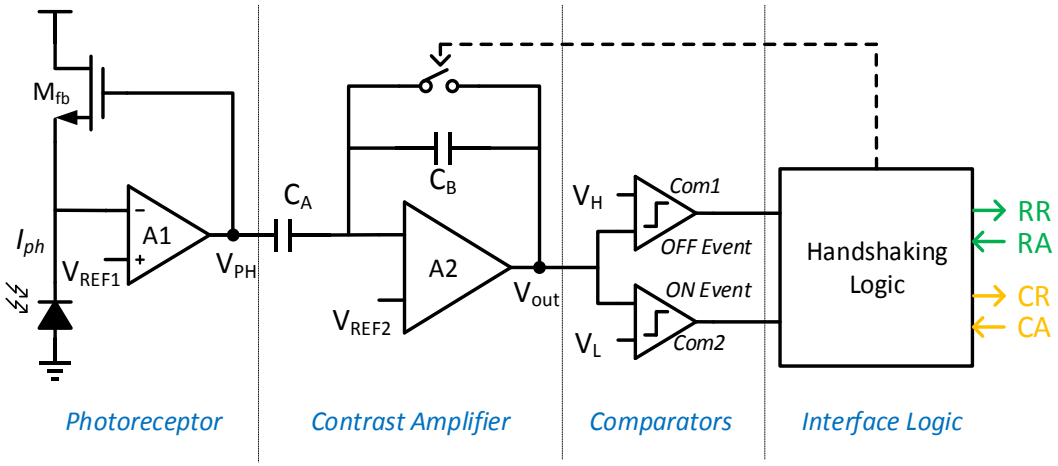


Figure 2.7 Schematic of a DVS pixel in block level.

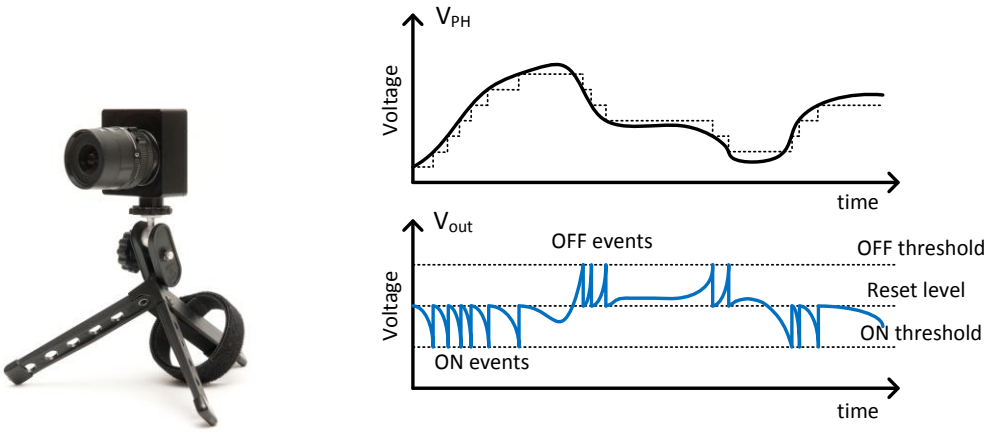


Figure 2.8 Operation principle of DVS [23].

changes result in corresponding variations of V_{PH}. The subsequent contrast amplifier (with a gain (C_A/C_B) of around 20) amplifies the V_{PH} variation and increases the temporal contrast sensitivity. The output of the contrast amplifier (V_{out}) is compared with a predefined reference voltages (V_H and V_L), which set the threshold of temporal contrast. An on/off event is generated when V_{out} reaches V_H or V_L. The pixel is then named to be fired and triggers the handshaking logic to send out request signals to peripheral modules to be processed.

The operation principle of the first-generation DVS is illustrated in Fig.2.8 showing an example waveform of two crucial signals V_{PH} and V_{out} of the DVS sensor. The output voltage of the logarithmic receptor, V_{PH}, continuously responds to the light intensity and the variation of V_{PH} represents the temporal changes of light intensity. This voltage is amplified by the subsequent contrast amplifier. The amplifier output voltage V_{out} is continuously compared by on/off comparators to detect pixel activity. Once the change of V_{PH} reaches the on/off threshold, the pixel is triggered to generate an on/off event, which initiates the four-phase handshaking communication and the peripheral circuits start to process pixel signals. Meanwhile, the in-pixel logic resets the contrast amplifier by pulling its output voltage V_{out} back to reset level V_{REF2}. It is observed that faster variation of V_{PH} results in a higher frequency of event generation. Timing diagram of four-phase handshaking with respect to processing one event is illustrated in Fig.2.9. The fired pixel first issues row request (RR) and when the row (of

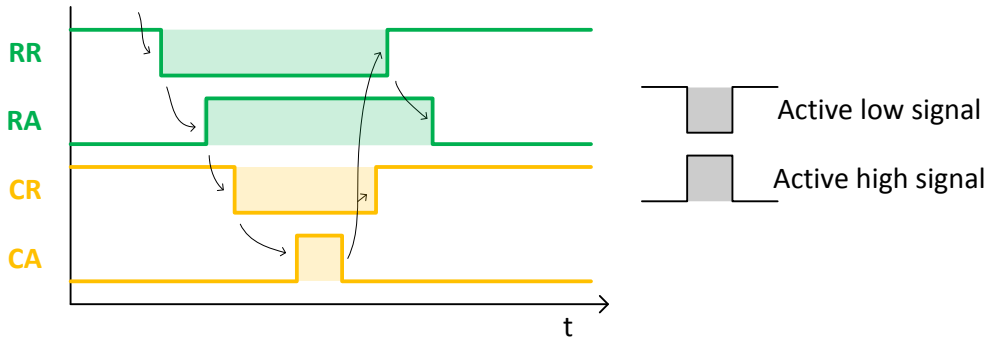


Figure 2.9 Timing diagram of four-phase handshaking protocol: RR-row request, RA-row acknowledgement, CR-column request, CA-column acknowledgement.

the fired pixel) receives row acknowledgement (RA) meaning that this row is selected to process, the fired pixel releases the column request (CR) in column direction. Once the column acknowledgement is granted to the column (of the fired pixel), the fired pixel (selected by both row and column acknowledgement signals) is being processed and its corresponding row and column numbers are encoded and output to the off-chip processor as address-events. Afterwards, the pixel goes back to the initial status (the status right after reset) and continues to detect light intensity changes. The arbiter tree is incorporated to avoid readout data collision when multiple pixels issue request simultaneously. The fired pixels would be processed sequentially based on the arbitration results of the row and column arbiter trees.

Address Event Representation (AER)

The Address-Event Representation (AER) is an asynchronous protocol for communication between different processing units, modeled after biological systems [176]. The AER contains three main types of handshaking blocks including the handshaking circuits within the pixel, the arbiter, and a chip-level handshaking logic block. The handshaking circuits within the pixel are responsible to store the pixel status, send reading requests (RR, CR), and receive acknowledgement signals (RA, CA). The arbiter receives the row or column request lines from the pixel array through the arbiter interface block and arbitrates amongst the active lines, as shown in figure 2.10. The arbiter interface block handles the handshaking signals from the row (or column) requests and acknowledges signals. For each dimension of row and column (assuming N rows or columns), the arbiter consists of a tree-like structure of $N - 1$ two-input

arbiter cells, arranged in a tree with $b = log2(N)$ levels required to arbitrate between N pixels. Each two-input arbiter gate outputs a Req to the two-input arbiter at the next level, if any one of its two input Req signals is active. For example, the input request signals are labelled as r11 and r12 for the two-input gate A1 and the corresponding Ack lines are a11 and a12. The output signal r21 from A1 goes to one of the two inputs of the next arbiter gate B1. This process continues to the top of the tree. The final selected output Req at the top of the tree then goes to an inverter, and the inverted output becomes the Ack signal that now propagates downward through the various gates until the corresponding RA line of an active RR line of a row of an array is activated by the arbiter interface block. For a 2D array, a second arbiter handles the CR and CA lines from the active pixels of the acknowledged row in the column dimension.

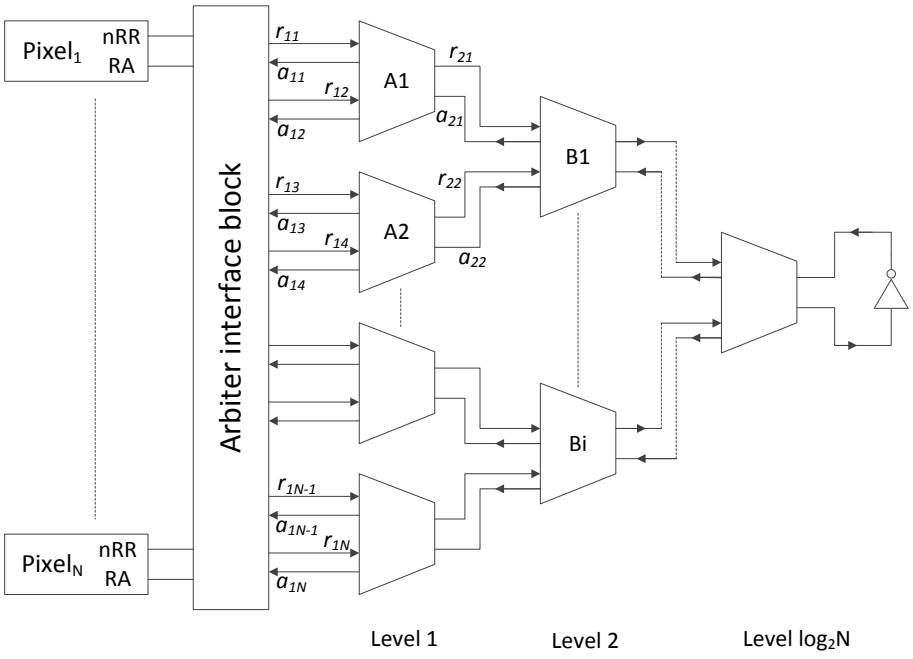


Figure 2.10 Communication interface between pixels of different rows and the arbiter tree of two-input

Arbiter circuits are designed to satisfy two criteria: low latencies and fairness. The first criteria ensure that incoming requests will be serviced as quickly as possible and the second ensures that requests are serviced in the order that they are received regardless of the position on the arbiter tree.

The original arbiter circuit used in the early neuromorphic chips is fair in its arbitration but slow in operation [179]. The communication cycle of this arbiter is slow because the arbiter block at the first level has to wait till the handshaking cycle is

completed in all the communicating processes of the arbiter tree before the acknowledge signal to this arbiter is activated. This design was used in the very early neuromorphic chips, including the retina of Mahowald [177]. The original arbiter circuit was modified by Boahen [178] to allow faster transmission of address events off-chip. As long as the alternate row request becomes active during the selection phase of the other input request, the request of another row will not be serviced thus reducing the handshaking time of the pixel by not having to switch to a different row. However, one disadvantage of this arbiter design is that a row with high activity tends to hold on to the arbitration bus leading to the transmission of events from only part of the chip. Later, Boahen [180] and Thu Linn [181] proposed different new arbiter structures to realize fair arbitration. In the new schemes, a pixel that makes consecutive requests to the arbiter is not revisited until the rest of the tree has been serviced. In the case of the chip with a fair arbiter, we see a profile where all pixels show a more even distributed recorded rate demonstrating that the pixels are serviced roughly equally often. The maximum spike rate of each pixel is limited by the refractory period of the event-generating circuit within the pixel.

Summary

Compared to the 3-T APS frame-based imagers, the creative design of DVS lies in the functional pixels and the scanning mechanism. To obtain light information, the APS sensors are forced and programmed to scan the whole pixel array periodically. However, the DVS is capable of autonomously providing relevant light information based on the light injection and selectively output pixel information. After the temporal contrast detection by the front-end circuitry, the pixel self-reports its information to the off-chip processor since the temporal contrast detection of the front-end circuitry in the pixel immediately triggers the pixel processing controlled by the peripheral logic through the four-phase handshaking protocol. The peripheral access circuitries no longer need to scan all pixels but only respond to active pixels. As a result, the efficiency of the DVS is dramatically improved compared to the frame-based image sensor. Moreover, the data transmitted to the off-chip processor is reduced to a great extent. The address-events are stamped temporal information when it is transmitted to processor. The dynamic visual image is reconstructed based on the address and temporal information

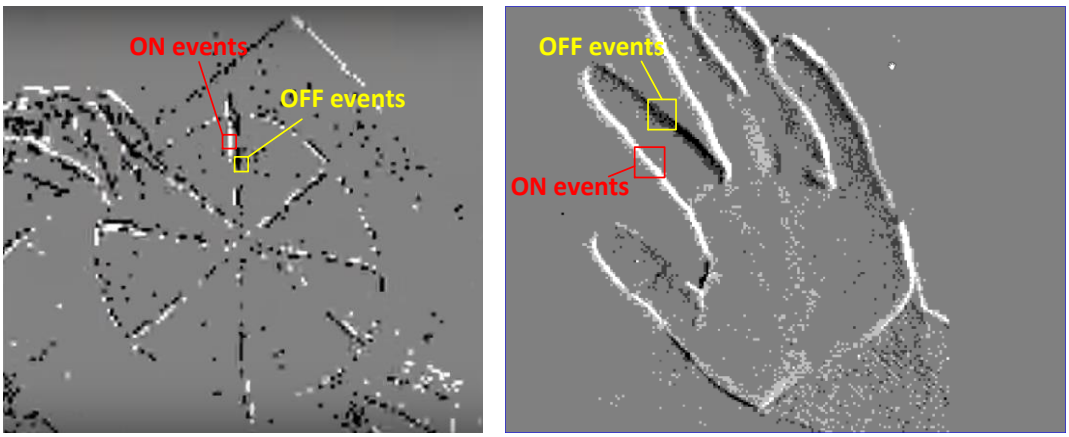


Figure 2.11 Examples of DVS result provided by IniLabs. (a) DVS result of a rotating circle and bars, (b) DVS result of a moving hand [44].

of the events, which is the so-called address-event representation (AER). Fig.2.11 shows two examples of DVS result provided by IniLabs. The left shows the result for rotating circle with four bars and the right figure shows the result of a moving hand.

Compared to frame-based image sensors, the DVS sensor [23] achieves much wider dynamic range (>120 dB), lower ($15\mu s$) event latency and lower mismatch (2.1% contrast). The short latency enables the sensor to perceive precise timing information of the events in observing fast moving objects. The output data volume, depends on the number of triggered events, is typically orders of magnitude lower than that of the frame-based counterpart. Combined with the lower power consumption, these significant features of DVS sensor making it suitable for low-power real-time applications in contrast to power-hungry data-redundant conventional APS imagers [45]. For example, a 128×128 DVS provides low event rate at $20k\ event/s$ (40 kB/s, 16-bit encoding for each event) and enables a robotic goalie system to achieve a mean latency of only $2.2ms$ from ball movement onset to generated motor command, whereas a comparable system using an APS imager with the same resolution running at 400fps would have a raw data rate of 6.6 MB/s (8-bit ADC).

TABLE I SPECIFICATION SUMMARY OF DVS IN DIFFERENT VERSIONS

	2015 [51]	2014 [49]	2013 [48]	2011 [47]	2011 [46]	2008 [23]
Technology	0.18 µm MM/RF 1P6M	0.18 µm IS 1P6M	0.35 µm IS 2P4M	0.35 µm MM/RF 2P4M	0.18 µm MM/RF 1P6M	0.35 µm MM/RF 2P4M
Resolution	60x30	240x180	128x128	128x128	304x240	128x128
Chip area(mm^2)	3.2x1.6	5x5	4.9x4.9	5.6x5.5	9.9x8.2	6.3x6
Pixel area (um2)	31.2x31.2	18.5x18.5	31x30	35x35	30x30	40x40
Fill factor (%)	10.3	22	10.5	8.7	10	8.1
Power @ 100eps (mW)	0.72	14	4	145	175	24
Power/Resolution (µW)	0.4	0.32	0.24	8.8	2.4	1.5
Contrast Sensitivity (%)	1	11	1.5	10	13	17
DR (dB)	130	130	120	100	N.A.	120
Intra-Scene DR (dB)	130	130	60	56	N.A.	120

2.1.2.3 Hybrid Dynamic Vision Sensor with Intensity Readout

According to the operation principle, the DVS sensors only report temporal contrast regardless of the illumination value. The absence of absolute light intensity information restricts its applications in some computer vision algorithms, including optical flow computation, which requires light intensity information to extract interested features. As a result, researchers proposed new architectures of hybrid dynamic vision sensors that integrate the DVS with the snapshot function of conventional frame-based imagers. Others attempted to improve the specifications of DVS including sensitivity, and power consumption etc., as summarized in Table I. State-of-the-art event-based image sensors which can provide absolute light intensity information are named asynchronous time-based image sensor (ATIS) based on pulse width modulation (PWM) and dynamic and active pixel vision sensor (DAVIS) through integration of DVS and APS.

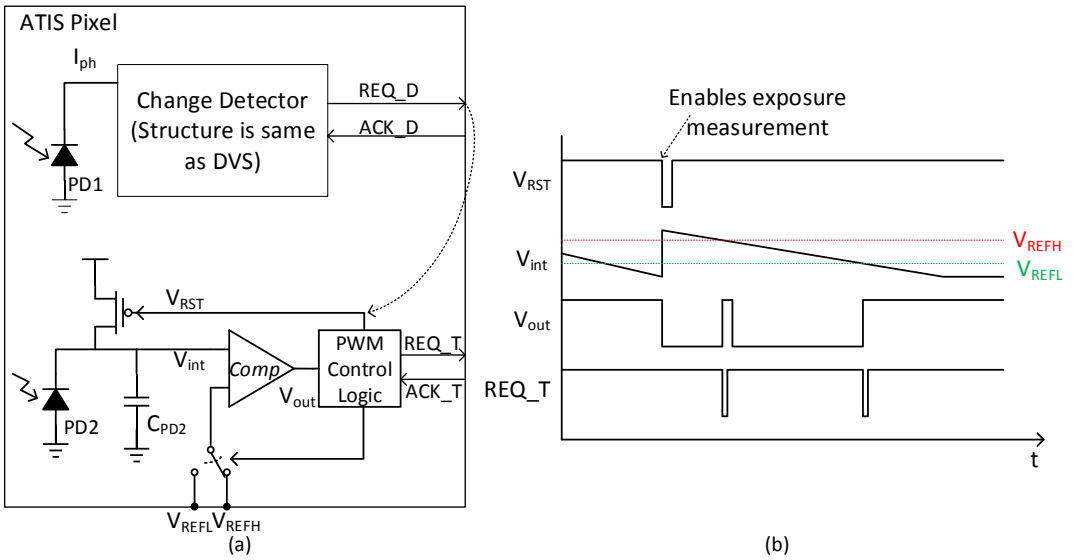


Figure 2.12 Architecture of ATIS pixel and timing diagram of operation principle. (a) Structure of ATIS pixel, (b) Timing diagram of exposure measurement unit.

Asynchronous time-based image sensor (ATIS)

Based on the DVS pixel, ATIS, introduced by Christoph Posch in 2011 [46], introduced an extra PWM imaging circuitry to measure the absolute light intensity sensed by an individual pixel. The structure of the ATIS pixel is shown in Fig.2.12 (a), which consists of a change detector and an exposure measurement unit. The change detector has the same structure as the DVS sensor and generate events when temporal contrast is detected. The generated event thus triggers the exposure measurement unit to measure the absolute light intensity based on PWM scheme. As a commonly-used current-measurement method, the PWM technique first converts the input current into a voltage with linear variation through an integrator. The produced voltage is then fed into comparators with constant threshold. The time instant when this voltage reaches the threshold is measured and used to evaluate the input current. The PWM method is effective in quantizing current and has been widely applied in the time-based image sensors to broaden the dynamic range.

The design of the exposure measurement unit makes an improvement on the traditional PWM circuits by employing two threshold voltages (V_{REFH} and V_{REFL}) allowing for the implementation of time-domain correlated double sampling (TCDS).

The operation principle could be explained with the timing diagram of the main signals as shown in Fig.2.12 (b). When the change detector reports an event, it generates a short reset pulse (V_{rst}) which enables the exposure measurement. The integration voltage V_{int} is first reset to VDD and then drops during the subsequent integration stage. The PWM control logic is first connected to the high threshold (V_{REFH}). When V_{int} reaches V_{REFH}, the comparator flips and the PWM control circuit sends out the first request (Req_H) to the external handshaking logic. At the same time, the threshold voltage of the comparator is switched to the lower threshold voltage (V_{REFL}). The second request (Req_L) is issued when V_{int} reaches V_{REFL}. The time interval (t_{int}) between the two requests is computed to evaluate the absolute light intensity. A longer t_{int} indicates lower illumination and vice versa.

The PWM scheme ensures the wide intra-scene dynamic range of the ATIS while the proposed TCDS circuit enhances the image quality with reduced fixed-pattern noise (FPN) and improved signal-to-noise ratio (SNR). In addition, the time-domain readout of the light intensity information through asynchronous request signals is fully compatible with that of the DVS sensor. This simplifies the peripheral logic design and the back-end signal processing since both the temporal contrast and the grayscale data are reported in terms of asynchronous events. Combined with the in-pixel conditional exposure measurement triggered by the front-end change detector, the ATIS could yield lossless high-quality video compression across the focal plane.

Dynamic and Active Pixel Vision Sensor (DAVIS)

DAVIS, proposed by Tobi Delbruck [49, 50], makes use of the conventional APS to achieve the absolute light intensity information. It is a compact design which integrates the conventional 3-T or 4-T APS with the DVS sharing a single photodiode. The design is based on the fact that the front-end photoreceptors of DVS code the light temporal contrast information without destructing the corresponding photocurrent. Therefore, the intact photocurrent could be integrated by the APS to generate a voltage representing the absolute light intensity information.

The pixel circuit is shown in Fig.2.13, which contains two parts: the APS (red) and DVS (blue). The APS circuit part could be implemented using either 3-T or 4-T APS

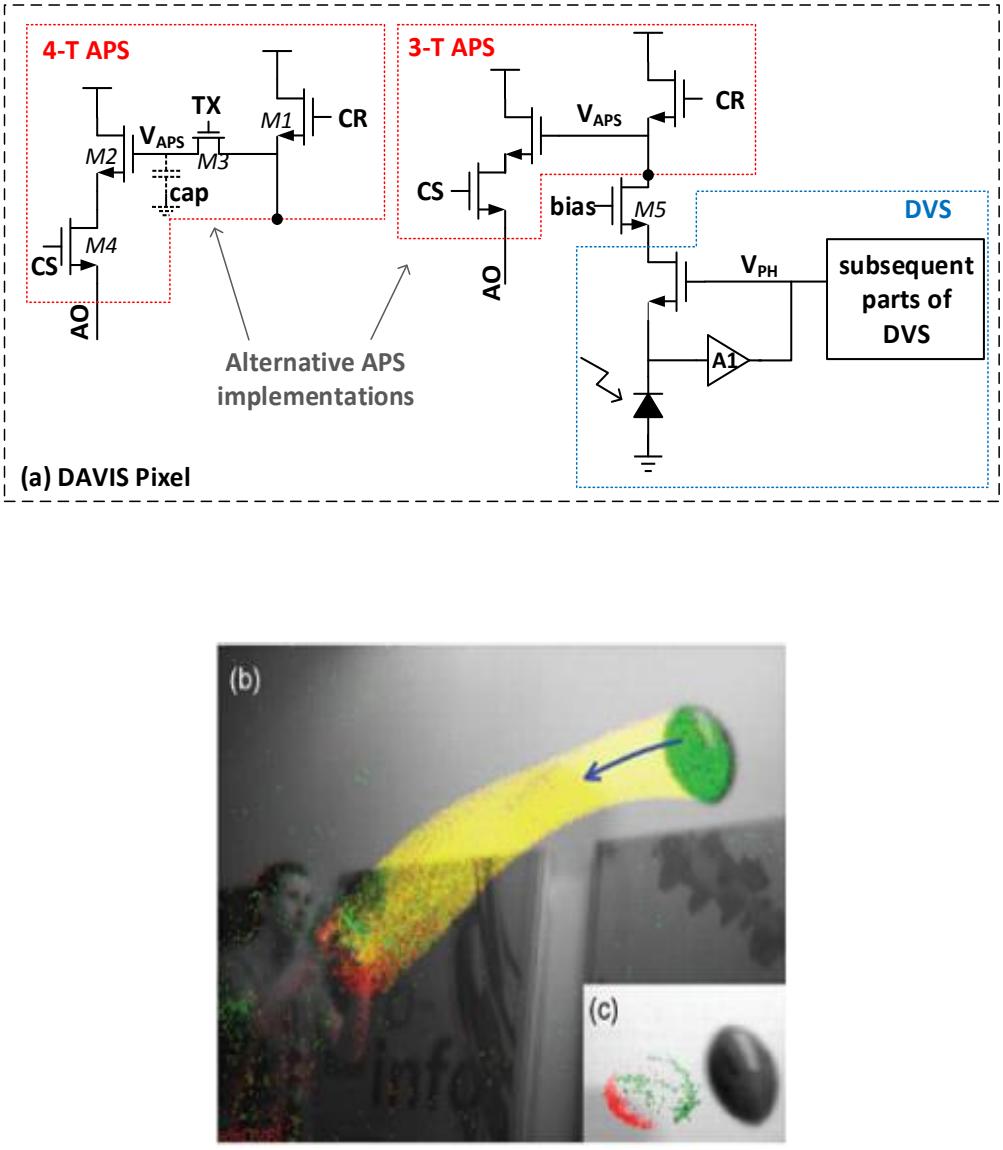


Figure 2.13 Architecture of DAVIS and an example. (a) Structure of DAVIS pixel, (b) an example of DAVIS result from the 240×180 sensor, including a full frame image with pixel intensities and binary on/off event marked as red/green events, (c) binary events and intensity of the region of interest (ROI) with an obvious lag.

with an extra cascade transistor ($M5$) as an isolation of these two parts. The design of this pixel has several advantages. Firstly, although combined in a single pixel, the APS and DVS could operate independently without interfering with each other. The 4-T APS ($M1 - M4$) is preferred in this design to implement the global shutter scheme, instead of rolling shutter, during the intensity readout mode. The use of global shutter is crucial to avoid the appearance of undesired motion blur in images since all pixels could be controlled synchronously to be exposed during the same time period. Therefore, the

outputs of DAVIS contain full frame gray images and binary dynamic events. Another advantage of this design is that it enables the readout of the absolute light intensity corresponding to the region of interest (e.g. motion-related), which makes this design possible to be applied in more computer vision applications.

Advantages and Limitations of ATIS and DAVIS

Although implemented differently, both ATIS and DAVIS make the DVS capable of achieving absolute light intensity information. The corresponding intensity readout mechanism of ATIS and DAVIS alleviates problems such as large data redundancy and power consumption from which the conventional frame-based imagers suffer. Moreover, such integration of functionalities of event-based and frame-based image sensors also broadens the application prospect of the DVS from the early high-speed motion detection and object tracking to more generalized machine vision algorithms such as object recognition and classification.

On the other hand, however, both designs still suffer from some limitations which might hinder their popularity to some extent. The most noticeable problem of ATIS is the requirement of large pixel area, which consumes $30 \times 30\mu m^2$ even using $0.18\mu m$ CMOS technology. A large part of the valuable pixel area is occupied by the adoption of two photodiodes, together with the complicated intensity readout circuitry. This poses great threats for the further implementation of sensors with larger resolution, which is usually a significant specification for some industrial applications. As for the DAVIS, although the output frame rate could be adjusted according to the dynamic contents of the visual scene, some degree of data redundancy still exists because a full image, instead of just the region of interest, is exported as a whole. What's more, the APS intensity readout method also restricts the intra-scene dynamic range to only $51dB$, much lower than that of the natural scene (> 120 dB) [23]. In addition to the limitations mentioned above, both designs require exposure time for light integration to achieve the brightness information. Though the output is in the form of asynchronous time spikes, the PWM technique adopted by the ATIS, is a variant of APS in essence but transforms the integration voltage into time-domain information. Therefore, both cases require some exposure time which in turn increases latency and degrades the temporal resolution of dynamic events, leading to the undesired discordance between the very

fast address-events output and the postponed intensity readout. This limits the potential advantages of the event-based system. For example, the address-event latency is in the order of several microsecond for both designs. While the time resolution for the intensity readout could be as long as tens of millisecond. Such mismatch could either limit the maximum achievable framerate of the sensor (ATIS) or cause obvious image lag between the dynamic and static information of the scene (DAVIS).

2.2 Object Visual Tracking

Object tracking is one of the most challenging tasks in the field of computer vision and finds a wide range of applications including video surveillance, robotics, medical imaging, human-computer interface and vehicle navigation. The goal of visual tracking is to locate a moving object over time. Object tracking algorithms have acquired priority due to the availability of highly sophisticated computers, good quality and inexpensive cameras. However, it is a big challenge to design a fast and robust tracker according to various critical conditions in visual tracking, such as deformation, illumination variation, occlusion, blur and fast motion, background clutter and so on. Traditionally, object tracking is conducted on videos, i.e. a series of image frames, which is captured using conventional frame-based image sensors. Over the past decades, various frame-based visual tracking algorithms have been proposed to cope with these issues. Some of the classical tracking algorithms use generative models [2-6], which perform tracking by searching the best-matching windows, and some algorithms use discriminative models [7-10], which learn to distinguish the moving target from background typically through a trained classifier. Discriminative methods are preferred especially when the input data size is tremendous [11, 12]. In particular, correlation filter-based trackers have been developed and made significant contributions in visual tracking since the proposal of Minimum Output Sum of Squared Error (MOSSE) filter [13]. The main achievement of correlation filter-based tracker is to speed up the post-processing by reducing the computational complexity using Fast Fourier Transform (FFT). However, such methods find it hard to handle fast moving object tracking. In recent years, deep learning methods have also been applied in object tracking and have shown great performance in benchmarks [14]. However, currently there is no standard offline training and testing sets for deep learning methods. There are yet no theoretical proofs developed for such methods. Therefore, discussions on deep learning based tracking methods will not be included in this work. Due to the development of event-based dynamic vision sensors, several event-based tracking methods have also been developed [25-26, 117].

In this chapter, I first review the key fundamentals of classical tracking methods, including object detection methods, feature extraction and popular classifiers. After that, the dominant frame-based trackers are reviewed followed by a review of current development of event-based tracking methods.

2.2.1 Fundamentals

The basic framework for object visual tracking is shown in Fig.2.14 [61]. The tracking flow is an iterative process and the tracking process in each frame typically first use extracted features and learning algorithms to recognize one or multiple target moving objects and then localize the target objects.

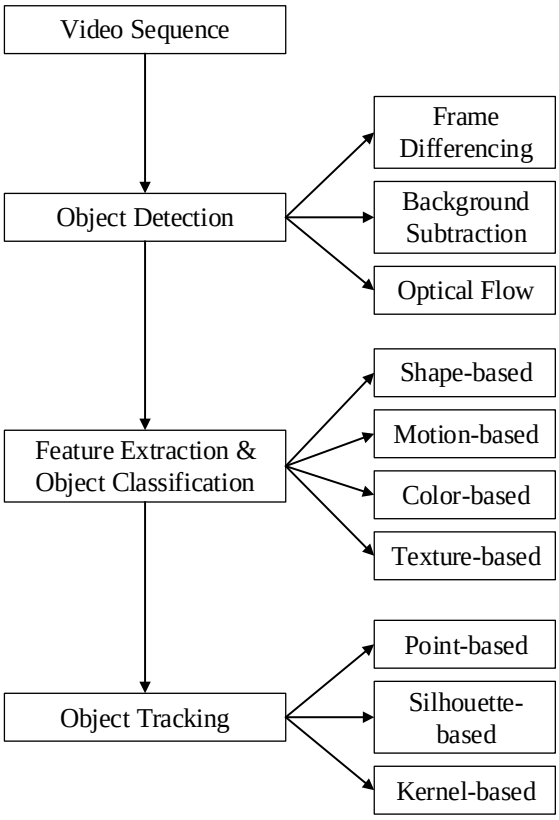


Figure 2.14 Basic steps for object tracking-by-detection [61].

The main steps of object tracking include data capturing, object detection, feature extraction and object classification, and object tracking. Object detection is the process of finding instances of real-world objects such as faces, bicycles, and buildings in images or videos. Object detection algorithms typically use extracted features and

learning algorithms to recognize instances of an object category. It is commonly used in applications such as image retrieval, security, surveillance, and advanced driver assistance systems (ADAS). A variety of models can be used to detect objects, including deep learning object detection, feature-based object detection, Viola-Jones object detection, SVM classification with histograms of oriented gradients (HOG) features, and image segmentation and blob analysis. In object classification, the detected objects are classified into different categories, such as birds, vehicles, and other moving objects. For single moving object detection, object classification is simplified to classify samples as target object or background. Visual tracking is the process of locating single or multiple objects over time from the data captured by a camera. The process of object tracking involves selection of region-of-interest (ROI) and localization updating over multiple frames.

2.2.1.1 Object Detection

Moving object detection is the initial step to start video tracking. It is conducted in each frame for detection based tracking, while for online learning trackers, object detection is only implemented in the first frame when the target object appears. Two widely used techniques are background subtraction method and frame differencing method [62]. Optical flow is another typically used detection method and will be reviewed in detail in the next part.

Temporal differencing

The temporal differencing methodology calculates the difference between two consecutive images in a video sequence. Then the moving objects are detected with the differences. This method is highly adaptive to dynamic environments and simple for implementation. Yet the detection results have a poor performance since it is tough to obtain a complete outline of the moving object. It may also result in small holes appearing in the detected moving objects for which a solution is mentioned in [7] based on temporal difference method. Devi et al. [63] presented motion detection using background frame matching. This is a very efficient method of comparing image pixel values in subsequent still frames captured after every two seconds from the camera. As shown in Fig.2.15 (a), two frames are required to detect movement. The first frame is

called reference frame and the second frame, which is called the input frame contains the moving object. The two frames are compared and the differences in pixel value are determined. The functionality of the temporal differencing and DVS are similar with the differences that DVSs sense and detect intensity change in sub-pixel level rather than frame level and have a much higher temporal resolution compared with the inter-frame time interval used in temporal differencing.

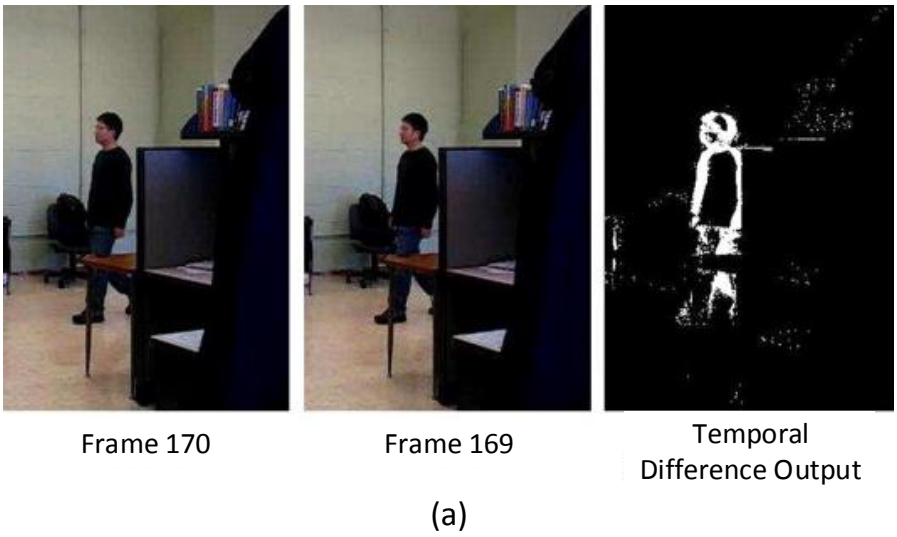


Figure 2.15 Illustration of object detection algorithms. (a) Temporal differencing conducted on two consecutive frames [64], (b) background subtraction method by subtracting the referenced background model from the current frame [65].

Background subtraction

Background subtraction, also known as foreground detection, is a widely used technique for detecting moving objects in videos from static camera. Generally, background subtraction is conducted by subtracting the current frame from a pre-captured static background frame or a background model, illustrated in Fig.2.15 (b). Background modelling can be categorized into two categories which are non-recursive and recursive techniques [61]. Recursive algorithms include frame differencing, adaptive background, Gaussian mixture model and approximate median. These approaches do not buffer a static background image, but recursively update a single background model based on each input frame. Therefore, the current reference background is affected by the past input frames. This algorithm requires less storage, however, if error occurs in estimation of background, it will exist for longer time. For non-recursive algorithms, the background is estimated by taking use of a sliding window. It stores a buffer of previous L frames and uses the variation of each pixel to estimation the background image. Such techniques are more adaptable since they do not depend on frames beyond those in buffer. Nonetheless, if large buffer is required to cope with slow-moving traffic, the storage requirement is prominent.

Rakibe and Patil [66] proposed an algorithm based on background subtraction for detecting moving objects with a static background scene. The algorithm first builds a reliable background updating model based on statistical followed by morphological filtering to remove noise and solve the background interruption difficulty. Kim [67] presented a method for tracking multiple objects with static background scene by combining both the background modelling and particle filter. The searching area is restricted by the background subtraction such that the particle filter can efficiently prediction the location of each moving object. Tang, Zheng, et.al [68] developed an innovative system for robust object segmentation and tracking, which dynamically controls the decision thresholds of background subtraction and shadow removal around the adaptive kernel regions based on the preliminary tracking results. The evaluation of the method demonstrates a good performance in similar-colour background and shadow areas. Background subtraction methods are generally the simplest approaches for object detection. However, they are more applied for tracking on moving object with static

background. What's more, these methods are highly sensitive to image noise, gradual variations of the lighting conditions in the scene and small movements of non-target objects such as tree branches blowing in the wind. Another critical issue is that most of the time shadow regions projected by foreground are detected as moving objects.

Optical flow

Optical flow method is to compute the optical flow field between images, and do cluster processing according to the optical flow distribution characteristics of images. It can get the complete movement information and detect the moving object from the background better [62]. Shafie et al. [69] presented motion detection using optical flow method. Optical flow can arise from the relative motion of both the objects and the viewer, so it can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement. Discontinuities in the optical flow can help in segmenting images in to regions that correspond to different objects. However, compared with frame difference methods and background subtractions, optical flow is more complex, and the details of optical flow algorithms will be reviewed in next part.

TABLE II COMPARISON OF OBJECT DETECTION METHODS

Methods	**Merits**	**Demerits**
Frame Differencing	- Adaptive to dynamic environments. - Simple for implementation	Detection result has low accuracy
Background Subtraction	- Low memory requirement. - It does not require sub sampling of frames for creation of an adequate background model.	- Its computation requires a buffer with the recent pixel values. - It does not cope with multimodal background - Sensitive to image defects
Optical Flow	- It can provide complete movement information. - The motion information contained can simplify the following processing such as image segmentation	Require large amount of calculation

Combined motion detection methods

The above reviewed three object detection methods have their own merits and demerits, which are listed in Table II. Recently, the combinations of these approaches for moving object detection have caused widespread attention. An improved object detection method based on background model and inter-frame difference is proposed in [70], which reduced the number of ghosts in the background subtraction using Gaussian mixture model and the serious cavities in the inter-frame difference method and effectively detected moving objects under the condition of the background mutations. However, the detection is not accurate for complex environment. Chen, et al. [71] presented a technique which improved the frame difference method by first classifying the blocks in the frame as background and others using correlation coefficient. Yuan, et al. [72] put forward a new moving object detection method, which calculated optical flow information with selecting the image representative Harris corner pixels, simplified the calculation of optical flow and three-frame difference method was introduced as a supplement, while the optical flow was more complex. Guo, et al. [73] introduced the image repair and morphological processing to an improved object detection method based on frame-difference and background subtraction which can eliminate noise and fill cavities quickly and detect more complete moving objects.

2.2.1.2 Feature Extraction

The detected region of moving object using object detection methods are then processed to extract features, which are representations of original image regions and are used for classification to categorize the detected objects. Classification is a process in which individual items such as objects, patterns, pixels, etc. are grouped and categorized based on the similarity between the items and the feature descriptions of the groups. The accuracy of object classification thus depends on the type of classifier and the extracted object features used for classification. According to feature descriptions, classification in computer vision are generally classified as shape-based, motion-based, colour-based and texture-based classifications.

Shape-based features

Shape-based features are purely applied to the object geometry instead of structural analysis. It extracts geometry features of target image regions like boxes, blobs, etc. Input features are the mixture of image-based and scene-based object parameters such as image blob area, apparent aspect ratio of blob bounding box and camera zoom. Classification is performed on each blob in every frame and the results are kept in histogram. Hota, et al. [74] proposed an online feature selection method which selects a good subset of shape features while the machine learns the classification tasks and use the selected features for object classification. Cho, et al. [75] presented an effective method for detecting and classifying card images in which shape features are used to filter out regions with low possibility. Jose, et al. [76] conducted event classification using shape-based data analysis by constructing the events into shape spaces. The shape features are aligned with time-series measurements for better feature selection in power systems.

Motion-based features

In image based object recognition, shape features are mostly used while for video-based moving object analysis, motion features are also employed. The motion-based features of moving objects, such as motion history image (MHI), motion energy image (MEI) and optical flow-based motion descriptors, have a strong cue of periodic property shown by a non-rigid articulated motion of the object. Rigidity analysis and periodicity of moving entities are done by residual flow. The non-rigid dynamic objects such as human being can have greater average residual flow and display a periodic component, while the rigid objects are expected to have little residual flow [11]. Optical flow is helpful for motion-based object classification since it provides complete motion information of pixels or blobs [77-83].

Colour-based features

Colour space or colour features can be easily acquired and are relatively more stable under viewpoint variations, compared to other image features. Low computational cost of algorithms proposed makes colour a desirable feature to exploit when appropriate. In real-time object detection and tracking system with content-based image retrieval

(CBIR) system, colour-based searching is the most efficient and simple searching approach [79]. Various methods have been developed to extract colour features, including colour histogram, colour correlogram and colour moments [80].

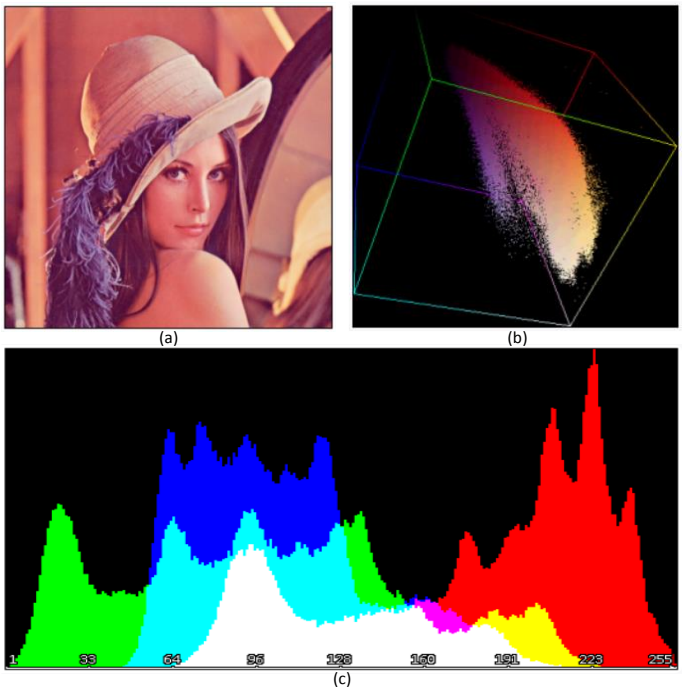


Figure 2.16 (a) Original Lena image, (b) 3D colour distribution of the original Lena image, (c) colour histogram of the original Lena image [81].

Swain and Ballard [82] illustrated the use of colour histogram and histogram intersection for efficient indexing of large databases. Given a colour space (e.g., red-green-blue (RGB)), a colour histogram makes statistics of the frequency of each colour (or each range of colour) occurs in the whole image. Fig.2.16 shows an example of colour histogram and 3D colour distribution of original *Lena* image (generated by G'MIC [81]). The RGB colour space is partitioned into a number of bins. The size of the ball located in a bin represents the frequency of the corresponding range of colour. Colour histogram is an efficient representation of colour images. It is widely used in content based image retrieval (CBIR) system, which is a computer system for searching and retrieving images from a large database of digital images [83]. Wang [84] proposed a robust CBIR approach based on local colour histogram. The image was first divided into several blocks and a colour histogram was calculated in each block. Similar technique was adopted in [85], where Wang et al. computed colour histogram on local feature regions which were obtained by using multi-scale Harris-Laplace detector and

feature scale theory. Han et al. [86] presented a new histogram, the so-called fuzzy colour histogram, and demonstrated its superior retrieval performance than conventional colour histogram. Huang et al. [87] introduced spatial correlation of colour into colour histogram and defined a new image feature called colour correlogram. Colour correlogram-based object tracking was illustrated in [88].

Texture-based features

An image texture is a set of metrics calculated in image processing to qualify the perceived texture of an image. It provides information about the structural arrangement of colour or intensities in an image or selected region of an image. It also describes the relationship between foreground object and background environment. In visual tracking, texture features are important features of the target description. Compared to other features, texture features are more robust towards image noise. Qian, et al. [89] proposed a hyperspectral feature extraction and pixel classification method based on 3D discrete wavelet transform texture features, which allows the capture of geometrical and statistical spectral-spatial structures of the target surface. Brewster, et al. [90] presented a new approach to aggregate a set of texture features, including grey-level co-occurrence matrix (GLCM), local binary patterns (LBP), and local directional patterns (LDP), to retrieve a better representation of texture such that classifiers can more accurately distinguish between the textures of targets and non-targets. The performance of the new texture feature extraction method is evaluated by a support vector machine (SVM).

After decades of research on texture feature extraction, various methods have been developed in the field of computer vision. Among these methods, two approaches have become most widely used for object classification and detection in recent years, which are Histogram of Oriented Gradients (HOG), and Haar-like feature extraction [91]. These three approaches provide good robustness, and high potential in real-time performance. Thus, it is necessary to fully understand the function of these features and their respective characteristics in object classification.

HOG features

HOG is a feature descriptor used to detect objects in computer vision and image processing, given the fact that local object appearance and shape can often be well characterized by the distribution of local intensity gradients or edge directions, even when there is no precise knowledge of the corresponding gradient or edge positions [92]. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image – detection window, or region of interest (ROI). The implementation of the HOG descriptor algorithm, as shown in Fig.2.17, contains following steps:

1. Divide the image into small connected regions called cells, and for each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell.
2. Discretize each cell into angular bins according to the gradient orientation. Each cell's pixel contributes weighted gradient to its corresponding angular bin.
3. Groups of adjacent cells are considered as spatial regions called blocks. The grouping of cells into a block is the basis for grouping and normalization of histograms.
4. Normalized group of histograms represents the block histogram. The set of these block histograms represents the descriptor.

HOG features play an important role in detection and tracking systems. Wang, et al. [93] proposed a human detection approach which is capable of handling partial occlusions. In their research, HOG features and LBP features are integrated as new global- or local- feature descriptors which are learned by a linear SVM. Based on the response of each block of the HOG feature to the global detector, the occluded regions can be inferred. Zeng, et al. [94] proposed an extended correlation filter-based tracking system using multi-channel correlation, filters which is inspired by single-channel correlation filters, such that multi-channel HOG descriptors are used to represent the image patch. Compared with conventional correlation filter-based tracking algorithms, which use linear classifier and process raw image, the new method significantly improved the robustness of filters.

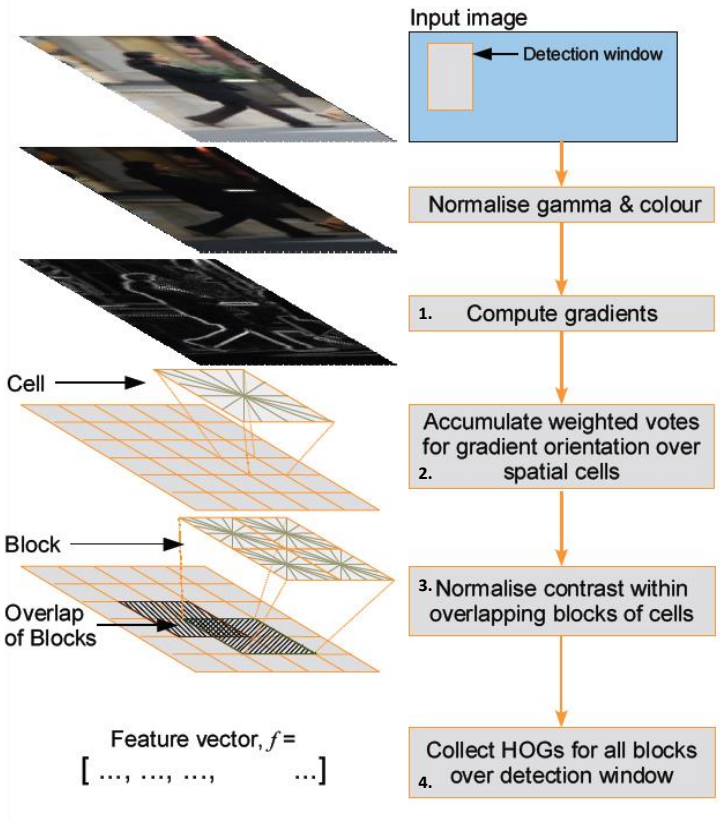


Figure 2.17 An overview of static HOG feature extraction. The detector window is tiled with a grid of overlapping clocks. Each block contains a grid of spatial cells. For each cell, the weighted vote of image gradients in orientation histograms is performed. These are locally normalized and collected in one big feature vector [92].

Haar-like feature

Haar-like features are intuitively similar with Haar wavelets and were used in the first real-time face detector [95]. Haar-like features are an over complete set of two-dimensional (2D) Haar functions, which can be used to encode local appearance of objects. The extraction of each Haar feature refers to two or more rectangular regions enclosed in a temple. Some of the classical templates are shown in Fig.2.18. Each template contains two parts, white rectangles and black rectangles. The feature value of each template, which is applied to a specific region in the original image, is computed by subtracting the sum of respective pixels inside the black rectangles from the sum of pixels in the white rectangles. Each template is designed to extract a specific feature. The main differences between these templates lie in the number of rectangles and the orientation of the rectangles with respect to the template.

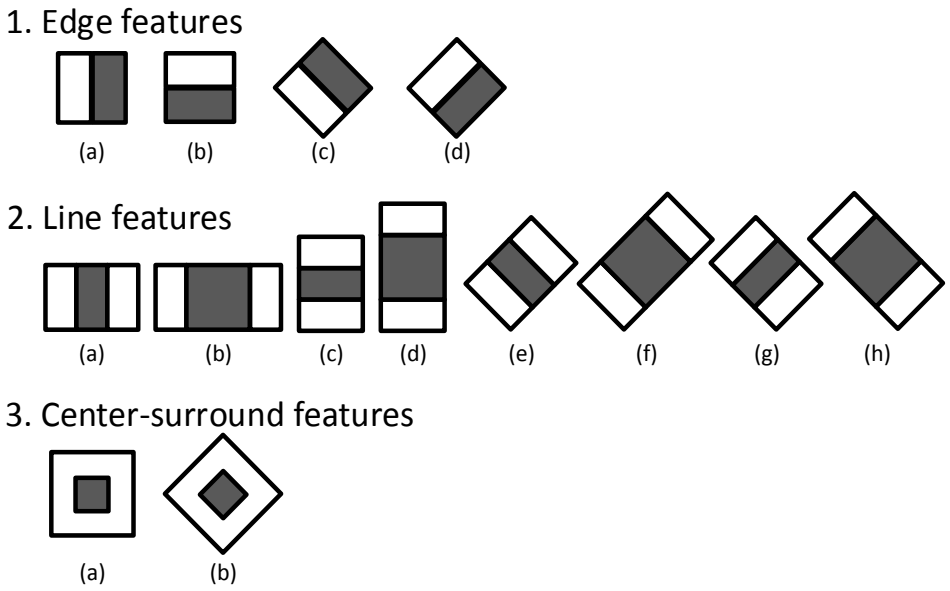


Figure 2.18 Example templates of Haar-like features.

One of the main reasons for the popularity of the Haar-like features is that they provide good performance on the trade-off between accuracy and speed of evaluation. The computation of Haar-like features has high efficiency with the help of integral images, proposed by Viola and Jones [95]. Integral images can be defined as 2D lookup tables in the form of a matrix with the same size of the original image. Each element of the integral image contains the sum of all pixels located on the up-left region of the original image. The integral images allow to compute sum of rectangular areas in the image, at any position or scale, using only four lookups. To illustrate, Fig.2.19 shows an image and its corresponding integral image. The integral image is padded to the left and the top for calculation. The pixel value at $(3, 2)$ in the input image becomes the pixel value at $(4, 3)$ in the integral image after adding the pixel value to the left and up. Using an integral image, summations over image sub-regions can be rapidly calculate. For example, the summation of the blue-shaded region in the input image becomes a simple calculation using four reference values of the rectangular region in its corresponding integral image. The calculation becomes, $46 - 22 - 20 + 10 = 14$. Therefore, the computation of sum of rectangular areas in the image, at any position or scale, can be expressed as following:

$$sum = I(A) + I(C) - I(B) - I(D) \quad (2\text{-}1)$$

where points A, B, C, D belong the integral image I. Each Haar-like feature may need more than four lookups, depending on how it was defined.

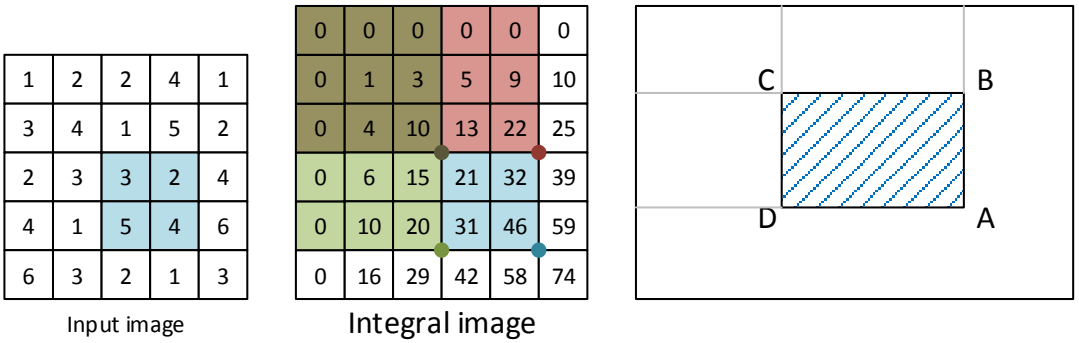


Figure 2.19 An example of the Haar-like feature calculation using integral image.

2.2.1.3 Classification

After the review of various kinds of feature extraction approaches, this section will discuss multiple popular classifiers which play a dominant role in computer vision tasks. Classification itself is a process of machine learning that it assigns an input instance to one of a fixed set of classes as output by learning a model from the training instances. In supervised learning using statistical methods, classifiers can be classified into generative classifiers and discriminative classifiers based on the type of data modelling, i.e. generative models and discriminative models. The target of all classifiers is to learn the probability of an instance belonging to each class, i.e. conditional probability distribution $P(Y|X)$. A generative classifier learns a model of joint probability $P(X,Y)$. They first assume some parametric form for $P(X,Y)$and prior probability (Y) . Then conduct parameter estimation of $P(X,Y)$ and $P(Y)$ from the training data. Once achieved the prior probabilities, the target post priority $P(Y|X)$can be calculated by using Bayes rule. Representative generative classifiers include Naïve Bayes, Hidden Markov Models (HMM), and Gaussian Mixture Model (GMM). In contrast, a discriminative classifier directly assumes some parametric form for the target post priority $P(Y|X)$, followed by a parameter estimation from the training data. Typical discriminative classifiers include Logistic Regression, Support Vector Machine (SVM), and traditional Neural Networks.

The principal advantage of discriminative classifier is that it tends to be more robust than generative models to violations of their independence assumptions. It is proved considering the family of joint distributions whose conditionals all take the 'logistic regression form'. But there are many other joint models, some with complex dependencies among X, whose conditional distributions also have the same form. By modelling the conditional distribution directly, we can remain agnostic about the form of $P(X)$. The principal advantage of generative classifier is that it has the elegant probabilistic concepts of priors, structure, uncertainty, and so forth, which give it explanatory power and a convenient way to insert prior knowledge. In discriminative classifier, alternative notions of penalty functions, regularization, kernels and so forth are used instead. Thus, discriminative classifiers feel like black-boxes where the relationships between variables is not as explicit as in generative models. Furthermore, discriminative approaches may be inefficient to train since they require simultaneous consideration of all data from all classes. On the other hand, generative models are typically more flexible than discriminative models in complex learning tasks. Moreover, most discriminative models are inherently supervised and cannot easily support unsupervised learning. However, for classification tasks that do not require the joint distribution, discriminative models can yield superior performance in part because they have fewer variables to compute. An analysis of discriminative and generative classifiers was made by Andrew et al. [96] by comparing logistic regression and naïve Bayes. From their study, it was proved that generative and discriminative models have their own advantages and disadvantages, and are complementary to each other to a certain extent. Therefore, hybrid generative-discriminative appearance models were proposed to fuse useful information from both models [97].

Since discriminative classifiers are widely used in classification and object tracking systems, this work contains a detail review of two discriminative classifiers, which are popularly applied in object tracking, including k-Nearest Neighbour (k-NN) and Support Vector Machine (SVM).

k-Nearest Neighbour (k-NN)

In classification, the *k*-NN algorithm is a simple non-parametric method that classifies an unknown example based on the consensus of its *k* closest training samples in feature

space [98]. *k*-NN is a typical lazy learning method which does not use the training data to implement generalization. Therefore, the training phase of *k*-NN is pretty fast. On the other hand, *k*-NN makes decisions based on the entire training data set which means it stores the feature vectors and labels of all training samples that are needed during the testing phase. This results in a costly testing phase, in terms of both time and memory. In the testing phase, an input instance is classified by finding its k nearest neighbour feature vectors in the feature space and do a majority voting. *k*-NN is an example-based classification approach which requires search among the training samples. The user defined constant '*k*' is usually chosen as an odd number for binary (two classes) classification to avoid tie votes. When *k*=1, it becomes the nearest neighbour algorithm, in which the unknown example is classified by directly assigning to it the label of its nearest neighbour in feature space. For different sets of training data, with respect to data size and distribution, the constant '*k*' usually need to be carefully tuned. Fig.2.20 shows examples of decision boundaries under different *k* values. It is observed that the decision boundary becomes smoother with increasing value of *k*.

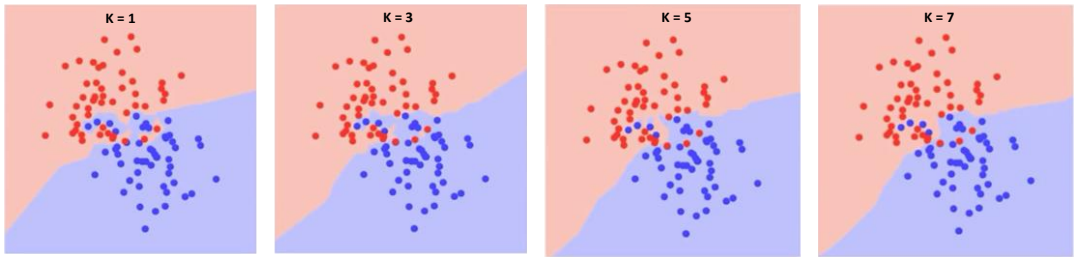


Figure 2.20 Decision boundaries with k=1,3,5,7 respectively. The decision boundaries become smoother with increasing value of k. [99]

Support Vector Machine (SVM)

SVM is another widely used supervised learning method used for classification based on statistical learning theory, which are considered as heuristic algorithms [100]. A simple SVM is a linear binary classifier. In this algorithm, each data item is plot as a point in an n-dimensional feature space (where n is the number of features used). If these feature points are linearly separable, they can thus be classified by finding a hyperplane that separates the two classes. There could be infinite hyperplanes that can separate the two classes and the target of SVM is to find the optimum hyperplane that

can maximize the distance of each class to the classification plane, as shown in Fig.2.21. The target optimum hyperplane is also referred to as maximum-margin hyperplane. The points which constrain the margin are called the support vectors. New testing examples are then mapped into the same space and labelled based on which side they fall.

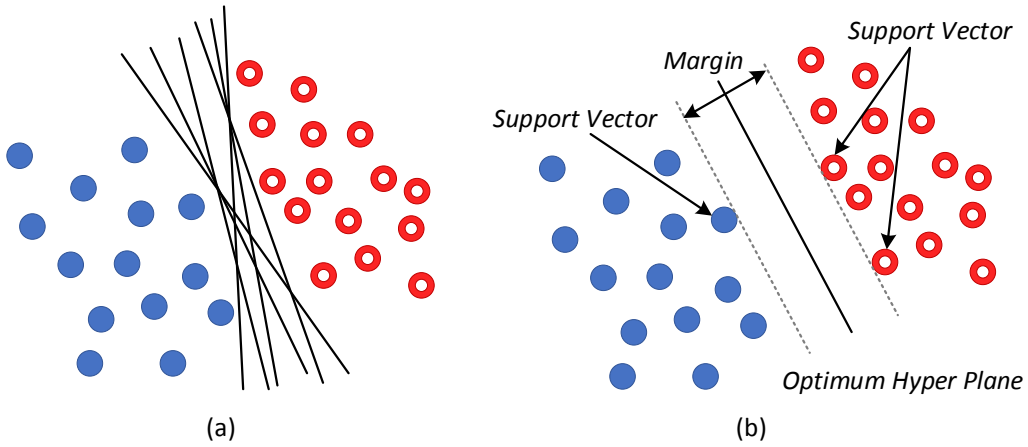


Figure 2.21 (a) Hyper planes for linearly separable data, (b) optimum hyperplane with maximum margin for an SVM trained with sampled from two classes. Samples on the margin are called support vectors.

If the input data vectors are not linearly separable, which is common in practical problems, SVM can also find the classification plane by introducing the concept of kennel. Kernels (linear, polynomial and radius kernels) enable mapping the nonlinear input data to a higher dimensional feature space in which the data vectors can be linearly separated, and thus classification issues can still be solved by a linear SVM, as illustrated in Fig.2.22. For different classification problems, the choice of kernel functions is a key element of classification accuracy [101]. Types of popular used kernels are listed as follows,

- Linear kernels, $k(x, x') = x^T x'$
- Polynomial kernels, $k(x, x') = (1 + x^T x')^d$ for and > 0 , containing all polynomial terms up to degree d.
- Gaussian kernels $k(x, x') = \exp(-\|x - x'\|^2 / 2\sigma^2)$ for $\sigma > 0$, providing a feature space with infinite dimensions. The most popular Gaussian kernel used in SVM classification is Radial Basis Function (RBF).

To find the optimum hyperplane for SVM classification is an optimization problem. In addition to kernel function, there are some other extensions to this problem including soft margins and duality solution.

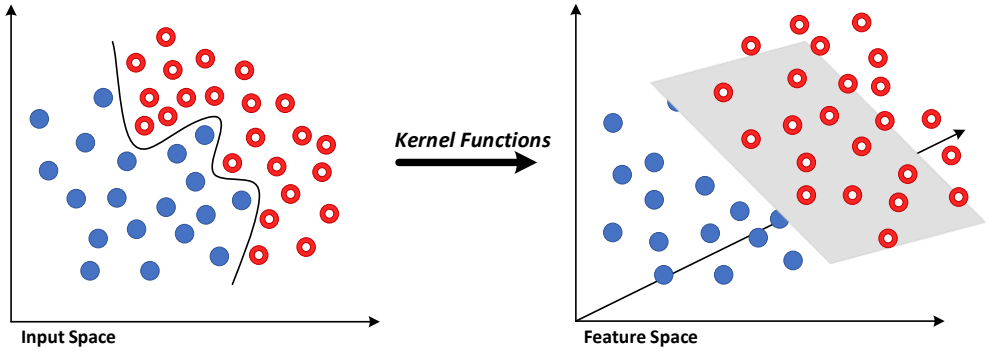


Figure 2.22 Functionality of kernel functions. Nonlinearly separable data can be mapped into linearly separable vectors in higher dimensional space by applying kernel functions.

Besides the two kinds of classifiers mentioned above, there are still many other classification approaches, such as Naive Bayes, Decision Tree (DT), and neural networks. Naive Bayes classifier is a simple probabilistic classifier based on applying Bayesian decision theory with assumptions that the value a particular feature is independent of the value of any other feature, given the class variable. The posterior probability of each class can be described by the multiplication of prior probability and class conditional probability densities divided by evidence. The classification decision is then made according to the maximum posterior probability. The DT classifier has a hierarchical structure and classifies an instance in a top-down process from root to the leaf nodes. The paths from root to leaf nodes represent classification rules. The hierarchical essence ensures that the decision tree has a good testing performance, however, suffers from the overfitting issues and expensive computational cost due to the training of large number of parameters. A neural network consists of units arranged in multi-layers, including an input layer, one or more hidden layers, and an output layer, mimicking the vast network of neurons in human brain. Each unit on one layer generates an output to the next layer by applying a function (e.g. sigmoid function) to the input instance. Generally, the networks are defined to be feed-forward such that a unit feeds its output to the units on the next layer without feedback to the previous

layers. One of the most significant parameters of neural network is the unit weightings which are assigned to signals passing from one unit to another and trained in the training phase. Tuning the weightings are always essential for different testing scenarios and thus raise the training cost.

2.2.2 Frame-based Tracking-by-Detection Algorithms

Visual tracking is one of the most challenging computer vision problems, regarding the generation of an inference about the motion of an object given a sequence of images, captured by frame-based image sensors. In general, the definition of tracking can be simplified as an analysis of video sequences with the purpose of locating the target moving object over a sequence of frames or time starting from the position of the bounding box labelled in the start frame. Over decades of research, a large variety of trackers have been developed targeting to handle the issues in complicated realistic scenarios, such as illumination change, occlusion, deformation and low contrast. Smeulders, A. W., et.al. [102] reviewed the popular classical trackers and classified the tracking methods into two types, one is matching based approaches and the other one is discriminative classification based approaches, which are also known as tracking-by-detection methods. The matching based tracking methods can be further categorised into three types: tracking using matching, tracking using matching with extended appearance model and tracking using matching with sparse optimization constraints. Tracking using matching approaches, such as Kalman appearance tracker [103] and mean shift tracking [104], perform a matching of the representation of the target model built from the previous frames and make decisions by finding the candidate with highest similarity score. Trackers using matching with extended appearance model, including incremental visual tracking [105], tracking on the affine group [106], and tracking by sampling trackers [107], are especially designed for long term tracking. The extended appearance model is built based on the target's appearance or behaviour over the previous frames. The tracking process is still to find the candidate with best matching, however, it has to search both the previous image and the extended model of appearance variations. Another extension of matching based methods is to apply sparse representation and the representative trackers of this type are tracking by Monte Carlo

sampling [108], adaptive coupled-layer tracking [109] and L1-minimization tracker [110]. The application of sparse representation enables tracking of targets for which the object appearance drastically over time. On the other hand, tracking-by-detection methods first build a classifier to distinguish target pixels (foreground) from the background pixels and then update the classifier with new sample data over time. Typical tracking-by-detection approaches include super pixel tracking [111], tracking-learning-detection [4] and struck output tracking [1]. Tracking-by-detection approaches have become particularly popular recently partially due to the great deal of progress made recently in object detection, with most ideas being transferable to tracking. Moreover, the development of these approaches which employs online training of the classifiers providing a natural mechanism for adaptive tracking. Compared with matching based trackers, tracking-by-detection approaches generally have much better performance in terms of both accuracy and time cost.

For a comprehensive performance evaluation of different trackers, it is critical to evaluate the trackers on a standard and representative dataset that covers various complicated environmental situations. Early datasets for object tracking, such as the VIVID [112], CAVIAR [113], and PETS [114] databases, contains targets which are usually humans or small cars with static background and most sequences do not provided annotated ground truth. Wu, et al. [58] first collect and build a tracking dataset with 50 fully annotated sequences in 2013 (OTB-50) to facilitate the tracking evaluation and the dataset was extended to 100 sequences in 2015 (OTB-100) [59]. The OTB datasets have become the standard to evaluate the performance of a new developed tracker. They also set up a benchmark with the evaluation results of large scale experiments on 33 representative tracking algorithms with various evaluation criteria, as shown in Table III. In addition, the benchmark also integrates most of the publicly available trackers in one code library with unified input and output formats. The setup of OTB is a milestone in the field of object tracking as it promotes the progress of research to a great extent.

TABLE III SRER EVALUATION RESULTS ON THE TB-50 DATASET [59].

	All	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
STRUCK	57.5 / 3.6	59.3 / 3.3	52.4 / 4.6	55.6 / 3.8	57.0 / 3.4	59.0 / 3.3	59.1 / 3.9	59.9 / 2.8	55.9 / 4.1	57.3 / 3.7	58.9 / 3.4	57.8 / 3.6
SCM	54.4 / 4.1	61.3 / 2.9	51.5 / 4.8	42.8 / 6.5	51.8 / 4.3	61.1 / 3.1	61.7 / 2.5	45.2 / 5.9	56.8 / 3.8	57.0 / 3.8	56.4 / 4.5	55.8 / 3.9
ASLA	53.2 / 4.1	59.2 / 3.0	50.5 / 4.5	42.0 / 6.5	52.1 / 4.1	59.6 / 3.0	59.3 / 2.3	44.6 / 5.9	56.0 / 3.8	56.3 / 3.7	55.3 / 4.3	54.0 / 3.9
CSK	52.4 / 4.9	55.9 / 4.2	48.1 / 5.7	45.8 / 5.8	52.4 / 4.7	56.2 / 4.3	55.0 / 4.8	50.6 / 5.1	52.7 / 5.1	53.3 / 4.9	53.2 / 4.9	52.0 / 5.0
L1APG	50.8 / 4.8	54.8 / 4.2	44.6 / 5.8	44.6 / 6.0	52.0 / 4.5	52.9 / 4.7	59.5 / 3.2	49.2 / 5.2	50.9 / 4.8	50.9 / 5.0	55.5 / 4.4	51.0 / 4.7
OAB	50.3 / 5.3	50.1 / 5.1	46.0 / 6.0	47.4 / 5.7	51.0 / 4.9	48.3 / 5.8	56.2 / 4.5	49.9 / 5.1	49.2 / 5.6	49.2 / 5.6	50.0 / 5.6	50.8 / 5.1
VTD	49.3 / 5.2	55.1 / 4.2	46.2 / 5.5	41.7 / 6.8	50.2 / 4.6	53.7 / 4.6	47.1 / 5.6	43.5 / 6.3	52.3 / 4.9	53.7 / 4.6	51.5 / 5.4	48.9 / 5.4
VTS	49.1 / 5.1	54.7 / 4.2	46.3 / 5.5	40.6 / 6.8	50.0 / 4.5	53.7 / 4.4	47.1 / 5.6	42.6 / 6.2	52.2 / 4.8	53.4 / 4.5	51.2 / 5.3	48.8 / 5.3
DFT	49.0 / 5.6	53.2 / 4.7	48.1 / 5.5	41.7 / 6.3	50.7 / 5.1	53.0 / 4.7	47.7 / 6.3	46.7 / 5.4	52.7 / 5.1	53.2 / 5.0	55.2 / 4.4	47.9 / 5.9
LSK	48.6 / 5.0	53.4 / 4.1	45.1 / 5.5	38.9 / 6.7	48.1 / 4.8	50.2 / 4.8	58.5 / 3.2	39.8 / 6.5	50.6 / 4.8	50.0 / 4.8	47.5 / 5.5	49.8 / 4.8
RS	48.4 / 5.6	48.6 / 5.1	50.5 / 5.3	46.7 / 6.3	47.0 / 5.8	47.6 / 5.4	44.5 / 5.8	46.4 / 6.0	52.2 / 5.0	51.9 / 4.9	53.2 / 5.0	47.9 / 5.7
MTT	48.3 / 5.2	50.7 / 4.9	40.1 / 6.6	40.6 / 6.8	51.5 / 4.4	50.2 / 5.0	57.8 / 3.6	45.5 / 6.2	47.3 / 5.5	48.2 / 5.3	50.4 / 5.6	48.2 / 5.3
LSHT	48.3 / 5.3	52.8 / 4.4	46.3 / 5.6	40.6 / 6.4	48.0 / 4.9	52.3 / 4.5	49.0 / 5.6	42.9 / 5.6	52.1 / 4.9	53.7 / 4.5	53.4 / 4.2	48.6 / 5.4
CXT	48.2 / 5.4	49.4 / 5.5	37.0 / 7.2	46.2 / 5.7	52.1 / 4.4	48.5 / 5.4	53.7 / 5.0	52.0 / 4.7	45.7 / 6.0	47.2 / 5.7	51.0 / 5.3	48.7 / 5.5
LSS	47.6 / 5.6	52.8 / 4.7	42.7 / 6.3	39.5 / 6.6	47.1 / 5.5	51.4 / 5.2	56.2 / 4.1	42.7 / 6.1	51.2 / 5.1	50.2 / 5.4	55.2 / 4.6	48.5 / 5.3
TLD	46.8 / 5.5	48.3 / 5.4	37.4 / 7.2	44.6 / 5.5	48.9 / 4.9	46.7 / 5.6	53.3 / 4.9	51.0 / 4.5	45.2 / 5.9	46.0 / 5.7	50.2 / 5.0	47.1 / 5.6
TM	46.7 / 6.0	49.4 / 5.4	40.2 / 6.6	44.8 / 6.2	47.2 / 5.7	45.7 / 6.2	55.6 / 4.8	48.1 / 5.5	46.8 / 5.8	46.3 / 6.1	52.6 / 5.1	47.6 / 5.8
PD	46.6 / 5.8	45.1 / 6.1	46.4 / 6.0	47.6 / 5.6	48.5 / 5.2	45.1 / 6.0	43.2 / 6.1	50.6 / 4.7	49.2 / 5.6	48.9 / 5.6	52.4 / 5.2	46.3 / 5.9
IVT	46.4 / 5.5	51.6 / 4.6	40.5 / 6.4	37.3 / 6.9	46.4 / 5.3	51.2 / 4.8	55.8 / 3.7	41.3 / 6.2	49.3 / 5.1	49.0 / 5.3	52.3 / 4.9	47.1 / 5.3
VR	45.9 / 6.0	44.8 / 6.2	45.3 / 6.2	47.1 / 5.9	47.9 / 5.5	43.7 / 6.4	42.4 / 6.0	49.7 / 5.3	48.1 / 5.9	48.2 / 5.9	50.8 / 5.5	45.8 / 6.2
MIL	45.9 / 6.1	48.6 / 5.3	45.7 / 5.9	44.1 / 6.5	45.7 / 5.9	47.1 / 5.6	43.5 / 6.8	43.7 / 6.3	47.6 / 5.8	48.9 / 5.7	52.7 / 5.1	44.5 / 6.5
LOT	45.5 / 5.9	48.6 / 5.2	42.4 / 6.5	47.1 / 5.5	43.2 / 6.3	45.4 / 5.8	38.1 / 6.1	46.9 / 5.3	49.5 / 5.2	48.5 / 5.5	49.2 / 4.8	45.6 / 5.8
FOT	44.3 / 6.6	51.6 / 5.3	38.8 / 7.2	40.8 / 6.7	46.1 / 5.8	50.8 / 5.6	42.1 / 8.0	44.9 / 6.2	44.7 / 6.6	47.0 / 6.1	49.0 / 5.5	43.8 / 6.7
FRAG	44.2 / 6.6	46.1 / 5.9	41.8 / 6.8	44.8 / 6.4	43.3 / 6.7	42.6 / 6.6	42.6 / 7.3	46.1 / 6.1	46.6 / 6.2	46.1 / 6.4	50.1 / 5.5	44.2 / 6.8
PCOM	44.0 / 6.0	49.2 / 5.0	39.0 / 6.4	37.6 / 6.8	43.7 / 5.9	47.0 / 5.8	49.4 / 5.3	41.7 / 6.3	47.4 / 5.5	46.3 / 5.9	50.4 / 5.2	44.8 / 5.7
KMS	43.8 / 6.6	44.1 / 6.2	44.8 / 6.4	44.6 / 6.3	42.7 / 6.8	41.5 / 6.8	39.4 / 7.3	43.9 / 6.1	46.6 / 6.2	45.8 / 6.5	45.8 / 6.2	44.0 / 6.6
CPF	43.5 / 6.6	42.8 / 6.4	44.2 / 6.8	42.9 / 6.9	43.1 / 6.4	41.0 / 6.9	43.6 / 5.3	39.8 / 7.1	47.9 / 5.8	47.6 / 5.9	46.4 / 6.0	44.1 / 6.6
ORIA	42.4 / 6.4	47.6 / 5.9	34.8 / 7.8	32.8 / 7.6	44.9 / 5.5	46.9 / 5.9	50.8 / 4.4	36.8 / 7.3	44.4 / 6.3	44.9 / 6.1	46.9 / 6.2	43.4 / 6.3
CT	40.4 / 6.6	43.2 / 5.9	39.4 / 6.8	35.7 / 7.5	42.8 / 6.0	42.5 / 6.2	40.0 / 6.8	37.1 / 7.2	43.7 / 6.4	44.9 / 6.1	47.3 / 5.9	40.2 / 6.8
SBT	37.3 / 7.4	37.3 / 7.4	28.0 / 8.5	36.7 / 7.3	35.5 / 7.5	34.7 / 7.7	48.0 / 6.3	38.3 / 6.8	36.8 / 7.4	35.8 / 7.7	41.0 / 7.2	38.7 / 7.2
MS	35.6 / 7.9	36.7 / 7.4	32.8 / 8.0	40.5 / 7.1	36.8 / 7.8	34.6 / 7.8	28.4 / 9.2	41.2 / 6.8	37.4 / 7.7	37.3 / 7.9	41.0 / 6.9	36.0 / 7.9
BSBT	31.4 / 7.8	30.8 / 7.8	20.6 / 8.7	30.8 / 7.6	31.2 / 7.6	28.9 / 8.0	42.4 / 6.8	32.3 / 7.4	32.0 / 7.7	30.4 / 7.9	36.3 / 7.2	32.5 / 7.7
SMS	29.0 / 8.2	24.2 / 8.4	29.7 / 8.5	31.6 / 7.7	30.4 / 8.1	26.7 / 8.3	31.7 / 8.0	33.0 / 7.2	33.3 / 7.9	31.5 / 8.2	31.0 / 8.2	29.9 / 8.3

Each entry contains the average overlap in the percentage and the average number of failures in 1,000 frames at the overlap threshold of 0.5. The trackers are ordered by the average overlap scores, and the top five methods in each attribute are denoted by different colours: red, green, blue, cyan, and magenta.

The Visual Object Tracking (VOT) challenge is another platform for performance evaluation of visual object trackers [14]. It is initiated in 2013 to address performance evaluation of short-term trackers. From then on, five challenges have taken place with one challenge conducted per year and all challenges consider single-camera, single-target, model-free, causal trackers, applied to short-term tracking. The VOT challenge contains established datasets, evaluation measures and toolkits. It also creates a public platform for discussions on tracking issues. A large number of trackers have actively participated in the VOT challenge, such as there are 51 trackers (containing 38 entries and 13 baseline trackers) tested on the VOT2017 challenge, and most of the trackers keep to the state of the art. From 2015 onwards, trackers build on correlation filter and CNN gradually capture more and more attentions of researchers.

It is worth mentioning that a new benchmark, namely NFS benchmark, for tracking on high framerate image sequences was built in 2017 [151]. The dataset consists of 100 videos in total captured with the currently wide available higher frame rate (240fps)

cameras from real world scenarios. The benchmark extensively evaluated the most recent and state-of-the-art trackers based on correlation filter and deep learning on sequences with both high framerate (240fps) and low framerate (30fps). The trackers are ranked according to their tracking accuracy and real-time performance. With the experimental results, Gallogahi et al. [151] have drawn a conclusion that tracking accuracy on higher framerate sequences has a significant improvement compared with that on lower framerate sequences but in the expense of higher computational cost. They also concluded that at higher framerates, simple trackers outperform complex methods based on deep networks. For the research of event-based tracking, the NFS benchmark is significant and valuable. Compared to the event-based tracking benchmark [120] simulated by capturing 30fps sequences using DVS, event-based simulations on high framerate sequences are more closer to the observation of real world scenarios by DVS.

Tracking evaluation measures

Many tracking evaluation measures have been proposed typically by comparing the tracking results with the ground truth, considering the target presence and position. Smeulders, A. W., et.al. [102] summarized the most common measures used in single target tracking as summarised in Table IV.

TABLE IV OVERVIEW CHARACTERISTICS OF THE EVALUATION METRICS [116]

Name	**Equation**	**Aim**	**Measure**
F-score [108]	$2 \cdot \frac{precision \cdot recall}{precision + recall}$	Accuracy	Threshold precision and recall
*F*1-score [115]	$\frac{1}{N_{frames}} \sum_i 2 \cdot \frac{p^i \cdot r^i}{p^i + r^i}$	Accuracy	Precision and recall
OTA [116]	$1 - \frac{\sum_i (n_{fn}^i + n_{fp}^i)}{\sum_i g^i}$	Accuracy	False positive and false negative
OTP [117]	$\frac{1}{\lvert M_S \rvert} \sum_{i \in M_S} \frac{\lvert T^i \cap GT^i \rvert}{\lvert T^i \cup GT^i \rvert}$	Accuracy	Average overlap over matched frames
ATA [115]	$\frac{1}{N_{frames}} \sum_i \frac{\lvert T^i \cap GT^i \rvert}{\lvert T^i \cup GT^i \rvert}$	Accuracy	Average overlap
Deviation [118]	$1 - \frac{\sum_{i \in M_S} d(T^i, GT^i)}{\lvert M_S \rvert}$	Location	Centroid normalized distance
PBM [115]	$\frac{1}{N_{frames}} \sum_i [1 - \frac{Distance(i)}{T_h(i)}]$	Location	Centroid L1-distance

It is notably that OTB and VOT have created their own measurement system. For OTB, the evaluation methodology contains an accuracy evaluation consisting of the precision and success rate for quantitative analysis and a robustness evaluation consists of temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). Precision plot or centre location error plot, one the most widely used evaluation metric on tracking precision, is defined as the average Euclidean distance between the centre locations of the tracked targets and the manually labelled ground truths. A threshold of pixel error is commonly given such that the precision plot reflects the percentage of frames whose centre location error are within the given threshold. A typical threshold is 20 pixels [119]. Success plot is another commonly used evaluation metric, measuring the overlap score of a tracked bounding box and the ground-truth bounding box. A threshold of overlap score is predefined such that the success plot reflects the percentage of frames whose overlap score are larger than the defined threshold. To evaluate the robustness of a tracker, TRE and SRE measures were proposed in OTB. Unlike one-pass evaluation, TRE and SRE evaluate each tracker numerous times by starting from different frames in temporal domain or spatially from different bonding boxes in the start frame. For each type, the average of all evaluation tests is taken as the final TRE/SRE score. The accuracy measurement methodology adopted in VOT is the same as that of OTB, which contains both centre location error and bounding box overlap measurement. However, the robustness measurement is different from TRE and SRE. During the evaluation test of a tracker, whenever there is a zero overlap between a tracked bonding box and the ground-truth bonding box, it is recorded as a failure and the tracker is reinitialized five frames after the failure. The total number of failures is summarised as the robustness score of a tracker.

2.2.3 Event-based Object Tracking Methods

With the development of DVS, several object tacking approaches based on only event driven image sensor or combined event- and frame-based data acquisition mechanisms have been developed. Benosman, R. et.al. [25] proposed a computationally efficient and robust pattern-based tracking method using ATIS, the asynchronous time-based image sensor which outputs asynchronous events with delayed event intensity. The

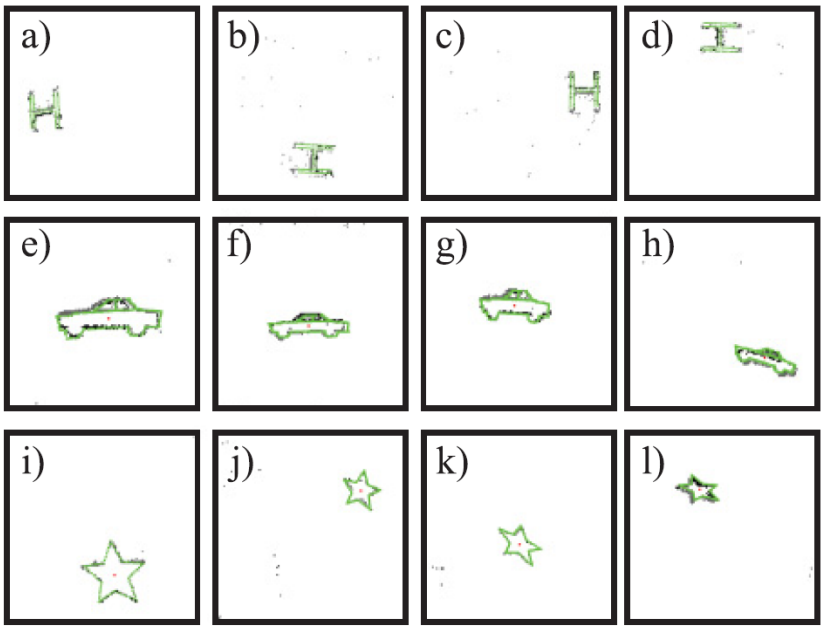


Figure 2.23 Indoor experiments of tracking the patterns on a rotating white plate. Affine transformations including the rotation, translation, and scaling in the x, y-dimensions are applied and estimated simultaneously as shown in the second, third and fourth columns.[25]

primary target of pattern-based tracking is to establish the correspondence between the target and a known model. Since the event-driven image sensor reduces the data redundancy by filtering out the static background and directly extracts motion information, it solves the limitations of framerate using conventional frame-based sensors in real-time applications. The proposed algorithm is designed as an event-adaptive pattern-based tracking method embodied in three main aspects, pattern representation, transformation model and the tracking criterion for estimation of the optimal transformation parameters. The algorithm processes each output event once received and iteratively updates the model location and orientation based on the arrival of events. The pattern model is set up based on a cloud of events (2D point sets), taking use of the position and intensity information. For each arrived event, spatial matching is firstly conducted based on the spatial-temporal characteristic of the event followed by a geometric transformation for model updating. The algorithm potentially solves the ambiguous cases of object occlusions that are poorly handled by frame-based cameras. The group conducted various indoor and outdoor experiments and Fig.2.23 shows some experimental results with respect to indoor shape tracking with affine tracking. The experiment results indicate that the algorithm allows shape tracking at an equivalent

framerate of 200kHz and validate the application of event-based cameras for machine vision.

Another application of object tracking based on event-driven camera is the feature detection and tracking using DAVIS developed by David, T. et.al. [26]. The DAVIS combines a standard frame-based camera with an asynchronous event-based sensor and outputs both asynchronous binary on/off events and gray-level full frames on demand by synchronous control of all pixels. The proposed algorithm is the first approach to detect and track features taking use of both frame and event outputs of DAVIS. The feature tracking process contains two parts, firstly, features are detected in the grayscale frames and then tracked from the event stream. The detected features in this algorithm include edges and corners because the characteristics of these features are invariable as the DAVIS pixels responses to the illumination variation. The tracking process of this algorithm is similar to the shape tracking mentioned above in [25], which build models based on the detected features and the feature models are continuously updated with the newest arriving event. Fig.2.24 shows a grayscale image of *Lucky Luke* dataset captured by DAVIS for initialization, the centre pointes of the detected edge or corner features by Harris detector and Canny edge detector and the space-time view of the feature tracking results as well.

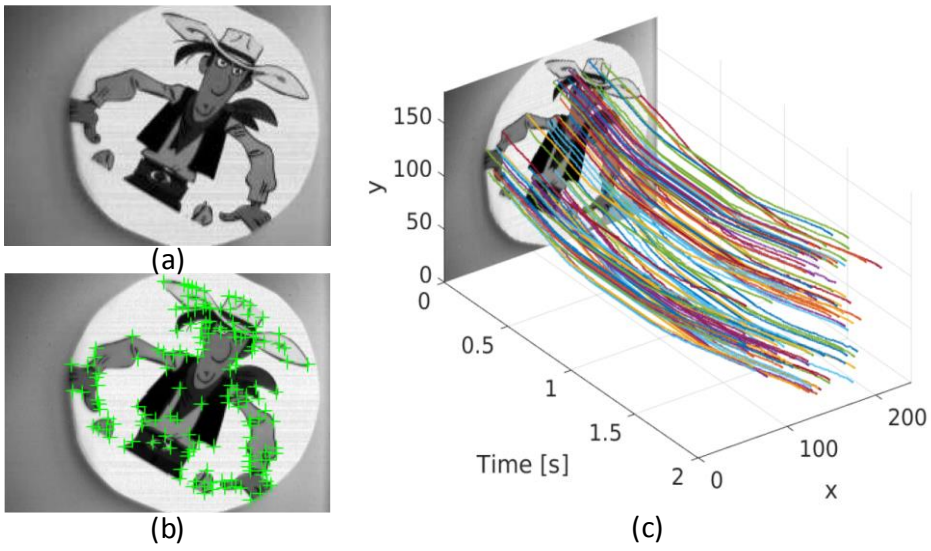


Figure 2.24 Feature detection and tracking on Lucky Luke dataset. (a) DAVIS frame for initialization, (b) frame with centres of detected features (green crosses), (c) space-time view in the image plane of the tracked features' trajectories. [26]

The hybrid image sensor, DAVIS, was employed by Liu, et.al. [140] who proposed that the DVS events could be used to guide the application of Convolutional Neural Network (CNN) technology to DAVIS for object detection and tracking. The tracking process contains three steps: firstly, the regions of interest (ROIs) are generated by a cluster-based tracking method guided by the binary events of DAVIS, then the roughly target locations are roughly detected by CNN based on the grayscale frames captured by DAVIS and finally a particle filter is used to infer the target location from the ROIs. The experimental result shows that by combining event- and frame-based cameras, the speed is estimated to be boosted by a factor of 70 compared with a full-frame CNN-based tracking. The measurement of tracking performance of the above-mentioned methods all employed the centre location error of accuracy measurement which is subpixel accuracy measurement compatible with the output pixel events of DVS. The same group also setup a DVS benchmark datasets for object tracking, action recognition and object recognition by capturing the public image sequences using DAVIS [120]. The primary target of the benchmark is to help the development of algorithms processing event-based vision input, allowing for a comparison of different approaches with a uniform test standard. However, there is a significant deficiency in data acquisition since the event streams in this dataset is generated by capturing image sequences displayed on a monitor. The image sequences chosen for event stream generation are typically recorded at 30fps. As a result, the functionality of DVS that can continuously record and output data in temporal domain is highly limited.

2.2.4 Advantages and Limitations of Event-based Tracking Methods

Based on the understanding of the functionality of event-based cameras and the review of both frame-based and event-based tracking methods, there are some advantages of event-based tracking methods over the conventional frame-based tracking approaches. One significant advantage is the tremendous reduction of computational cost such that the data processing speed of event-driven methods can achieve dozens of times higher than frame-based methods. Therefore, event-based data acquisition provides a potential solution to the applications highly require real-time performance like navigation of autonomous vehicles. Another advantage lies on the complementary information

contained in dynamic DVS events. The conventional frame-based cameras are limited by framerate and they do not provide any information in the blind time between subsequent frames. On the contrary, the DVSs are able to capture the full motion trajectory in nature, since the DVSs can continuously detect and report motion in temporal domain.

On the other hand, the research and development of frame-based tracking methods are much more mature than event-based tracking mainly because that most of the currently event-based vision sensors are only available as R&D prototypes. Another limitation lies in the asynchronous acquisition of binary events and intensity information. The experimental results have implied the significance of intensity for machine vision, therefore, using a single traditional DVS with binary events barely finds its place in the field of complicated vision tasks, such as object tracking and optical flow. Even though the issue is mitigated with the development of DAVIS, output binary events and grayscale full frames, and ATIS, output grayscale events with a relatively large latency due to light integration, there is an essential to develop a new event-based vision sensor which can integrate the event- and frame-based sensors in a better format such that it can generate both grayscale frames on demand and grayscale events with low or without latency.

2.3 Optical Flow Algorithms

2.3.1 Fundamentals

The optical flow technique has been developed for over two decades and is crucial to autonomous applications, e.g. tracking moving objects, autonomous navigation and obstacle avoidance. State-of-the-art optical flow calculations are robust for a variety of applications, yet still far from being fully satisfactory, as they are not reliable in flow field real-time estimation for fast moving objects. Optical flow implementations can be classified into two main types in term of image acquisition method by employing frame-based or event-based image sensors. Traditional optical flow calculations are frame-based which have several main drawbacks as large data redundancy, high power consumption and time-consuming software computation, which are not adapted for real-time applications such as mobile autonomous robots and vehicles. The frame-based optical flow computation is limited by the framerate of the used image sensor, which is usually less than 60 Hz. To develop real-time optical flow methods, event-based methods have been developed in recent years by replacing the frame-based image sensor with event-based image sensors [27-29].

Figure 2.25 Example applications based on optical flow estimation technology.

2.3.1.1 Optical Flow Concepts

Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image. Optical flow can rise from relative motion between an observer (an eyeball or a camera) and the scene. The main target of optical flow is to compute an approximation of 2-D motion field from spatiotemporal patterns of image intensity.

A 2-D motion field is a projection of 3-D velocities of surface points onto the imaging surface. The calculated optical flow field contain two dimensional vectors that indicating motion direction and speed. Fig.2.26 shows the optical flow field of a rotating object which is calculated between the subsequent two frames. Variation of optical flow corresponding to different objects helps with the image segmentation. Optical flow may also be used to perform motion detection, obstacle avoidance, tracking objects and navigation of autonomous robots and vehicles.

Figure 2.26 An example optical flow field in vector expression of a rotating object.

The concept of optical flow has been described with respect to different properties from literature. Helmholtz described that "My belief too is that it is mainly by variations of the retinal image due to bodily movements that one-eyed persons are able to form correct apperceptions of the material shapes of their surroundings." [121] He mainly concerned about depth perception and described optical flow as a result of the combination of body motion, distance and rigidness of the environment. On the other hand, Gibson expressed his view about optical flow in 1966 that "analytically, this total transformation of the array appears to mean that the elements of this texture are displaced, the elements being considered as spots. Introspectively, the field is everywhere alive with motion when the observer moves." [122] In his point of view, the optical flow is considered as a transformation of the optic structure displacement caused by the motion of observer. A more recent definition of optical flow is from Horn that "The apparent motion of brightness patterns observed when a camera is moving relative to the objects being imaged is called optical flow." This definition attributes the "motion of brightness patterns" in an image to the relative change between observer

and scene, which becomes the foundation of the perception and extensive development of optical flow.

2.3.1.2 Optical Flow Sensors

An optical flow sensor is a vision sensor that is capable to measure the observed motion through optical flow computation and output a feature measurement based on the optical flow. Optical flow image sensor has various configurations, of which one configuration is to connect an image sensor to a separate programmable processor to implement optical flow algorithms as shown in Fig.2.27 (a). Another configuration is to integrate an image sensor and the processor onto a single vision chip, allowing for compactable implementation, e.g. PX4FLOW smart optical flow camera shown in Fig.2.27 (b). Optical flow sensors are also used in computer optical mice as the main sensing component for the mouse motion measurement across a surface. Other applications that primarily have the requirement to detect object motion or relative motion between objects such as unmanned aerial vehicles (UAVs) and collision-avoidance system provide a market for the optical flow sensors [123]. In recent years, research in an optical flow sensor starts to employ neuromorphic engineering techniques to implement circuits responding to optical flow [124].

(a) (b)

Figure 2.27 Examples of optical flow sensors. (a) APM optical flow sensor v10 [125], the image sensor and processing unit are separable, (b) PX4FLOW smart optical flow camera [126] which integrate the image sensor and programming processor onto a single vision chip.

2.3.1.3 Restricted Problems

Aperture problem

Aperture problem refers to the fact when optical flow is not corresponding to the motion field. A typical example of aperture problem is the rotating barber's pole as shown in Fig.2.28. When the pole is rotating in a counter-clockwise direction from the top view, the motion direction of the patterns on the pole is actually pointing to the right while the observed optical flow field has a direction pointing upwards. The analysis of such problem is illustrated in Fig.2.29. When a triangle is moving upwards and only one

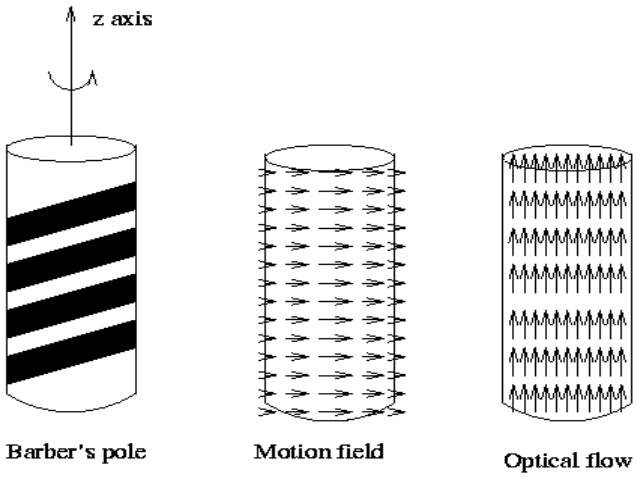


Figure 2.28 Motion field and optical flow of a barber's pole [127].

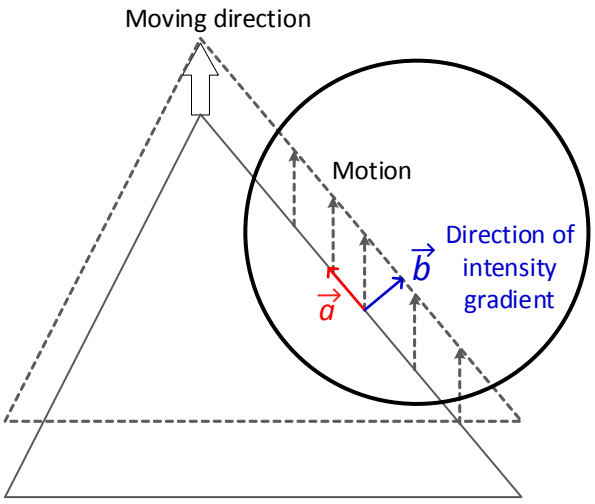


Figure 2.29 Illustration of the aperture problem.

edge is observed by an aperture, the motion vector can be decomposed into two vectors $\vec{a}$ and $\vec{b}$, correspondingly referring to the direction perpendicular to the intensity gradient and the direction parallel to the intensity gradient. Since the vector $\vec{a}$ provides zero gradient which cannot be detected, only the component vector $\vec{b}$ can be measured.

Other restrictions (Shading effects)

Apart from the aperture problem, optical flow cannot be used to perceive motion due to shading effects. Optical flow field is derived through the projection of the 3-D motion. Conversely to the optical flow derivation, calculated optical flow field is used to perceive motion. However, a moving object may give rise to a pattern with constant brightness. For example, rotation of a uniform sphere which exhibits a fixed shading pattern under stationary light injection give rise to zero optical flow at all the points in the image, since light intensity detected by each pixel doesn't change. Another problem is the occluding edges of objects, because discontinuities of optical flow occur at object boundaries. For convenience, a particularly simple visual scene is analyzed where the apparent velocity of brightness patterns can be directly identified [15]. To avoid the discontinuities of brightness caused by shading effects, it is always initially assumed that the surface being imaged is flat and the incident illumination is uniform across the surface.

2.3.1.4 Main Constraints for Optical Flow Estimation

Brightness constancy constraint

Under most circumstances, the surfaces persist and hence the apparent brightness of a small region on the moving objects remains constant over a relatively short time interval. Consider a patch of the brightness pattern that is displaced a distance Δx in x-direction and Δy in y-direction during a short time period Δt, as shown in Fig.2.30. The brightness of the patch is assumed to be conserved such that

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{2-2}$$

By applying Taylor expansion to the right side of Eq. (2-2), we get

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \varepsilon \tag{2-3}$$

where ε contains second and higher order terms in Δx, Δy and Δt. By subtracting $I(x, y, t)$ from both sides of Eq. (2-3) and dividing through by Δt we have,

$$\frac{\delta x}{\delta t}\frac{\partial I}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} + \epsilon(\delta t) = 0 \tag{2-4}$$

where $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are partial derivatives of image brightness with respect to x, y and t, respectively. If we let

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t} \; and \; u = \frac{\delta x}{\delta t}, v = \frac{\delta y}{\delta t} \tag{2-5}$$

Then it is easy to derive a single linear equation with two unknown variables u and v,

$$I_x u + I_y v + I_t = 0 \tag{2-6}$$

The Eq. (2-6) can be alternatively expressed in vector domain so that

$$\left(I_x, I_y\right) \cdot (u, v) = -I_t \tag{2-7}$$

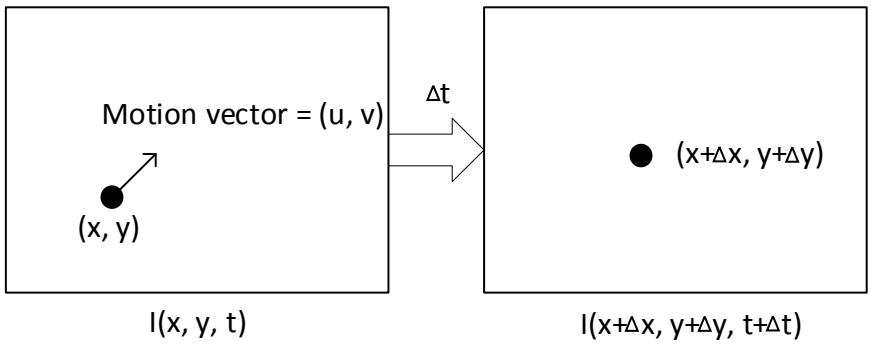


Figure 2.30 Illustration of brightness constancy constraint.

Smoothness constraint or spatial coherence

The smoothness constraint comes from the observation that neighboring points in the visual scene typically belong to the same surface and hence tend to perform similar motion. As shown in Fig.2.30, since they are projected to the nearby points in the image, it is expected that pixels in a small window, usually 3×3 or 5×5, have same motion in the image flow computation, i.e. spatial coherence.

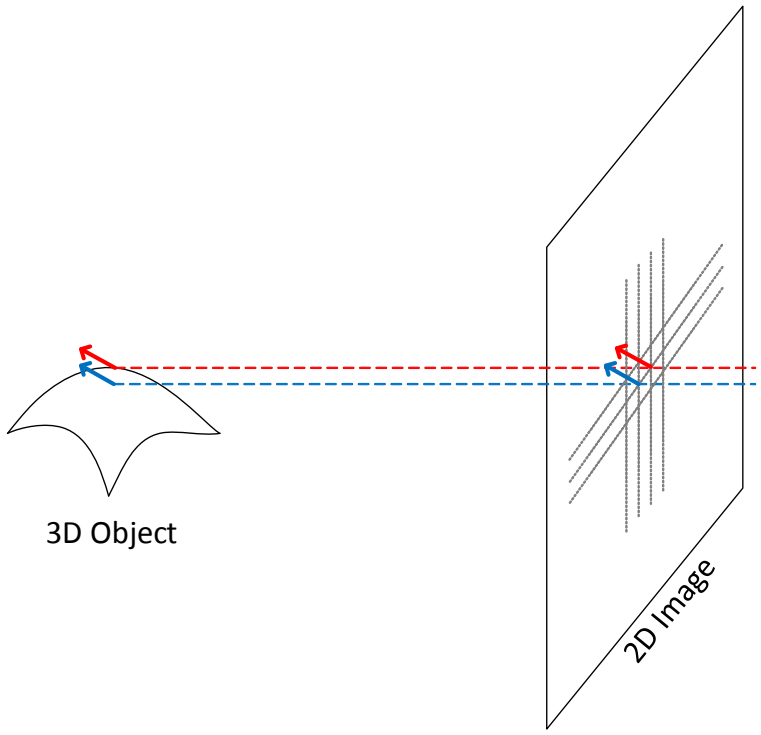


Figure 2.31 Smoothness constraint, based on the fact that neighbouring two points in the visual scene are projected to the nearby two pixels in the image.

2.3.2 Frame-based Optical Flow Algorithms

Traditionally, optical flow is calculated between consecutive image frames. Existing optical flow techniques are originated from two main concepts, one is the energy minimization framework introduced by Horn and Schunck in 1980 and the other one is the coarse-to-fine image warping proposed by Lucas and Kanade in 1981 to handle large displacements. Since the results of the original methods are significantly influenced by the outliers, statistical techniques are used to extend the concepts and achieve reliable results even in the situations when occlusions or motion discontinuities occur [128, 129]. Barron, Fleet and Beauchemin implemented and compared different optical flow methods, including gradient or differential methods, region-based matching, energy-based and phase-based techniques [17]. Sun et al. suggested that little has changed in the typical formulation of Horn and Schunck method in the domain of differential computation [18]. Some typical methods of each algorithm implementation are described as following.

2.3.2.1 Differential Methods

Differential methods are the most widely used techniques for optical flow computation using image sequences. Differential methods can be classified into local methods and global methods. The representative method of local methods is the Lucas-Kanade technique, which proposed a coarse-to-fine warping scheme and focus on the optical flow computation of small motion. The typical global methods include Horn-Schunck technique with first order derivative smoothness and other typical methods like Nagel method and Uras method which conduct global smoothness using higher order derivatives. The global techniques yield dense optical flow while the local techniques are more robust under noise. The details of the typical methods are summarized in this part.

Horn and Schunck [15]

Horn and Schunck method was the first proposed variational method of optical flow computation which combines the brightness constancy constraint with a global smoothness term to constrain the estimated velocity. Commonly observed objects of finite size in the visual scene undergo rigid motion or deformation. It is reasonably assumed that neighbor pixels on the objects perform similar velocities and the relative velocity field varies smoothly (spatial coherence). Discontinuities in optical flow is expected where object occlusion occurs. Horn and Schunck method express the second constraint by minimizing the square of the magnitude of the gradient of the optical flow velocity

$$||\nabla u||_2^2 = \left(\frac{\partial \boldsymbol{u}}{\partial x}\right)^2 + \left(\frac{\partial \boldsymbol{u}}{\partial y}\right)^2 \text{ and } ||\nabla v||_2^2 = \left(\frac{\partial \boldsymbol{v}}{\partial x}\right)^2 + \left(\frac{\partial \boldsymbol{v}}{\partial y}\right)^2$$

The main target is to minimize the sum of the errors in the brightness constancy constraint equation and the measure of the departure from smoothness in the velocity flow.

$$\int_D (\nabla I \cdot \boldsymbol{v} + I_t)^2 + \lambda^2 \left(\left|\left|\nabla u\right|\right|_2^2 + \left|\left|\nabla v\right|\right|_2^2 \right) dx \qquad (2\text{-}8)$$

defined over domain D, where the velocity vector $\boldsymbol{v}(x, y, t) = (u(x, y, t), v(x, y, t))$, the magnitude of λ reflects the influence of the smoothness term which varies for different test cases. Iterative solution is used to solve the problem.

$$u^{k+1} = \bar{u}^k - \frac{I_x[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{a^2 + {I_x}^2 + {I_y}^2}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{a^2 + {I_x}^2 + {I_y}^2} \quad (2\text{-}9)$$

Where *k* denotes the iteration number, $u^0$and v^0 are the initial velocity value which are usually set to zero and $\bar{u}^k$, $\bar{v}^k$ denote the neighborhood average value of u^k,v^k. Notice that the new estimates of a particular point do not depend directly on the previous estimate at the same point. Intensity derivatives was originally estimated by using first-order derivatives [15], and then improved through pre-smoothing methods and applicable kernels [17].

Lucas and Kanade [16]

Lucas and Kanade method focuses on the computation of small motions and is the most commonly applied method which implements a weighted least-square fit of local first-order constraints to a constant model for $\boldsymbol{v}$ in each small spatial neighborhood Ω by minimizing

$$\sum_{x\in\Omega} W^2(q)[\nabla I(q,t)\cdot v + I_t(q,t)]^2 \quad (2\text{-}10)$$

where W denotes a window function that contributes more influence on constraints at the center than those at periphery. The window size is often set to 3×3 or 5×5. The solution to Eq. (2-10) is given by

$$A^T W^2 A v = A^T W^2 b \quad (2\text{-}11)$$

where for n points $q_i \in \Omega$ at time point t,

$$A = [\nabla I(q_1), \nabla I(q_2), \dots, \nabla I(q_n)]^T$$

$$W = diag[W(q_1), W(q_2), \dots, W(q_n)]$$

$$b = -(I_t(q_1), I_t(q_2), \dots, I_t(q_n))^T$$

The solution to Eq. (2-11) is $v = [A^T W^2 A]^{-1} A^T W^2 b$, which is solved in closed form when $A^T W^2 A$ is nonsingular, since it is a 2×2 matrix:

$$A^T W^2 A = \begin{bmatrix} \sum W^2(q) I_x^2(q) & \sum W^2(q) I_x(q) I_y(q) \\ \sum W^2(q) I_x(q) I_y(q) & \sum W^2(q) I_y^2(q) \end{bmatrix} \qquad (2\text{-}12)$$

where all sums are taken over points in the neighborhood Ω with the smoothness constraint applied.

Nagel [130]

Nagel was one of the first to use second order derivatives to measure optical flow. Similar to Horn and Schunck method, Nagel used a global smoothness constraint by integrating basic measurements. As an alternative to Eq. (2-8), Nagel suggested an oriented-smoothness constraint in which smoothness is not imposed across steep intensity gradients in an attempt to handle occlusions. The problem is formulated as the minimization of the function

$$\iint (\nabla I^T \mathrm{v} + I_t)^2 + \frac{\alpha^2}{||\nabla I||_2^2 + 2\delta} \begin{bmatrix} \left(u_x I_y - u_y I_x\right)^2 \\ +\left(v_x I_y - v_y I_x\right)^2 \\ +\delta\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) \end{bmatrix} dxdy \qquad (2\text{-}13)$$

Same as the Horn and Schunck method, Nagel method employed Gauss-Seidel iterations to solve this Eq. (2-13).

Uras, Girosi, Verri and Torre [131]

The other 2nd-order techniques considered here are based on a local solution to Eq. (2-14). Following Uras et al. [131], Eq. (2-14) may be solved for v wherever the Hessian H of $I(q,t)$ is nonsingular. In practice, for robustness, they divide the image into 8x8 pixel regions. From each region, 8 estimates are selected that best satisfy the constraint $||\mathrm{M}\nabla\mathrm{I}|| \ll ||\nabla I_t||$, where $\mathrm{M} \equiv (\nabla \mathrm{v})^T$. Of these they choose the estimate with the smallest condition number $k(H)$ of the Hessian as the velocity for the entire 8×8 region.

$$\begin{bmatrix} I_{xx}(q,t) & I_{yx}(q,t) \\ I_{xy}(q,t) & I_{yy}(q,t) \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + \begin{pmatrix} I_{tx}(q,t) \\ I_{ty}(q,t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad (2\text{-}14)$$

Eq. (2-14) is the second-order derivatives (the Hessian of I) to constrain 2-d velocity. It is noticed that if the aperture problem prevails in a local neighborhood (i.e. if intensity is effectively one-dimensional), then the 2nd-order derivatives cannot usually be measured accurately enough to determine the tangential component of $\boldsymbol{v}$ because of the sensitivity of numerical differentiation. As a consequence, velocity estimation from 2nd-order methods are often assumed to be sparser and less accurate than estimation from 1st-order methods.

In addition to the above mentioned differential methods with local or global smoothness, it has been found that the optical flow field computed using gradient method is restricted by the intensity structure and the displacement of motion. The optical flow result is better for image patches with higher texture and the typical operating range is motion of one pixel based on the fact that linearization of brightness is only suitable of small displacements. Moreover, temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity. As shown in Fig.2.32, the estimation of motion using local methods is correct for small motion while aliasing appears when using local methods to tackle large displacements.

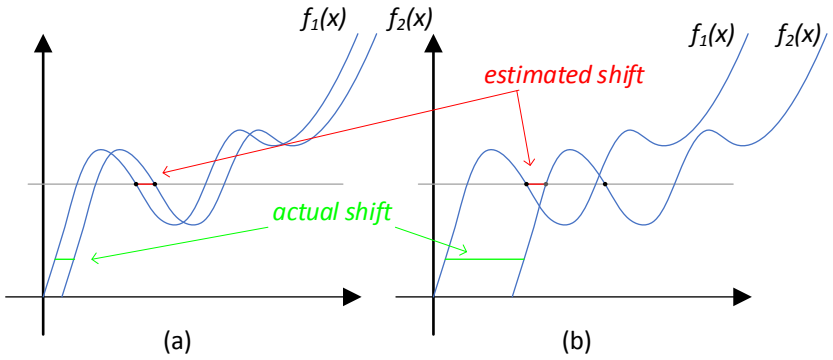


Figure 2.32 Illustration of image aliasing. (a) Nearest match is correct (no aliasing), (b) nearest match is incorrect (aliasing).

To solve the aliasing problem, coarse-to-fine estimation is used to warp images which progressively estimate the flow from coarse to fine levels within Gaussian pyramid of two images. The idea is to create a pyramid of coarse-to-fine down-sampled versions of the original image. At the coarsest level, the linearity domain of the image encompasses large displacements and the estimation can be based on the brightness constancy constraint. The estimations at coarser levels serves to warp the image at

subsequent finer levels, where the original estimation is reduced to a search for small motion increments. The solution is iteratively refined at each level until reaching the full image resolution. It is possible to continue the process in still higher resolutions created by interpolation, as proposed in the coarse-to-fine approach [132]. An example of using such estimation is illustrated in Fig.2.33. The estimation starts at the coarsest level ($k = 3$) which is the warp image with a random initialization and then apply the iterative LK estimation until achieving a convergence $\boldsymbol{v3} = (u3, v3)$. Then the coarse level convergence is used to warp the next level ($k = 2$) using same iteration to obtain a new convergence $\boldsymbol{v2}$. The estimation is repeated until warping level $k = 0$, the original image and get the final convergence $\boldsymbol{v0}$ [18]. The larger number of warping levels can provide more accurate estimation but also introduce large amount of computation load due to the iterative calculation. Therefore, for usual applications, the level number is less than 10 and some may use only 3 warping levels to gain faster estimation.

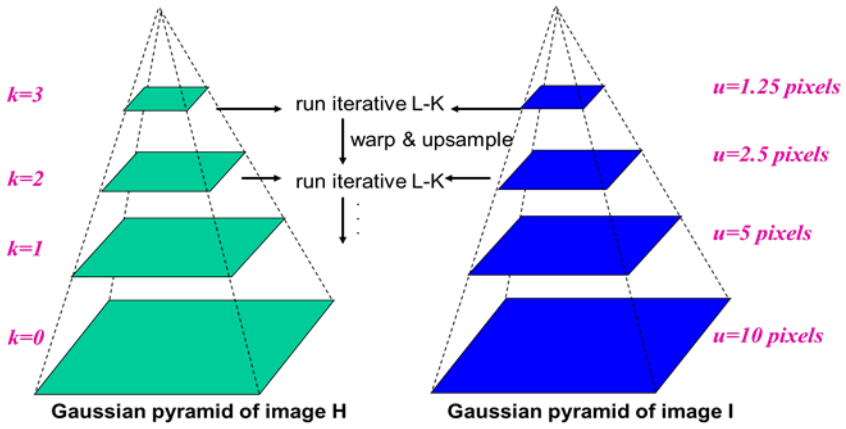


Figure 2.33 A simple example of coarse-to-fine optical flow estimation steps.

To deal with optical flow estimation with large displacements, almost all differential approaches resort to the multiscale solution. However, the main drawback of using coarse-to-fine strategy is that the flow estimation on the coarsest level could result in loss of small and fast-moving objects in the final estimated flow field. If the size of the object is smaller than its displacement, it is likely to be smoothed out when down-sampled to the coarse level. The issue is essentially caused by the data acquisition using frame-based sensors which cannot capture the full trajectory of the moving object

and can be solved in nature by a DVS sensor since it continuously captures and reports the position of the moving object with a temporal accuracy as high as microsecond.

2.3.2.2 Other optical flow techniques

Region-Based Matching

Such approaches define velocity v as a shift $d = (dx, dy)$ that yields the best fit between image regions at different times. Finding the best match amounts to maximizing a similarity measure (over d), such as the normalized cross-correlation or minimizing a distance measure, such as the sum-of-squared difference (SSD). Region-based matching method is more favorable as the accurate numerical differentiation may be impractical due to noise, limit number of frames or image aliasing.

Typical methods: Anandan is one of the typical groups who applied region-based matching approaches. Anandan method employs a Laplacian pyramid and a coarse-to-fine SSD-based matching strategy, allowing the handling of large displacements between frames [133, 134].

Energy-Based Methods

The energy-based methods are also called frequency-based methods because of using velocity-tuned filters in the Fourier domain. The Fourier transform of a translating 2-d pattern is

$$\hat{I}(\mathrm{k}, \omega) = \widehat{I_0}(\mathrm{k})\delta(\omega + v^T \mathrm{k}), \tag{2-15}$$

where $\widehat{I_0}(k)$ is the Fourier transform of $I(\mathrm{x}, 0)$, $\delta(\mathrm{k})$ is a Dirac delta function, ω denotes temporal frequency and $\mathrm{k} = (k_x, k_y)$ denotes the spatial frequency. It shows that all nonzero power associated with a translating 2-d pattern lies on a plane through the origin in frequency space. It has been shown that certain energy-based methods are equivalent to correlation-based methods or the gradient-based methods.

Typical methods: Heeger's method formulated as a least-squares fit of spatiotemporal energy to a plane in frequency space. In order to extract local energy, the method employs 12 Gabor-energy filters even distributed over spatial and tuned to different spatial orientations and different temporal frequencies [135, 136].

Phase-Based Techniques

The phase-based techniques were first introduced by Fleet and Jepson and have been shown to be more precise and robust due to the fact that phase information is more robust to changes in contrast, scale, orientation, and speed. In phase-based techniques, velocity is defined in terms of the phase behavior of band-pass filter outputs. Therefore, the spatiotemporal filter is usually carefully designed in algorithms [17, 137]. Phase-based optical flow algorithms are characterized by high computational requirements and thus barely find their places in real-time applications.

2.3.2.3 Evaluation Measurements and Benchmarks

Besides the development of algorithms for optical flow estimation, much attention has been paid to the design of appropriate evaluation measurements for these algorithms. Currently, there are two main techniques of visualization to express the estimated flow field, arrow visualization and color coding visualization as shown in Fig.2.34. The arrow visualization expresses the motion vectors in arrows which can directly indicate the flow direction and the scale of the arrow tell the flow speed. On the counterpart, the length of arrow vectors need to be normalized for a clean display. The color coding visualization takes use of the color hue and saturation to express the direction and speed of flow field. It allows for expressions of dense flow estimation and provides a better visual perception of subtle differences among neighbor motion vectors.

The performance of optical flow is commonly evaluated by comparing the direction and speed of the estimated flow vectors with the ground-truth. Thus, two error measures regarding to direction and speed respectively are designed, i.e. Angular Error (AE) and Endpoint Error (EPE). If an estimated motion vector is expressed as $w_{est} = (u_{est}, v_{est})^T$ with a referenced ground-truth motion vector $w_{gt} = (u_{gt}, v_{gt})^T$, then the AE of this motion vector is defined by the 3-d angle created by the extended vectors $(u_{est}, v_{est}, 1)^T$ and $(u_{gt}, v_{gt}, 1)^T$ with an expression,

$$AE = cos^{-1}\left(\frac{u_{est}u_{gt}+v_{est}v_{gt}+1}{\sqrt{u_{est}{}^2+v_{est}{}^2+1}\cdot\sqrt{u_{gt}{}^2+v_{gt}{}^2+1}}\right) \quad (2\text{-}16)$$

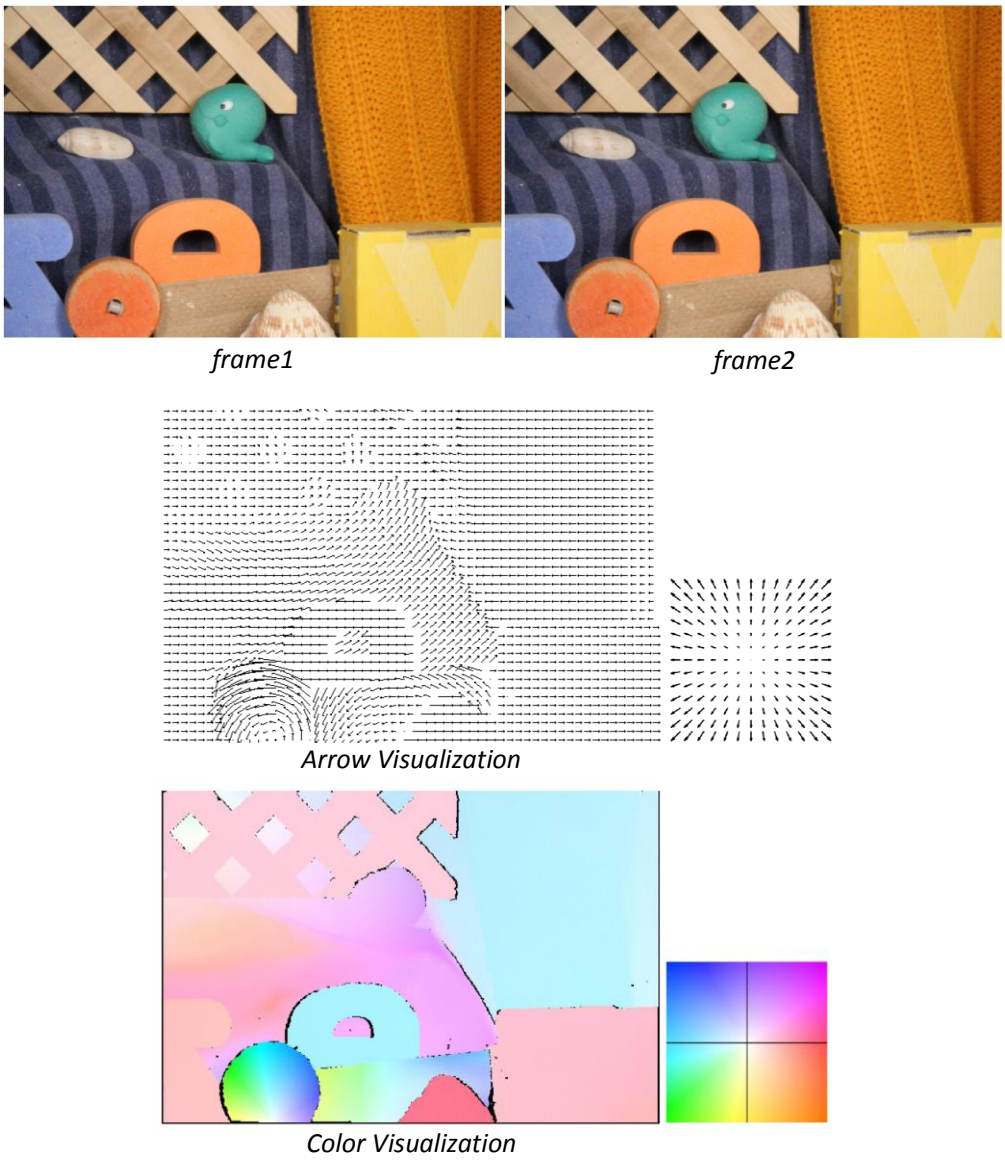


Figure 2.34 Two types of visualization of the motion field transforming from frame1 to frame2 [138].

The EPE is defined as the Euclidean distance between the endpoints of the two vectors:

$$EPE = \sqrt{(u_{est} - u_{gt})^2 + (v_{est} - v_{gt})^2} \tag{2-17}$$

Based on the performance measurements methods, several benchmarks have been built with ground truth which require tremendous amount of work especially for dense optical flow estimation. Targeting at different applications, the three most popular benchmarks which are Middlebury benchmark [138], MPI Sintel benchmark [139], and KITTI benchmark [140], are designed for specific applications. The Middlebury benchmark contains more challenging sequences which contain smooth deformations

Middlebury Benchmark [138]

MPI Sintel Benchmark [139]

KITTI Benchmark [140]

Figure 2.35 Examples of frames and ground truth from the main optical flow evaluation benchmarks.

such as the translation or rotation to real images, also involve motion discontinuities and motion details. Part of the test sequences are synthetic, and some others are real

images captured under a strictly controlled experimental situation allowing for generation of ground truth. In contrast, the image sequences in MPI Sintel benchmark are all extracted from a synthetic movie and raise up the issues mainly related to large displacements, occlusions, illumination changes, and effects like blur or defocus. Moreover, the KITTI benchmark provides real-world datasets captured by driving around the city of Karlsruhe, in the rural areas and on highways and the ground truth is provided by a Velodyne laser scanner and a GPS localization system. This benchmark is mainly design for assisted vehicle driving. The typical testing sequences and the corresponding ground truth provided are shown in Fig.2.35.

2.3.2.4 Advantages and Limitations of Frame-based Optical Flow Methods

Frame-based optical flow techniques have been developed over twenty years and have become quite sophisticated. A large amount of open source databases and test benches can be found online. Some techniques are capable to achieve reliable results in situations with occlusions or motion discontinuities [128,129]. Some other techniques introduce gradient consistency to improve the robustness of illumination change, which is constrained by small displacements [141]. Some methods employ multi-scale techniques to handle large displacements, which compute the flow through coarse-to-fine scales. These downscaling techniques first use coarser scales to initialize the global computation as global spatial smoothing, which smooth away the motion of small structures. Thus, the computed optical flow is dominated by large motion structures and is not accurate towards small motions [142]. Techniques based on matching and frequency are more robust at the expense of lower accuracy. The main advantage of such techniques is to take into account the small structures, which could be missed by the coarse-to-fine methods. In order to extract accurate information of small motions, sparse hypotheses are integrated into variational approaches to conduct local optimization. Some optimizations estimate dense optical flow with subpixel accuracy [143,144], however, is greatly influenced by the outliers and scale selection. Such outliers can be removed through the application of the coarse-to-fine estimation.

Since the frame-based optical flow estimation employs the frame-based image acquisition, the post processing speed is limited by the frame rate. Referring to the

characteristic, light intensity is integrated over time interval and thus cannot capture dynamic light intensity change. As a result, some information of fast moving objects would be missing. In addition, different illumination conditions force the camera system to globally adapt the operating range using different exposure times or apertures, which increase the power consumption and processing time. In most cases, when calculating optical flow of moving targets, the observed background remains stationary and provide non-useful information. However, frame-based cameras always capture full images providing tremendous redundancy data which burden the flow computation. Last but not least, the frame-based processing algorithms rely on the iteration of all the image pixels several times and repeat it for each frame and thus have computational cost. Due to the above described limitations, frame-based optical flow processing has high power consumption and is not fast enough for real-time applications.

2.3.3 Event-based Optical Flow Algorithms

Since frame-based image acquisition and processing techniques are not computationally efficient, recently several adaptive event-based optical flow algorithms have been proposed. These techniques compute dense optical flow by employing an asynchronous DVS instead of the frame-based cameras. This section summarizes the event-based optical flow algorithms with the testing results compared to the frame-based optical flow.

The first event-based optical flow algorithm was proposed by Benosman [143]. The algorithm is based on the LK differential method and computes spatial and temporal gradients by accumulating events during a short time period. The main steps of this algorithm are described as follows,

1. Attain the spatial-temporal gradients by using instantaneous activities of each individual pixel instead of grey level. Since the DVS only reports temporal contrast, the number of events represent the changing speed of light intensity instead of absolute light intensity information.
2. Based on local constant assumption, apply the attained gradients to a defined $n \times n$ window and derive n^2 optical flow equations.

3. Solve the equation system by using least square error minimization techniques.

As introduced in the traditional optical flow, the optical flow constraint can be formulated in an event-based manner as well [15]. The main difficulty being that event-based retinas do not provide gray levels. The absence of gray levels makes it difficult to provide a spatial gradient computation. Nevertheless, for adjacent active pixels it is possible to provide an estimation of the spatial gradient by comparing their instantaneous activities:

$$\begin{cases} \frac{\partial e(x,y,t)}{\partial x} \sim \sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x-1,y,t) \\ \frac{\partial e(x,y,t)}{\partial y} \sim \sum_{t-\Delta t}^{t} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x,y-1,t). \end{cases} \quad (2\text{-}18)$$

Δt is a temporal interval of few μs and is generally set to $50\mu s$. The estimation of the temporal gradient is expected to be more precise than in the frame-based framework due to the high temporal accuracy of the DVS. It can be written as

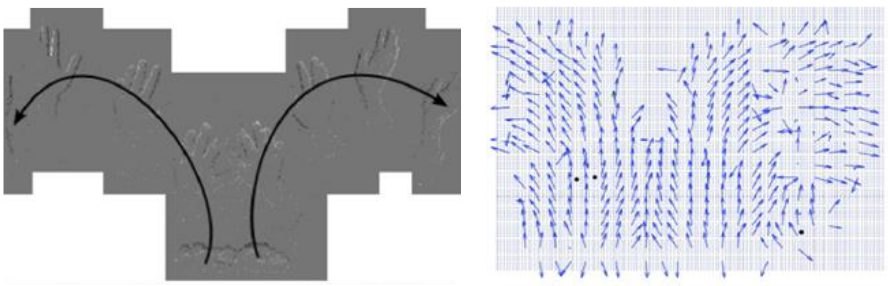
$$\frac{\partial e(x,y,t)}{\partial t} \sim \left(\sum_{t-\Delta t}^{t_1} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x,y,t)\right) / t - t_1 \text{ , with } t_1 < t. \quad (2\text{-}19)$$

By substituting Eq. (2-18) and Eq. (2-19) into Eq. (2-6), the j th line of the matrix equality can be reformulated using events:

$$\left(\sum_{t-\Delta t}^{t} e(x_j, y_j, t) - \sum_{t-\Delta t}^{t} e(x_j - 1, y_j, t)\right) v_x + \left(\sum_{t-\Delta t}^{t} e(x_j, y_j, t) - \sum_{t-\Delta t}^{t} e(x_j, y_j - 1, t)\right) v_y$$

$$= \sum_{t_1}^{t} e(x_j, y_j, t) \Big/ t - t_1$$

Since



(a) (b)

Figure 2.36 Example of optical flow estimation using event-based algorithm. (a) DVS result of moving hands, (b) optical flow estimation in arrow visualization.

$$\sum_{t-\Delta t}^{t_1} e(x,y,t) - \sum_{t-\Delta t}^{t} e(x,y,t) = \sum_{t_1}^{t} e(x,y,t).$$

Fig.2.36 shows the optical flow field of hands with curve motion using the event-based algorithm. Several tests were conducted by this group and the results compares the computational time required by the frame-based and event-based optical flow methods. It shows that the computation time consumed by event-based algorithm is approximately 25 times lower than frame-based method.

The same group later proposed another event-based algorithm similar with the event LK method but adopts a plane fitting method [144] based on the spatial-temporal character of the DVS events, as shown in Fig.2.36. The surface of active events $\sum e$ is derived to provide an estimation of amplitude and orientation of motion. Since the derivatives of the points on the fitted plane can always be calculated, iterative computation is enabled in this algorithm. The algorithm starts by collecting instantaneous activities of each individual pixel. For each pixel event $e(x, y, t)$, define a spatiotemporal neighborhood, centered on $e(x, y, t)$ of spatial dimensions $n \times n$ and duration $[t - \Delta t, t + \Delta t]$. Then a robust plane fitting is applied through least square minimization techniques and iterative computation. The plane fitting functions as flow regularization. On the one hand, the plane fitting regularization rejects the events that are too far from the plane to improve the flow estimation robustness against noise. On the other hand, it compensates for absent events in the neighborhood of active events. Heiko, et.al [29] proposed an approach that exploits the local plane using direction-selective filters which contain a family of superposed spatial-temporal filters that

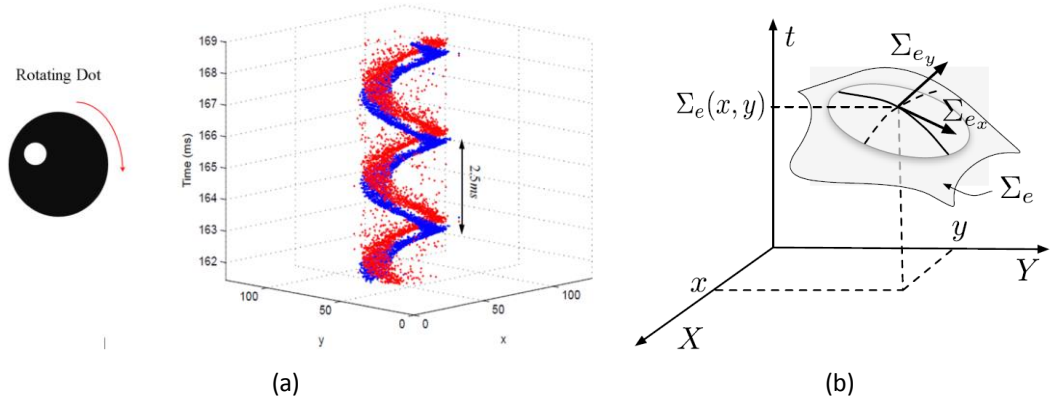


Figure 2.37 (a) Accumulated events of a rotating dot in spatial-temporal domain, (b) general principle of event-based visual flow using plane fitting method.

consist of quadrature pairs of spatial weighting profiles and mono/bi-phasic temporal profiles.

2.3.4 Advantages and Limitations of Event-based Optical Flow Estimation

According to the study of the event-based optical flow, optical flow estimated using adapted methods with asynchronous event-based acquisition has higher dynamic and lower cost computation compared to frame-based optical flow. Since the dynamic vision sensors only detect illumination changes and trigger events to process, the stationary background no longer produces redundant data that reduces the computation cost and power consumption as well. Another advantage of using event-based sensor is to overcome the framerate limitation as the DVSs allow for high temporal resolution. Time interval between two frames cannot be less than several milliseconds while that between two event-slices (accumulated events during a certain time interval) could be reduced to tens of micro seconds. The high temporal resolution allows faster flow small calculation and thus high responding speed towards dynamic motions. The testing results show that event-based optical flow techniques outperform the frame-based techniques in relation to large displacements and subpixel accuracy.

The above described event-based optical flow algorithms all employed the traditional DVS as data acquisition. High sparseness of such DVS when detecting high speed moving object introduces accuracy issues, because single triggered event cannot provide information on spatial gradient. Since each pixel processes individually, events have to be accumulated over a certain interval and post-processed synchronously as event slices. Therefore, there is a need to introduce new framework to increase accuracy on the basis of the event-based optical flow and take advantage of the asynchronous character of DVS, which can make it more valuable in real-time analysis of high-speed events.

2.4 Summary

In this chapter, we have reviewed the structures and mechanisms of frame-based cameras and event-based cameras. We also reviewed and compared frame-based approaches and event-based approaches in object tracking and optical flow. Frame-based architectures have been dominant in machine vision nowadays, but they carry hidden costs because they are based on a series of snapshots taken at a constant rate. The pixels are sampled repetitively even if their values are unchanged. Short-latency vision problems require high frame rate and produce massive output data (e.g., 1 GB/s from 384×320 pixels at 10 kfps). Pixel bandwidth is limited to half of the frame rate and reducing the output to a manageable rate by using region-of-interest readout usually requires complex control strategies. Dynamic range is typically limited by the identical pixel gain, the finite pixel capacity for integrated photocharge, and the identical integration time. For machine vision in uncontrolled environments with natural lighting, limited dynamic range and bandwidth can compromise performance.

Instead of taking frames, pixels of event-based vision sensors respond asynchronously to intensity change. The event sensor output is an asynchronous stream of pixel address-events that directly encode scene reflectance changes, thus reducing data redundancy while preserving precise timing information. The dynamic events record precisely timed information about object and image motion, simplifying post-processing with direct motion extraction. Therefore, from the review of object tracking and optical flow approaches, event-based algorithms are observed to have lower algorithmic complexity and computational cost. Moreover, due to less data redundancy, the event-based approaches require less data storage and the sensors consume less power compared with frame cameras. In addition, the event cameras achieve wide dynamic range (>120dB), low latency (15 μs), low power consumption (23mW), and low mismatch (2.1%) [23]. The high dynamic range and bandwidth make event cameras become an efficient solution to dynamic vision problems.

On the other hand, there are still some limitations of event cameras when applied in vision processing, where intensity information is essential for image content interpretation. Even though two types of hybrid event and frame cameras have been

designed, there are latency and mismatch between events and intensity information leading to inaccurate feature extraction. Basic processing steps, e.g. gradient extraction, histogram feature extraction and local Gaussian filtering, require intensity information for pixels within a local window. However, the output data of event cameras are sparse pixel events that not all the target pixels can collect useful information for post-processing. Based on these limitations, it is essential to design novel event-cameras to break the limitations and facilitate the application of event cameras in computer vision.

Chapter 3 Motion Sensor with Asynchronous Grayscale Event-guided High-Speed Object Tracking

3.1 Motivation

It was reviewed in chapter 2.2 that it is still a big challenge for the state-of-the-art tracking algorithms to realize real-time tracking of super-fast-moving objects, such as ball strike, rotating disk and high speed moving vehicles. The main factors that limit accuracy and real-time performance are motion blur and large movement displacement. Motion blur is typically caused by the structural characteristic of conventional frame-based cameras, as elaborated in chapter 2.1. With the presence of motion blur, it is hard to recognize and extract a clear contour of the target object. Moreover, the feature vectors extracted from a predicted 'region-of-interest' cannot well represent the objects and thus introduce high inaccuracy in tracking results. On the other hand, video sequences for tracking are usually captured by frame-based cameras with a frame rate of 30Hz. For a $33ms$ time interval, a super-fast-moving object can move a long distance, i.e. a large pixel distance when reflected in an image. The movement displacement is adaptive within a wide range especially in long term tracking. However, existing trackers usually predefine the searching area with a constant radius. If the searching radius is small, the searching area may lose the target object when it moves fast. If the radius is large, it introduces huge redundant computational cost since it uniformly searches within the searching area.

To solve the aforementioned problem, a new dynamic vision sensor with asynchronous pixel intensity was proposed to help with guiding an adaptive searching area in object tracking. Since dynamic vision sensors are sensitive to motion and have high temporal resolution, they can easily capture the trajectories of fast moving objects which can be used in object tracking. However, traditional dynamic vision sensors used in event-based tracking only output binary events. Lack of asynchronous pixel intensity

greatly limits the application of traditional event-based motion sensors in image processing tasks. Therefore, an asynchronous intensity readout scheme with low latency was proposed. With the novel designed motion sensor, an event-guided structured output tracking method was designed for the tracking of super-fast-moving objects.

The structure of this chapter is organized as following. Section 3.2 elaborates the proposed event-guided tracking algorithm. Section 3.3 describes the architecture and design of the novel motion sensor in detail. Section 3.4 introduces the chip implementation and measurement results. Section 3.5 discusses experimental results and section 3.6 summarizes this chapter.

3.2 Image Sensor Design

3.2.1 Sensor Architecture

The main target of the new motion sensor is to asynchronously output events with both pixel position and illumination. It should also be able to generate grayscale full frame images on demand. As shown in Fig.3.1, the design ideas contain three main parts, (a) directly take the output of logarithmic detector as intensity output by introducing a buffer and readout switch; (b) introduce a global control signal *forcefire* to generate full-frame images on demand; (c) the buffered analog intensity value is converted and outputted through a specifically designed column analog readout path.

As reviewed in the literature, state-of-the-art dynamic vision sensors use logarithmic detectors to convert input photocurrent signals into voltage signals. The log detectors are used only to detect intensity change in either pure DVSs or DAVIS and ATIS. The output voltage of log detector, V_{PH} in Fig.3.1, continuously responds to the photocurrent in a logarithmic manner. Since the photocurrent is proportional to the input illuminance, the output voltage of the log detector can directly reflect the input illuminance. Therefore, if the V_{PH} of each active pixel can be buffered out and measured, the illuminance of the pixel can be estimated. Both minimum address-event latency and temporal resolution for intensity readout are in the order of microseconds. The

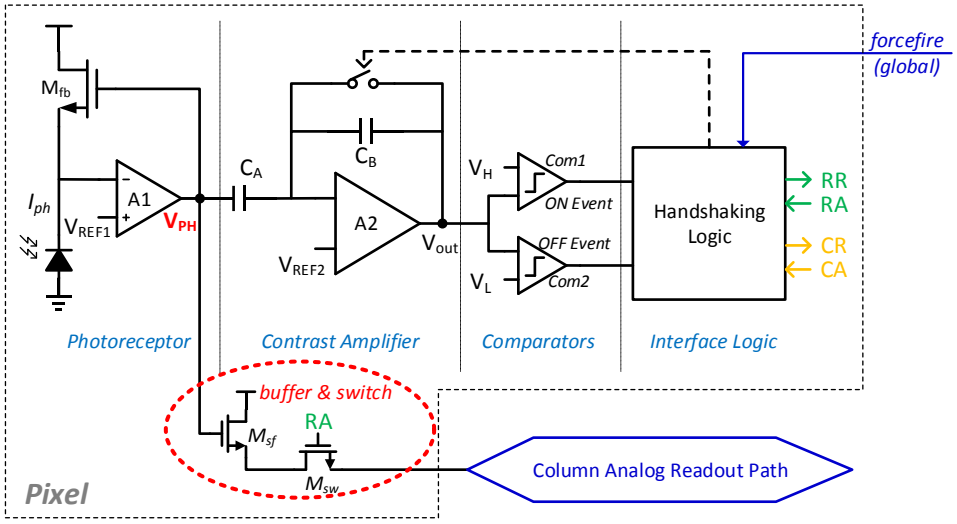


Figure 3.1 Design ideas contain three main parts, (a) directly take the output of logarithmic detector as intensity output by introducing a buffer and readout switch; (b) introduce a global control signal to generate full-frame images; (c) the buffered analog intensity value is converted and outputted through a specifically designed column analog readout path.

aforementioned undesired discordance between the fast address-events and the postponed intensity output of the ATIS and DAVIS are settled accordingly. In addition, a global control signal *forcefire*, which globally and simultaneously controls the status of each pixel in the pixel array, is introduced to generate full-frame images with gray intensities, as shown in Fig.3.1. For vision-based processing tasks, e.g. object tracking, the research work has developed over decades and frame-based image processing algorithms are much more sophisticated than pure event-based methods. Therefore, the capability of the new motion sensor to acquire full-frame images allows the motion sensor to be more compatible with the advanced algorithms. To output the buffered analog intensity value, a column analog readout path is used to convert the analog signals into digital signals.

The architecture of the whole sensor is shown in Fig.3.2, taking a 4×4 pixel array as an example. The whole sensor consists of a pixel array, readout handshaking logic, column intensity readout circuitry and system control. Blocks not found in the traditional DVS are labelled in blue. Referring to the design idea, modifications are conducted in each pixel, column processing unit and system level control block. The next sub-parts will describe the novel sensor design in detail.

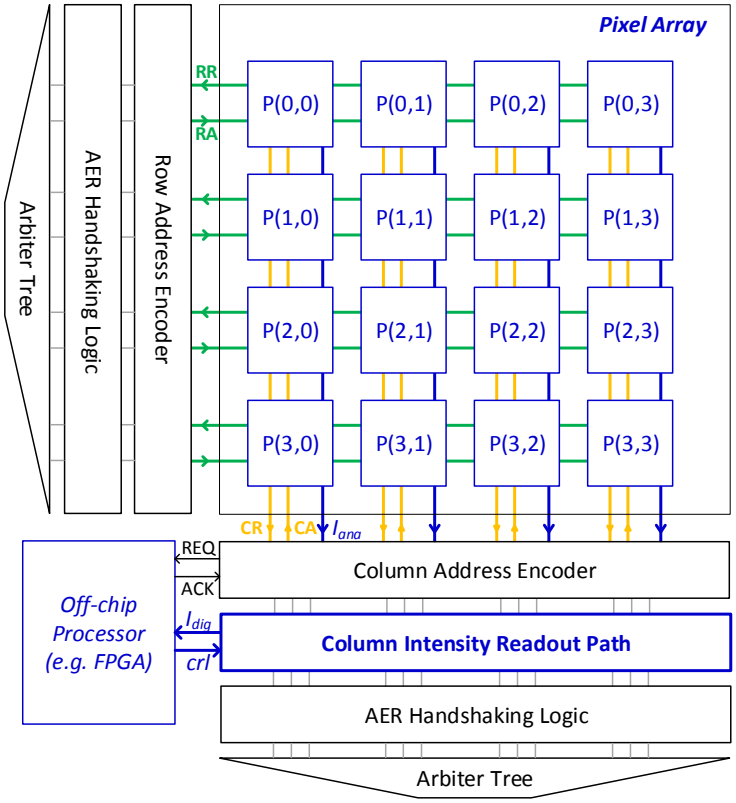


Figure 3.2 Sensor architecture with a 4 × 4 pixel array taken as an example. Based on traditional DVS (Fig.2.6), modifications are conducted in pixel, column readout processing and signal control on system level, which are labelled in blue.

3.2.2 Intensity Readout Path

The details of the intensity readout path are illustrated in Fig.3.3. The path contains two main sections: buffer and switch transistors in pixel and a column intensity readout circuit with signal amplifier and analog-digital conversion (ADC), which are responsible for conditioning and quantizing the output voltages from activated pixels in a column.

The complete intensity readout path is illustrated in Fig.3.4. For simplicity, only the front-end logarithmic detector together with the buffer and switch transistors (M_{sf} and M_{sw}) are shown in pixel. The incident light is first detected and converted into voltage signals through a logarithmic detector. If the intensity change is larger than the pre-defined threshold, the pixel generates an event and becomes active. The generated event signal sends row and column requests to the peripheral circuits through the handshaking logic and waits for row and column acknowledge signals to process the

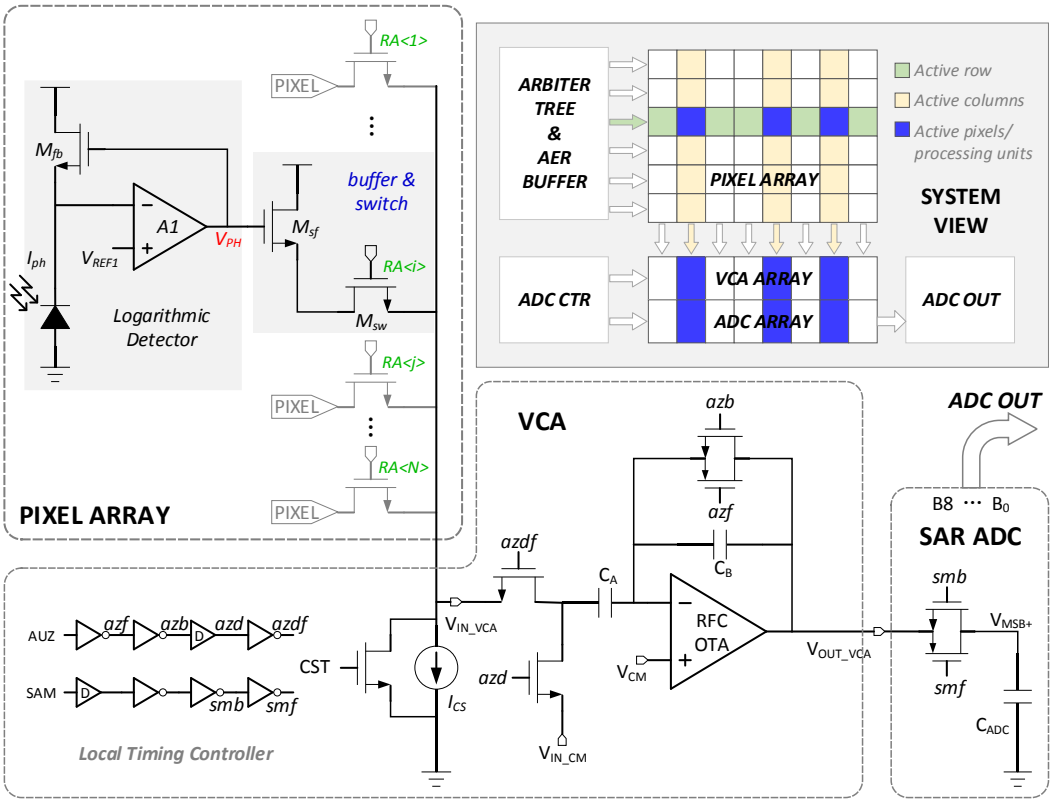


Figure 3.3 Schematic of a complete intensity readout path.

active pixel. For intensity readout of this active pixel, two transistors are added to buffer the log-intensity, one is a source follower and the other one is a switch controlled by row acknowledgement signal. The source follower continuously samples the output voltage of the log detector and once row acknowledgement signal (*RA*) arrives, the voltage is buffered out to a column shared bus followed by a column-parallel intensity processing circuit, which consists of a block of voltage calibration and amplification (VCA) and a subsequent ADC. The buffered analog intensity signal is first amplified by the VCA and then quantized into a digital signal by the ADC. Once the analog-digital conversion is completed, the derived digital signal is outputted using the column acknowledge signal. The main purpose of using VCA is to ensure good quantization especially when the intensity range in the observed scene is small. In that case, the voltage range of the buffered log-intensity is even smaller due to the logarithmic compression. The input range of the ADC is designed considering the dynamic range, which is much larger than the voltage range in a common scene. Therefore, the VCA amplifies the input voltage and increases the useful input range for the ADC. In addition, the SCA adjusts the common-mode level of pixel output voltage to accommodate the

quantization range of the ADC. Since the ADCs are column-parallel with a pitch as small as 30um, an area-efficient SAR ADC with significantly simplified capacitance-DAC was adopted in this design. The detailed circuit design and operation principles of VCA and ADC will be introduced later in this section.

3.2.2.1 Voltage Calibration and Amplification (VCA)

As mentioned previously, the column-parallel VCA is the interface between the pixel and ADC. The output voltage of a pixel is calibrated and amplified according to the input voltage range of the subsequent ADC, which is defined based on the operation principle and characteristics of the logarithmic detectors in pixels.

The front-end circuit employs a logarithmic receptor to detect light intensity. The photocurrent output is a function of the incident light intensity. As shown in Fig.3.4, the main functions of the amplifier (*A1*) and feedback transistor (M_{fb}) is to clamp the photodiode node voltage (V_{PH}) at a predefined reference voltage (V_{REF1}). Since the photocurrent is small and usually ranges from several pico-amperes to several nano-ampere, the feedback transistor (M_{fb}) works in the subthreshold region. For a transistor falling into subthreshold region, its drain current is derived as

$$I_{ph} = I_0 exp \frac{V_{GS_{fb}} - V_{TH_{fb}}}{\zeta V_T} \tag{3-1}$$

where I_0 is the transition current, also the drain current when $V_{GS_{fb}} = V_{TH_{fb}}$, $\zeta > 1$ is the subthreshold slope factor and $V_T = kT/q \approx 26\text{mV}$ under room temperature. For AMS 0.35μm technology, ζ is about 1.3. To derive the output of the logarithmic receptor, $V_{GS_{fb}}$ can be derived from Eqn. (3-1),

$$V_{GS_{fb}} = V_{TH_{fb}} + \zeta V_T ln \frac{I_{ph}}{I_0} \tag{3-2}$$

and thus V_{PH} is calculated as

$$V_{PH} = V_{REF1} + V_{GS_{fb}} = V_{REF1} + V_{TH_{fb}} + \zeta V_T ln \frac{I_{ph}}{I_0} \tag{3-3}$$

It is observed that output voltage V_{PH} logarithmically responds to the drain current I_{ph}, as shown in Fig.3.4.

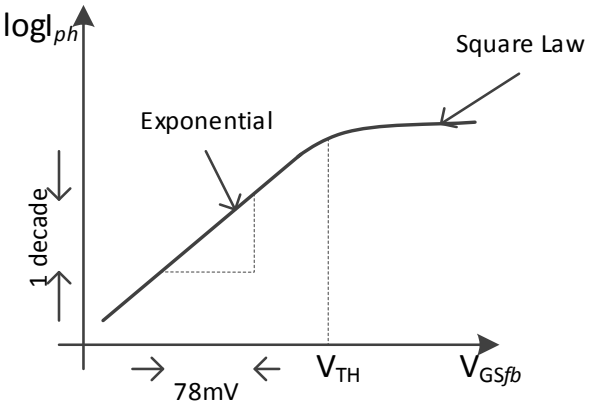


Figure 3.4 $logI_{ph}$ against $V_{GS_{fb}}$ of transistor M_{fb} under subthreshold conduction.

For I_{ph} changes over one decade from I_{ph1} to $I_{ph2} = 10I_{ph1}$, the change of V_{PH} is

$$\Delta V_{PH} = V_{PH}|_{I_{ph2}} - V_{PH}|_{I_{ph1}}$$
$$= \zeta \cdot V_T \cdot \ln(I_{ph2}/I_{ph1}) = 1.3 \cdot 26\text{mV} \cdot \ln 10 \approx 78\text{mV} \quad (3\text{-}4)$$

Therefore, for every decade change of photocurrent, the output voltage of the logarithmic receptor changes by $78mV$. Since the detected photocurrent ranges from several pico-amperes to several nano-amperes, the total range of V_{PH} is logarithmically compressed to a range over about $500mV$. Therefore, an amplifier is needed before ADC quantization as the ADC input quantizing range is much larger than $500mV$ (usually set as $2V$). Once the row acknowledge signal (e.g. $RA < i >$ in Fig.3.3) is received, the voltage V_{PH} is buffered out to a shared bus, which is biased by a current source I_{CS} and connected to the column-parallel VCA. The column current source (I_{CS}) providing bias current for the in-pixel source follower is chosen as $3\mu A$. To accelerate the amplification, an extra switch controlled by CST is introduced in parallel with I_{CS}. It provides a low-resistive discharging path for the large parasitic capacitance along the column readout bus during the switching between two different rows. It resets V_{IN_VCA}

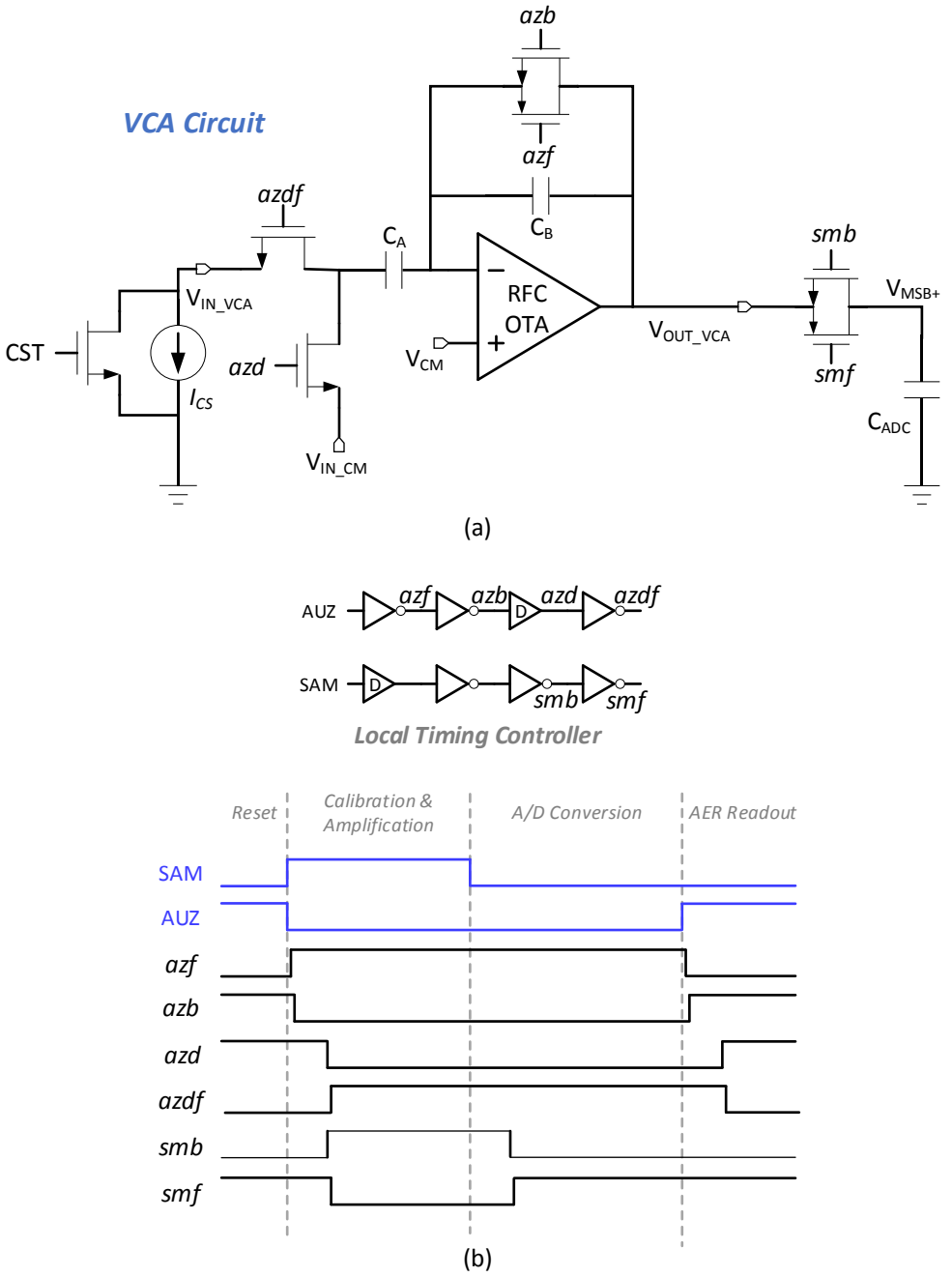


Figure 3.5 (a) Schematic of VCA (voltage calibration and amplification), (b) timing diagram of control signals generated by the Local Timing controller.

to GND and thus shortens the amplification procedure since the speed of in-pixel source follower in charging this capacitance is much higher than during discharge.

Fig. 3.5 shows the schematic and timing diagram of VCA. The VCA block consists of two parts: DC level calibration (transistors M1 and M2) and a recycling folded cascode (RFC) OTA [147] with a gain provided by capacitors C_A and C_B. The output of the OTA is directly connected to the capacitance-DAC of the subsequent ADC, which

would convert this voltage into digital codes. The operation of VCA contains four statuses: reset, calibration and amplification, A/D conversion and AER readout. The operation of the VCA is mainly controlled by local-generated control signals. These control signals are generated by a local timing adjustment circuit taking use of two global digital signals *SAM* and *AUZ*. The timing diagram of the control signals is shown in Fig.3.5(b). During the reset period ($SAM = 0/AUZ = 1$), the amplifier is reset such that the output voltage $V_{OUT_{VCA}}$ equals to the reference voltage of the amplifier V_{CM}. Also, the left plate of the capacitor C_A keeps sampling the input reference voltage $V_{IN_{CM}}$. Once the VCA receives the output voltage from the selected pixel, V_{IN_VCA}, it starts calibration and amplification by first turning off the reset switch of the amplifier and the sampling switch of $V_{IN_{CM}}$, then turning on the transistor M1 to sample the input voltage to C_A. After amplification, the output voltage $V_{OUT_{VCA}}$ is expressed as

$$V_{OUT_{VCA}} = V_{CM} - \frac{C_A}{C_B}\left(V_{IN_{VCA}} - V_{IN_{CM}}\right) \tag{3-5}$$

Where V_{CM} is the reference voltage of the amplifier and is set to $VDD/2$ $(1.6V)$ to achieve a large output swing; $V_{IN_{CM}}$ is the reference value of input voltage and it is set to the middle of the range of $V_{IN_{CM}}$ such that the calibrated voltage range, i.e. the input voltage range of the amplifier, centres at V_{CM}. The operation principle of VCA with respect to V_{CM} and $V_{IN_{CM}}$ is illustrated in Fig.3.6. The red and blue segments on the VCA Input Range represent the voltage range of $V_{IN_{VCA}}$ when testing under two situations with different strength of mean illumination. The VCA is designed to process the illumination with a dynamic range (DR) as high as 120dB, however, the DR of a general testing situation only covers a small part of the whole input range.

The gain value $\frac{C_A}{C_B}$ of the amplifier is designed based on the voltage range of $V_{IN_{CM}}$ and the input voltage range of ADC. When the output voltage of the logarithmic detector V_{PH} is buffered out, there is a voltage drop due to the source follower. According to Eqn. (3-3), the relationship between $V_{IN_{VCA}}$ and V_{PH} can be expressed as

$$\begin{aligned} V_{IN_{VCA}} &= V_{PH} - V_{GS_{sf}} \\ &= V_{REF1} + V_{TH_{fb}} + \zeta V_T ln\frac{I_{ph}}{I_0} - V_{TH_{sf}} - \sqrt{\frac{2I_{CS}}{\mu_n \cdot C_{ox} \cdot S_{sf}}}, \end{aligned} \tag{3-6}$$

where $V_{TH_{sf}}$ and S_{sf} are the threshold voltage and aspect ratio (W/L) of M_{sf}, μ_n is the electron mobility and C_{ox} is the unit area gate-oxide capacitance of MOSFETs. As a result, V_{OUT_VCA} could be rewritten as

$$V_{OUT_{VCA}} = V_{CM} - \frac{C_A}{C_B}\left(V_{REF1} + V_{TH_{fb}} + \zeta V_T ln\frac{I_{ph}}{I_0} - V_{TH_{sf}} - \sqrt{\frac{2I_{CS}}{\mu_n \cdot C_{ox} \cdot S_{sf}}} - V_{IN_{CM}}\right) \quad (3\text{-}7)$$

Based on this equation, several key points with respect to the intensity readout strategy of this sensor could be emphasized as follows. (1) $V_{OUT_{VCA}}$ is inversely proportional to $\ln I_{ph}$, which means that a brighter pixel (with stronger incident light and larger I_{ph}) leads to smaller $V_{OUT_{VCA}}$ and the corresponding ADC output code. On the other hand, the image sensor achieves very high dynamic range due to the logarithmic compression of I_{ph} but at the expense of relatively low sensitivity. (2) With the gain value $\frac{C_A}{C_B}$ of the

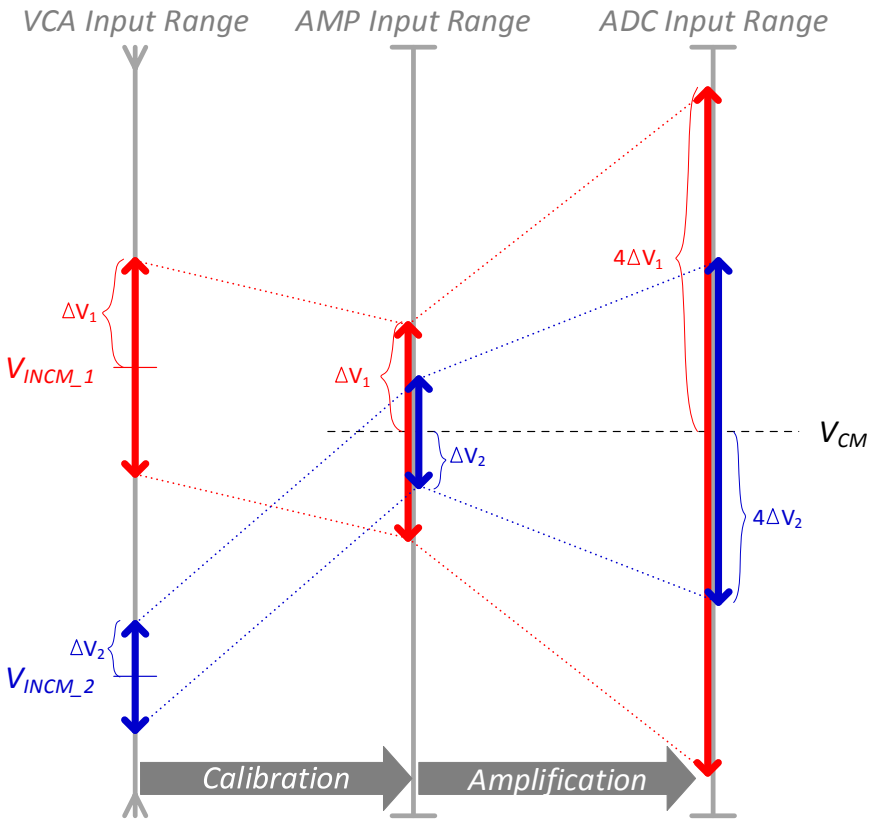


Figure 3.6 Illustration of the operation principles of VCA. V_(IN_CM) is set to the middle level of VCA input voltage range, which is adjustable according to the test situation. V_CM is set at the centre of ADC input range, which is also the reference voltage of OTA.

amplifier, the OTA amplifies the input signal ($V_{IN_{VCA}}$) and maps it with the full-scale range of the subsequent ADC. In simulation, for a decade increase of I_{ph} under room temperature, V_{PH} increases about $78mV$ and $V_{IN_{VCA}}$ increases about $65mV$ (due to the gain loss of the source follower). As a result, for I_{ph} ranging about 6 decades ($120dB$ dynamic range), the swing of $V_{IN_{VCA}}$ is about $390mV$. By setting the gain value $\frac{C_A}{C_B}$ to 4 ($C_A = 400fF, C_B = 100fF$), the range of $V_{OUT_{VCA}}$ is a little smaller than $1.6V$, which is close to the full-scale range of ADC ($3.3V$). (3) The DC level of $V_{OUT_{VCA}}$ could be adjusted by $V_{IN_{CM}}$. As mentioned above, in general, $V_{IN_{CM}}$ should be in the middle of the range of $V_{IN_{VCA}}$. Since the voltage range of $V_{IN_{VCA}}$ varies in different observation scenarios, $V_{IN_{CM}}$ is adjustable according to different applications.

3.2.2.2 Column-Parallel SAR ADC

A high quality analog signal path is required to directly readout the analog voltage of the logarithmic photo detector. One well-known drawback associated with logarithmic pixel is the poor response at low light, which corresponds to a small voltage swing. Therefore, a 4-gain amplifier is first used to amplify the signal before feeding it into a column parallel ADC processing unit. The column parallel scheme allows high throughput by parallelly processing AD-conversion, which requires several clock cycles, of the active pixels in the selected row. The ADC processing units are activated adaptively according to the pixel status, such that only active pixels trigger the AD-conversion to save power.

Several ADC architectures are popular for column parallel readout, such as single-slope ADC, algorithmic (or cyclic) ADC and successive approximation register (SAR) ADC. Single-slope ADC is very hardware-efficient, but this comes at the expense of long conversion time. Algorithmic ADC works much faster by means of a switched capacitor circuit to implement an efficient binary search algorithm. However, it usually requires a high performance operational transconductance amplifier (OTA), which typically burns a lot of power. SAR ADC is a better choice due to its combined advantages of relatively high conversion speed and low power dissipation. Conventional SAR ADC employs a set of binary weighted capacitances to adjust the

input sampling voltage. For 9-bit resolution, it would need 512-unit capacitances in total. Differential architecture would consume even more capacitances. It will be very difficult to implement such a large number of capacitances within a narrow column slice with good matching. In order to limit the use of capacitances, we combined a top-plate sampling [145] and a two-step architecture [146]. The designed 9-bit SAR ADC only needs 14-unit capacitances.

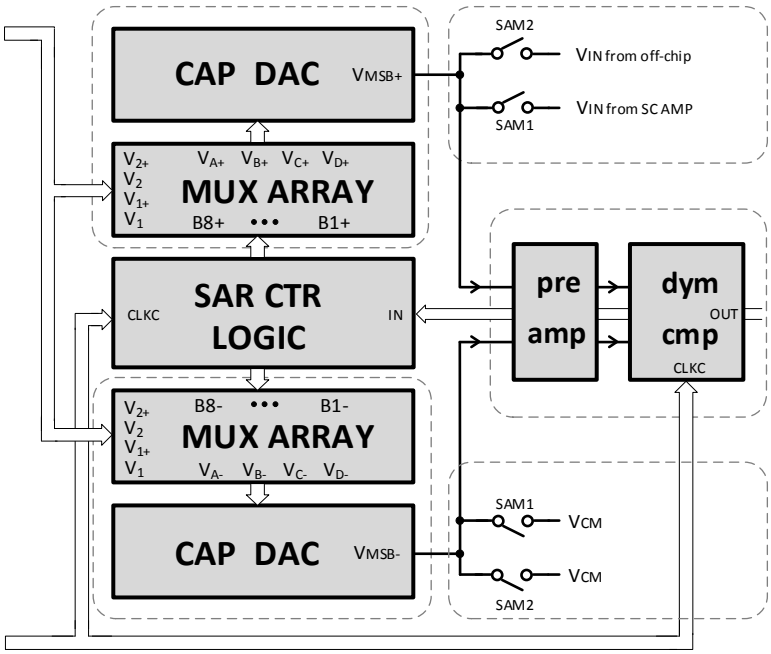


Figure 3.7 Block diagram of the 9-bit SAR ADC.

Fig.3.7 shows the block diagram of the SAR ADC. It mainly consists of four building blocks, namely cap-DAC, reference voltage MUX, SAR control logic and comparator. The control logic generates a series of sequential pulses with each pulse controlling a conversion bit. The comparator is composed of a classic sense amplifier with a preceding preamplifier to reduce its offset voltage and kick back noise. In order to suppress the common-mode noises from the power supply and substrate, pseudo-differential architecture is adopted with one cap-DAC sampling the input voltage while the other one sampling the common-mode voltage V_{CM}.

The ADC operates in two phases, namely signal sampling and AD conversion, while the latter can be further divided into coarse and fine conversion sub-phases. As shown in Fig.3.8(a), each cap-DAC is a 4-bit capacitance array. It consists of two identical sub 2-bit binary-weighted capacitance arrays bridged by an attenuation

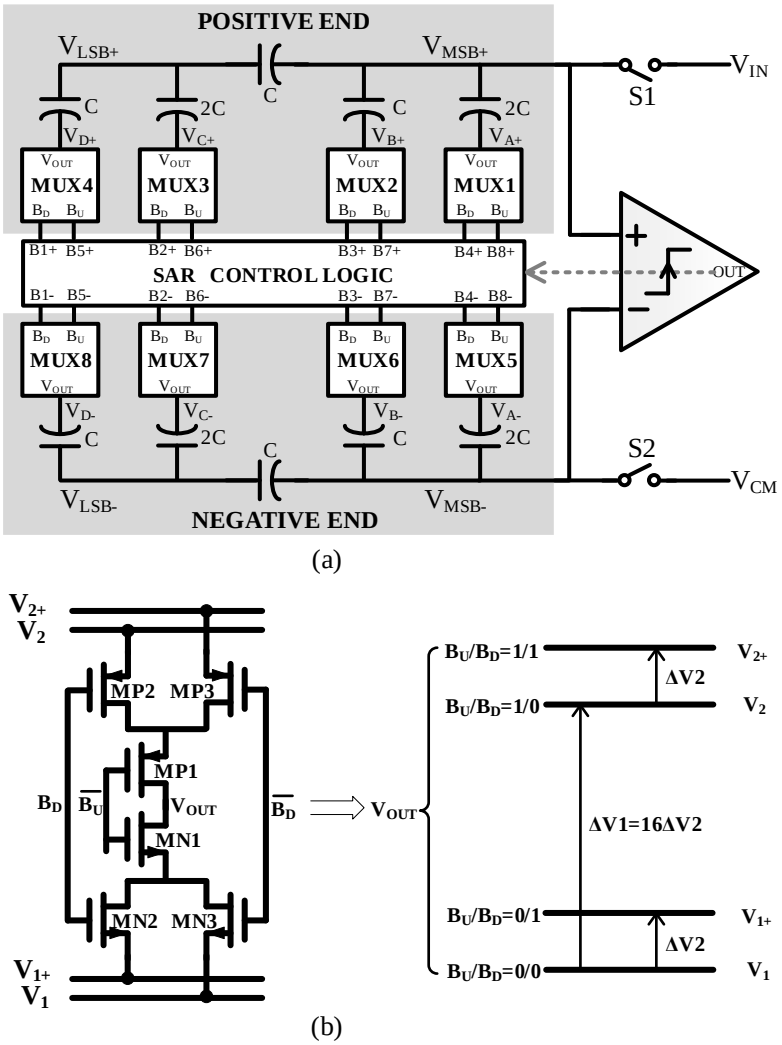


Figure 3.8 (a) Schematic of the 9-bit ADC, (b) reference voltages MUX.

capacitance. There is a total of 14-unit capacitances in this design. Each bottom plate of the capacitances in the cap-DAC is driven by an analog MUX circuit, which could switch between 4 reference voltages, namely V_1, V_{1+}, V_2, and V_{2+}. The two control bits B_U and B_D come from the SAR control logic. The output of the MUX V_{OUT} with respect to the different combinations of the B_U/B_D is shown in Fig.3.8(b). In the sampling phase, both control bits (B_U and B_D) are 0 thus V_{OUT} is clamped at V_1. During the conversion phase, there are three possible switching schemes of the MUX: (1) V_1 to V_2; (2) V_1 to V_{1+}; and (3) V_2 to V_{2+}. The first switching scheme of the MUX is used in the coarse conversion phase (conversion of $B8$ to 4), while the latter two switching schemes are used in the fine conversion phase (conversion of $B3$ to $B0$). Since the output of the

MUX always jumps to a higher reference voltage, this ensures the monolithic change of the input sampling voltages (V_{MSB+} and V_{MSB-}).

Fig.3.9 shows an example of ADC conversion. During the sampling phase, all control bits $Bi+$ and $Bi-$ are reset to 0 by the ADC control logic. Therefore, all the bottom plates of the capacitances V_{A+} to V_{D+} and V_{A-} to V_{D-} are clamped at V_1. Meanwhile, the input voltages are sampled on the top plates of the positive and negative cap-DAC as V_{IN} and V_{CM}. The coarse conversion phase begins immediately after S1 and S2 are switched off and the first bit $B8$ could be directly obtained by comparing V_{MSB+} with V_{MSB-} without switching the reference voltage MUX. Since V_{MSB+} is smaller than V_{MSB-}, $B8$ is 0. Then the ADC control logic would set $B8+$ to 1. This causes V_{A+} to jump from V_1 to V_2 (V_{A+} remains at V_1), which in turn elevates the V_{MSB+} by approximately legging half of ΔV_1 to make it approximate V_{MSB-}. After V_{MSB+} settles down, the second comparison could be executed and $B7$ is converted to be 1 in this case. Thus, the V_{B-} would switch to V_2, which increases the V_{MSB-} by approximately a quarter of V_1. The same comparison procedures are executed until $B4$ has been converted, which ends the coarse conversion phase. In this example, the first 5 bits $B8$ to $B4$ are 01011 and accordingly $B8+$, $B7-$, $B6+$ and $B5-$ are pulled up to 1. The fine conversion phase is responsible for the conversion of the lower 4 bits $B3$

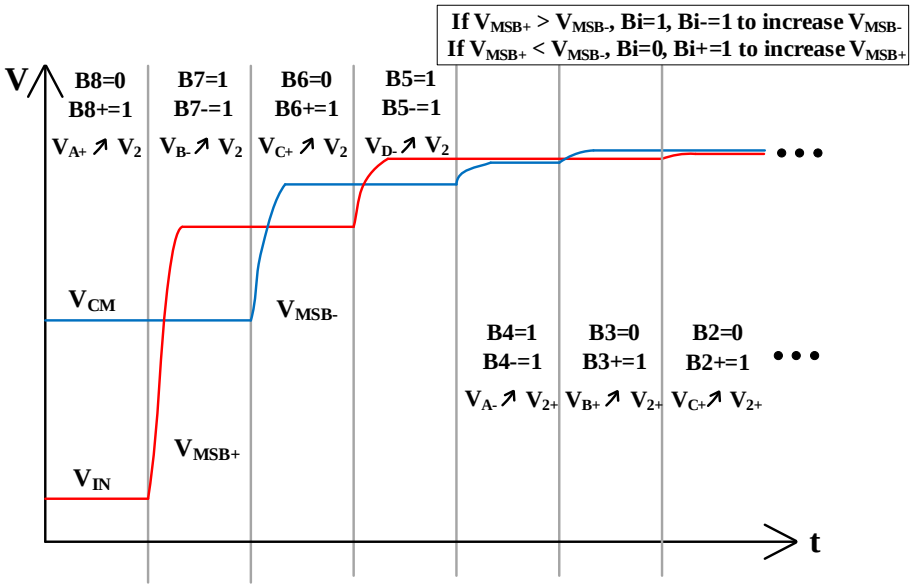


Figure 3.9 An example of ADC conversion.

to $B0$. The same cap-DAC arrays are used in this phase but with scaled reference voltages (V_1 to V_{1+} or V_2 to V_{2+}) to implement the binary scaled voltage step. Since $B4$ is 1, the $B4-$ would set to 1 while $B8-$ remains 0. Therefore, the MUX5 switches V_{A-} from V_1 to V_{1+}. By setting ΔV_2 equal to $1/16$ of ΔV_1, the V_{MSB-} would increase about $1/32$ of ΔV_1, which is consistent with the previous binary-scaled ratio. For the next bit, $B3$ is converted as 0 and thus $B3+$ is pulled up to 1. Then the MUX2 switches the V_{B+} from V_1 to V_{1+}. This same procedure is executed until the LSB $B0$ is converted, which indicates the completion of the AD conversion.

3.2.3 System Control and Handshaking Protocol

In order to achieve the address-event and capture the corresponding intensity information simultaneously, the original 4-phase handshaking scheme has been modified, as shown in Fig.3.10. The fired pixel first sends RR to the external row handshaking logic and after being selected by the row arbiter tree, it will receive RA. This would trigger the pixel to issue CR in the column direction. As for the column AER logic circuit in this case, it would not acknowledge this CR immediately (by giving out the corresponding CA). Instead, it would spare some time for the column readout circuit to quantize the logarithmic output voltage from the fired pixel. In order to implement this, an additional ADC control logic was incorporated at the top of the column arbiter tree. What's more, an extra control signal ADC_BUSY is introduced and

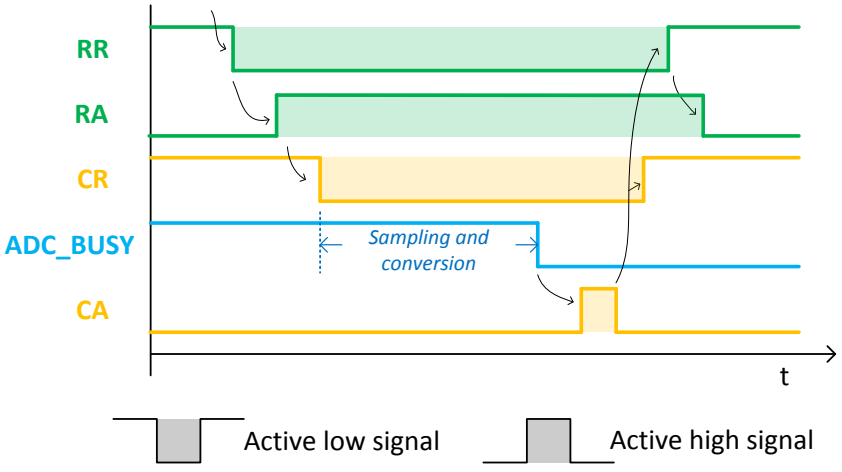


Figure 3.10 Handshaking protocol between blocks for the new DVS sensor.

first set to 1 to block the normal handshaking communication from CR to CA. The falling edge of CR will trigger the ADC control logic to generate a series of control signals for the column readout circuits to perform a complete A/D conversion of the logarithmic output voltages from the fired pixels. After that, ADC_BUSY would be pulled down to 0, which recovers the previously blocked acknowledgement to CR. The corresponding CA will be issued to process the fired pixel accordingly. Since the sensor is self-acknowledged, it would also send CA as a $VALID$ signal to the off-chip processor which would sample the address-event and the corresponding intensity information simultaneously when $VALID$ is high. In reality, it is noteworthy that the fired pixels are processed on the basis of an entire row. After RA is issued, all the fired pixels in this row would be processed in sequence. The column readout circuits also work in parallel to quantize the output voltages from all the fired pixels of this row at the same time. Since pixel output voltage responds to light intensity continuously, the integration procedure is no longer needed. The delay of this sensor to achieve the brightness information is only the time spent on the A/D conversion. Such procedure usually takes sub-microseconds, which brings about negligible effect to the final address-event latency (usually in the order of several μs).

3.3 Sensor Implementation and Measurement

The sensor was implemented using AMS 0.35 μm 2P4M CMOS process with an array of 384×320 pixels. The layout of the whole image sensor is shown in Fig.3.11 with a total chip area of $12.1 \times 13.2\ \text{mm}^2$. Thanks to the simple logarithmic intensity readout strategy, each pixel occupies $30 \times 30\ \mu\text{m}^2$ with 12% fill factor. The extra source follower and access switch only account for 5.5% of the total pixel area. The analog and digital parts of pixel are separated from each other to mitigate the coupling from the noisy digital circuits to the sensitive analog parts. Besides, the power metals are also paved above the analog circuits to shield the potential coupling from the above digital handshaking buses. The whole pixel array is surrounded by a dedicated guard-ring circuit with large decoupling capacitances. This not only isolates external noises but also provides low-resistive power rails (VDD and GND) for pixels. The column

readout circuit containing the SC amplifier, SAR ADC and AER handshaking logic was designed into a slice of 30×1020 $\mu m2$. The two-step monotonic switching SAR ADC occupies about 30×700 $\mu m2$ with 26 % area allocated to the cap-DAC. In addition, all the bias and reference voltages for the column slice are generated on chip. Specifically, four on-chip high-resolution DACs are adopted to generate the required reference-voltages for column-parallel SAR ADC array. They could be easily programmed by off-chip processor according to the standard Serial Peripheral Interface (SPI) protocol.

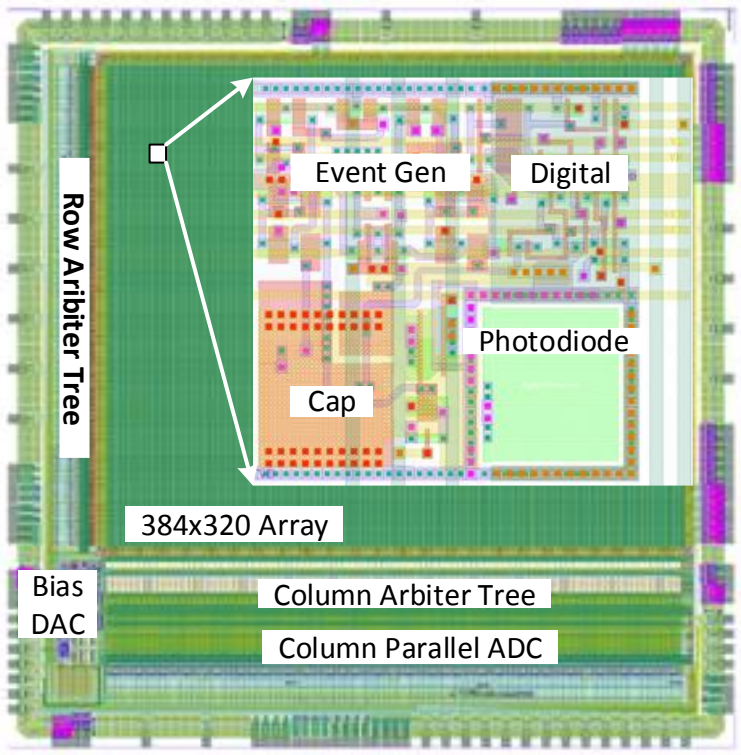


Figure 3.11 Layout of the motion sensor with asynchronous grayscale events.

An external global control signal, namely $forcefire$ was also added in this design to force the pixel to generate a request unconditionally through the AER handshaking logic circuits. It ensures the capability of on-demand full-picture acquisition of this sensor. When $forcefire$ is activated, all pixels of the sensor will be activated and processed in sequence, a full frame can thus be produced accordingly. In summary, this sensor could work in three different modes: the analog, digital and progressive mode. For the analog mode, the sensor works as the conventional frame-based image sensor with periodic $forcefire$ stimulus and can generate a maximum of 300 frames per second. This mode makes the sensor compatible with mainstream image processing algorithms. With respect to the digital mode, the sensor

works as a pure DVS sensor as it only reports the binary address information of events. The intensity readout path (column SC amplifier and SAR ADC), along with the reference and bias voltage generator, is disabled in this mode to reduce the power consumption of the whole system. Finally, the sensor could also be configured into the progressive mode during which the address and intensity information of the fired pixels are reported simultaneously. Since the intensity readout is only trigged by corresponding events ($forcefire$ disabled in this mode), the data volume that needs to be communicated is reduced to a great extent according to the dynamic range of the visual scene. Such event-driven intensity readout mechanism leads to lossless video compression on focal-plane, which enables the sensor to detect high-speed moving object with precise time information.

To measure the hardware performance of this motion sensor, a FPGA-based testing platform was developed. The testing platform contains an Opal Kelly FPGA board, which is configured to provide input control signals, temporarily store the data and communicate with a local host PC through USB link. The sensor is powered by 3.3V

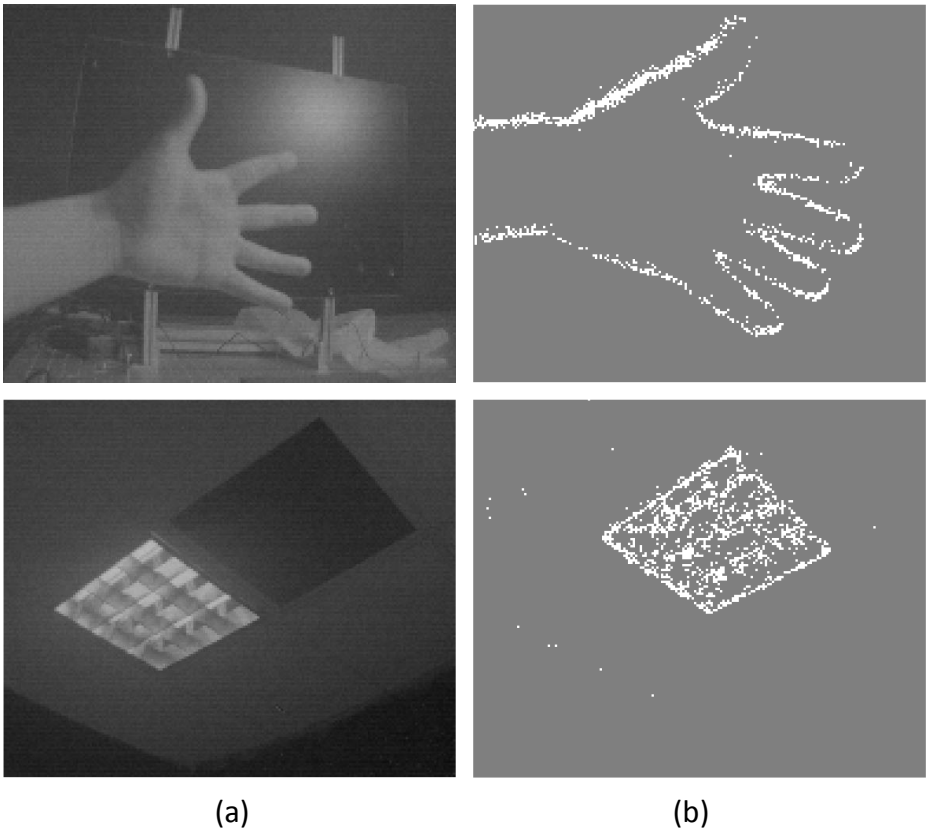


Figure 3.12 Sample images of the image sensor. (a) Full-frame images captured by forcefire signal, (b) reconstructed images using binary events within a 20ms period.

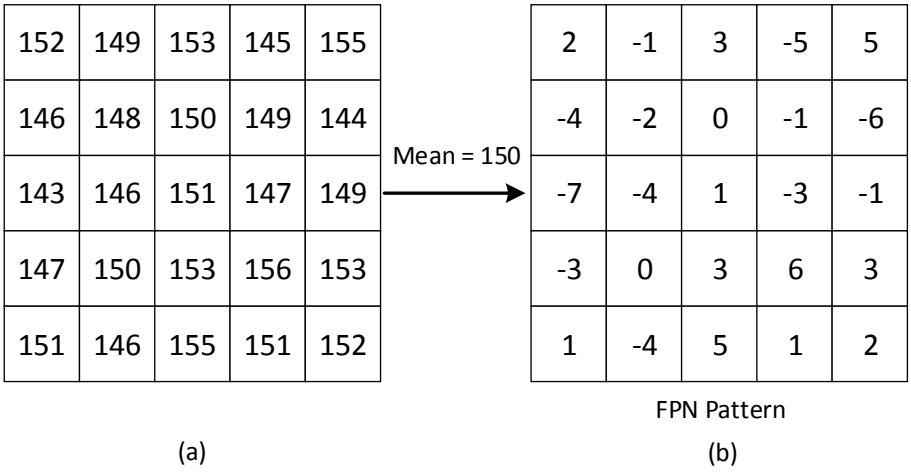


Figure 3.13 Illustration of FPN extraction. (a) Full-frame reference image capture under uniform light condition, (b) extracted FPN pattern by subtracting the mean pixel value from each pixel.

and clocked at 20MHz. Each of the received event packet consists of three elements: pixel address, timestamp and its illumination. The received events can describe the contour of a moving object. The computer is able to command the sensor to produce a full-frame picture, which is required by most computer vision algorithms. This is done by issuing a $forcefire$ signal, all the pixels on the sensor will be activated and readout one by one. Therefore, the motion sensor can operate in two modes, digital mode and full-frame mode. During the period of a full-frame readout, the pixels on the rows, which have been processed, will turn into digital mode at once. Fig.3.12 reports a couple of sample images.

The FPN of the sensor was characterized by taking a full-frame image under uniform light condition and working out the standard deviation of the output data from all pixels. A back-end correlated double sampling (CDS) method is used to remove FPN by firstly taking a full-frame reference image under uniform light condition. After that, the FPN information is extracted by removing the average value of the whole reference image from each pixel, as illustrated in Fig.3.13. With the extracted FPN information, the newly received pixel events first subtract the previously calculated FPN accordingly and then display. For FPN analysis, full-frame images were taken under 60 and 2klux light condition for FPN measurement. Multiple images were taken repeatedly and only the mean value of each pixel was used for FPN extraction. This is necessary to remove the adverse effect of temporal noise. The mean value of the data

is 303.5 LSB for 60 lux and 166.9 LSB for 2klux. The corresponding standard deviation (also FPN) is 17.5 and 17.9 LSB respectively. After applying the back-end FPN-cancellation, the standard deviation is significantly reduced to 1.6 LSB, only 9% of the previous 17.7 LSB (average of 17.5 and 17.9 LSB). For a tenfold range of the illumination, this corresponds to about 2.1 % of the useful signal range. As a result, the back-end CDS method is very effective in cancelling FPN for this design.

Based on the measurement methods illustrated in [23], some basic parameters with respect to the sensor in detecting temporal contrast were also determined. Firstly, the contrast sensitivity, which defines the minimum detectable contrast change, was characterized with a moving gradient bar, which would trigger the pixels to be fired continuously. The higher contrast sensitivity is desired in some applications to recognize more details of the moving objects at the expense of larger readout data bandwidth and power consumption. As for this sensor, the contrast sensitivity is recognized as 30% with around 150 mV threshold window. This sensitivity is lower than that of the ATIS (13%) and DAVIS (11%) because the gain of the in-pixel difference amplifier is set to only 7 to save the valuable pixel area. In addition, the minimum event latency and the illumination time resolution are also measured. By

TABLE V SPECIFICATIONS COMPARED WITH PRIOR DVS IMPLEMENTATIONS

	This work	Delbruck et al. [49]	Posch et al. [46]
Functionality	async. temporal contrast + async intensity + full frame	async. temporal contrast + APS	async. temporal contrast + level crossing intensity
Process Technology	AMS 0.35um 2P4M OPTO	0.18um 1P6M MIM CIS	0.18um 1P6M MIM
Array Size	384 × 320	240 × 180	304 × 240
Pixel Size μm^2	30 × 30	18.5 × 18.5	30 × 30
Fill Factor	12%	22%	20%, 10%
FPN	0.38%	0.5%	0.25%
Supply Voltage	3.3V	1.8/3.3V	3.3V analog, 1.8V digital
Power @100eps mW	70	145	175

means of the adopted continuous-time pixel intensity readout strategy combined with the column-parallel ADC array, the sensor is capable of reporting the address-event and its corresponding intensity information simultaneously. Therefore, the minimum event latency and illumination time resolution are the same for this sensor and measured to be 6 μs. Finally, the dynamic range of the sensor is also estimated as high as 120 dB since the sensor works well from very bright illumination (>10k lux) down to very dark condition (0.1 lux). The main parameters of the motion sensor are summarized and compared with DAVIS and ATIS in Table V.

3.4 Proposed Event-guided Structured Output Tracking Algorithm

As discussed in the literature review on the state-of-the-art tracking algorithms, it is still a big challenge when tracking high speed moving objects in terms of tracking accuracy and computational cost. Currently, most of the released tracking sequences in public benchmarks such as VOT2016 [148], UVA123 [149], OTB100 [59] and ALOV300 [102] are captured at 30fps, which results in motion blur for high speed moving objects. Traditional object tracking is implemented based on the concept of 'frames'. The tracking accuracy has achieved a fairly high level (60%) [148] for objects moving at normal speed. However, the accuracy drops to around 30% when tracking high speed moving objects [150]. This is expected because motion blur is introduced when capturing fast motion under standard frame rate. Moreover, the searching area in traditional tracking methods is commonly set to be a constant, which could be too large for slow motion or too small for fast motion. Accuracy of tracking decreases if objects' displacement distance outreaches the search area and thus it becomes much more difficult to track fast moving objects. Nevertheless, such problems can be solved by increasing the capture rate. Galoogahi, HK, et al. [150] have released a new benchmark containing sequences captured at $240fps$. The time interval between two consecutive

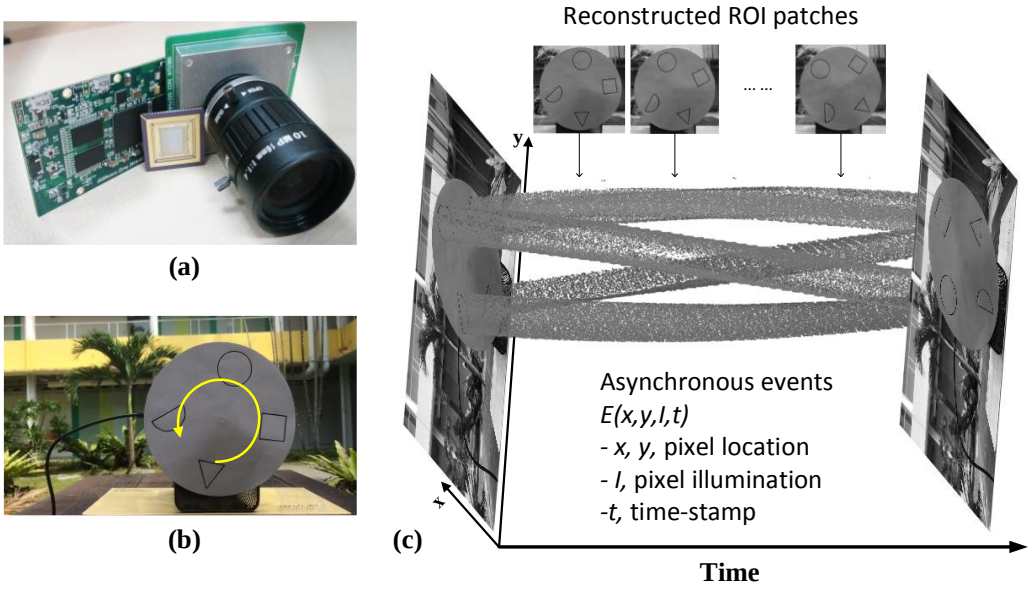


Figure 3.14 Overview of the designed motion sensor. (a) Designed motion sensor with 384×320 pixels. (b) Scenario of a rotating plate with four different geometrical patterns. (c) The spatial-temporal spacing of events generated in response to the rotating plate: pixels independently generate asynchronous event packages. Static background can be captured at first and small patches of region-of-interest (ROI) can be reconstructed by updating the intensities of dynamic events on the pre-captured background.

frames is approximately $4ms$, which decomposes a large displacement into small ones. In this way, the tracking accuracy of most trackers are improved by at least 50% and the state-of-the-art trackers are able to achieve a success rate of 60% [150]. Nonetheless, the accuracy improves at an expense of large computational cost. Compared with $30fps$ sequences, the input data of $240fps$ sequences are eight times larger. For sub-step tasks like sample searching and feature extraction, series of computationally demanding operations need to be performed on each acquired frame. Therefore, tracking on high-frame rate sequences increases the computational cost and thus makes it a challenge to implement real-time tracking.

Considering the characteristics of the designed dynamic motion sensor, as shown in Fig.3.14, an event-guided structured output tracking method is proposed taking use of the new motion sensor with asynchronous pixel intensity extraction, as elaborated in the previous part. The motion sensor is not limited by the concept of 'frame'. When a pixel senses an intensity variation that is larger than threshold, it becomes active and outputs the pixel intensity together with the encoded pixel address. Meanwhile, a

timestamp is added into the event package by FPGA. Therefore, each pixel package contains three information, pixel address, pixel intensity and a timestamp.

The idea of this event-guided tracking algorithm is making use of the characteristics of the motion sensor to provide useful information about fast moving objects and to guide a frame-based tracking algorithm to achieve better accuracy and real-time performance. Taking use of the pixel location in event packages, we first propose an Event Position-guided Search Localization (EPSL) method to guide the moving object localization. The EPSL is able to provide adaptive search area and guide potential motion direction, such that it increases the searching accuracy and reduces the computational cost. Furthermore, using the intensity information in event packages, an Event Intensity-guided Sample Supplement (EISS) method is proposed to supplement more effective samples for online tracker learning and updating.

The motivation behind the algorithm is to achieve real-time tracking of fast moving objects with high accuracy while maintaining the computational cost at a low level with the help of a functional motion sensor. To prove that the motion sensor has outstanding advantages in reducing redundancy and capturing fast motion, a classical frame-based tracking method, Structured Support Vector Machine (SSVM), is chosen as the main tracking algorithm. Among classical tracking methods, SSVM has a significant good performance on normal speed moving objects according to Wu's research [58]. However, the computational cost of SSVM is much larger than the current popular correlation-filter based algorithms. The experiments will show that by incorporating asynchronous and high temporal output events of motion sensor with low frame rate full-frame images, the event-guided SSVM (ESSVM) can achieve the best trade-off between accuracy and real-time performance while outperforming state-of-the-art trackers. In the following section, the traditional SSVM algorithm is first elaborated followed by the details of the proposed event-guided tracking method, containing EPSL and EISS.

3.4.1 About *Struck* Tracking Algorithm

The tracking algorithm *Struck* was first proposed in 2011 [1] and extended in 2016 [151] to achieve better performance. The tracker is based on a kernelized structured support vector machine (SSVM) which generalizes the SVM classifier. Traditional adaptive tracking-by-detection algorithms regard tracking problem as a classification task to distinguish a target object from its background and use online learning scheme to repeatedly update the object model. To realize the model updating, the estimated object positions must be converted into a set of labelled training examples and the labels are always binary or discretely distributed such that the labelled predictions are not explicitly coupled to the accurate estimation of object position. Therefore, the *Struck* tracker was proposed to allow the output space expressing the estimation of target object more explicitly by measuring a score function.

In the traditional adaptive tracking-by-detection algorithms, a classifier is learned online to distinguish a target object from its background. During tracking, the classifier is used to estimate object position by searching for the maximum classification score within a local region around the estimated position in the previous frame. The searching process is typically conducted using a sliding window. Given the estimated object position, a set of binary labelled training samples are generated by traditional classifiers and used for model updating. Therefore, the tracking process is divided into two parts, one is the generation and labelling of samples and the other one is classifier model updating. By analysis, the objective of the classifier, which is to correctly label the samples is not explicitly coupled with the objective of the tracker, which is to precisely estimate the target object location. The strategy for generating and labelling samples should always be carefully designed according to different applications to decide whether a sample should be labelled positive or negative and the binary labelling scheme is too simple to precisely estimate the object location since it provides the same weightings for similar samples, positive or negative, depending only on the predefined labelling strategy without considering other constraints such as transformations.

The structured output scheme is a qualified solution to the issues caused by binary labelling. The propose of the structured output SVM tracking framework was based on

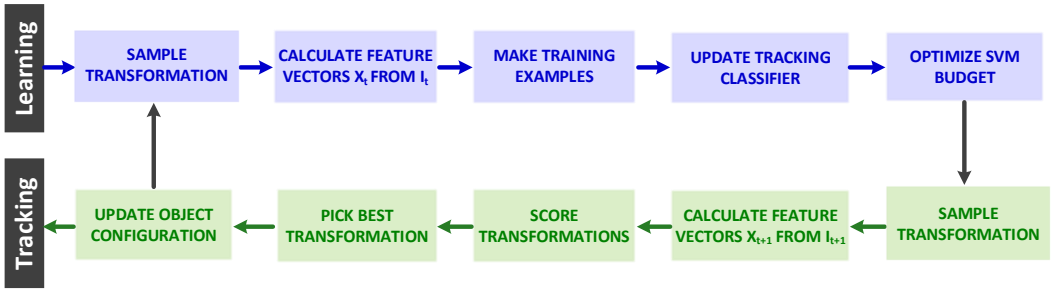


Figure 3.15 Structure of the tracking algorithm based on structured support vector machine (SSVM).

the high performance of SVM in object detection research, which shows good generalization, robustness to label noise, and flexibility in object representation by using kernels. Compared to traditional tracking-by-detection methods, one of the main contributions of the structured output tracking approach is to avoid offline data labelling for training. This is realized by applying a structured output SVM that learns a prediction function to directly predict the object transformation between frames instead of learning a classifier with binary labels. Another contribution is to apply a fixed support vector budget for online learning of classification SVMs such that it ensures computational efficiency since the number of support vectors will not grow to a tremendous amount during the kernelized online learning process. The typical structure of the structured output tracking algorithm is shown in Fig.3.15.

The objective for the tracker is to estimate a transformation (e.g. translation, rotation) $y_t \in Y$ such that the new position of the object is approximated by the transformation $P_t = P_{t-1} \circ y_t$. Y denotes our search space and its form depends on the type of transformation to be tracked. For most tracking-by-detection approaches this transformation is a 2D translation, in which case $Y = (u, v) \mid (u^2 + v^2 < r^2)$, where r is a search radius. During tracking, it is assumed that a change in position of the target can be estimated by maximizing the scoring function $F(X_t^{P_t})$, which represents the similarity between candidate patches on the feature map X of frame t, $X_t^{P_{t-1} \circ y}$, and the target tracking object in position P_{t-1} in frame $t-1$, $X_{t-1}^{P_{t-1}}$. The *Struck* directly estimates the transformation y_t between consecutive images by a prediction function f, which is based on the scoring function F. In this way, the transformation y_t from P_{t-1} to P_t can be predicted according to:

$$y_t = f\left(X_t^{P_{t-1}}\right) = \arg\max_{y \in Y} F(X_t^{P_{t-1}} \circ y) \quad (3\text{-}8)$$

The SSVM solver [151, 152] has been widely used in tracking for its outstanding performance. However, two significant limitations exist in the transformation generation and sampling. Firstly, given the estimated object position p, the searching area Y is generally set to be a rectangular region centered at location p with a constant radius which is fixed for each test sequence and commonly much larger than the size of the target object. It thus results in unnecessary computational cost and inaccurate tracking results. Secondly, the blind time between subsequent frames results in insufficient training sample updating because it losses the object feature between X_{t-1} and X_t which could be variable and significant especially in tracking fast moving objects. This is the reason why trackers show better performance on video sequences with higher framerate, e.g. 240fps [150].

3.4.2 Event-guided Structured output tracking (ESSVM)

3.4.2.1 Overview

The proposed ESSVM algorithm follows the main structure of the SSVM tracker, but with two event-based guiding methods, as demonstrated in Fig.3.15. Firstly, to adaptively guide the search space Y of moving object transformation, the EPSL model is proposed to generate a guided search space Y_G taking use of the location information of the grayscale events. Traditional frame-based sensors cannot capture the object motion between two consecutive frames, while the dynamic detection feature of our motion sensor enables recording of full object motion trajectory by continuously generating events in temporal domain, as shown in Fig.3.13. For events $E_{[t-1,t]}$ generated during time interval from $t-1$ to t, their location set is $L_{[t-1,t]}$. Different from the traditional SSVM tracker which uses fixed search radius around the object location P_{t-1} in previous frame, the proposed algorithm uses an adaptive search region based on three location regions: the previous object location P_{t-1}, the predicted location P_t^{Pre} in frame t calculated by the traditional SSVM, and the event location region $L_{[t-1,t]}$. The final search space Y_G is calculated as:

$$Y_G = \{P_{t-1} \cup P_t^{Pre} \cup L_{[t-1,t]}\} \quad (3\text{-}9)$$

Secondly, for handling the fast representation updating of moving object and acquiring more up to date positive and negative supporting vectors from training samples, the EISS model is proposed to generate complementary moving object samples taking use of the intensity information of the grayscale events. As shown in Fig.3.14, the event packages generated by the motion sensor contain not only pixel locations but also intensities. By exploiting the motion sensor's high temporal resolution, one can acquire an intermediate frame at a temporal precision Δt as accurate as $50ns$. This means any intermediate frames t' can be reconstructed between frame $t-1$ and t using the event intensities to acquire a higher sampling rate because events are continuous with $50ns$ accuracy in temporal domain. Once the temporal precision

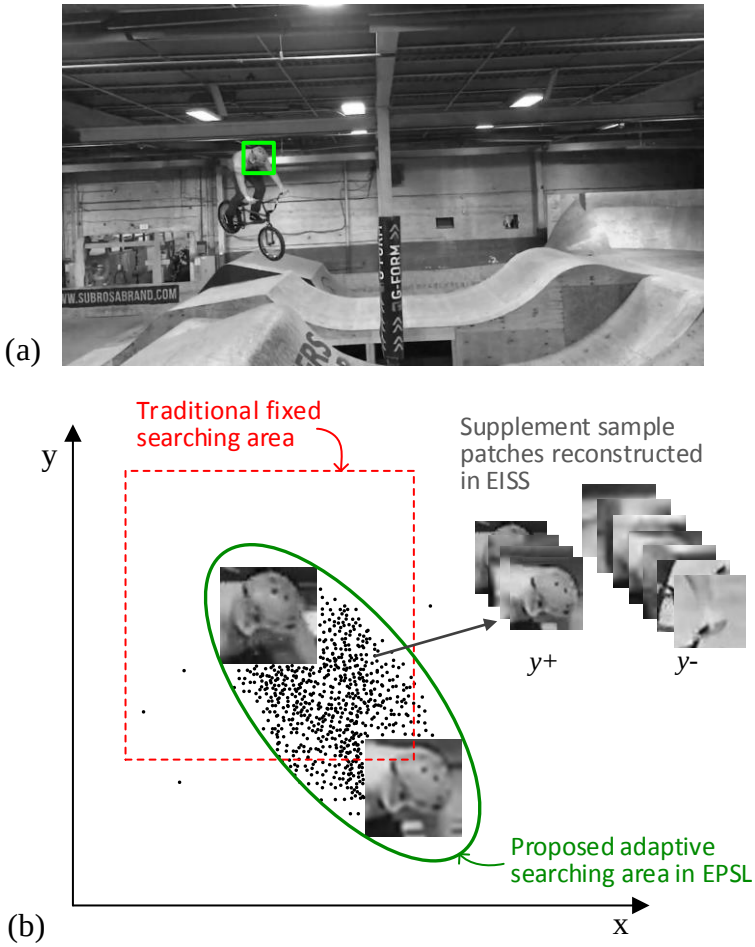


Figure 3.16 Demonstration of the proposed tracking method using ESSVM. (a) Sample image from *biker_head* sequence with the bonding box of ROI, (b) illustration of a fast-moving object tracking process, including demonstrations of EPSI and EISS.

Algorithm 1

Input: P_{t-1}, object location in previous frame $t-1$
$(X_{t-1}, y) \in S_{t-1}$, samples within the maintained set of support vectors after processing frame $t-1$
X_t, training sample in current frame t

Event Position-guided Search Localization (EPSL)

1. $y_t^{Pri} = \underset{y \in Y}{\operatorname{argmax}} F(X_t^{P_{t-1}}, y)$
2. $P_t^{Pri} = P_{t-1} \circ y_t^{Pri}$
3. $Y_G = \{P_{t-1} \cup P_t^{Pre} \cup L_{[t-1,t]}\}$

Event Intensity-guided Sample Supplement

4. for $j = 1$ to $\frac{t-(t-1)}{\Delta t}$
5. $t' = t - 1 + \Delta t \cdot j$
6. $y_{t'} = \underset{y \in Y_G}{\operatorname{argmax}} F(X_{t'}^{P_{t-1}}, y)$
7. $P_{t'} = P_{t-1} \circ y_{t'}$
8. $(t, y_+, y_-) \leftarrow GSS(X_{t'}^{P_{t'}}, y_{t'})$
9. if $(t' = t)$ $P_t^{final} = P_{t'}$
10. end for
11. Optimize ()
12. Return P_t^{fin}, S_t

Δt has been set, $(\mathrm{T_t} - \mathrm{T_{t-1}})/\Delta \mathrm{t}$ frames could be reconstructed and injected into the original frame sequence according to the required temporal precision.

These reconstructed frames can be used to generate supplement sample patches and update the tracker model by selecting more up to date positive and negative supporting vectors with respect to the appearance variation of the tracked moving object over time from T_{t-1} to T_t. In general, the supporting vectors $y+$ and $y-$ are selected by the following algorithm from all the feature vectors $X_{t'}$, which is the feature vector calculated from frame $I_{t'}$.

$$y+ = \underset{y \in Y}{\arg\min}\{\Delta(y, \bar{y}) + F(X_{t'}^{P_{t-1} \circ y})\} \tag{3-10}$$

$$y- = \underset{y \in Y}{\arg\max}\{\Delta(y, \bar{y}) + F(X_{t'}^{P_{t-1} \circ y})\} \tag{3-11}$$

Here, $\Delta(\mathrm{y}, \bar{y}) = 1 - S_{P_t}^{o}(y, \bar{y})$ is the loss function regulated to $[0, 1]$ for penalizing the position shift from the real object position $p^{\bar{y}}$ to the sample position p^{y}. $S_{P_t}^{o}(y, \bar{y})$ is the transformation similarity function with overlap measurement, which measures the degree of overlap between two bounding boxes and is defined by

$$S_{P_t}^{o}(y, \bar{y}) = \frac{y(p) \cap \bar{y}(p)}{y(p) \cup \bar{y}(p)} \tag{3-12}$$

The loss function $\Delta(\mathrm{y}, \bar{y})$ will be set to 0 if the two positions are totally the same; and set to 1 if there is no overlap between the two positions. $F(X_t, y)$ is the scoring function used in Eq. (3-8), which calculates the score related to the similarity between the candidate sample $X_{t'}^{P_{t-1} \circ y}$ and the tracked sample in the previous frame $t-1$. Combining both EPSL and EISS, the proposed method ESVM, calculates the ultimate tracking result P_t^{fin} based on the proposed adaptive search space and supplement support vectors. The detailed event-base guiding methods of ESSVM is shown as Algorithm 1:

3.4.2.2 Primal SVM Formulation

The target of ESSVM is to learn a prediction function $f: T \rightarrow Y$, which can directly estimate the object transformation between subsequent frames and thus object location in each frame. $f\left(X_t^{P_{t-1}}\right)$ is the required transformation y_t from p_{t-1} to p_t and $X_t^{P_{t-1}}$ is the set of features from the bonding box in frame t at position P_{t-1}, which is the object location in the previous frame $t-1$. The prediction function f is learned by making use of a scoring function $F: T \times Y \rightarrow \mathbb{R}$ which measures the compatibility of the features extracted from two locations after transformation. The f function can be generalized with an expression as

$$y_t = f(X_t) = \arg\max_{y \in Y} F(X_t, \boldsymbol{y}) \tag{3-13}$$

To formulate the SVM, the scoring function is first restricted to be a linear function $F(X_t, y) = \langle w, \Phi(X_t, y) \rangle$, where w is the weight vector in SVM and $\Phi(X_t, y)$ is the joint kernel map which maps the input data (X_t, y) in a lower dimension space into a

suitable feature space with higher dimension and endowed with the dot product $\langle\cdot,\cdot\rangle$. The kernel mapping function Φ is explicitly defined as

$$K(x, y, \bar{x}, \bar{y}) = \langle \Phi(x, y), \Phi(\bar{x}, \bar{y}) \rangle \tag{3-14}$$

where $x_1, \dots, x_n \in X$ are training features and $y_1, \dots, y_n \in Y$ are their corresponding transformations. The use of kernel mapping is to solve the non-linearly separable issues. The score function F can be learned by solving the quadratic program

$$\min_{w} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \tag{3-15}$$

$$s.t. \forall i = 1 \dots n : \ \xi_i \geq 0$$

$$\forall i, \forall y \neq y_i : \langle w, \delta\Phi_i(y_i, y) \rangle \geq \Delta(y, \bar{y}) - \xi_i$$

where $\delta\Phi_i(y_i, y) = \delta\Phi(y_i) - \delta\Phi(y)$ and the constant number C is set to 100. The quadratic problem is similar to the SVM classification problem, and can be solved using its equivalent dual form, which will be explained in the next section. The target of the optimization is to ensure that when $y \neq y_i$, the score function value for the i^{th} training sample $F(X_i, y_i)$, extracted from either full frames from frame cameras or reconstructed frames from the motion sensor, must be larger than $F(X_i, y)$ by a margin $\Delta(y, \bar{y}) - \xi_i$, where $\Delta(y, \bar{y})$ is a loss function defined based on the transformation similarity function (TSF) and is used to provide non-binary weightings for each training sample. There are multiple different ways to define the TSF such as bonding box overlap measurement and centre distance measurement. In this algorithm, the overlap measurement function $S^o_{P_t}$, see Eq. (3-12), is applied as suggested [151,168] and the loss function is defined as

$$\Delta(\mathrm{y}, \bar{y}) = 1 - S^o_{P_t}(y, \bar{y}) \tag{3-16}$$

The application of the proposed adaptive searching area (EPSL) restricts the traditional searching area to a much smaller region, and thus, improves the efficiency of sample generation and reduces the computational cost of object location estimation. This is realized because of the characteristics of the motion sensor which continuously reports the moving object locations. Therefore, during a short time period within the blind time interval between consecutive frames captured by frame-based cameras, the

region where events appeared around the object location in previous frame indicates the direction of the target motion.

3.4.2.3 Dual SVM Formulation

The primal quadratic problem in Eq. (3-15). can be solved using standard Lagrangian duality techniques and can be converted into the equivalent dual form

$$\max_{\alpha} \sum_{i,y\neq y_i} \Delta(y,y_i)\alpha_i^y - \frac{1}{2}\sum_{\substack{i,y\neq y_i \\ j,y\neq y_j}} \alpha_i^y \alpha_i^{\bar{y}} \langle \delta\Phi_i(y_i,y), \delta\Phi_j(y_j,y)\rangle \quad (3\text{-}17)$$

$$s.t. \forall i, \forall y \neq y_i : \alpha_i^y \geq 0$$

$$\forall i : \alpha_i^y \leq C$$

And the scoring function is expressed as

$$F(X_t,y) = \sum_{i,y\neq y_i} \alpha_i^{\bar{y}} \langle \delta\Phi_i(y_i,y), \Phi(X_t,y)\rangle \quad (3\text{-}18)$$

One of the main benefits of the dual representation is to make it quite easy to apply kernel tricks since the primal equations and the equations with kernel mapping have almost the same dual formulation except that the problem is solved by computing the kernel function instead of the ordinary dot-product.

This dual problem can be considerably simplified according to [153] by reparametrizing it with nk variables β_i^y and defined as

$$\beta_i^y = \begin{cases} -\alpha_i^y & if\ y \neq y_i \\ \sum_{\bar{y}\neq y_i} \alpha_i^{\bar{y}} & otherwise \end{cases} \quad (3\text{-}19)$$

Note that $\beta_i^{y_i}$ are positive and $\sum_y \beta_i^y = 0$. The above dual equation can thus be simplified to a much simpler expression as

$$\max_{\beta} -\sum_{i,y} \Delta(y,y_i)\beta_i^y - \frac{1}{2}\sum_{i,y,j,\bar{y}} \beta_i^y \beta_j^{\bar{y}} K(X_{t_i},y,X_{t_j},\bar{y}) \quad (3\text{-}20)$$

$$\text{s.t.}\ \forall \text{i}, \forall \text{y}: \ \beta_i^y \leq \delta(y,y_i)C$$

$$\forall \text{i}: \sum_y \beta_i^y = 0$$

where $\delta(y, y_i) = 1$ if $y = y_i$ and 0 otherwise. The discriminant function is then simplified to

$$F(X_t, y) = \sum_{i,\bar{y}} \beta_i^{\bar{y}} \langle \Phi(X_{t_i}, \bar{y}), \Phi(X_t, y) \rangle \quad (3\text{-}21)$$

In this form, pairs $(X_{t_i}, \bar{y})$ whose parameter $\beta_i^{\bar{y}} \neq 0$ are referred to as support vectors and X_{t_i} included in one or more support vectors are referred as support patterns. For a given support pattern X_{t_m}, only the support vector (X_{t_m}, y_m) will have $\beta_m^{y_m} > 0$ and is referred as positive support vectors, while any other support vectors $(X_{t_m}, \bar{y})$ where $\bar{y} \neq y_m$ will have $\beta_m^{\bar{y}} < 0$ are referred as negative support vectors. With the derived positive and negative support vectors, the sequential minimal optimization (SMO) approach introduced by Platt [154] is used to update the SVM such that the ESSVM always maintain a budget with 100 updated support vectors (the number of the stored support vectors keeps increasing over the first few frames until it reaches the limit of the budget).

3.2.3.5 Other Considerations

At an implementation level, besides the adaptive search space, there are some other considerations including the specific choices for kernel functions and image features. As discussed in the literature review, these two parts directly influence the tracker performance. Although taking use of the linear kernels can provide efficient computation in feature vector estimation, it shows poor performance handling the input data in a non-linearly separable feature space, which is common in tracking tasks. In addition, linear kernels have limit ability to incorporate the various appearance of the moving object. Therefore, the Gaussian kernel (e.g. RBF kernel) is chosen in ESSVM to map the feature vectors into a high-dimensional space where the linearly separability could be easier to achieve. Moreover, the types of image features used in this algorithm contain raw intensity, Haar features, and histogram features as reviewed in chapter 2.2. The sample patches used for object location prediction are extracted from both full-frame images at low frame rate and the supplemental images reconstructed from the dynamic neuromorphic motion sensor. The raw intensity features are obtained by scaling an image patch to 16×16 pixels, whose grayscale intensities are first

normalized to [0,1] and then scanned in raster order. The derived results are converted to a 256-D vectors and stored as raw features. The Haar features are extracted by using 9 different types of feature templates arranged at 5 scales on a 4×4 grid, resulting in 750-D feature vectors with each vector normalized to [0,1]. The Histogram features are 16-bin intensity histograms extracted from a spatial pyramid of 4 levels, which results in 480-D feature vectors. In the experiments, it was found that for some sequences, the combination of two or three of the above features would give a better result than single type feature application. However, the results were reverse for some other sequences in terms of tracking accuracy and speed. Therefore, the feature bench for each testing sequence is selected accordingly.

3.5 Experiments and Results

3.5.1 Experiment Setup

The tracking experiments are conducted on the original image sequences from public tracking benchmarks combined with the frame/event data recording by the novel designed motion sensor. The setup of generating recordings, illustrated in Fig.3.17, consists of a LED monitor, the motion sensor and a PC for sensor control and data communication. The workstation is placed in a darkroom such that during the recording process the motion sensor will only detect light from the specified LED monitor. In this experiment, the NFS benchmark which contains testing sequences with high frame rate (240fps) is used to generate event-based sequences since the images with higher frame rate are more compatible with the dynamic characteristic of the neuromorphic motion sensor in terms of generating event-based simulation data. For each testing scenario, the NFS contains three image sequences, which are captured at high framerate (240fps), low frame rate (30fps) without motion blur in which the frames are extracted by taking one frame from every eight frames of the corresponding 240fps sequence, and low frame rate (30fps) with motion blur which is captured by a frame-based camera with 30fps. To generate event sequences, the original 240fps video sequences were displayed using a consumer-grade DELL U2414H LED monitor with a refresh rate of

60Hz and the native resolution of 1920×1080. To implement tracking on NFS sequences at 30fps, the events generated by every seven frames in eight of 240fps sequences are used to guide tracking and reconstruct patches with gray intensities for sample supplement in ESSVM.

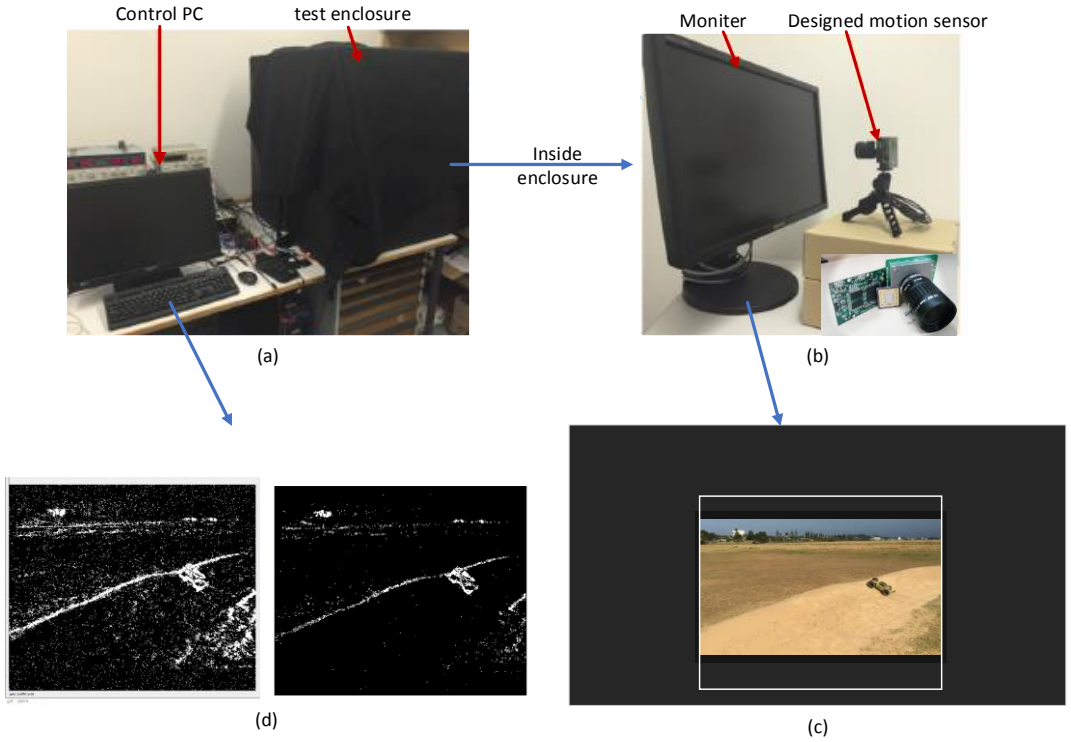


Figure 3.17 (a) Workstation for event generation, (b) a monitor is used for displaying images that are observed by the designed motion sensor, (c) sample observed frame on the display monitor, (d) events are collected and reconstructed to frames, left image uses raw events while the right image shows the reconstructed frame after temporal noise filtering.

3.5.2 Experiments on benchmark sequences

The experiments are designed to validate that the proposed ESSVM algorithm can remarkably improve the comprehensive performance of tracking accuracy and efficiency with respect to fast moving objects. We use three test sequences where two of them, *biker_head* and *car_rc_rolling*, are high frame rate sequences taken from NFS benchmark [150] with fast motion label and *bolt*, is from OTB [59]. We compare the tracking results of ESSVM with those of nine state-of-the-art baseline trackers, i.e. HDT [155], SiameseFc [156], Staple [157], FCNT [158], HCF [159], KCF [160], LCT [161], MEEM [162] and DSST [163]. Additionally, in order to research the

effectiveness of EPSL and EISS models in the proposed ESVM, we also conducted experiments using SSVM without event guidance (SSVM in Table VI). The method SSVM with only adaptive search space EPSL is referred to as ESSVM − and SSVM with both EPSL and EISS is referred to as ESSVM. Event packages for bolt are from DVS tracking benchmark [120], while the event packages for NFS sequences *biker_head* and *car_rc_rolling* are generated using the method mentioned above. It is noted that the framerate of *bolt* is originally 30fps such that events are generated only by the 30fps frames and no event package can be generated between two consecutive frames. As a result, the adaptive search area (EPSL) in current frame can still be guided by the event location corresponding to the current frame, however, the supplement patches EISS cannot be applied due to insufficient grayscale events. Therefore, there is only ESVM − result for *bolt* sequence. Taking the same measurement methods in [25-27], the tracking performance of the event-guided algorithm is measured using location error and processing speed. The results of average location error for both event-guided algorithms and baseline trackers are summarized and compared in Table VI.

TABLE VI FLOW ESTIMATION RESULTS

Performance comparison on two tracking scenarios including higher framerate tracking (240fps) and lower framerate tracking (30fps). Results are reported as the average location error (ALE).

	biker_head		*car_rc_rolling*		*bolt*
	30fps	**240fps**	**30fps**	**240fps**	**30fps**
HDT	422.8563	25.9101	135.5448	66.9254	5.0181
SiameseFc	431.3484	20.0044	136.4567	67.3311	125.17
Staple	*3.7215*	3.8445	351.9668	143.129	*4.136*
FCNT	24.9628	18.1085	136.0955	67.6556	7.2892
HCF	5.2031	3.9494	108.5254	*20.2911*	6.8574
KCF	500.3417	4.9028	397.3422	375.525	6.7622
LCT	357.4497	4.9020	377.4301	103.915	5.2233
MEEM	4.2633	*3.4865*	266.3398	102.105	15.5379
DSST	62.0206	3.5718	307.9312	119.366	4.5785
ESSVM	**3.8400**	**3.8559**	***83.3213***	**80.9876**	-
ESSVM −	4.3060	-	104.9961	-	**5.4555**
SSVM	428.5034	-	320.9378	-	370.206

Table VI lists the tracking results of object location precision for the proposed ESSVM tracker and the baseline trackers mentioned in NFS benchmark. For the

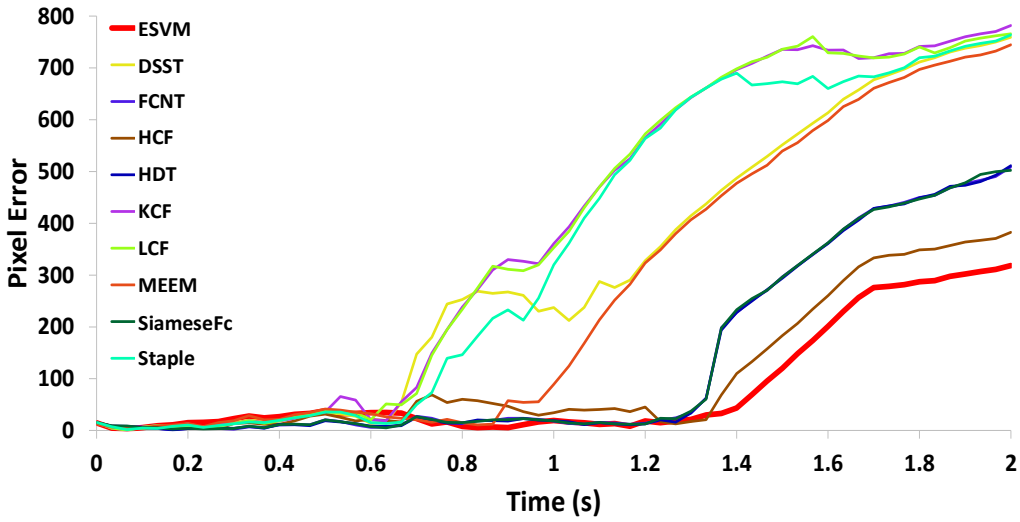


Figure 3.18 (a) Tracking errors for both ESSVM and state-of-the-art algorithms on car_rc_rolling sequence at 30fps.

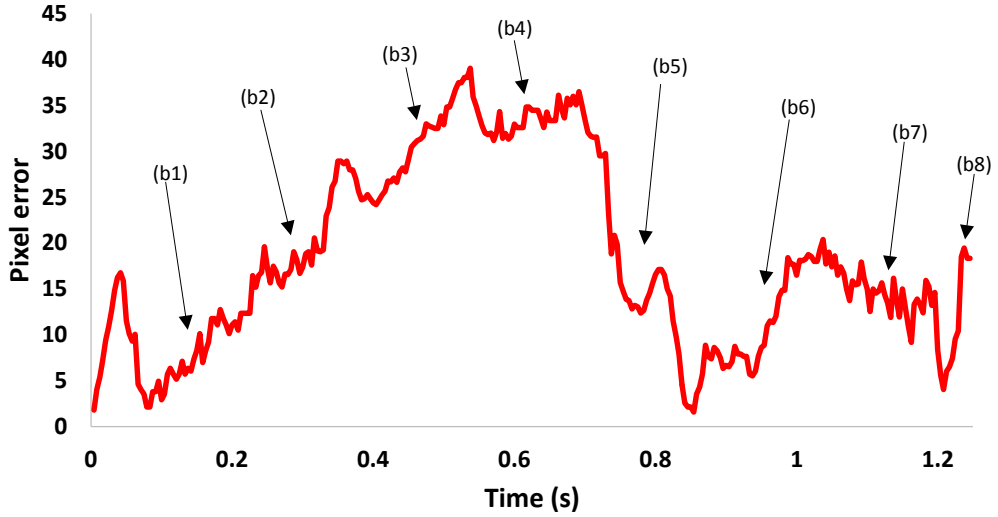


Figure 3.18 (b) Tracking errors of ESSVM for the first 1.2s.

car_rc_rolling 30fps sequence, ESSVM ranks the first with respect to location accuracy followed by ESSVM – method. The average location error (ALE) using ESSVM is 83.321 which is 25 pixels smaller than the best baseline tracker HCF, whose ALE is 108.525. For the *biker_head* 30fps sequence, ESSVM performs better than most baseline trackers and is comparable to the first with a minor difference of 0.12 pixel. With the help of EISS, ESSVM is capable of predicting locations for 240fps, which is used as reference. By comparing the ALE of trackers on 30fps and 240fps sequences,

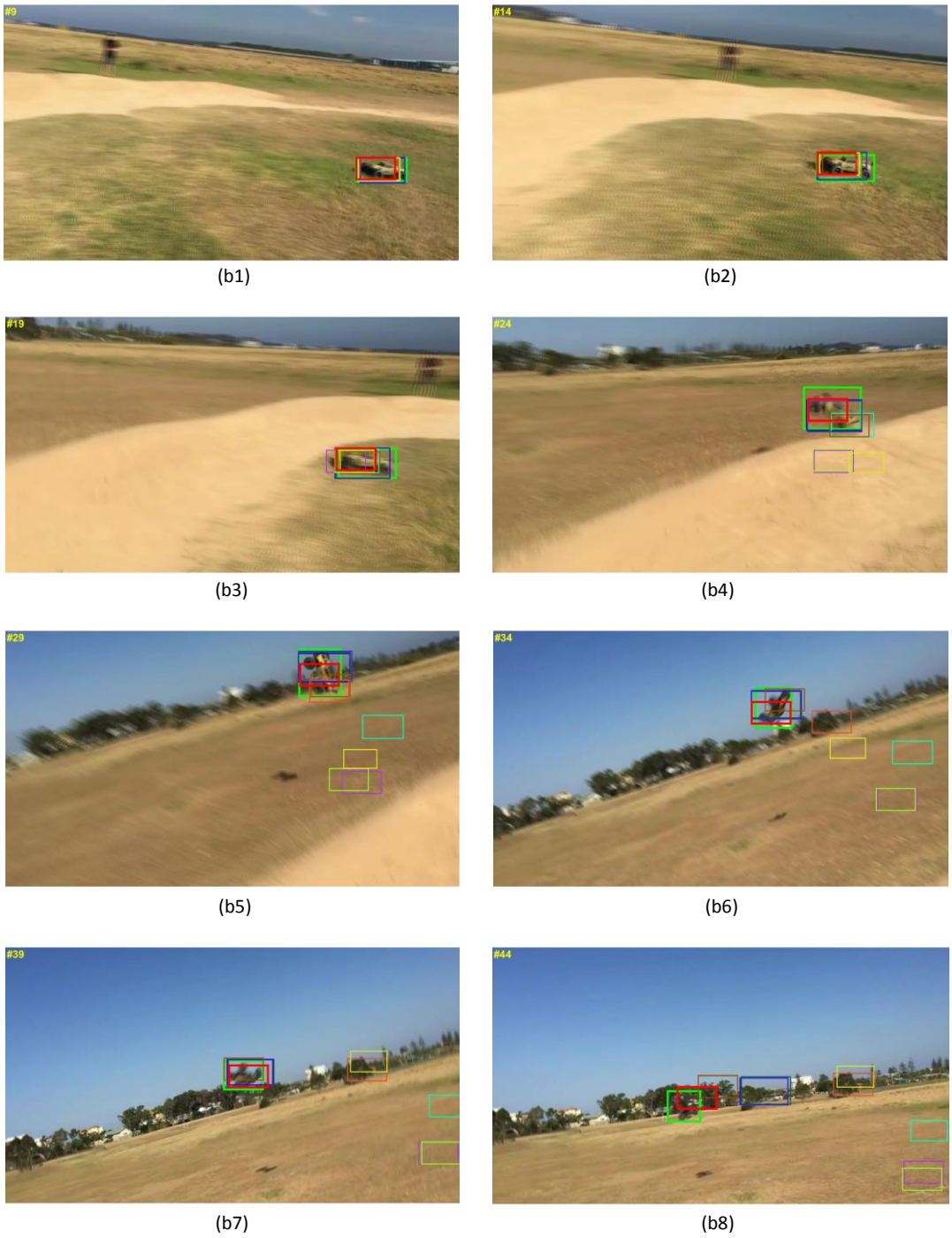


Figure 3.18 Tracking performance of car_rc_rolling sequence. (a) Tracking errors for both ESSVM and state-of-the-art algorithms on car_rc_rolling sequence at 30fps. (b) Tracking errors of ESSVM for the first 1.2s and (b1-b8) are sample frames showing tracking performance of all listed trackers, the colors of bonding boxes are consistent with the legend in (a).

the accuracy of ESSVM tracker on 30fps is as good as other trackers' performance on 240fps. Therefore, the proposed tracking algorithm ESSVM, which employs event-based motion sensor to guide object tracking in low frame rate videos, demonstrates

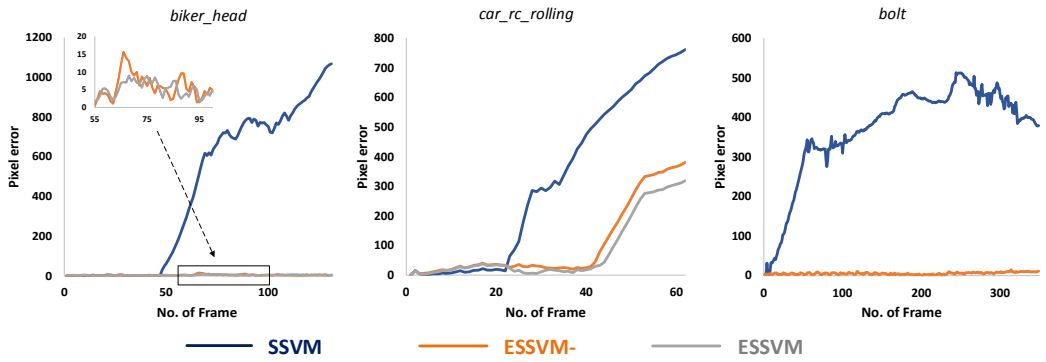


Figure 3.19 Evaluation of event-guiding methods by comparing the ALE for SSVM, ESSVM- and ESSVM on the testing sequences.

state-of-the-art performance. Moreover, the results show that ESSVM – remarkably improves the tracking accuracy compared with traditional SSVM while ESSVM performs even better. The location precision plot of the *car_rc_rolling* sequence is demonstrated in Fig.3.18(a). The proposed ESVM tracker outperforms all the other trackers. Fig.3.18(b) shows the location error on each frame using ESSVM and Fig.3.18(b1-b8) are eight sample frames with bonding boxes of all listed trackers. The accuracy results have shown that the proposed event-guided tracking algorithm outperforms most of the current advanced trackers with respect to tracking fast moving objects. It is observed that all the trackers lost tracking of the target rolling car. The result is caused by two reasons. Firstly, the car_rc_rolling sequence is captured using a moving camera, which results in generated events from both moving background and object. The second reason, which is also the main cause of tracking loss, is that the grayscale intensity of the grove in images has high similarity with that of the rolling car, which results in false decisions of positive and negative samples. To intuitively understand the contribution of event-guiding methods in terms of the tracking accuracy, the location error plots of the three testing sequences using SSVM, ESSVM –, and ESSVM are shown in Fig.3.19. By observing the pixel error plots of SSVM, the pixel error rates become serious from the early part of a whole sequence and cannot be recalled. It is thus concluded that the traditional SSVM can poorly cope with long-term tracking of fast moving objects. In contrast, the ESSVM – and ESSVM have much better performance. For *biker_head* and *bolt* sequences, the event location-guided method ESSVM – never lost tracking of the target object. The difference of tracking

results by ESSVM − and ESSVM is not obvious for *biker_head* sequence since the target tracking region is small. Nevertheless, with the amplified view of pixel error results, the tracking accuracy of full event-guiding methods (ESSVM) is more stable than guiding with only event locations (ESSVM −) and the same conclusion can be derived from the results of *car_rc_rolling* sequence.

Apart from tracking accuracy, real-time performance is another important metric for object tracking. In this work, the machine we used runs at 3.6 GHz with an NVIDIA GeForce GTX Titan Black GPU, and the system is under Ubuntu 14.04. For traditional SSVM tracker, we use the default parameters including Haar features, a Gaussian kernel with $\sigma = 0.2$ and a budget of 100 support vectors. Comparing to the speed of other baseline trackers [150], the processing speed of ESSVM, which reaches 93.8fps, is much faster than most of the others, as shown in Fig.3.20.

When taking both accuracy and processing speed into consideration, EESVM ranks first as illustrated in Fig.3.19. The location error for each tracker is the average value of location error on all 30fps sequences. Performance value is computed by dividing processing speed by 30fps. A value larger than 1 indicates that trackers have potential to process low frame-rate sequences in real-time. Obviously, ESSVM ranks first because it performs the best in terms of balancing the accuracy and real-time performance. In contrast, KCF is the fastest tracker, however, its accuracy is not satisfactory.

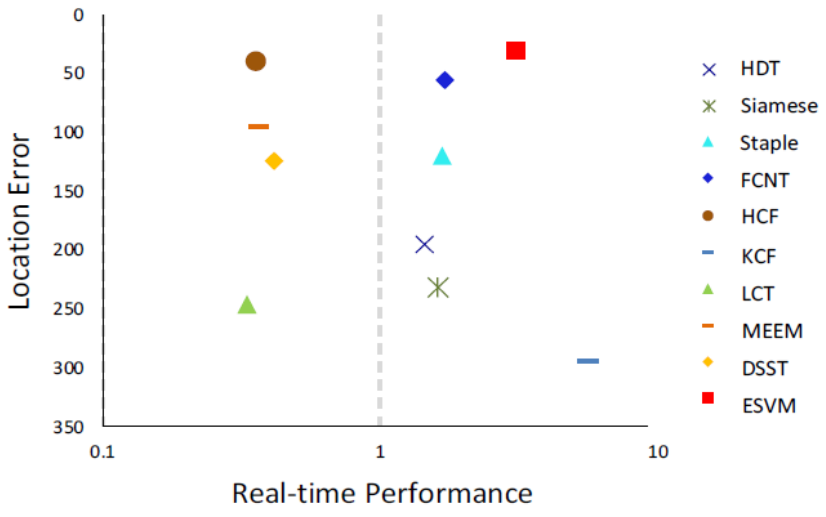


Figure 3.20 Plot of average location error over 30fps sequences and real-time performance.

3.6 Summary

In this chapter, a novel motion sensor hybrid standard frame-based and event-based dynamic vision sensor was proposed. The primary target of this proposal is to solve the accuracy limitations caused by traditional dynamic imagers with binary events. The proposed sensor has independent pixels that asynchronously report intensity changes ('events') with corresponding pixel intensities. The output of each event package contains pixel location and pixel instantaneous illumination. As a result, the proposed sensor directly extract motion with intensities that are essential for vision-based applications. In addition, pixels in the designed sensor are globally controlled by $forcefire$ signal to synchronously generate image frames on demand, contributing to the environmental perception in computer vision applications. The pixel intensity is readout through a low latency and low power intensity readout scheme, which has been elaborated in detail. A prototype of the proposed sensor has been fabricated and tested. Besides low latency and high temporal resolution, both in micro-seconds, the designed sensor has a high dynamic range (120dB). Moreover, compared with DAVIS and ATIS, it's free of exposure, thus avoids time lag between the readout of events and intensity.

In addition, this chapter also describes the proposed event-guided discriminative tracking method for the detection and tracking of high speed moving objects. Object tracking is one of the most complex tasks in computer vision, especially for fast moving objects. To achieve high accuracy and real-time performance, the proposed motion sensor was applied in this field and the adaptive event-guided tracking algorithm was designed. The algorithm integrates full-frames at low frame rate with the grayscale event stream that is temporally continuous. The high temporal resolution of event stream enables the capture of entire motion trajectory while standard frame-based cameras record discrete object locations. The event locations are used to generate an adaptive searching area for target localization. Meanwhile, the embedded intensities in asynchronous events contribute to feature reconstruction for target detection. The proposed algorithm provides high computational efficiency and experiments over sequences from multiple tracking benchmarks demonstrate its superior accuracy and real-time performance compared to state-of-the-art trackers.

Chapter 4 Motion Sensor with Pixel-Rendering Module for Optical Flow Extraction

4.1 Motivation

Real-time optical flow computation is still a crucial requirement for vision tasks such as optical flow-based segmentation, motion detection, object tracking and obstacle avoidance for aircraft and vehicles. In real-time applications where optical flow is a fundamental, decisive element, there is a pressing need to speed up flow computation while maintaining a high degree of accuracy.

Optical flow estimation has been studied extensively over the past 35 years. Classical methods for computationally based flow estimation are normally classified into four types: energy-based or frequency-based methods [134, 135], phase-based methods [17], correlation-based or region matching methods [165, 166] and gradient-based methods [15, 16]. Energy-based and phase-based methods usually achieve higher accuracy but involve such a high computational cost that they are not suitable for real-time processing. Correlation-based or region-matching methods are relatively faster but lack robustness. In contrast, gradient-based methods provide a better trade-off between accuracy and processing speed. Despite the large number of algorithms that have been proposed (127 were listed on the Middlebury website at the time of submission), most of these solutions are very similar to Horn and Schuncks' original formulation [15]. State-of-the-art techniques have gradually raised accuracy to remarkable new levels, as evidenced by evaluations on the Middlebury benchmark [139]. These methods process data on multiple frames captured by frame-based image sensors, repeatedly generating motion irrelevant data (that is to say, stable backgrounds) for post processing.

As a result, the processing speed of flow estimation methods is greatly limited by large data redundancy and massive computational costs. The development of DVS [46–51] in recent years, however, has provided an excellent potential solution to this problem. Unlike conventional frame-based image sensors, a DVS asynchronously reports the position of pixels, detecting variations in their brightness. As the data collected is highly relevant to the motion being observed, this considerably reduces data redundancy and dramatically increases post processing efficiency. In addition, a DVS is able to achieve super-high temporal resolution in nano-seconds [15]. Benosman et al. [16] first proposed computing event-based optical flow using the Lucas Kanade (LK) method [8] as a point of departure. They used the number of accumulated events, which actually represent variations in intensity, to simulate pixel intensity. Later, they proposed a local plane fitting method to estimate flow in the space-temporal domain [17]. Delbruck et al. recently evaluated state-of-the-art event-based optical flow estimation techniques with a generated ground truth in 8 directions [18]. All these developments are indicative of the potential of event-based techniques to resolve major problems in conventional, frame-based flow estimation. However, the lack of absolute intensity results in inaccuracy, especially when texture is present. Flow estimation accuracy is also limited by the sparseness of events, as the information from a single event is insufficient to compute spatial-temporal gradients.

This chapter presents a novel motion sensor with proposed pixel rendering module (PRM) to improve the accuracy of flow estimation based on events while retaining low computational cost. The PRM enables an active pixel to communicate with its neighbor pixels instead of working independently. The sensor outputs asynchronous event packages. Each event package contains information on the pixels position and the corresponding captured intensity together with an event-triggered time-stamp. Unlike other state-of-the-art DVS systems, the proposed sensor outputs asynchronous gray-level events from both the active pixel and its neighbor pixels, providing sufficient information for gradient computation. The design idea was inspired by the LK method [16] and by MRF modeling of frame-based optical flow estimation [169–171]. The small neighborhood window selected in the LK method and the MRF concept of neighborhood system fit well with the local events generated by a DVS. By

incorporating absolute intensity information together with its PRM, the proposed sensor improves the accuracy of event-flow estimation compared with a normal DVS, meanwhile, maintaining the real-time performance.

The rest of this chapter is structured as follows. Section 4.2 explains the design concepts behind the proposed sensor in detail. Pixel design and sensor implementations are described in section 4.3, while section 4.4 discusses the characterization of the sensor followed by experimental results in section 4.5. Finally, some conclusions are drawn in Section 4.6.

4.2 Design Concepts

4.2.1 Lucas Kanade Method (LK)

The LK method is one of the classical methods for optical flow estimation, which functions as the basic step for a wide range of frame-based flow estimation approaches. The traditional LK method has also been applied in event-based optical flow computation. The LK method focuses on local estimation, which is compatible with the sparseness output of a neuromorphic motion sensor. As reviewed, the classical assumption in optical flow estimation is brightness constancy constraint. It assumes that the intensity is constant along the pixels trajectory. The intensity of pixel $p = (p_x, p_y)$ at time t is given by the function

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{4-1}$$

Where $\delta x = u \cdot \delta t$, $\delta y = v \cdot \delta t$. Here, u, v represent components of the motion vector in horizontal and vertical directions respectively. By applying Taylor expansion to the above equation, the following equation is obtained

$$\frac{\delta x}{\delta t}\frac{\partial I_{p,t}}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I_{p,t}}{\partial y} + \frac{\partial I_{p,t}}{\partial t} + \epsilon(\delta t) = 0 \tag{4-2}$$

where $\frac{\partial I_{p,t}}{\partial x}$, $\frac{\partial I_{p,t}}{\partial y}$, and $\frac{\partial I_{p,t}}{\partial t}$ are spatial-temporal gradients. The error $\epsilon(\delta t)$ is a term of higher order δt and is negligible for short time interval. Hence, the Eq. (4-2) is of higher accuracy for higher temporal precision.

Single constraint cannot determine a unique optical flow and thus introduce the aperture problem. The Lucas Kanade (LK) method takes spatial smoothness as the second constraint. It is based on the observation that neighbour pixels in the visual scene typically belong to the same object and thus they are assumed to have the same dominant flow vector. The key idea of the LK method is to combine information from pixels in a small neighbourhood window, usually 3 × 3 or 5 × 5 as shown in Fig.4.1, and use least-square method to solve the simultaneous equations

$$\begin{aligned} \frac{\delta x}{\delta t}\frac{\partial I_{p_1,t}}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I_{p_1,t}}{\partial y} &= -\frac{\partial I_{p_1,t}}{\partial t} \\ \frac{\delta x}{\delta t}\frac{\partial I_{p_2,t}}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I_{p_2,t}}{\partial y} &= -\frac{\partial I_{p_2,t}}{\partial t} \\ \cdots & \\ \frac{\delta x}{\delta t}\frac{\partial I_{p_n,t}}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I_{p_n,t}}{\partial y} &= -\frac{\partial I_{p_n,t}}{\partial t} \end{aligned} \tag{4-3}$$

The LK method is a popular classical gradient-based method because of its low complexity. The method itself is not accurate enough when dealing with large displacement since it is a pure local method. In frame-based optical flow estimation, pyramid computation is usually combined with LK method to resolve the problem. In contrast, event-based DVS instinctively solves the problem because of their high temporal precision as accurate as several micro seconds. In most scenarios, the travel

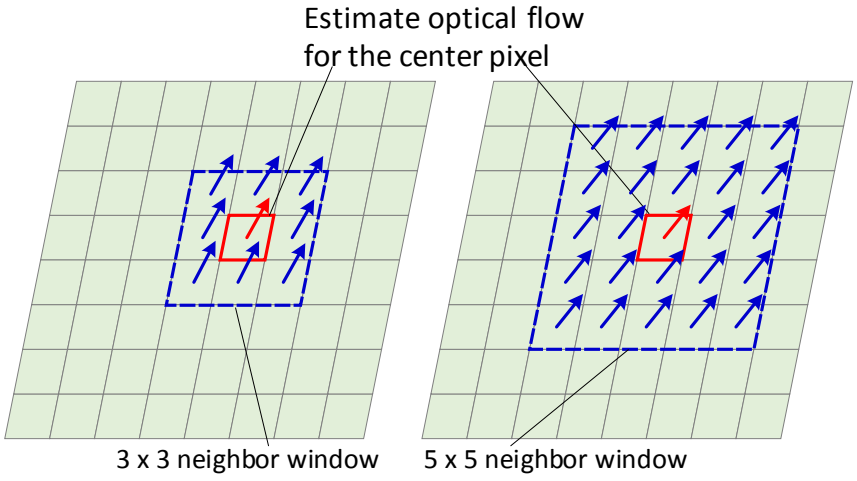


Figure 4.1 Flow vectors in a neighbourhood window, usually 3x3 or 5x5, are assumed to be the same.

distance of a fast-moving object during tens of micro second is limited within a neighborhood window.

4.2.2 Markov Random Field (MRF)

MRF optimization is widely used in the probabilistic optical flow estimation [169-172] and the optical flow itself is modelled as a Markov random field [173]. Let S be the finite index set which enumerates the pixels of a video frame. For MRF, a pixel position $p \in S$ is called a site or location, and a family of random variables $\{F_p : p \in S\}$ into a finite set Λ is called a random field. The smoothness constraint for flow estimation using MRF requires neighbourhood structures. MRF defines a neighbourhood system containing a collection $N = \{N_p, p \in S\}$ if the sites satisfy the following conditions

- $p \notin N_p, and\ if\ symmetry\ holds, i.e.$
- $p \in N_p \Rightarrow q \in N_p$

If $q \in N_p$A, p and q are called neighbours with a symbol $p \sim q$. Finally, a subset of S is called clique if any two of its elements are neighbours. The objective of MRF is to

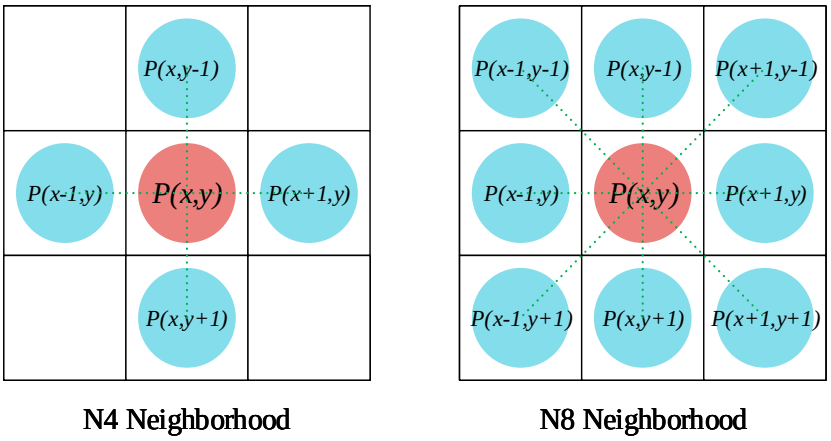


Figure 4.2 Pixel connection of N4 neighbourhood system (left) and N8 neighbourhood system (right).

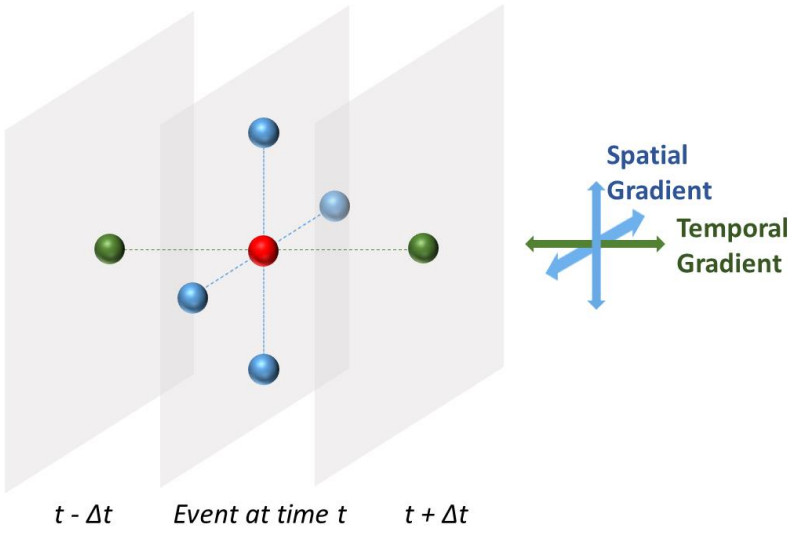


Figure 4.3 MRF neighbourhood system in spatial-temporal domain. If each point represents a pixel with absolute intensity, the blue pixels construct MRF in spatial domain for spatial gradient calculation while green pixels construct MRF in temporal domain for temporal gradient calculation.

generalise the well-known Markov chains to 2-D structures. As known, a Markov chain is a sequence of random numbers, such that a present random number depends only on the immediate previous one while it is independent from the other predecessors. To generalize the Markov chains, a random field is a Markov random field if its random variables satisfy Markov properties stating that a variable is highly dependent on its neighbours. The neighbourhood structure is defined by the Euclidean distance between neighbours $p \sim q$

$$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} < c \tag{4-4}$$

where $p = (p_x, p_y)$, $q = (q_x, q_y)$ and $c \in \mathbb{R}$ is a constant usually set to 1 or 2. For each site, the local characteristic of a Markov field is defined as

$$\prod_p(f) = P(F_p = f_p | F_p = f_p, \forall q \in N_p) \tag{4-5}$$

The whole family $\{\prod_p : p \in S\}$ is known as the local specification of the MRF. Combined with the satisfaction of positivity condition, which is usually satisfied in practice, the distribution of an MRF is uniquely determined. In spatial domain, the neighbourhood systems typically employ 4-neighbourhood (N4) or 8-neighbourhood (N8) structure, as visualized in Fig.4.2. In temporal domain, a present random variable

depends only on the immediate previous one (same location in the previous frame). Fig.4.3 illustrates a spatial-temporal neighbourhood system centred at the red point, which consists of a N4 system (blue points) and two-neighbouring points along time axis (green points).

4.2.3 Pixel Rendering Module (PRM)

Both LK method and MRF focus on the pixels in a local area. The original LK method construct simultaneous equations based on at least nine pixels in a 3×3 window. Combining these two concepts in optical flow estimation, the number of simultaneous Eq. (4-3) from LK method can be reduced to five based on the five pixels in MRF neighborhood system (N4). Inspired by this, a pixel rendering module (PRM) is heuristically designed based on the DVS structure. Each pixel is connected to its four neighbor pixels in N4. The functionality of PRM is intuitively illustrated in Fig.4.4.

Since the current event-based flow estimations are conducted taking use of the events of traditional DVSs, as illustrated in Fig.4.4, a comparison of functionality

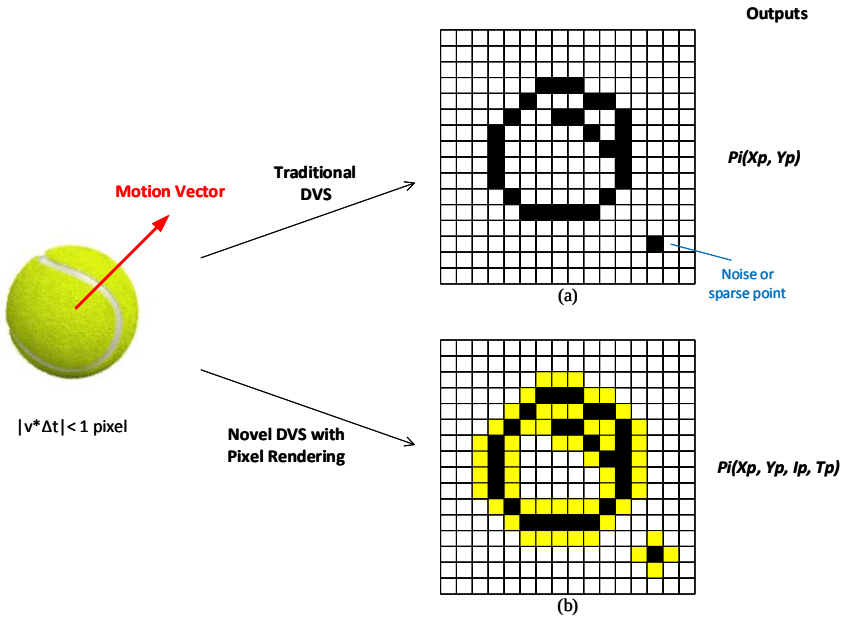


Figure 4.4 Functionality comparison between a traditional DVS and the proposed optical flow motion sensor with PRM. (a) Traditional DVSs only output the position (X_p, Y_p) of pixels labeled in black, (b) novel motion sensor with PRM outputs the event packages from both active pixels and their neighbor pixels as labeled in both black and yellow. Each event package consists of pixel position (X_p, Y_p), captured intensity (I_p) and event-triggered time-stamp (T_p).

between a traditional DVS and the proposed sensor by showing the possible output data of the event-based sensors when observing a moving ball. The block arrays on the right side represent the pixel arrays in image sensors. Assume a tennis ball with white background is moving towards the upper right, pixels in a normal DVS are triggered following the detection of sufficient intensity variation. Such active pixels with positions (X_p, Y_p) will output binary events with encoded pixel positions and are marked black as shown in Fig.4.4(a). In contrast, the novel designed motion sensor outputs not only active pixels, as for the traditional DVS, but also their neighbor pixels, marked as black and yellow in Fig.4.4(b). For each active pixel, if pixels in its four-neighborhoods (left, right, top and bottom) are not active, they will be force activated by the active pixel. In this way, one can collect information from both active pixels and their neighbor pixels. As shown in Fig.4.3, for a single sparse event, data from five pixels are collected instead of one. The design of PRM guarantees the sufficiency of data for optical flow estimation such that the accuracy issues caused by the event sparseness can be mitigated especially in the flow estimation of fast motions. To validate this design, we evaluated optical flow result using simulated data from a normal DVS and our motion sensor based on Middlebury database [138] and MPI database [139], as detailed in section 4.5.

4.2.4 Asynchronous Grayscale Events

Event-based flow estimation approaches [143,144] compute normally optical flow based on LK method with spatial-temporal gradients extraction. The number N_e of accumulated events $E(x, y)$ for a single pixel $P(x, y)$ during a short time interval Δt is used to simulate the intensity of this pixel in gradient extraction. However, the generation of an event by DVS is caused by intensity variation ΔI, the accumulated event number N_e essentially reflects the total intensity variation $N_e \times \Delta I$ during Δt. As a result, such algorithms are susceptible to scenarios with high texture in foreground and background. To mitigate this issue, the optical flow motion sensor adopts the asynchronous event intensity extraction mechanism proposed in chapter 3. With the assistance of PRM and pixel intensities, the spatial-temporal derivatives, which are essential in optical flow computation, can be easily derived. In addition, with the

extraction of absolute illumination, the optical flow motion sensor thus can be applied to a wider range of gradient-based flow estimation algorithms.

4.3 Image Sensor Design

4.3.1 Sensor Architecture

Based on the study and analysis of frame-based and event-based optical flow algorithms, gradient-based algorithms are more robust and widely implemented in real-time applications. In order to improve the performance of the whole optical flow computation system, part of the gradient-based algorithm, which is the base of the brightness constancy constraint, is implemented using a dynamic vision sensor (DVS) during the dynamic image acquisition. According to the brightness constancy constraint equation:

$$I_x u + I_y v = -I_t \tag{4-6}$$

where I_x, I_y are the spatial gradients in x and y directions respectively, and I_t is the temporal gradient. When detecting high speed moving objects, the event spikes in DVS become sparse and thus the spatial gradient cannot be derived with a single pixel. To solve this issue, each pixel is designed to communicate with its neighbouring pixels. Fig.4.5 is used to illustrate the extraction of spatial-temporal gradients for a specified pixel P_c. Taking a 5×4 pixel array as an example, at $T_0 = 0$, all pixels are reset to idle status. Assuming that a single pixel P_c (P_{32}) is triggered and turns into an active pixel at $T_1 = t_1$ (red P_c at T_1), it sends a pulse signal to its four-neighbour pixels (left, right, top and bottom) immediately and simultaneously, which forces the neighbouring pixels (green pixels) to generate events and record the light intensity at $t_1 + \delta t$. δt is the time delay caused by the communication circuit between pixels and is simulated to be $1 -$

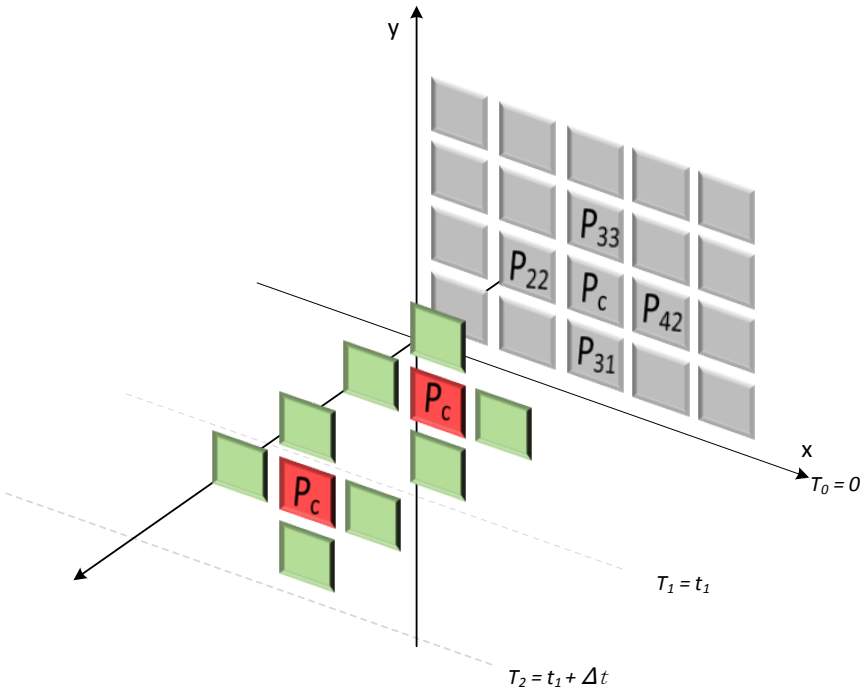


Figure 4.5 Design idea of the pixel rendering module (PRM). A single activated pixel P_c triggered by intensity variation will simultaneously activates its neighbour pixels $(P_{22}, P_{31}, P_{42}, P_{33})$ to generate grayscale events.

$2ns$. Since the shortest time interval to read the same pixel twice is several microseconds, the time delay δt is negligible. The spatial gradients for $P_{c|T_1}$ is thus calculated as

$$g_x|_{c|T_1} = \frac{1}{2}\left(I_{22|T_1} - I_{42|T_1}\right) \tag{4-7}$$

$$g_y|_{c|T_1} = \frac{1}{2}\left(I_{31|T_1} - I_{33|T_1}\right) \tag{4-8}$$

The calculation of temporal gradient is the difference between the intensities of the same pixel at two consecutive timepoints,

$$g_t|_{c|T_1} = \left(I_{c|T_2} - I_{c|T_1}\right) \tag{4-9}$$

A prototype of this optical flow motion sensor with a 64x64 pixel array was implemented with AMS 0.35µm CMOS technology and the system architecture is shown in Fig.4.6 with novel designed modules labelled in blue. Each pixel is connected with a row AER control logic bus, column control logic bus, and column readout circuitry. In addition, each pixel has four communication lines connected with its neighbouring four pixels to enable the communication between the pixels. The row

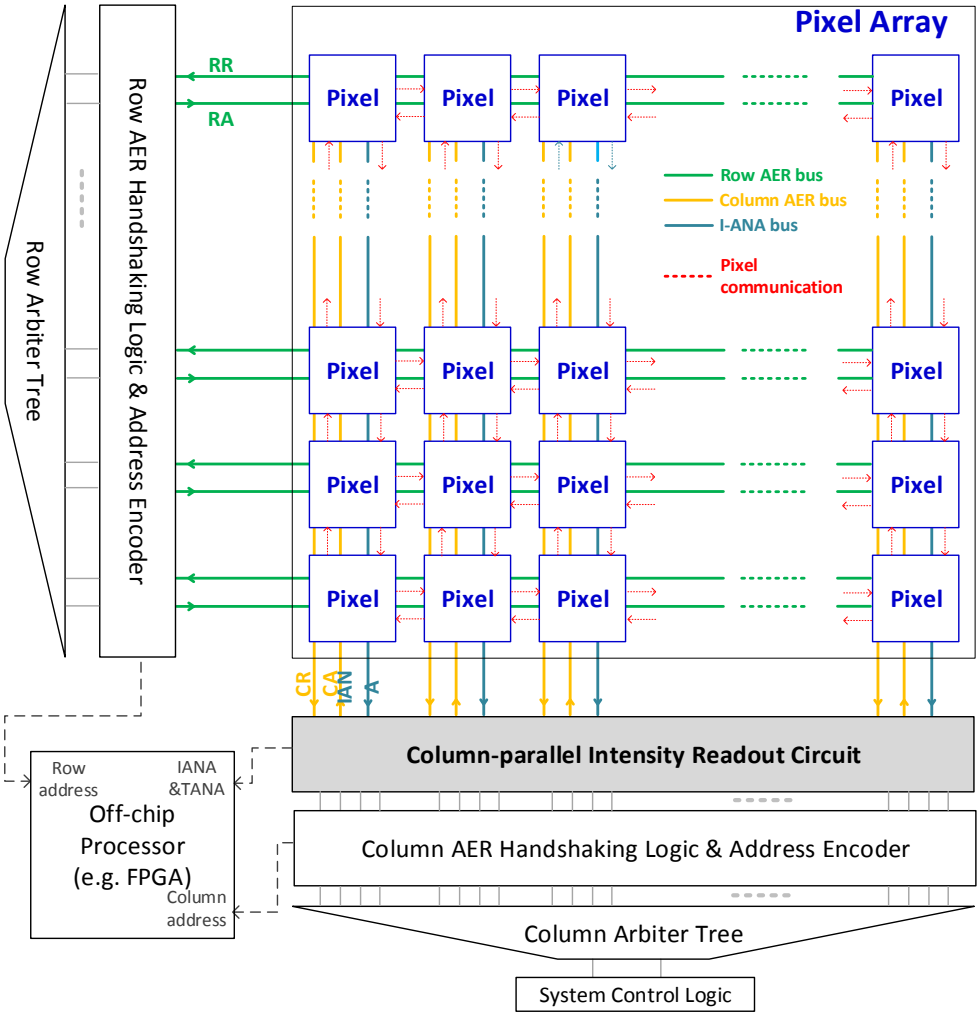


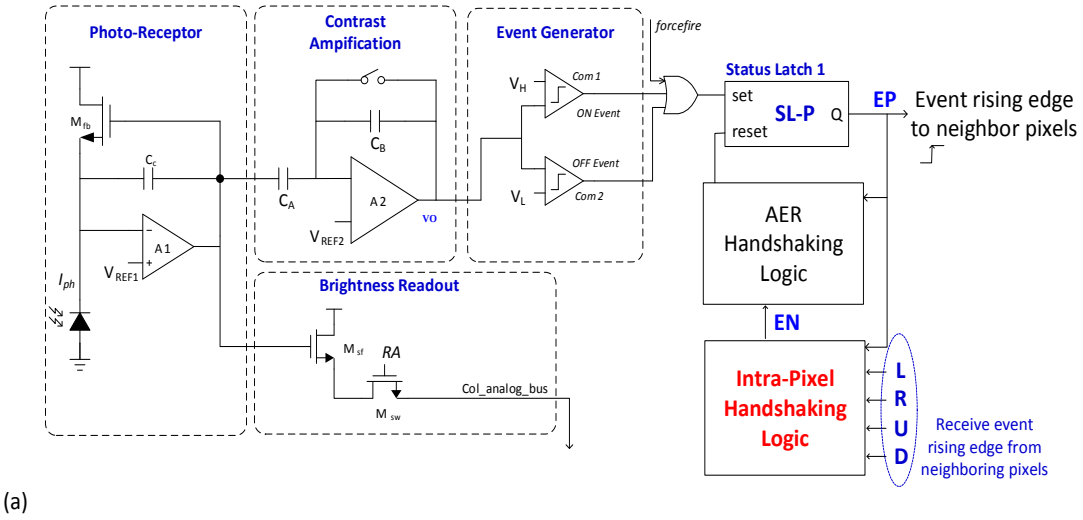
Figure 4.6 Architecture of the optical flow oriented motion sensor.

AER control logic includes a row address encoder and a row arbiter tree. This work proposed a new structure of arbiter tree, which can process arbitrations among multi-requests in a fair manner and solve the double-line deadlock issue caused by the arbiter tree in old version. The row AER control logic is responsible for handshaking with pixel signals in the row direction. The row arbiter tree receives the multiple row requests and is designed to randomly respond to the row requests and send row acknowledgement one at a time. Sending out a row acknowledgement indicates that a certain row of the pixel array is selected, and the in-row active pixels will be processed. During the row processing, the row encoder encodes the row address of the row being processed and the result is transferred off-chip for post processing. The column control

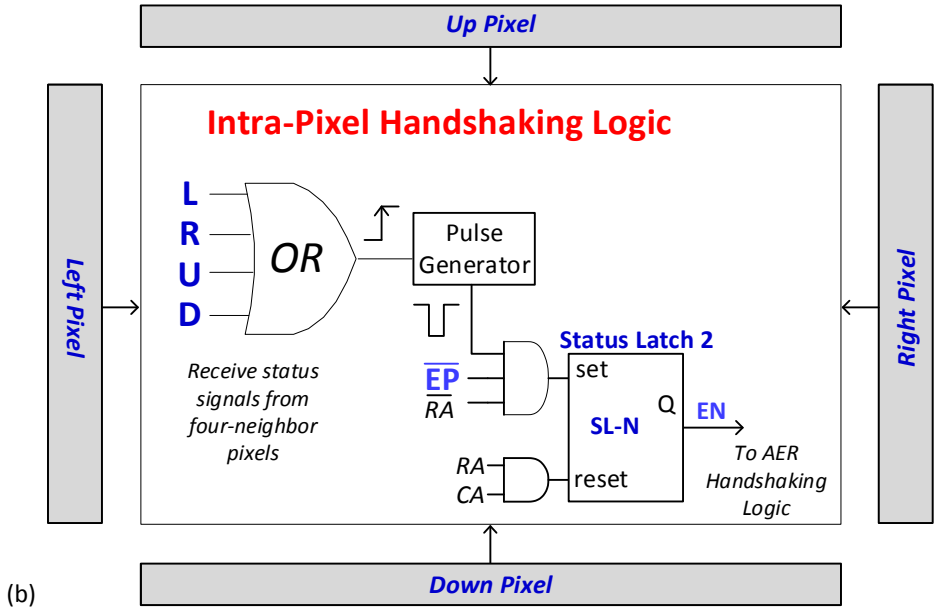
logic is in charge of the pixel communications in the column direction and contains signal readout logic apart from the column address encoder and column arbiter tree. When the pixel array receives the row acknowledgement, the pixels on the selected row output column requests as well as the light intensity analog value. The column readout circuit is the same as elaborated in chapter 3. The function of the column arbiter tree is the same as the row arbiter tree, which receives column requests and sends back a column acknowledgement. The column acknowledgement indicates that the pixel on the selected column is being processed and can be used to encode the column address by the column address encoder. Therefore, at each time point of pixel processing, there is only one pixel who receives both row and column acknowledgement signals being processed. In this sensor, we design an arbiter tree structure that the arbitration priority of each arbiter cell is toggled in a fair manner, such that a pixel that makes consecutive requests to the arbiter is not revisited until the rest of the tree has been serviced.

4.3.2 Pixel Design

The pixel architecture is illustrated in Fig.4.7(a). The pixel can be simplified into several blocks including photo-receptor, event generator, logarithmic pixel readout, AER handshaking logic and intra-pixel handshaking logic. The photo-receptor consists of one NMOS transistor with photo-diode and a negative feedback amplifier. Its logarithmic response provides high dynamic range, at the s temporal contrast sensitivity at high illumination. The logarithmic photo-receptor transforms the intensity to a voltage signal in real-time. The transformed intensity signal is fed into a delta modulator followed by two comparators. An OR gate merges the V_H and V_L events which corresponds to light changes from either dark to bright or vice versa. An external forcefire signal is also merged to unconditionally force all pixels to set their respective status latch 1 (SL-P) and generate requests through the AER handshaking circuits. SL-P can be set by either the output of the V_H or V_L comparator or the external forcefire signal. The output signal of SL-P (EP) is active high and connected with its neighbor pixels. An intra-pixel handshaking logic is introduced to monitor primary event status (EP) of its four neighbor pixels (Left, Right, up and down). If one or more neighbor



(a)



(b)

Figure 4.7 (a) Pixel architecture of the novel designed vision sensor. (b) Architecture of the intra-pixel handshaking logic of Pixel Rendering Module. L, R, U, D represents the EP signal received from its left, right, up and down neighbour pixels.

pixels become active, i.e. signal EP switches from low to high, the intra-pixel handshake generates a signal EN, indicating that at least one of the neighbor events is active, and initials the AER handshaking circuits. The AER handshaking logic can thus be initialized by intensity variation detection, $forcefire$ control or intra-pixel handshaking logic.

The pixel takes the voltage on the logarithmic photo-receptor as its intensity information. No additional exposure time is needed and thus it takes no effort to match the event position and intensity information, in both spatial and temporal domain. The logarithmic photo-receptor encodes the intensity information to electronic signal in real-time. Since the photo-receptor voltage is logarithmically related to the light intensity, a wide dynamic range up to $120dB$ can be achieved.

Pixels of the current dynamic image sensors work individually leading to highly sparse pixel output. In the new motion sensor, an intra-pixel handshaking is introduced such that the pixels communicate pixel status with each other and no longer behave independently. Fig.4.7(b) shows the schematic of the intra-pixel handshaking logic. The signal EP in each pixel is connected to the intra-pixel handshaking logic of its neighbor pixels. When a pixel detects intensity change, the status latch is set and generates a rising edge. The intra-pixel handshaking logic in each pixel continuously monitors the status of its neighbor pixels. An OR gate merges the status signals and transmits the rising edge to a pulse generator if at least one of the neighbor pixels is activated. The

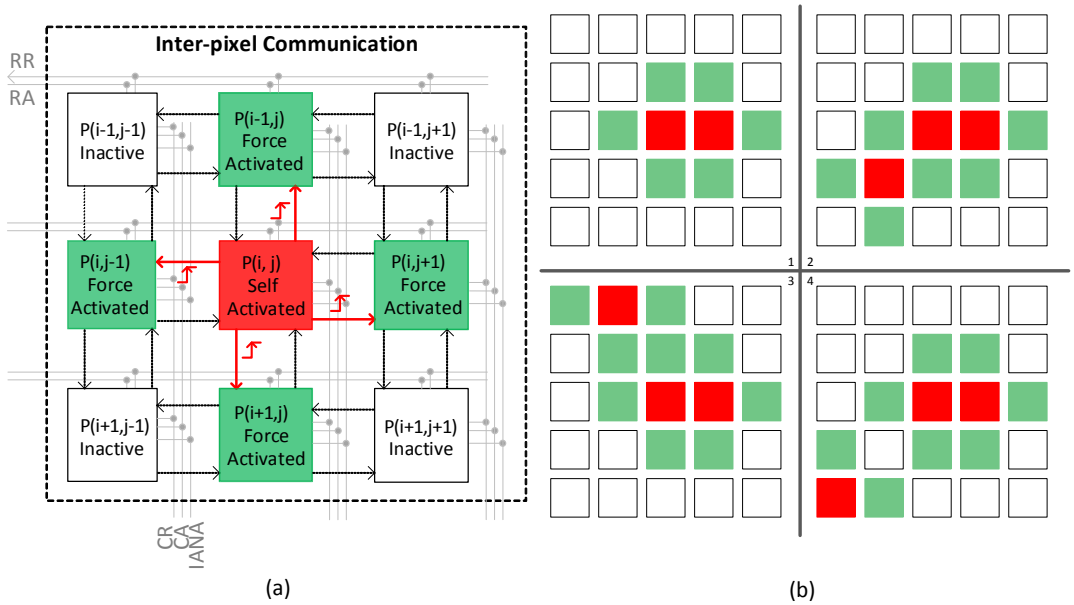


Figure 4.8 Inter-pixel communication due to PRM. (a) Illustration of communications between pixels under pixel rendering mechanism. RR, RA, CR and CA are signal for AER handshaking, while pixels in the same column share the $IANA$ buses for intensity analog value readout, (b) four examples of pixel arrays with different pixel status pattern, red pixels represent the pixels activated by illumination detector and green pixels represent pixels activated by self-activated (red) pixels.

generated pulse sets the status latch 2 (SL-N) under the conditions that the pixel is neither activated due to intensity change nor selected for processing. The pixel resets both status latches when it is processed. The signals EP and EN can be used to identify whether an event is triggered by the pixel's illumination detector or by its neighbors.

The function of PRM is intuitively elaborated in Fig.4.8(a), taking a 3x3 pixel array as an example. Pixels are marked as red, green and white, representing three possible status of pixels, i.e. self-activated, force-activated and inactive pixels respectively. A self-activated pixel refers to a pixel activated by its illumination change detector, while a pixel is force-activated if triggered by its neighbor pixels instead of intensity change. It is noted that intensity change always has a higher priority such that a force-activated pixel can still detect intensity change and turn into a self-activated pixel. The two statuses are distinguished in function as only self-activated pixels (in red) can activate neighbor pixels. Accordingly, the timing diagram of main signals of this example is shown in Fig.4.9. Suppose only the central pixel ($P(i,j)$) detects intensity variation and the output signal of delta modulator (VO), see Fig.4.7(a), exceeds the contrast threshold (V_H). When VO meets V_H, the status signal EP of the central pixel, $EP(P(i,j))$, is set to high and this rising edge triggers the status signals EN of its neighbor pixels to high. The transfer latency is less than $1ns$ and thus can be ignored. Both self-activated status and force activated status are reset (falling edge of EP and EN) when the pixel is processed, i.e. receiving row acknowledgement (RA) and column acknowledgement

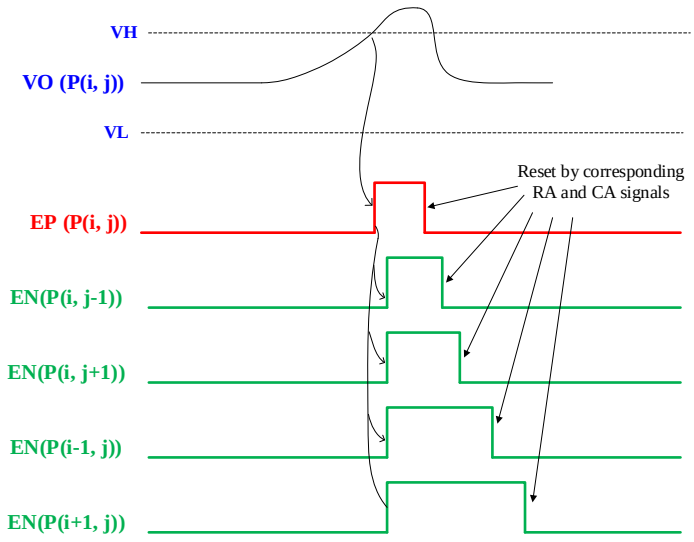


Figure 4.9 An example of main signal timing diagram according to Fig.4.8 (a).

(CA) signals simultaneously. Meanwhile, the signal VO is reset to a reference level. Fig.4.8(b) shows four more examples with different pixel status patterns. With the inter-pixel handshaking logic, the sensor collects more supplementary information to compute the flow of self-activated events. For each time a single pixel is self-activated, the sensor collects information from five pixels, including the self-activated pixel and the force-activated neighboring pixels. Each pixel outputs an event package consisting of pixel position, captured intensity information and time-stamp.

4.4 Sensor Implementation and Measurement

A 64×64 test prototype of dynamic vision sensor with PRM was designed and fabricated in AMS 0.35um 2P4M process. Fig.4.10 shows a micro photograph of the die, of size $4.2 \times 5.9\ \text{mm}^2$. Pixel area is $40 \times 40\ \text{mm}^2$. The capacitors are placed next to each other for better matching. They are also shielded from signal wires with GND metal plates. To measure the hardware performance of this motion sensor, a FPGA-

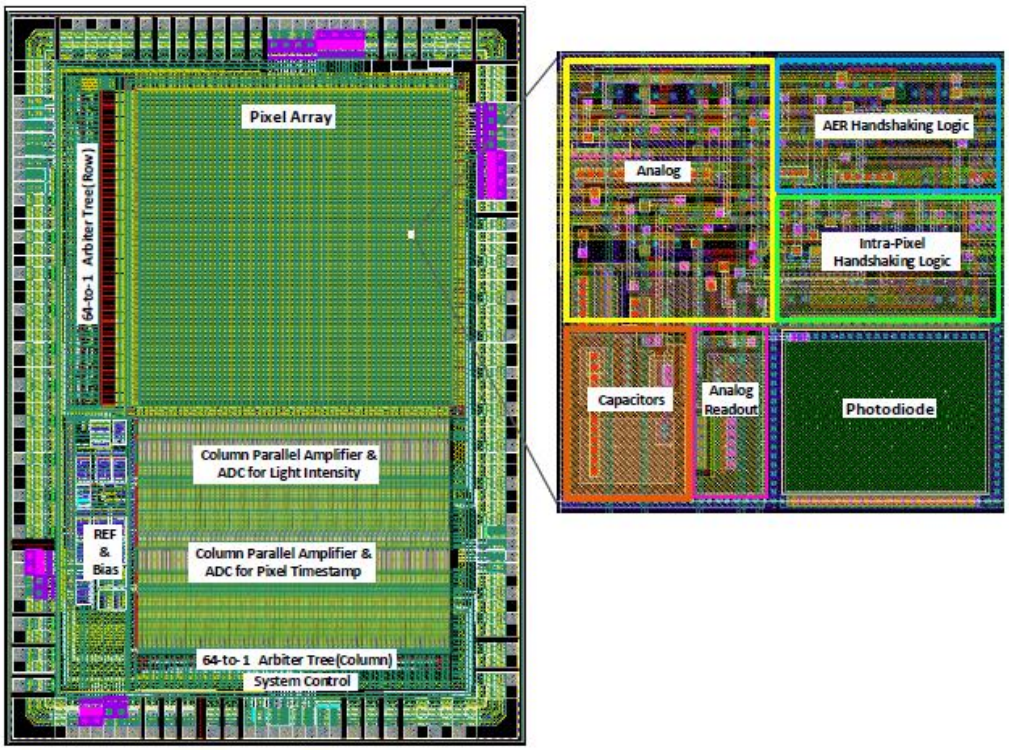


Figure 4.10 (left) Top level layout of the fabricated motion sensor with pixel rendering module, (right) pixel layout.

based testing platform was developed. The testing platform contains an Opal Kelly FPGA board, which is configured to provide input control signals, temporarily store the data and communicate with a local host PC through USB link. The FPGA board adopted is Opal Kelly USB-FPGA module XEM3050. It features a Xilinx Spartan 3 FPGA (XC3S4000), a phase-locked loop (PLL) for clock configuration, an on-board 9Mb synchronous static random-access memory (SSRAM) and two 32MB synchronous dynamic random-access memories (SDRAMs) for large data storage. The FPGA board also uses a Cypress USB controller to handle the USB protocol. The sensor is powered by 3.3V and works well with maximum 50MHz. A GUI is developed using C++ for the convenience of sensor control and data analysis. Table VII summarizes the sensor specifications and its main features. According to the test results, the sensor is clocked at 40MHz during the experiments. Compared with the first designed sensor detailed in Chapter 3, the optical flow motion sensor has a larger fill factor at the expense of larger pixel size due to the introduction of PRM.

4.5 Experiments and Results

First part of the experiments presents experiments using simulated data for different types of motion sensors. This part also includes a comparison of optical flow estimation accuracy between a typical DVS and the proposed optical flow oriented

TABLE VII SPECIFICATIONS COMPARED WITH PRIOR DVS IMPLEMENTATIONS

	This work	Design work 1	Delbruck et al. [49]	Posch et al. [46]
Pixel Functionality	Pixel Rendering	Individual	Individual	Individual
Functionality	async. temporal contrast + async intensity + full frame	async. temporal contrast + async intensity + full frame	async. temporal contrast + APS	async. temporal contrast + level crossing intensity
Process Technology	AMS 0.35 μm 2P4M OPTO	AMS 0.35 μm 2P4M OPTO	0.18 μm 1P6M MIM CIS	0.18 μm 1P6M MIM CIS
Array Size	64×64	384×320	240×180	304×240
Pixel Size μm^2	40×40	30×30	18.5×18.5	30×30
Fill Factor	17.7%	12%	22%	20%, 10%
Power Supply	$3.3V$	$3.3V$	$1.8/3.3V$	$1.8/3.3V$
FPN	0.79%	0.38%	0.5%	0.25%
Power @100eps mW	55	70	145	175

motion sensor. The second part of experiments shows the test platform and examples of flow estimation in real scenarios using the fabricated motion sensor.

4.5.1 Methodology

The method employed for flow estimation is the $classic + NL$ approach. Among classical methods, it exhibits superior performance with an exhaustive analysis and public source code. Based on observations about median filtering and L1 energy minimization, the $classic + NL$ integrates median filtering into the original, classical objective function whose spatially discrete form can be expressed as

$$\mathrm{E(u,v)} = \sum_{i,j}\left\{\rho_D\left(I_1(i,j) - I_2\left(i+u_{i,j}, j+v_{i,j}\right)\right) + \lambda\left[\rho_S\left(u_{i,j}-u_{i+1,j}\right) + \rho_S\left(u_{i,j}-u_{i,j+1}\right) + \rho_S\left(v_{i,j}-v_{i+1,j}\right) + \rho_S\left(v_{i,j}-v_{i,j+1}\right)\right]\right\} \quad (4\text{-}10)$$

Where u and v are the horizontal and vertical components of the optical flow field to be estimated from images I_1 to I_2, λ is a regularization parameter, and ρ_D and ρ_S are the data and spatial penal functions defined as $\rho(x) = \log(1 + \frac{x^2}{2\sigma^2})$, which is related to a standard pairwise MRF based on a 4-neighborhood. The application of median filtering during optimization effectively improves the robustness of the classical methods. In particular, the optimization of Eq. (4-10), with interleaved median filtering, approximately minimizes

$$\begin{aligned}\mathrm{E}(\mathbf{u},\mathbf{v}) = \sum_{i,j}\Big\{&\rho_D\left(I_1(i,j) - I_2\left(i+u_{i,j}, j+v_{i,j}\right)\right)\\ &+ \lambda\left[\rho_S\left(u_{i,j}-u_{i+1,j}\right) + \rho_S\left(u_{i,j}-u_{i,j+1}\right) + \rho_S\left(v_{i,j}-v_{i+1,j}\right)\right.\\ &\left.+ \rho_S\left(v_{i,j}-v_{i,j+1}\right)\right]\Big\} + \lambda_N \sum_{i,j}\sum_{(i',j')\in N_{i,j}}\left(\left|u_{i,j}-u_{i',j'}\right| + \left|v_{i,j}-v_{i',j'}\right|\right)\end{aligned} \quad (4\text{-}11)$$

where $N_{i,j}$ is the set of neighbors of pixel (i,j) in a possibly large area and λ_N is a scalar weight. The neighbor size in this algorithm is set to 5×5, which is the same size as the median filter. The algorithm is summarized in Algorithm 2.

Algorithm 2

Input: Image pairs, (I_i^0, I_{i+1}^0)
Output: Optical flow field $\boldsymbol{u} = (u, v)$
Parameters: $\alpha = 0.8$ (scale factor), $N_{GNC} = 2$, $N_{warp} = 3$, $N_{iter} = 20$, λ , $\varepsilon = 0.01$
Initialization: $\boldsymbol{u}^0 = 0$

1. compute the number of levels N_{level} according to scale factor
2. for scale j = 0 to N_{level} do
3. $\quad I_i^j = I_i^j * LoG$
4. $\quad I_{i+1}^j = I_{i+1}^j * LoG$

5. for structure k = 0 to N_{GNC} do
6. $\quad$ for level j = N_{level} to 0 do
7. $\quad\quad u^j = u^0$
8. $\quad\quad$ for $l = 1$ to N_{warp}
9. $\quad\quad\quad$ warp image I_{i+1}^j with flow field u^j
10. $\quad\quad\quad m = 1$
11. $\quad\quad\quad$ do
12. $\quad\quad\quad\quad$ compute u^j using energy minimization of $E(u^j)$ in Eq. (4-11)
13. $\quad\quad\quad\quad m = m + 1$
14. $\quad\quad\quad$ while ((m $\leq N_{iter}$) and ($\left\|u^j - u^0\right\|^2 > \varepsilon$))
15. $\quad\quad\quad$ Update $u^0 = u^j$
16. $\quad\quad\quad$ perform weighted median filtering
17. $\quad\quad$ if j > 0 then
18. $\quad\quad\quad u^0 = \text{median}(\uparrow \circ \frac{u^j}{\alpha})$
20. $\quad\quad$ else
21 $\quad\quad\quad$ return optical flow (u = u^0)

4.5.2 Flow Estimation using Simulated Data

To validate that the newly designed motion sensor with pixel rendering mechanism contributes to the accuracy improvement in event-driven optical flow estimation, experiments were first designed to estimate optical flow using simulated event data. The experimental data were generated to simulate three types of event sensor. One of them (Type I motion sensor) is the event mode of DAVIS [49, 50], which is widely used in the current event-based optical flow estimation with the output of sensor comprising asynchronous events with binary polarity. The second type of event sensor

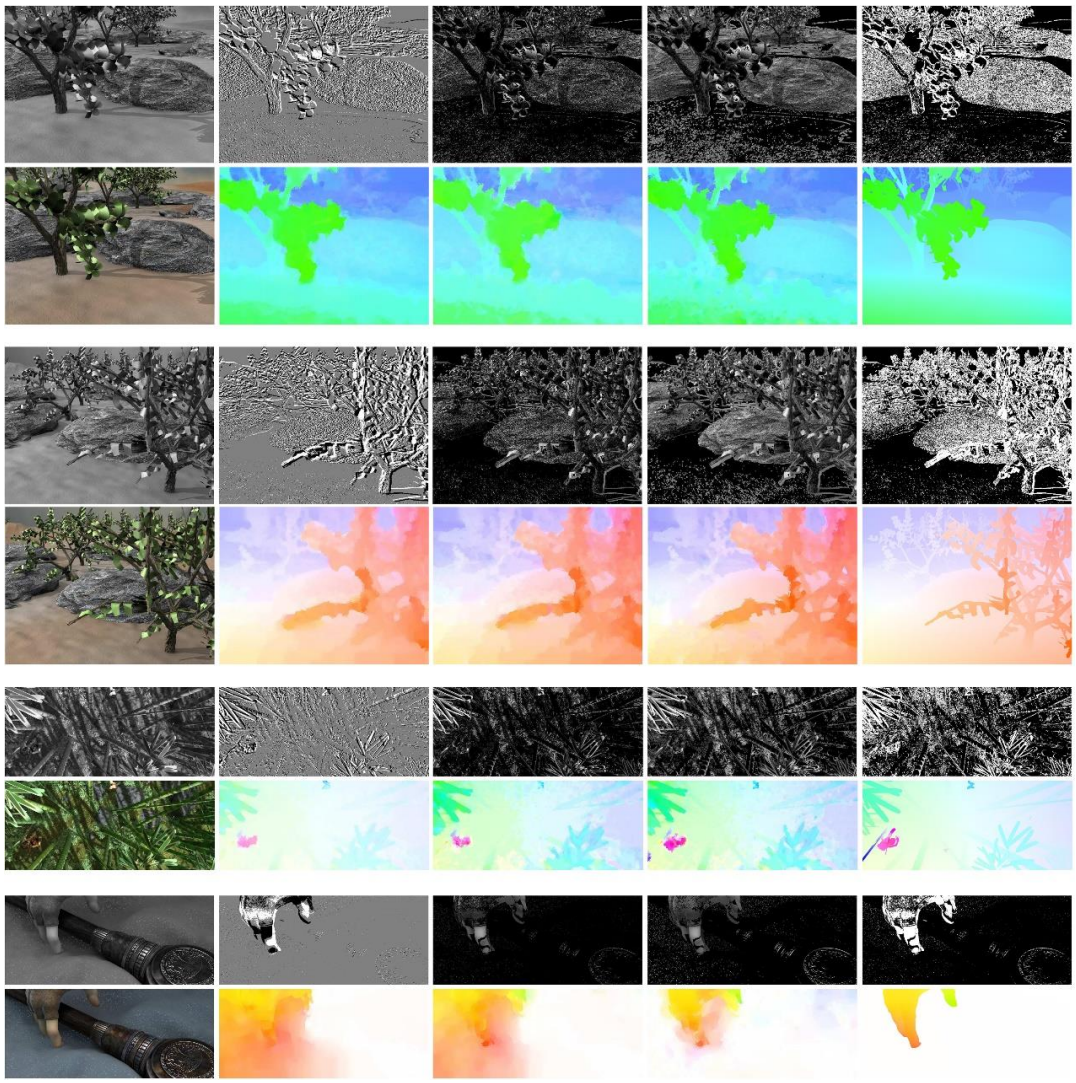


Figure 4.11 Four sets of experiment data and results. The test sequences from top to bottom are Grove2, Grove3 (from Middlebury database [138], bamboo-1 and ambush 7 (from MPI database [139]). For each data set, the first row from left to right: original image in gray-level events under PRM and a binary map indicating the active events without pixel rendering; the second row from left to right: original color image, flow using event with intensity variation, flow using event with absolute intensity, flow using events from our sensor and ground truth.

(Type II) is the motion sensor described in chapter 3, named grayscale motion sensor (GMS) for short, which outputs asynchronous grayscale events, and the last one (Type III) is the optical flow motion sensor (PRM-GMS) presented in this work. The testing sequences were taken from the well-known Middlebury database [138] and MPI-Sintel database [139]. Fig.4.11 shows some examples of the simulated data together with the flow estimation results. The test sequences from top to bottom are Grove2, Grove3 (from Middlebury database, bamboo-1 and ambush 7 (from MPI database). For each data set, from left to right the first row first shows the original image in gray-scale

followed by the events with binary polarity simulating DAVIS events. The third image shows the grayscale events followed by the grayscale events with pixel rending and the last one shows a binary map indicating the active events without pixel rendering. The second row, from left to right, shows the original color image, flow using event with intensity variation, flow using event with absolute intensity, flow using events from optical flow motion sensor and ground truth.

The synthesized data is generated as follows: a delta map is generated at first by subtracting two consecutive images from a sequence. Each value in the delta map indicates the intensity variation of corresponding pixel. The sign of value (positive or negative) is used to label output events as ON events or OFF events. A contrast threshold of 15 is set to simulate the threshold of comparators ($V_H - V_{REF2}$ or $V_{REF2} - V_L$). This contrast threshold is experimentally determined. As mentioned in chapter 2.3, the prior work of optical flow estimation based on traditional DVS simulates pixel intensity using the number of accumulated events, which actually reflects the intensity variation. Therefore, the delta map is used to simulate the output of a normal DVS (see Fig.4.11 row 1, col 2 of each test sequence). In contrast, the output events of our sensor have no polarity. When a pixel is activated, it captures the absolute intensity and also activates its neighbor pixels. To simulate the output of our sensor, we first set up a binary map according to the delta map. If the absolute intensity variation of a pixel is larger than the predefined contrast threshold, the pixel is considered to be active and its corresponding binary value is one, otherwise is considered inactive with a binary value zero. In the binary may, we search for zeros in the neighborhood of each active pixel and manually set to ones. Then we get a second binary map containing both active pixels and their force activated neighbor pixels. Next for each pixel with binary one in the second binary map, we extract its intensity value in the first frame (consecutive two frames were used to compute optical flow). Finally, we set up a new figure containing events with absolute intensity (see Fig.4.11 row 1, col 4 of each test sequence). In order to demonstrate the contribution of PRM in flow estimation, we also simulated the output data of a motion sensor with only gray-level events (see Fig.4.11 row 1, col 3 of each test sequence). The events contain the pixel positions according to the first binary

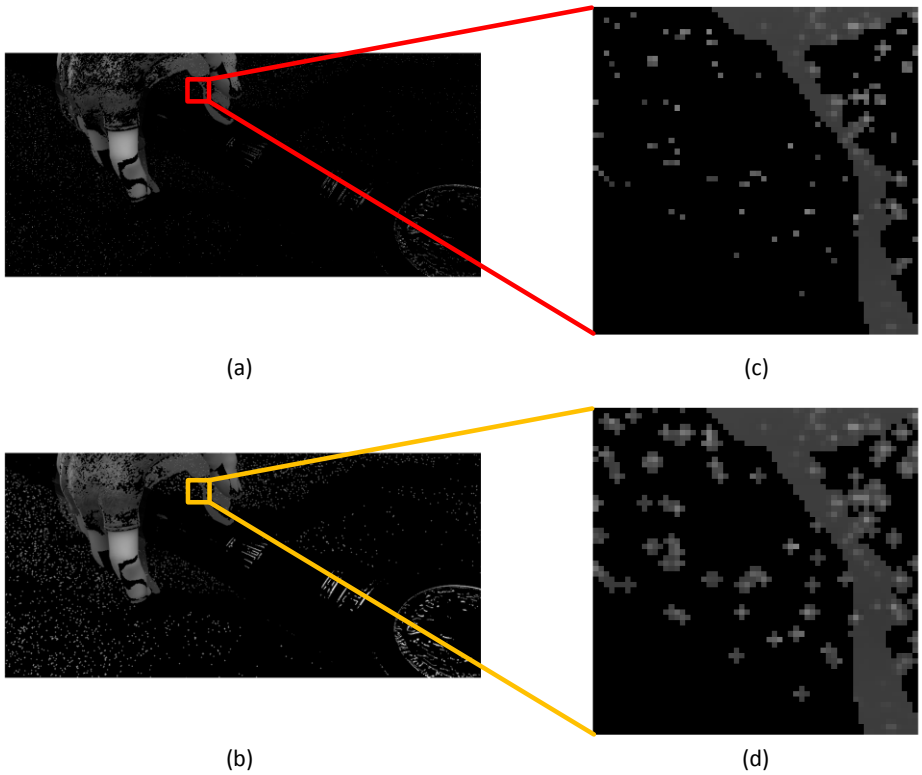


Figure 4.12 The difference of output events with or without pixel rendering.

map as well as their corresponding intensity. Fig.4.12 shows the difference of synthesized data with or without pixel rendering.

To evaluate the flow estimation performance of different types of motion sensor, we generated event slices using the simulated events and estimated flow on the slices using the classical $classic + NL$ method. Two event slices were generated from three consecutive images in each test sequence. We used $classic + NL - fast$ [19] to estimate flow, whose source code is available, and the same method was employed in [27] for comparison. In practice, the accumulated events corresponding to the moving objects usually have a small motion due to high temporal resolution and thus pyramid layers can be highly reduced to retain small motions. In this experiment, pyramid layers are necessary as the movements between some images are larger than ten pixels and considered as large displacement. The simulation results indicate that the pixel rendering scheme together with asynchronous absolute intensity information plays an important role in flow estimation and significantly improves the accuracy of event flow estimation. The experiment covers the testing of 27 sequences from both Middlebury

database [138] and MPI database [139]. Fig.4.11 shows an example of flow results of four sequences. The color-coding flow results use the same color coding method as database [138]. From direct observation of the flow result, the flow estimation using only gray-level events (GMS) is seen to perform better than using polarity events, yet the improvement is not obvious. In contrast, the introduction of PRM, apparently improves the accuracy performance of flow results and provides higher accuracy in detail even in highly textured scenarios (e.g. Grove3).

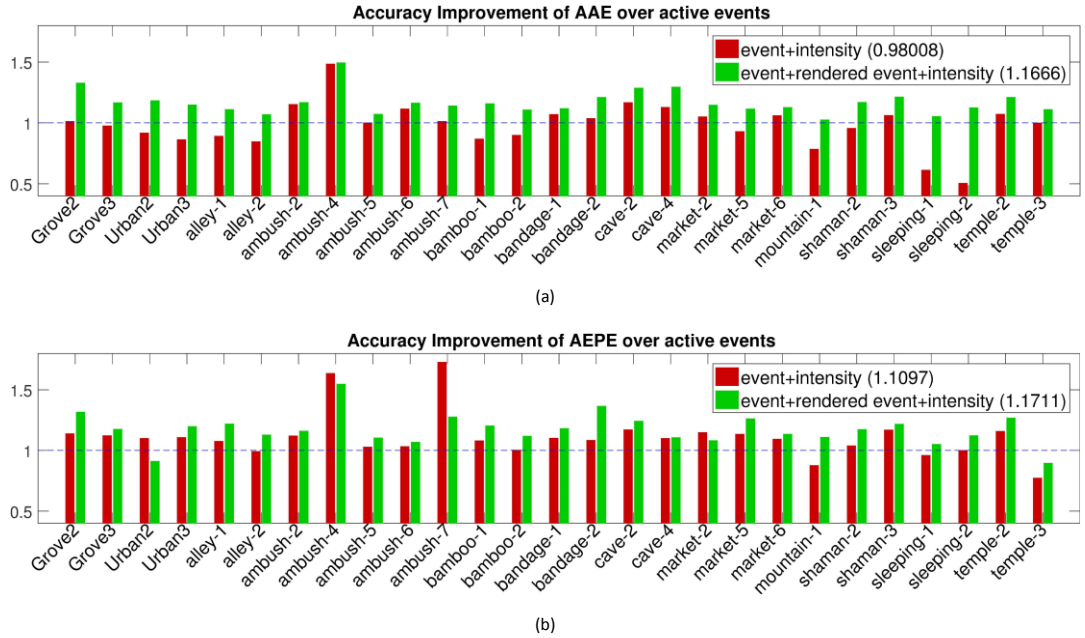


Figure 4.13 Evaluation results of the simulated event sensors in terms of average angle error (AAE) and average endpoint error (AEE).

Using the same measurement method as [138], the flow accuracy is evaluated by computing their average angle error (AAE) and average endpoint error (AEE) based on the open source ground-truth. We collected the accuracy results and made a comparison to statistically show the accuracy improvement due to the optical flow motion sensor in contrast to the other two types of DVS, as shown in Fig.4.13. The value of each bar represents the relative accuracy compared to the result using a normal DVS (Type 1 motion sensor). The measurement result represents the estimation error and thus a smaller value indicates higher accuracy. The relative accuracy evaluation $RAAE$ and $RAEE$ are defined as

$$RAAE\% = \frac{AAE1-AAE2}{AAE1} * 100\% + 1, \quad (4\text{-}12)$$

$$RAEE\% = \frac{AEE1-AEE2}{AEE1} * 100\% + 1, \quad (4\text{-}13)$$

where $AAE1$ or $AEE1$ is the accuracy measurement result using normal DVS. To compute the relative accuracy of GMS, replace $AAE2$ or $AEE2$ with the corresponding accuracy result using GMS, simulated motion sensor with only grayscale events. The relative accuracy of our sensor uses the same equations as Eq. (4-12) and Eq. (4-13), replacing $AAE2$ or $AEE2$ with the accuracy result using PRM-GMS, motion sensor with both asynchronous absolute intensity and pixel rendering module.

The $RAAE$ and $RAEE$ of GMS (legend $event + intensity$) and PRM-GMS (legend $event + rendered\ event + intensity$) are shown as the red bars and green bars respectively in Fig.4.13. Each figure shows the accuracy results for the tested 27 sequences. We take the accuracy result of normal DVS as the baseline, labeled as the blue dashed line in each figure (y = 1). Therefore, GMS or PRM-GMS outperforms the normal DVS if the bar value is larger than one, otherwise underperformed. According to the statistics, the $RAEE$ for GMS equals to 1.1 (slightly higher than one), while its $RAAE$ equals to 0.98, slightly worse than the result of normal DVS. It is observed that, for some sequences, e.g. Urban3, mountain-1 and sleeping, the angle errors of GMS are more significant. For such sequences, the frames selected for testing are observed to have relatively cleaner background. The main difference between GMS and normal DVS is that GMS computes the spatial-temporal derivatives using absolute intensity named as first-order derivative and the normal DVS uses intensity variation which is considered as the second-order derivative. In the scenarios with clean background, the second-order derivative provides more robustness. However, if high texture is present, e.g. ambush and cave sequences, the intensity variation is not able to describe the moving object correctly. In general, the accuracy results of GMS and normal DVS do not exhibit much difference.

In contrast, by analyzing the value of green bars, our sensor improves both AAE and AEE by around 17% (see the legends). The PRM collects intensity information from neighbor pixels to assist in the computation of the flow of central active pixel. Pixels in DVSs become active due to intensity change, which usually occur at

discontinuities such as object contour or texture. It is common for a normal DVS to collect sparse events. These cause accuracy issues in flow estimation, especially when observing fast moving objects. We thus introduce the PRM to collect additional pixel information that assists in estimating the flow of self-activated pixels. Taking use of the intensities of neighboring pixels, the computation of spatial-temporal derivatives is available for even a single event, which benefits both the local feature extraction and global estimation. The test results using synthesized data from 27 sequences meet our expectations. Fig.4.11 shows the flow estimation results of four examples (2 from Middlebury database and 2 from MPI database). From the color coding of flow, the

TABLE VIII ACCURACY AND EVENT-DENSITY FOR SAMPLE SEQUENCES

Seq. name	AAE-DVS	AAE-GMS	AAE-PRM-GMS	AEE-DVS	AEE-GMS	AEE-PRM-GMS	Den-DVS/GMS	Den-PRM-GMS	RD
Grove2	5.048	4.988	3.396	0.342	0.349	0.234	0.186	0.388	2.091
Urban2	6.296	6.818	5.144	1.223	1.384	1.332	0.096	0.169	1.768
bamboo-1	9.69	10.975	8.164	0.732	0.757	0.583	0.187	0.285	1.526
cave-2	7.249	6.038	5.176	3.074	2.724	2.333	0.351	0.604	1.72

AAE-, average angular error; AEE-, average endpoint error; Den-, event density defined as (number of active events during a short time period / image resolution); RD, relative event density (Den-PRM-GMS / Den-DVS or GMS)

accuracy improvement of our sensor is significant. Static accuracy result for some selected sequences is shown in Table VIII.

Compared to conventional frame-based cameras, motion sensors only output motion-related data and greatly reduce the data redundancy. A reduced data size enables lower computational cost and higher processing speed. Another advantage of motion sensor in flow estimation is their high temporal resolution. The data transmission speed of the state-of-art motion sensor is as high as $200Meps$ [174]. They are equivalent to high frame rate cameras as they accumulate events during a short time interval. Taking the 768×640 motion sensor in [174] for example, if 20% pixels detect motion and trigger events during the accumulation time, it generates $0.1M$ events and the motion sensor is thus equivalent of generating 2000 frames per second. The percentage of self-activated pixel (20%) is based on the test result, as shown in Table VIII. The resolution of the test sequences is 640×480 for Middlebury sequences and 1024×436 for MPI sequences. According to the test result, the average event density without pixel

rendering effect is 0.21. For each self-activated pixel, the PRM force-activates its four neighbor pixels, which are inactive, to generate event packages. As a result, it generates five event packages for a single self-activated pixel. The event density considering the pixel rendering effect is also measured through the synthesized data and the average value is 0.34. It shows that our motion sensor generates about 1.7 rather than 5 times events compared to normal DVS. This result is expected as moving objects usually activate multiple pixels in their neighborhood. The intensity of such neighbor pixels can be used directly to compute spatial derivatives and the PRM only generates additional data when there is insufficient data to calculate spatial derivatives.

4.5.3 Flow estimation using real data

In addition to the experiments and analysis on simulated data, experiments were also conducted on real data output from the fabricated optical flow motion sensor. Fig.4.14 shows the whole test platform placed on an optical table in a dark room. The testing contains two parts. In the first part, the target pattern is fixed on a stationary black board.

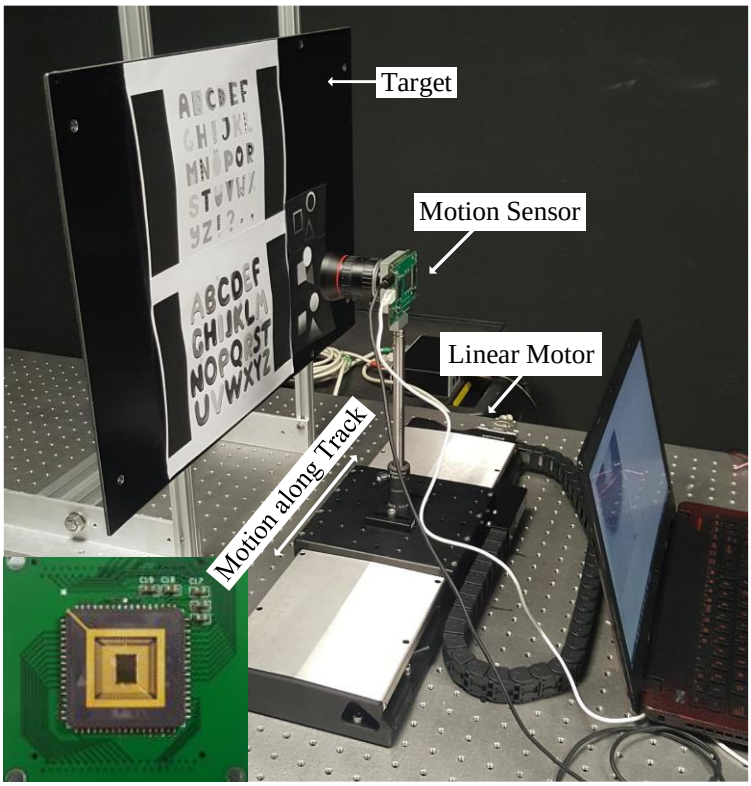


Figure 4.14 Testing platform including the optical flow motion sensor, a static board with testing patterns and a linear motor controlled by a PC. The whole testing platform is placed in a dark room.

The proposed pixel-rendering motion sensor was mounted on a camera holder fixed to a linear motor [175]. The high precision linear motor provides uniform motion with a speed of 0.5 meter per second. In the second part, a high textured background picture was placed on the black board while a magic cube was fixed on the linear motor. By adjusting the camera position, the object is observed to move left and right or front and back.

Fig.4.15 presents samples of flow estimation using our sensor. The full frame image in the first column is generated through the forcefire control signal, which forces all pixels to become active and capture intensity. During motion, event packages are generated continuously in spatial and temporal domain. To estimate the spatial gradients, slices are generated by accumulating events during last $100\,\mu s$. The

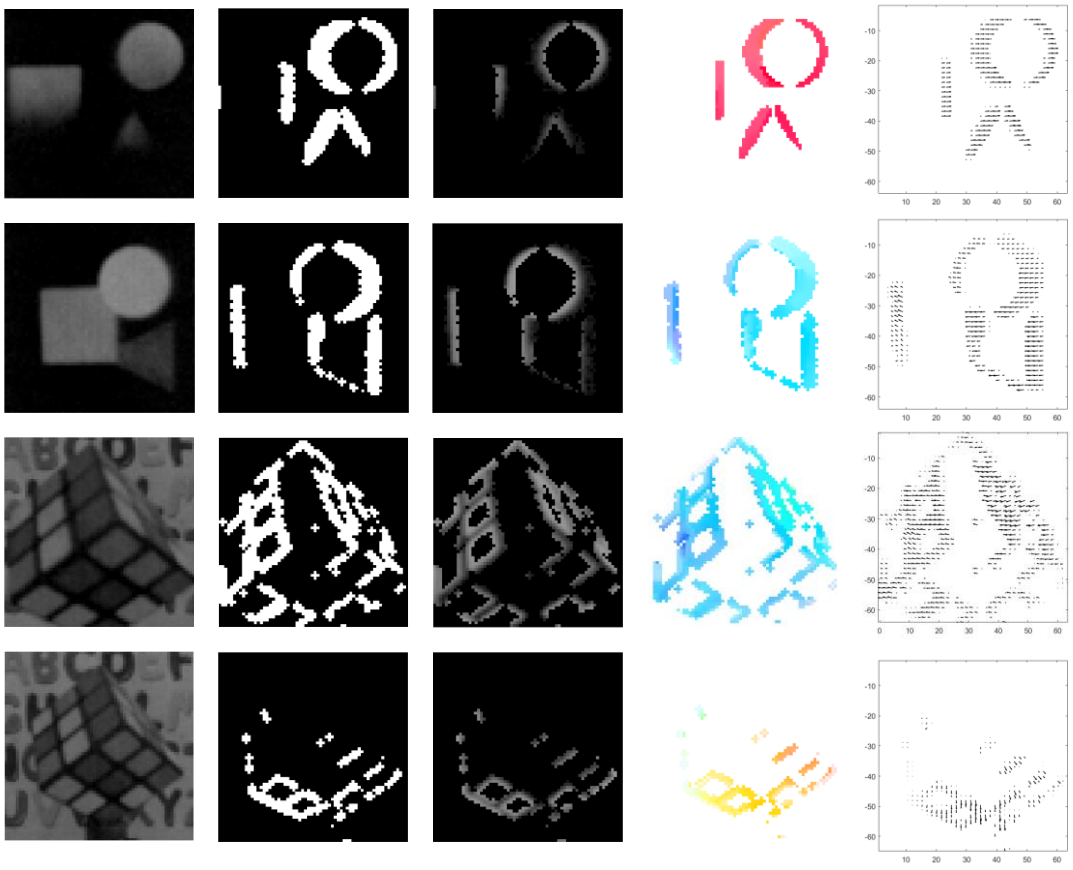


Figure 4.15 Flow estimation results of four examples using real data. Col-1 shows the full frame images of each testing scenario captured by the optical flow motion sensor controlled by the forcefire signal. Col-2 images are example event slices containing the binary events triggered due to moving objects. Col-3 images show the corresponding grayscale events. Col-4 shows the optical flow results in colour coding and Col-5 are the optical flow results in motion vectors.

accumulation time is experimentally determined, such that the accumulated events after noise filtering is able to construct clear features. The second column shows binary value of sample slices. The binary event image is made up of crosses due to the pixel rendering module. The third column shows the respective intensity of accumulated events and is used to compute optical flow. The $classic + NL - fast$ method, same as the method used in simulation experiment, is used to verify the sensor operation. The flow results in color coding are shown in column four and column five shows the flow in vector, indicating clearly the moving direction of each object. The pattern in the first target is made up of a square, a circle, a triangle and a black background. Patterns are gradient-filled as a whole and thus the lower parts of the square and triangle have similar gray value than the printed-black background. Low contrast patterns cannot trigger events and the motion sensor can only detect the upper part of triangle. The sensitivity is adjustable by tuning the comparator thresholds, yet a smaller comparator threshold window introduces more noise as a trade-off. In the third and fourth tests, we used a highly textured background. The motion sensor is able to filter out the static background. During event accumulation, if multiple events are generated at the same pixel position, the latest value is assigned to the pixel. Therefore, accumulating multiple events at the same pixel position is no longer essential since the collected intensity can be directly used to compute spatial derivatives.

4.6 Summary

Event-based flow estimation using a conventional DVS sensor is not sufficiently accurate, especially in scenarios with high texture levels and event sparseness. As a step forward from traditional approaches to processing spatial-temporal events, we proposed the first optical flow motion sensor with pixel rendering mechanism and a capability to evaluate asynchronous absolute intensity, and estimate optical flows for moving objects more accurately. Besides the detection of illumination changes, pixels in the proposed motion sensor are interconnected and communicate event status with each other through a pixel rendering module. Each active pixel together with its four-neighbour pixels report grayscale events to provide sufficient data in gradient extraction which is

essential in flow estimation. Thus, the proposed sensor fundamentally addresses the accuracy problems, which are caused by event sparseness and lack of event intensity, of existing event-driven optical flow estimation. A prototype of the optical flow motion sensor has been designed and fabricated. The increment in pixel circuit complexity results in higher temporal noise and larger fixed-pattern noise variation, which can be reduced by post processing. The proposed sensor offers real-time flow estimation, despite the fact that size of sensor output data (resulting from the pixel rendering mechanism) is about 1.7 times larger than traditional dynamic imagers. Experimental results for flow estimation, conducted on both synthesized data based on Middlebury and MPI database and real data collected from the designed sensor, have been elaborated and discussed in this chapter. The event-flow estimation results show an improvement of 17% in accuracy.

Chapter 5 Conclusions and Future Work

5.1 Conclusion

In the past decade, event-based dynamic vision sensors have drawn increasing attention in the field of machine vision. Real-time performance has always been a major concern for computer vision applications based on standard frame-based imagers, which generate tremendous redundant data leading to high computational cost. As a feasible solution, event-based imagers asynchronously report illumination changes and reduce the input data redundancy by directly filtering out static background. Such imagers have been validated in computer vision tasks, including optical flow and object tracking, with real-time performance. However, the lack of intensity data and the sparseness of events result in accuracy issues. This thesis focuses on the design of smart event-based motion sensors and event-driven algorithms to achieve high accuracy and real-time performance for tracking and optical flow applications.

This work first investigates event-based dynamic sensors and conventional object tracking algorithms. This inspires the proposal of the first hybrid motion sensor generating both frames and asynchronous grayscale events. Besides the illumination change detection functionality, the pixels of the designed hybrid sensor also report instantaneous illumination together with events. Pixel intensity is read out through a system level intensity readout scheme, containing buffering, calibration, amplification and quantization blocks. With the low latency and low power intensity readout scheme, the proposed hybrid motion sensor reduces significantly redundant data from static background significantly and extracts essential information for motion analysis. In contrast to existing event cameras with intensity readout (DAVIS and ATIS cameras), the designed sensor is free of exposure time, contributing to fast-response imaging of high speed objects. In addition, pixels in the proposed sensor are globally controlled to capture full frames on demand, thus benefiting environmental perceptions in computer

vision tasks. The sensor was implemented using AMS 0.35μm 2P4M Opto process with an array of 384×320 pixels. The FPN of sensor can be reduced to 0.31% by a back-end CDS method.

Secondly, an event-guided discriminative tracking method (ESSVM) tailored for the designed hybrid motion sensor is proposed. The proposed algorithm incorporates two event-guiding methods, EPLS and EISS, with a traditional tracking-by-detection algorithm, SSVM, for the task of visual tracking. Compared to the classical trackers, the proposed ESSVM exploits an online adaptive searching area to achieve a more accurate localization. Moreover, the gray-level intensity of event packages generated by the designed hybrid motion sensor is utilized to reconstruct additional samples and update the tracker model over time. The experimental results indicate that the proposed algorithm presents high computational efficiency and outperforms state-of-the-art trackers in terms of accuracy and real-time performance.

The first optical flow motion sensor with pixel rendering mechanism was also designed and implemented. The designed PRM motion sensor is capable of evaluating asynchronous absolute intensity and estimating optical flow of moving objects with higher accuracy. Besides the detection of illumination changes, pixels in the proposed motion sensor are interconnected and communicate event status with each other through a pixel rendering module. Each active pixel together with its four-neighbour pixels report grayscale events to provide sufficient data in gradient extraction, which is essential in flow estimation. Thus, the proposed sensor fundamentally addresses accuracy problems, which are caused by event sparseness and lack of event intensity, of existing event-driven optical flow estimation. A 64×64 prototype of the PRM motion sensor was fabricated and implemented using AMS 0.35μm 2P4M Opto process. The total chip area is $4.2 \times 5.9\ mm^2$ with pixel pitch $40\mu m$. The temporal noise and fixed-pattern noise variation caused by the complex pixel circuit can be significantly reduced by post processing. Exploiting the pixel rendering mechanism, the proposed sensor is able to perform real-time flow estimation even though the sensor output data size is 1.7 times larger than that of the traditional dynamic imagers. Experiments were conducted on both synthesized data based on Middlebury and MPI

database and real data collected from the designed sensor, showing an improvement (by 17%) in event-flow estimation accuracy.

5.2 Future Research

Motion analysis has always been a challenge in computer vision applications. The proposed two motion sensors and event-driven algorithm have shown significant improvement in accuracy and real-time performance of object tracking and optical flow estimation. However, the proposed sensors and sensor-processing systems are not free of issues:

(1) The performance of proposed motion sensors can be improved if a more advanced technology node is used. The improvements include higher area efficiency, lower power consumption and larger pixel fill factor leading to higher light sensitivity. In addition, more advanced technology node usually combines additional potential advantages, e.g. micro lenses, anti-reflection coating, and annealing steps that reduce junction leakage currents. The improvements on image sensor performance will benefit the accuracy of post processing.

(2) The timestamps used in the applications are those stamped by FPGA when the events are collected. These timestamps are not exactly the moment when pixels are triggered because multiple pixels can become active simultaneously but can only be read out in serial. Therefore, there is still mismatch between the event intensity and time information even though it is ignorable for small resolution. If the design is scaled to large resolution, it is essential to design a pixel level timestamp generation circuitry to precisely record event timing.

(3) Higher circuit complexity of the pixels in optical flow motion sensor brings higher temporal and fixed pattern noise. The layout design can be improved to separate digital circuit and analog circuit as far as possible. One possible smart design is shown in figure 5.1, digital and analog circuits can be respectively placed together through flipping pixels horizontally and vertically, while the photodiodes are placed in the centre to guarantee uniform distribution.

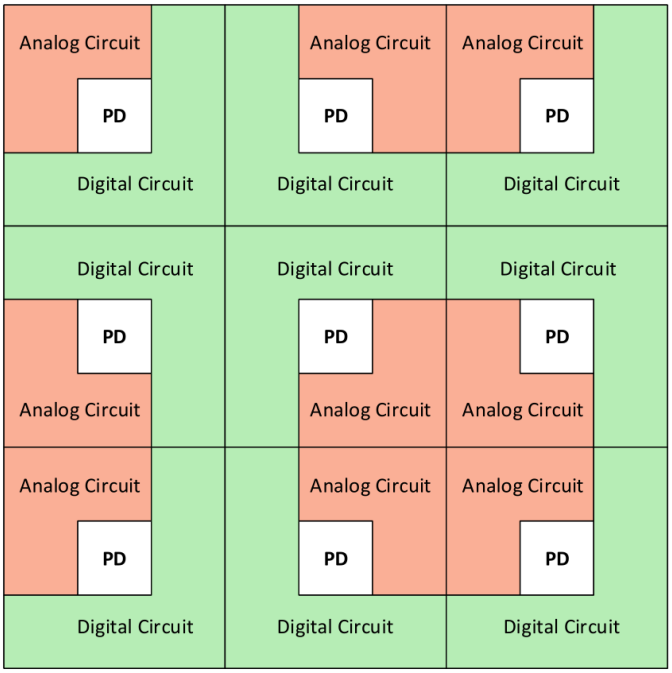


Figure 5.1 An example 3x3 array pixels with smart design

(4) Another typical problem is that output data of the motion sensor is highly dependent on illumination conditions. In the field of object tracking, the proposal of event-guided tracking algorithm is based on the assumption that the output events of the motion sensor are always related to target motion. However, motion sensors asynchronously response to illumination changes which may also be caused by static objects with varying illuminations. As a result, the proposed tracking system has superior performance on tracking of single or multiple moving objects with static background. Nevertheless, the proposed event-guiding method for adaptive searching area will not be applicable when both background and foreground targets are moving simultaneously.

Other future work concerns the deeper analysis of neuromorphic sensing system and its corresponding applications in machine vision tasks. For object tracking, experiments using both events and frames generated from the designed hybrid motion sensor can be further explored in the future. In the meanwhile, an event-guided tracking method based on neural networks is being developed. For optical flow estimation on sensor collected data, the performance could be better evaluated by quantitative analysis with ground-truth optical flow. Future research might dig into the development of machine vision algorithms, which are adaptive to neuromorphic vision.

Author's Publications

Journal Papers

(i) Huang, J., Guo, M., Wang, S., & Chen, S. A 64 x 64 Motion Sensor with On-Chip Pixel Rendering Module for Optical Flow Estimation. Submitted and under review.

(ii) Huang, J., Wang, S., Guo, M., & Chen, S. An Event-guided Structured Output Tracking of Fast Moving Objects using CeleX Sensor. Submitted and under review.

Conference Papers

(i) Huang, J., Guo, M., Wang, S. & Chen, S. (2018, May). A Motion Sensor with On-Chip Pixel Rendering for Optical Flow Gradient Extraction. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE. Approved.

(ii) Huang, J., Guo, M., & Chen, S. (2017, May). A dynamic vision sensor with direct logarithmic output and full-frame picture-on-demand. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on* (pp. 1-4). IEEE.

(iii) Guo, H., Huang, J., Guo, M., & Chen, S. (2016, May). Dynamic resolution event-based temporal contrast vision sensor. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on* (pp. 1422-1425). IEEE.

Bibliography

[1] Singh, S. (2015). *Critical reasons for crashes investigated in the national motor vehicle crash causation survey* (No. DOT HS 812 115).

[2] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 263–270.

[3] X. Wang, G. Hua, and T. X. Han, "Discriminative tracking by metric learning," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 200–214.

[4] Z. Kalal, J. Matas, and K. Mikolajczyk, "Pn learning: Bootstrapping binary classifiers by structural constraints," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 49–56.

[5] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 864–877.

[6] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Object tracking with joint optimization of representation and classification," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 25, no. 4, pp. 638–650, 2015.

[7] W. Shuigen, C. Zhen, L. Ming and Z. Liang, "An improved method of motion detection based on temporal difference," IEEE International Workshop on Intelligent Systems and Applications, Wuhan, pp. 1-4, 2009.

[8] Wu, Y., Pei, M., Yang, M., Yuan, J., & Jia, Y. (2015). Robust discriminative tracking via landmark-based label propagation. *IEEE Transactions on Image Processing*, *24*(5), 1510-1523.

[9] Ma, B., Huang, L., Shen, J., & Shao, L. (2016). Discriminative tracking using tensor pooling. *IEEE transactions on cybernetics*, *46*(11), 2411-2422.

[10] Hong, S., You, T., Kwak, S., & Han, B. (2015, June). Online tracking by learning discriminative saliency map with convolutional neural network. In *International Conference on Machine Learning* (pp. 597-606).

[11] Lipton, Alan J. *Local application of optic flow to analyze rigid versus non-rigid motion*. Carnegie Mellon University, The Robotics Institute, 1999.

[12] Chauhan, Abhishek Kumar, and Prashant Krishan. "Moving object tracking using gaussian mixture model and optical flow." *International Journal of Advanced Research in Computer Science and Software Engineering* 3.4 (2013).

[13] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[14] Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernández, G., ... & Pflugfelder, R. (2017). The visual object tracking 2017 challenge

results. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 1-24).

[15] Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, *17*(1-3), 185-203.

[16] Lucas, B. D., & Kanade, T. (1981, August). An iterative image registration technique with an application to stereo vision. In *IJCAI* (Vol. 81, No. 1, pp. 674-679).

[17] Barron, J. L., Fleet, D. J., & Beauchemin, S. S. (1994). Performance of optical flow techniques. *International journal of computer vision*, *12*(1), 43-77.

[18] Fleet, D., & Weiss, Y. (2006). Optical flow estimation. In *Handbook of mathematical models in computer vision* (pp. 237-257). Springer US.

[19] Sun, D., Roth, S., & Black, M. J. (2014). A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, *106*(2), 115-137.

[20] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., & Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, *92*(1), 1-31.

[21] Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., & Delbruck, T. (2014). Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, *102*(10), 1470-1484.

[22] Liu, S. C. (2015). *Event-based neuromorphic systems*. John Wiley & Sons.

[23] Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128× 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, *43*(2), 566-576.

[24] Delbrück, T., Linares-Barranco, B., Culurciello, E., & Posch, C. (2010, May). Activity-driven, event-based vision sensors. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (pp. 2426-2429). IEEE.

[25] Ni, Z., Ieng, S. H., Posch, C., Régnier, S., & Benosman, R. (2015). Visual tracking using neuromorphic asynchronous event-based cameras. *Neural computation*.

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

[27] Barranco, F., Fermüller, C., & Aloimonos, Y. (2014). Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, *102*(10), 1537-1556.

[28] Barranco, F., Fermuller, C., & Aloimonos, Y. (2015, June). Bio-inspired motion estimation with event-driven sensors. In *International Work-Conference on Artificial Neural Networks* (pp. 309-321). Springer International Publishing.

[29] Brosch, T., Tschechne, S., & Neumann, H. (2015). On event-based optical flow detection. *Frontiers in neuroscience*, *9*, 137.

[30] Ohta, Jun. *Smart CMOS image sensors and applications*. CRC press, 2007.

[31] W. S. Boyle; G. E. Smith (April 1970). "Charge Coupled Semiconductor Devices". *Bell Syst. Tech. J.* 49 (4): 587–593.

[32] Calebotta, Stephen. "CMOS, the Ideal Logic Family." *National Semiconductor CMOS Databook, Rev* 1 (1975): 2-3.

[33] G. Burgett, (2015, Sep 8). *Canon has developed an insane 250MP sensor with almost half a million pixels per square millimeter*, Retrieved from http://www.imaging-resource.com/news/2015/09/08/canon-has-developed-an-insane-250mp-sensor-with-almost-half-a-million-pixel

[34] D. Kearney (2017, Oct 25), "Newly Released, Industry's Highest Resolution CMOS Image Sensor for Automotive Cameras". *Products & Service News*, retrieved from https://incompliancemag.com/newly-released-industrys-highest-resolution-cmos-image-sensor-for-automotive-cameras/

[35] C. Mead, (1989). Analog VLSI and Neural Systems. Reading, MA, USA: Addison-Wesley.

[36] Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, *78*(10), 1629-1636.

[37] Maher, M. A. C., Deweerth, S. P., Mahowald, M. A., & Mead, C. A. (1989). Implementing neural architectures using analog VLSI circuits. *IEEE Transactions on circuits and systems*, *36*(5), 643-652.

[38] G. Daniel, Stugyblue website. Retrieved from https://www.studyblue.com

[39] Mahowald, M. (1994). The silicon retina. In *An Analog VLSI System for Stereoscopic Vision* (pp. 4-65). Springer US.

[40] Zaghloul, K. A., & Boahen, K. (2004). Optic nerve signals in a neuromorphic chip II: Testing and results. *IEEE Transactions on Biomedical Engineering*,*51*(4), 667-675.

[41] Grenet, E., Gyger, S., Heim, P., Heitger, F., Kaess, F., Nussbaum, P., & Ruedi, P. F. (2005, February). High dynamic range vision sensor for automotive applications. In *European Workshop on Photonics in the Automobile* (pp. 246-253). International Society for Optics and Photonics.

[42] Culurciello, E., & Andreou, A. G. (2006). CMOS image sensors for sensor networks. *Analog Integrated Circuits and Signal Processing*, *49*(1), 39-51.

[43] Moini, A. (2000). Vision Chips. Kluwer Academic Publishers.

[44] Picture retrieved from online source, https://inilabs.com/.

[45] R. Xu, W. C. Ng, J. Yuan, S. Yin, and S. Wei, "A 1/2.5-inch VGA 400 fps CMOS image sensor with high sensitivity for machine vision," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 342 – 2351, Oct. 2014.

[46] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and

time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259 – 275, Jan. 2011.

[47] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6 s latency asynchronous frame-free event-driven dynamic- vision-sensor," *IEEE J. Solid-State Circuits*, vol. 46, no. 6, pp. 1443 – 1455, Jun. 2011.

[48] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128 × 128 1.5% contrast sensitivity 0.9% FPN 3s latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 827 – 838, Mar. 2013.

[49] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 130dB latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333 – 2341, Oct. 2014.

[50] Berner, R., Brandli, C., Yang, M., Liu, S. C., & Delbruck, T. (2013, June). A 240× 180 10mW 12us latency sparse-output vision sensor for mobile applications. In *2013 Symposium on VLSI Circuits* (pp. C186-C187). IEEE.

[51] Yang, Minhao, Shih-Chii Liu, and Tobi Delbruck. "A dynamic vision sensor with 1% temporal contrast sensitivity and in-pixel asynchronous delta modulator for event encoding." *IEEE Journal of Solid-State Circuits* 50.9 (2015): 2149-2160.

[52] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1195–1202.

[53] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1910–1917.

[54] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2042–2049.

[55] T. Liu, G. Wang, L. Wang, and K. L. Chan, "Visual tracking via temporally smooth sparse coding," 2012.

[56] V. Belagiannis, F. Schubert, N. Navab, and S. Ilic, "Segmentation based particle filtering for real-time 2d object tracking," in *Computer Vision– ECCV 2012*. Springer, 2012, pp. 842–855.

[57] S. Kwak, W. Nam, B. Han, and J. H. Han, "Learning occlusion with likelihoods for visual tracking," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1551–1558.

[58] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2411–2418.

[59] Wu, Y., Lim, J., & Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1834-1848.

[60] Liu, H., Moeys, D. P., Das, G., Neil, D., Liu, S. C., & Delbrück, T. (2016, May). Combined frame-and event-based detection and tracking. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on* (pp. 2511-2514). IEEE.

[61] S. R. Balaji and S. Karthikeyan, "A survey on moving object tracking using image processing," *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, 2017, pp. 469-474.

[62] Singla, Nishu. "Motion detection based on frame difference method." *International Journal of Information & Computation Technology* 4.15 (2014): 1559-1565.

[63] K. Suganya Devi, N. Malmurugan, M. Manikandan, *Object Motion Detection in Video Frames Using Background Frame Matching*, International Journal of Computer Trends and Technology, Vol.4, Issue 6, pp 1928-1931, June 2013.

[64] Casares, M., Velipasalar, S., & Pinto, A. (2010). Light-weight salient foreground detection for embedded smart cameras. *Computer Vision and Image Understanding*, *114*(11), 1223-1237.

[65] OpenCV 3.0. "How to use background subtraction method". Retrieved from https://docs.opencv.org/3.2.0/d1/dc5/tutorial_background_subtraction.html

[66] RupaliS.Rakibe, BharatiD.Patil, *Background Subtraction Algorithm Based Human Motion Detection*, International Journal of Scientific and Research Publications, vol. 3, Issue 5, pp. 14, May2009.

[67] Kim, Intaek, Tayyab Wahab Awan, and Youngsung Soh. "Background subtraction-based multiple object tracking using particle filter." *Systems, Signals and Image Processing (IWSSIP), 2014 International Conference on*. IEEE, 2014.

[68] Tang, Zheng, et al. "Multiple-kernel adaptive segmentation and tracking (MAST) for robust object tracking." *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016.

[69] M. H. Ali, Fadhan Hafiz, A. A Shafie, *Motion Detection Techniques using Optical Flow*, World Academy of Science, Engineering and Technology, vol. 32, pp 559-561, 2009.

[70] Lan, Yongtian, et al. "Robot fish detection based on a combination method of three-frame-difference and background subtraction." China Control and Decision Conference, 2014:3905-3909.

[71] Chen J C, Zhang J H, Liu S J, et al. Improved Target Detection Algorithm Based on Background Modeling and Frame Difference[J]. Computer Engineering, 2011.

[72] Yuan, Guo Wu, et al. "A Moving Object Detection Algorithm Based on a Combination of Optical Flow and Three-Frame Difference." Journal of Chinese Computer Systems 34.3(2013):668-671.

[73] Guo, Jiajia, et al. "A New Moving Object Detection Method Based on Frame-difference and Background Subtraction." *IOP Conference Series: Materials Science and Engineering*. Vol. 242. No. 1. IOP Publishing, 2017.

[74] R. N. Hota, V. Venkoparao and A. Rajagopal, “Shape based object classification for automated video surveillance with feature selection,” IEEE, 10th International Conference on Information Technology, Orissa, pp. 97-99 Dec 2007.

[75] Phyo, Cho Nilar, et al. "Color and Shape based Method for Detecting and Classifying Card Images." (2017).

[76] Cordova, Jose, et al. "Shape-based data analysis for event classification in power systems." *PowerTech, 2017 IEEE Manchester*. IEEE, 2017.

[77] Doyle, Daniel D., Alan L. Jennings, and Jonathan T. Black. "Optical flow background estimation for real-time pan/tilt camera object tracking." *Measurement* 48 (2014): 195-207.

[78] Kolekar, Maheshkumar H., and Deba Prasad Dash. "Hidden Markov Model based human activity recognition using shape and optical flow based features." *Region 10 Conference (TENCON), 2016 IEEE*. IEEE, 2016.

[79] S. Sergyn, “Color content-based image classification,” 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics, Poprad, Slovakia, Jan 2007.

[80] Kumar, Mehta Priyanka Hemant, and Dr. Nimesh I. Modi. "A Survey on Content Based Image Retrieval System using Color and Texture." (2017).

[81] David T., “Visualizing the 3D point cloud of RGB colours.” (2014). Retrieved from http://opensource.graphics/visualizing-the-3d-point-cloud-of-rgb-colors/.

[82] M. J. Swain and D. H. Ballard, “Color indexing,” *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.

[83] MS. Pragati Ashok Deole, PROF. Rushi Lonngadge, “Content Based Image Retrieval using Color Feature Extraction with KNN Classification”, International Journal of Computer Science and Mobile Computing, IJCSMC, Vol. 3, Issue. 5, May 2014, pg.1274 – 1280.

[84] S. Wang, “A robust CBIR approach using local color histograms,” Master’s thesis, University of Alberta, 2001.

[85] X.-Y. Wang, J.-F. Wu, and H.-Y. Yang, “Robust image retrieval based on color histogram of local feature regions,” *Multimedia Tools Appl.*, vol. 49, pp. 323–345, August 2010.

[86] J. Han and K.-K. Ma, “Fuzzy color histogram and its use in color image retrieval,” *IEEE Transactions on Image Processing*, vol. 11, no. 8, pp. 944–952, 2002.

[87] J. Huang, S. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, “Image indexing using color correlograms,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR ’97)*, pp. 762 –768, Jun 1997.

[88] Q. Zhao and H. Tao, “Object tracking using color correlogram,” in *Proceedings of the 14th International Conference on Computer Communications and Networks*, (Washington, DC, USA), pp. 263–270, IEEE Computer Society, 2005.

[89] Yuntao Qian, Minchao Ye, and Jun Zhou. "Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features." *IEEE Transactions on Geoscience and Remote Sensing* 51.4 (2013): 2276-2291.

[90] Brewster, E., J. M. Keller, and M. Popescu. "A new approach for extracting texture features to aid detection of explosive hazards using synthetic aperture acoustic sensing." *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXII*. Vol. 10182. International Society for Optics and Photonics, 2017.

[91] Abrah&atildeo, W., et al. "A comparison of Haar-like, LBP and HOG approaches to concrete and asphalt runway detection in high resolution imagery."

[92] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

[93] Wang, Xiaoyu, Tony X. Han, and Shuicheng Yan. "An HOG-LBP human detector with partial occlusion handling." *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009.

[94] Zeng, Haihua, et al. "Visual tracking using multi-channel correlation filters." *Digital Signal Processing (DSP), 2015 IEEE International Conference on*. IEEE, 2015.

[95] Viola and Jones, "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001

[96] Ng, Andrew Y., and Michael I. Jordan. *"On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes."* Advances in neural information processing systems. 2002.

[97] Li, Xi, et al. "A survey of appearance models in visual object tracking." *ACM transactions on Intelligent Systems and Technology (TIST)* 4.4 (2013): 58.

[98] P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.

[99] T. Srivastava, "Introduction to k-nearest neighbors: Simplified," *Analytics Vidhya* (2014), https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/

[100] Vapnik, Vladimir. *The nature of statistical learning theory*. Springer science & business media, 2013.

[101] Kavzoglu, Taskin, and I. Colkesen. "A kernel functions analysis for support vector machines for land cover classification." *International Journal of Applied Earth Observation and Geoinformation* 11.5 (2009): 352-359.

[102] Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., & Shah, M. (2014). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(7), 1442-1468.

[103] H. T. Nguyen and A. W. M. Smeulders, "Fast occluded object tracking by a robust appearance filter," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1099–1104, Aug. 2004.

[104] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE CVPR*, Hilton Head Island, SC, USA, 2000.

[105] D. A. Ross, J. Lim, and R. S. Lin, "Incremental learning for robust visual tracking," *IJCV*, vol. 77, no. 1–3, pp. 125–141, 2008.

[106] J. Kwon and F. C. Park, "Visual tracking via geometric particle filtering on the affine group with optimal importance functions," in *Proc. IEEE CVPR*, Miami, FL, USA, 2009.

[107] E. Maggio and A. Cavallaro, "Tracking by sampling trackers," in *Proc. IEEE ICCV*, Barcelona, Spain, 2011, pp. 1195–1202.

[108] J. Kwon and K. M. Lee, "Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling," in *Proc. IEEE CVPR*, Miami, FL, USA, 2009.

[109] E. Maggio and A. Cavallaro, *Video Tracking: Theory and Practice*. 1st ed. Oxford, U.K.: Wiley, 2011.

[110] X. Mei and H. Ling, "Robust visual tracking using L1 minimization," in *Proc. IEEE 12th ICCV*, Kyoto, Japan, 2009.

[111] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Proc. IEEE ICCV*, Barcelona, Spain, 2011.

[112] R. Collins, X. Zhou, and S. K. Teh, "An open source tracking testbed and evaluation web site," in Proc. IEEE Int. Workshop Perform. Eval. Tracking Surveillance, 2005, pp. 17–24.

[113] R. B. Fisher, "The PETS04 surveillance ground-truth data sets," in Proc. IEEE Int. Workshop Perform. Eval. Tracking Surveillance, 2004, pp. 1–5.

[114] J. Ferryman and A. Shahrokni, "PETS 2009: Dataset and challenge," in Proc. IEEE Int. Workshop Perform. Eval. Tracking Surveillance, 2009, pp. 1–6.

[115] B. Karasulu and S. Korukoglu, "A software for performance evaluation and comparison of people detection and tracking methods in video processing," *MTA*, vol. 55, no. 3, pp. 677–723, 2011.

[116] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear MOT metrics," *EURASIP J. IVP*, vol. 2008, no. 1, p. 246309, Feb. 2008.

[117] R. Kasturi *et al.*, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 319–336, Feb. 2009.

[118] A. Sanin, C. Sanderson, and B. C. Lovell, "Shadow detection: A survey and comparative evaluation of recent methods," *PR*, vol. 45, no. 4, pp. 1684–1695, 2012.

[119] B. Babenko, M.-H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. PAMI, 33(7):1619–1632, 2011.

[120] Hu, Y., Liu, H., Pfeiffer, M., & Delbruck, T. (2016). Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in neuroscience*, *10*.

[121] Von Helmholtz, H. (1925). *Helmholtz's treatise on physiological optics* (Vol. 3). Optical Society of America.

[122] Gibson, J.J. (1966). The senses considered as perceptual systems. Houghton Mifflin, Boston.

[123] Zufferey, F, Srinivasan, S. (2006). Flying Insects and Robotics.

[124] Stocker, A, Wiley, &Sons. (2006). Analog VLSI Circuits for the Perception of Visual Motion.

[125] APM 2.5 2.6 2.8 optical flow sensor Optical Flow v10 no GPS point. Retrieved from http://www.aliexpress.com/item/Optical-Flow-v10-Sensor-No-GPS-Point-for-APM-2-5-2-6-2-8-Flight/32276453701.html

[126] PX4 autopilot. Retrieved from https://pixhawk.org/modules/px4flow

[127] Owens, R. (1997, Oct 29). *Optical flow*. Retrieved from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html#SECTION00040000000000000000

[128] Black, M., & Anandan, P. (1996). The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. Computer Vision and Image Understanding, CVIU, 75–104.

[129] Memin, E., & Perez, P. (1998). Joint estimation-segmentation of optic flow. In ECCV'98.

[130] Nagel, H. H. (1987). On the estimation of optical flow: Relations between different approaches and some new results. *Artificial intelligence*, *33*(3), 299-324.

[131] Uras S., Girosi F., Verri A. and Torre V. (1988) A computational approach to motion perception. Biol. Cybern. 60, pp. 79-97

[132] T. Amiaz, E. Lubetzky, and N. Kiryati. Coarse to over-fine optical flow estimation. Pattern Recognition, 40(9):2496–2503, 2007.

[133] Anandan, P. (1987). Measuring visual motion from image sequences.

[134] Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*,*2*(3), 283-310.

[135] Heeger, D. J. (1987). Model for the extraction of image flow. *JOSA A*, *4*(8), 1455-1471.

[136] Heeger, D. J. (1988). Optical flow using spatiotemporal filters. *International journal of computer vision*, *1*(4), 279-302.

[137] Pauwels, K., & Van Hulle, M. M. (2008, June). Realtime phase-based optical flow on the GPU. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on* (pp. 1-8). IEEE.

[138] S. Baker, M. J. Black, J. Lewis, S. Roth, D. Scharstein, and R. Szeliski. A database and evaluation methodology for optical flow. In International

Conference on Computer Vision ICCV), pages 1–8, Rio de Janeiro, Brazil, October 2007.

[139] D. J. Butler, J.Wul_, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In European Conference on Computer Vision (ECCV), pages 611–625. Springer-Verlag, 2012.

[140] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In Computer Vision and Pattern Recognition (CVPR), pages 3354–3361, 2012.

[141] Brox, T., Bruhn, A., Papenberg, N., & Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In LNCS: Vol. 3024. European conference on computer vision, ECCV, (pp. 25–36). Prague, Czech Republic: Springer.

[142] Fransens, R., Strecha, C., & Gool, L. V. (2007). Optical flow based super-resolution: a probabilistic approach. Computer Vision and Image Understanding, CVIU, 106(1), 106–115.

[143] Benosman, R., Ieng, S. H., Clercq, C., Bartolozzi, C., & Srinivasan, M. (2012). Asynchronous frameless event-based optical flow. *Neural Networks*,*27*, 32-37.

[144] Benosman, R., Clercq, C., Lagorce, X., Ieng, S. H., & Bartolozzi, C. (2014). Event-based visual flow. *IEEE transactions on neural networks and learning systems*, *25*(2), 407-417.

[145] Chun-Cheng Liu; Soon-Jyh Chang; Guan-Ying Huang; Ying-Zu Lin, "A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure," IEEE JSSC, vol.45, no.4, pp.731-740, April 2010.

[146] Shin, M.-S.; Kwon, O.-K., "14-bit two-step successive approximation ADC with calibration circuit for high resolution CMOS imagers," Electronics Letters, vol.47, no.14, pp.790-791, July 2011.

[147] Assaad, R.S.; Silva-Martinez, J., "The Recycling Folded Cascode: A General Enhancement of the Folded Cascode Amplifier," in *IEEE Journal of Solid-State Circuits*, vol.44, no.9, p.2535-2542, Sept. 2009

[148] Kristan M, Leonardis A, Matas J, Felsberg M, Pflugfelder R, ehovin L, Vojr T, Hger G, Lukei A, Fernandez Dominguez G, Gupta A, Petrosino A, Memarmoghadam A, Garcia-Martin A, Sols Montero A, Vedaldi A, Robinson A, Ma A, Varfolomieiev A and Chi Z. (2016). The Visual Object Tracking VOT2016 challenge results. 777-823.

[149] M. Mueller, N. Smith, and B. Ghanem. A benchmark and simulator for uav tracking. In ECCV, pages 445461, 2016.

[150] Galoogahi, H. K., Fagg, A., Huang, C., Ramanan, D., & Lucey, S. (2017). Need for Speed: A Benchmark for Higher Frame Rate Object Tracking. *arXiv preprint arXiv:1703.05884*.

[151] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M. M., Hicks, S. L., & Torr, P. H. (2016). Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, *38*(10), 2096-2109.

[152] Zhang, S., Sui, Y., Zhao, S., & Zhang, L. (2017). Graph-Regularized Structured Support Vector Machine for Object Tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, *27*(6), 1249-1262.

[153] A. Bordes, L. Bottou, P. Gallinari, and J. Weston, "Solving multiclass support vector machines with LaRank," in ICML, 2007.

[154] J. C. Platt, Fast Training of Support Vector Machines Using Sequential Minimal Optimization. MIT Press, 1999, pp. 185–208.

[155] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang. Hedged deep tracking. In CVPR, pages 43034311, 2016.

[156] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In ECCV, pages 850865, 2016.

[157] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In CVPR, June 2016.

[158] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In ICCV, pages 31193127, 2015.

[159] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In CVPR, pages 30743082, 2015.

[160] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Highspeed tracking with kernelized correlation filters. PAMI, 37(3):583596, 2015.

[161] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In CVPR, pages 53885396, 2015.

[162] J. Zhang, S. Ma, and S. Sclaroff. MEEM: robust tracking via multiple experts using entropy minimization. In ECCV, pages 188203, 2014.

[163] M. Danelljan, G. Hager, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In BMVC, 2014.

[164] Watson, Andrew B., and Albert J. Ahumada Jr. "A look at motion in the frequency domain." (1983): pp. 1-10.

[165] Sutton, M. A., et al. "Determination of displacements using an improved digital correlation method." Image and vision computing 1.3 (1983): 133-139.

[166] Banks, Jasmine, and Peter Corke. "Quantitative evaluation of matching methods and validity measures for stereo vision." The International Journal of Robotics Research 20.7 (2001): 512-532.

[167] Baker, Simon, and Iain Matthews. "Lucas-kanade 20 years on: A unifying framework." International journal of computer vision 56.3 (2004): 221-255.

[168] Blaschko, M. B., & Lampert, C. H. (2008, October). Learning to localize objects with structured output regression. In *European conference on computer vision* (pp. 2-15). Springer, Berlin, Heidelberg.

[169] Blake, Andrew, Pushmeet Kohli, and Carsten Rother, eds. Markov random fields for vision and image processing. Mit Press, 2011.

[170] Komodakis, Nikos, Nikos Paragios, and Georgios Tziritas. ”MRF energy minimization and beyond via dual decomposition.” IEEE transactions on pattern analysis and machine intelligence 33.3 (2011): 531-552.

[171] Chen, Qifeng, and Vladlen Koltun. ”Full flow: Optical flow estimation by global optimization over regular grids.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[172] Rueckauer, Bodo, and Tobi Delbruck. ”Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor.” Frontiers in neuroscience 10 (2016).

[173] Konrad, Janusz. ”Multigrid Bayesian estimation of image motion fields using stochastic relaxation.” Proc. 2rd Int. Conf. Computer Vision. 1988.

[174] Jing Huang, Menghan Guo and Shoushun Chen,”A Dynamic Vision Sensor with Direct Logarithmic Output and Full-Frame Picture-on-Demand,” IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, US, May 2017.

[175] “PImicos Motion Control.” Available: http://www.piusa.us/pdf/Motion_Control_Catalog_PI_miCos.pdf, pp. 90-91.

[176] Boahen KA. 1999.Athroughput-on-demand address-event transmitter for neuromorphic chips. Proc. 20th Anniversary Conf. Adv. Res. VLSI, Atlanta, GA, pp. 72 86.

[177] Mahowald M. 1994. An Analog VLSI System for Stereoscopic Vision. Kluwer Academic, Boston.

[178] Boahen KA. 2000. Point-to-point connectivity between neuromorphic chips using address-events. IEEE Trans. Circuits Syst. II 47(5), 416 434.

[179] Martin AJ and Nystrom M. 2006. Asynchronous techniques for system-on-chip design. Proc. IEEE 94(6), 1089 1120.

[180] Boahen KA. 2004. A burst-mode word-serial address-event link I: transmitter design. IEEE Trans. Circuits Syst. I: Reg. Papers 51(7), 1269-1300.

[181] Aung Myat Thu Linn, Anh Tuan Do, Shoushun Chen and Kiat Seng Yeo, "Adaptive Priority Toggle Asynchronous Tree Arbiter for AER-based Image Sensor, " IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SOC), pp. 66 - 71, Hong Kong, Oct. 2011.