A biologically inspired human posture recognition system

Zhao, Bo

2014

Zhao, B. (2014). A biologically inspired human posture recognition system. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/59546>

<https://doi.org/10.32657/10356/59546>

A BIOLOGICALLY INSPIRED HUMAN POSTURE RECOGNITION SYSTEM

ZHAO BO

School of Electrical and Electronic Engineering

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

2014

Acknowledgements

I would like to take this chance to thank Assistant Professor Chen Shoushun, for his advices on my research and for leading me into the world of neuromorphic vision processing. I also would like to thank him for sharing his scientific thinking and inspiring me with new ideas and constructive suggestions.

Thanks are extended to the following members in the same research group with me: Zhang Xiangyu, Qian Xinyuan and Yu Hang. I would like to thank them for numerous discussions and various contributions in my research work.

Thanks also go to my lab mates and friends, for making my study and research in VIRTUS IC Design Center of Excellence enjoyable.

Last but not least, thank my family for their persistent love and support.

Contents

Acknowledgements	i
List of Figures	vi
List of Tables	x
List of Abbreviations	xi
Summary	xiv
1 Introduction	1
1.1 Motivation and Objective	1
1.1.1 Application of Human Posture Recognition	1
1.1.2 Challenges and Progress of Human Posture Recognition	2
1.1.3 Event-based Processing and Neuromorphic Systems	4
1.1.4 Objective and Contribution	7
1.2 Report Organization	8
2 Literature Review	10
2.1 Data Acquisition	10
2.1.1 Wearable Sensors	11
2.1.2 Vision-based Sensors	12
2.1.3 Summary	17
2.2 Preprocessing	18
2.3 Feature Extraction	19

2.3.1	Color, Texture and Shape	19
2.3.2	Shape-based Representations	22
2.3.3	Motion-based Representations	31
2.3.4	Handcrafted vs. Biologically Plausible	32
2.3.5	Biologically Inspired Models	33
2.4	Classification	40
2.5	Summary	44

3	Bio-inspired Human Posture Recognition using Temporal Difference Im- age Sensor	46
3.1	System Architecture	47
3.2	Temporal Difference Image Sensor	47
3.3	Bio-inspired Line Feature Extraction	50
3.3.1	Proposed MAX-like Convolutional Network For Line Extraction .	51
3.3.2	Simulation Results	55
3.3.3	Discussion	56
3.4	Classification Using Line-segment Hausdorff Distance	57
3.4.1	Line-segment Hausdorff Distance	58
3.4.2	Size and Position Variance	59
3.4.3	Size- and Position- Invariant Recognition	59
3.4.4	Clustering-based Size and Position Calculation	60
3.5	System Performance Evaluation	68
3.6	Computational Cost of the System	73
3.6.1	Computational Cost Of Line Segment Extraction	73
3.6.2	Computational Cost Of Classifier	74
3.7	Hardware Implementation of The Frame-based Recognition System . .	76

3.7.1	Bio-inspired Line Feature Extraction	76
3.7.2	Clustering-based Size and Position Calculation	81
3.8	Discussion	88
4	Feedforward Human Postures Recognition on AER vision events	91
4.1	Introduction	91
4.2	About Address Event Processing	92
4.3	System Architecture	96
4.4	Feature Map Construction and Neuron Competition	98
4.4.1	Proposed Cortex-like Feature Extraction	98
4.4.2	Simulation Results	101
4.4.3	Discussion	104
4.4.4	Summary	104
4.5	Motion Symbol Detection and Feature Spike Generation	106
4.5.1	Motion Symbol Detection	106
4.5.2	Feature Spike Generation	109
4.5.3	Summary	110
4.6	Categorization by Spiking Neural Network	110
4.6.1	About Spiking Neural Network	111
4.6.2	Tempotron Learning Rule	112
4.6.3	Virtually Fully Connected Tempotron Network	114
4.7	Experimental Results	116
4.7.1	Using AER Motion Sensor as input	116
4.7.2	Using Standard Dataset as Input	123
4.8	Hardware Implementation of The Frame-free AER Recognition System	125
4.8.1	AER Recognition System Architecture	125

4.8.2	Memory and Computation Cost of the AER Recognition System	126
4.8.3	FPGA-based Software-Hardware Co-processing	131
4.9	Summary	136
5	Conclusions and Future Work	137
5.1	Conclusion	137
5.2	Future Work	142
	Appendices	144
A	Line-Segment Hausdorff Distance	145
	Author's Publications	152
	Bibliography	154

List of Figures

1.1 Applications of human posture recognition.	2
1.2 Example of one piece of address events captured by an AER vision sensor	6
2.1 Examples of sensor motes.	11
2.2 Placement of sensor motes.	12
2.3 Influence of smoke in both color and thermal images	15
2.4 Segmentation of moving objects in video	19
2.5 An example of color histogram	21
2.6 Gabor filters for texture analysis	22
2.7 Basic chain code directions	27
2.8 Examples of star skeleton	28
2.9 Examples of Shape context	29
2.10 Examples of eigenfaces	31
2.11 Ventral stream and dorsal stream in primate cortex.	34
2.12 The Hubel and Wiesel hierarchical model of building complex cells from simple cells.	35
2.13 Neuron latencies at different cortical stages in rapid categorization . . .	37
2.14 HMAX model proposed by Riesenhuber and Poggio.	39
2.15 The extended HMAX model proposed by Serre et al.	41

3.1	Block diagram of the overall human posture recognition system.	47
3.2	Temporal difference image sensor adopted in the proposed system . . .	49
3.3	MAX-like convolutional network for edge feature extraction	51
3.4	A test image with a horizontal bar of length 7.	53
3.5	Some of the $S1$ feature maps for a test image	53
3.6	MAX over local neighborhood	54
3.7	MAX over orientations and scales	55
3.8	Edge feature extraction results	56
3.9	Comparison of MAX operation in the proposed algorithm and that in Serre's model	57
3.10	The classical Hausdorff distance between two geometrical shapes . . .	58
3.11	Size- and position- invariant recognition	60
3.12	Updating an existing cluster upon the arrival of a new active motion event	62
3.13	Illustration of instant merging.	63
3.14	Evolution of the intermediate clusters during the sequential scanning of a temporal difference image.	65
3.15	Definition of localization error	66
3.16	RMSE versus $Th0$ for a test image sequence	67
3.17	RMSE versus number of clusters	68
3.18	Comparison of the width and height of the computed bounding box to the ground truth box for the test library.	69
3.19	RMSE versus noise density.	70
3.20	Localization results of a sample image under different noise conditions.	70
3.21	Comparison between the proposed algorithm, original HMAX and T. Serre's Model	73
3.22	Block Diagram of the feature extraction unit	78

3.23 Hardware reuse methodology for MAX operation	79
3.24 Block diagram of the object localization unit.	82
3.25 Arithmetic operations for $(a < b)$ and $abs(a - b)$	83
3.26 Generation of the clustering flags.	85
3.27 Update of cluster-1 registers. Clustering flags control a series of tri-state buffers to choose the right boundary update results.	86
3.28 Chip microphotograph of the image sensor with motion detection and object localization functionality.	89
4.1 One example of jAERViewer	94
4.2 One piece of address events and some accumulated pseudo frames . .	95
4.3 Architecture of the categorization system	97
4.4 On-the-fly convolution with a forgetting mechanism	100
4.5 MAX over local neighborhood	100
4.6 one example piece of address events used for simulation	101
4.7 S1 maps and C1 maps at the selected “good” timing	102
4.8 Bars reconstructed from surviving C1 neurons at the selected “good” timing	103
4.9 S1 maps and C1 maps at the selected “bad” timing	105
4.10 Bars reconstructed from surviving C1 neurons at the selected “bad” timing	106
4.11 Motion Symbol detector	108
4.12 The dynamics and learning rule of tempotron neuron	113
4.13 Full Tempotron Network and Virtually Fully Connected Tempotron Net- work	115
4.14 Some reconstructed frames from the posture dataset	117
4.15 Some examples of the AER data viewed in jAERViewer.	117
4.16 On-the-fly centroid calculation	118

4.17 Simulation results of on-the-fly centroid calculation on a stream of address events	120
4.18 The performance of the proposed algorithm on the posture dataset . . .	122
4.19 Some sample images from MNIST hand-written digits dataset.	124
4.20 Architecture of the frame-free AER Recognition System.	126
4.21 Block Diagram of Xilinx MicroBlaze soft processor.	133
4.22 MicroBlaze embedded system.	133
4.23 FPGA-based Software-Hardware Co-processing architecture.	134
A.1 Definitions of the distances $d_{\perp}$, $d_{ 1}$ and $d_{ 2}$ for two parallel line segments.	146
A.2 Example of modified line-segment Hausdorff distance calculation. . . .	148
A.3 Example for illustrating the meaning of penalty.	149
A.4 Results of MATLAB simulations conducted to find the optimal value for <i>penalty</i>	150
A.5 Definitions of Δp and Δc for two line segments for the simplified distance in equation Eq. A.9.	151

List of Tables

3.1	Characteristics of the temporal difference image sensor	49
3.2	Comparison of the recognition rates and computation time	56
3.3	Experimental results for images taken by a web camera.	71
3.4	Experimental results for images taken by a temporal difference imager .	71
3.5	Experimental results for images taken by asynchronous motion detec- tion image sensor	71
3.6	Experimental results for images from small NORB dataset	72
3.7	Hardware implementation results of the feature extraction unit	81
3.8	Comparison of hardware implementations for HMAX-like feature extrac- tion	81
3.9	Hardware implementation results of the clustering-based size and posi- tion module	88
3.10	Chip Characteristics of the image sensor with motion detection and ob- ject localization functionality	89
4.1	number of surviving $C1$ neurons	120
4.2	Parameters for posture dataset	121
4.3	Performance Comparison on the AER posture Dataset	123
4.4	Parameters for MNIST dataset	124
4.5	Resource Summary of AER Recognition System	135
4.6	Performance of AER Recognition System using XC3S4000	135

List of Abbreviations

ADL	Activities of Daily Living
AER	Address Event Representation
AIT	Anterior Inferotemporal (cortex)
ANN	Artificial Neural Network
ART	Angular Radial Transform
ASIC	Application Specific Integrated Circuits
BPNN	Back-Propagation Neural Network
BRAM	Block Random-Access Memory
CAVIAR	Convolution AER Vision Architecture for Real-Time
CBIR	Content Based Image Retrieval
CMOS	Complementary Metal Oxide Semiconductor
ConvNet	Convolutional Network
CPU	Central Processing Unit
DFT	Discrete Fourier Transform
ERP	Event-related potentials
FD	Fourier Descriptors
FNN	Feedforward Neural Network
FOV	Field of View
FPGA	Field Programmable Gate Array
FPN	Fixed Pattern Noise

FPS	Frames per second
GEI	Gait Energy Image
GPIO	General purpose input/output
GPU	Graphics Processing Unit
GLCM	Gray-Level Co-occurrence Matrix
HMAX	Hierarchical Model and X
I ² C	Inter-Integrated Circuit
IT	Inferotemporal (cortex)
KDE	Kernel Density Estimation
k-NN	k Nearest Neighbor
LHD	Line-segment Hausdorff Distance
LIF	Leaky Integrate-and-Fire
LMB	Local Memory Bus
LSE	Least Square Estimation
LTD	Long Term Depression
LTP	Long Term Potentiation
LUT	Lookup Table
MAC	Multiply Accumulator
MAX	Maximum operation
MEI	Motion Energy Image
MEMS	Microelectromechanical System
MHI	Motion History Image
MLP	Multilayer perceptron
MT	Middle Temporal (area)
OCR	Optical Character Recognition
OVA	One Versus All

OVO	One Versus One
PC	Personal Computer
PCA	Principal Component Analysis
PDF	Probability Density Function
PIT	Posterior Inferotemporal (cortex)
PLB	Processor Local Bus
PLL	Phase-locked Loop
PSP	Postsynaptic potential
RAM	Random-access Memory
RBF	Radial Basis Function
RGB	Red Green Blue
RISC	Reduced Instruction Set Computing
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SDRAM	Synchronous Dynamic Random-access Memory
SNN	Spiking Neural Network
SSRAM	Synchronous Static Random-access Memory
SVM	Support Vector Machine
TFS	Time to First Spike
UART	Universal Asynchronous Receiver/Transmitter
UWB	Ultra-wide Band
USB	Universal Serial Bus
VTU	View-tuned Units
WSN	Wireless Sensor Network
WTA	Winner Take All

Summary

Human posture recognition is gaining increasing attention nowadays, especially in the area of assisted living and elderly care. The aim of this thesis is to explore the design and implementation of a highly-efficient biologically-plausible engine for the categorization of objects, with real-time human posture recognition as the envisaged application.

Based on commercially available image sensors and powerful personal computers, an impressive series of systems have been reported for human posture recognition. However, the high complexity of these algorithms limits their usage on low-cost and lightweight embedded platforms. In addition, the conventional image sensors employed in these systems also contribute to lower energy efficiency, since their output contains a very high level of redundancy.

Unlike conventional cameras which have little computing capability, smart image sensors utilize novel focal-plane signal processing to improve computation efficiency. This thesis targets an innovative combination of a motion detection smart image sensor and a bio-inspired event-driven processing architecture. A biologically inspired human posture recognition system is proposed. The system was designed first using a frame-based temporal difference image sensor, followed by an improved version based on a frame-free asynchronous Address-Event-Representation (AER) vision sensor.

Using the frame-based temporal difference image sensor, a bio-inspired feedforward feature extraction approach was proposed. The feature extraction unit consists of a layer of simple cells (modeled by Gabor filters) and a layer of complex cells (featuring maximum operations). After feature extraction, each frame is represented as a set of vectorial line segments. A modified line segment Hausdorff-distance (LHD) classifier, combined with a clustering-based size and position calculation module, is then used for posture categorization. The system can achieve about 90% average successful rate in the categorization of human postures, while using only a small number of training samples.

Using the frame-free AER temporal contrast vision sensor, an improved event-based feedforward categorization system was further proposed. A similar cortex-like feature extraction method based on convolution and maximum operation was used, with the introduction of an event-driven processing technique to fully utilize the power of AER sensors. An asynchronous time-domain motion symbol detector was further proposed to detect a burst of motion events and then trigger the classification. The feature spikes are classified by a spiking neural network (SNN), namely tempotron. One appealing characteristic of this system is its fully event-driven processing. The input, the features, and the classifier are all based on address events (spikes). Experimental results on two datasets demonstrate the efficacy of the proposed system.

The proposed system performs well with single object/human posture in the scene. A number of challenging problems remain to be addressed, such as the view invariance, multiple objects tracking, and adaptation to high dynamic range lighting conditions. Resolving these issues will undoubtedly lead to a very promising new generation of categorization systems.

Chapter 1

Introduction

1.1 Motivation and Objective

1.1.1 Application of Human Posture Recognition

Understanding human activity is of great importance to a number of applications [1, 2] (see Fig. 1.1), such as assisted living and elderly care, security surveillance, intelligent visual human machine interaction, and video game systems, just to name a few.

For the scenario of elderly care, by using different kinds of sensors and recognition algorithms, the machines can be made to automatically recognize human postures and further record his/her daily activities. In addition, alarms can be triggered once accidental falls are detected. For the elderly who live alone, falls are a major health hazard. Approximately 30% of the 65 year old or older people fall each year [3]. How to efficiently monitor and analyze daily activities of the elderly in order to assist their living has become an important research topic.

Video monitoring is a commonly-used solution in nursing institutions. On one hand, considerable human resources are required in order to monitor the elderly's activities. On the other hand, due to privacy issues, the elderly may be reluctant to be monitored. Therefore, an automated system that locally conducts image/video parsing to understand the elderly's activities from their postures is needed. The outputs of such

a system, activities of daily living (ADL), can be used to provide an objective indication of the activity levels or restrictions experienced by patients or elderly. A detailed assessment of ADLs also helps to objectively evaluate the effectiveness of experimental manipulations in cases such as rehabilitation programs, surgeries, and medications.

Human activity analysis also plays an important role in security surveillance. Any unusual behavior of a person in the public area (say, a walking person suddenly lowers down himself to drop a package and then leaves immediately) should alert security staff. This kind of system can enable one guard to monitor more cameras at once, increasing work efficiency.

Figure 1.1: Applications of human posture recognition.

1.1.2 Challenges and Progress of Human Posture Recognition

Considerable work has been devoted to human posture recognition in the literature. Notably, wearable devices such as gyroscopes and accelerometers have been exploited. However, this kind of approach requires the devices being mounted to the patients and can only detect a few abrupt activities. Vision-based surveillance is a more popular, non-invasive solution. Unfortunately, the relative success of many vision systems is primarily based on laboratory measurements in well-controlled environments. For real-world applications, the obstacle lies mainly with the need of multiple sensors

to cover a large surveillance area and different view angles, together with the need of high frame rate video acquisition devices to detect fast events such as a fall. Conventional cameras, which exhibit little computing capabilities, fall short of meeting these requirements. Massive quantities of raw, redundant image data have to be transmitted and processed before the features of interest are obtained. These data put heavy pressure on the transmission bandwidth and the following processing stage.

Smart image sensors utilize novel focal-plane signal processing to improve computation efficiency. Among these are various image sensors for motion detection, adaptive resolution and even object tracking. For instance, an architecture of object tracking Complementary-Metal-Oxide-Semiconductor (CMOS) image sensor was presented in [4]. The sensor can switch between two operation modes: object acquisition and tracking. It first finds the N most salient targets in the field of view (FOV) and defines windows around their centroid coordinates. Once the sensor has acquired N salient targets, the predefined windows then serve as a spotlight in biological systems, such that only the regions inside the windows are processed. A spatial-temporal multi-resolution CMOS image sensor was reported in [5]. This image sensor can simultaneously generate two outputs: one at a low frame rate with the maximum spatial resolution for stationary backgrounds and the other at a high frame rate with reduced spatial resolution for moving objects in the region of interest (ROI).

On the signal processing side, existing systems have been focusing on algorithmic-software implementations, involving complex computations such as active contours, principle component analysis (PCA), and hidden Markov models (HMM) [1]. On the other hand, primate vision is extremely accurate and efficient in the categorization of objects. Therefore, it has been a hot topic to model the feature representations in visual cortex and design neuromorphic systems that mimic cortical information processing. The current theory of the cortical mechanism responsible for object categorization

has been pointing to a hierarchical and mainly feedforward organization [6]. This organization can provide hierarchical features of increasing complexity and invariance to size and position, making object categorization a multi-layered and tractable problem.

1.1.3 Event-based Processing and Neuromorphic Systems

By combining novel knowledge from neuroscience, researchers in neuromorphic engineering aim to build electronic systems that have the efficiency of biological computation [7,8]. Recent years have witnessed increasing efforts in event-based neuromorphic systems [9–16]. The aim is to emulate biological use of asynchronous, sparse, event-driven signaling as a core aspect of its computational architecture. Address-Event-Representation (AER) sensors naturally provide a way to incorporate demand-based computation, where an asynchronous stream of digital data encapsulates only the relevant information, i.e., “features”, for processing. In particular, AER temporal contrast vision sensors [17–19] allow pixel-parallel image processing at the focal plane. Each pixel in the sensor can individually monitor the temporal relative change in light intensity and report an event if a threshold is reached. Asynchronous row and column arbitration circuits process the pixel requests and make sure only one request is granted at a time in a fair manner when multiple requests are received simultaneously [20–22]. Once the arbitration process is completed, the pixel address is sent out, and the pixel will re-start its operation. The output of an AER vision sensor is a stream of address events. Each event has an address and a timestamp. The address indicates which pixel the event is from, and the timestamp represents the time of occurrence of the event.

Despite many institutions using the AER protocol, interfacing hardware remains difficult and it requires a deep understanding of all the components used. One drawback of AER “silicon retinas” is the high cost of silicon area per pixel. Limited feature

extraction can be done at the pixel level [23–25] and hence the output events are barely enough for direct input of classification algorithms. Additional preprocessing, such as segmentation, resizing, repositioning and even more complex high level feature extraction, is still needed. However, most of existing algorithms are based on conventional frame image sensors. In order to adopt these algorithms, a common practice (jAERViewer [26], for example) is to divide events into fixed time slices (20 ms, for example) and accumulate them into pseudo-pictures. Each incoming event is associated with an address, which is used to “light” the corresponding pixel in the picture. Fig. 1.2 shows a space-time scatter plot of one piece of address events captured by an AER vision sensor [27]. A person is doing stand-up and sit-down actions in this recording. The lower part of this figure shows several selected frames reconstructed from the address events. By inspecting these reconstructed frames, one can easily find that the human silhouette in some frames is incomplete or totally missing. The main difficulty arrives from the asynchronous nature of “motion” with respect to the time “slice”. A motion may fall into two time slices and neither of the pseudo picture tells a right story. In fact, this is a common problem when using frame-based image sensor for motion processing.

In order to fully utilize the power of AER sensors, the concept of “event” should be applied to every signal processing stage. Event-based object segmentation and tracking were studied in [28, 29]. In [28], an embedded vision system was designed and combined with an AER vision sensor to achieve real-time object tracking through efficient event-based clustering. Delbruck and Lichtsteiner [29] adopts similar algorithm for tracking and further use the tracking results to control a servo motor goalie. Running on a laptop computer, the system can track and block balls shot with small reaction latency (about 2.8ms). In addition, event-based convolution for feature extraction has been exploited in [30, 31]. AER 2D convolution chips for neuromorphic

Figure 1.2: Example of one piece of address events captured by an AER vision sensor. The lower part shows some reconstructed frames by dividing events into time slices and applying accumulation in each slice. One can find that the human silhouette in some frames is incomplete or totally missing. This is a drawback of frame-based processing on motion events.

spike-based cortical processing have been designed to accelerate convolutions of programmable kernels over the AER visual input. These convolution chips have been combined with other AER processing blocks to build larger neuromorphic systems [32–35]. “Convolution AER Vision Architecture for Real-Time” (CAVIAR) project [32] is such a massive neuromorphic vision system that performs sensory, processing, learning and actuating in a row under AER hardware framework. The system senses motion of objects with a temporal contrast silicon retina. It performs feature extraction through convolutional processing and winner-take-all (WTA). Learning chips based on spiking neurons are also included in CAVIAR for spatial-temporal pattern classification. The system can recognize and track a rotating dot of a certain size. Although this application is simple, CAVIAR does demonstrate the power and potential of the promising and effective AER processing. Moreover, event-based convolution was also applied to Convolutional Networks (ConvNets) in [34, 35] to generate a frame-free AER-based ConvNet for feature extraction and categorization on AER visual event

flow. The AER-based ConvNets have a similar architecture with conventional frame-based ConvNets [36], where convolution and subsampling modules interlace. Due to the frame-less event-based processing, AER-based ConvNets constitute a significant improvement in terms of input-output latency. However, the learning of the AER-based ConvNets is based on mapping from frame-based ConvNets but not naturally spike-based learning.

1.1.4 Objective and Contribution

This work targets an innovative combination of an event-based motion detection sensor and a bio-inspired processing architecture. The system will be first evaluated in modular algorithmic routines and then implemented into event-based VLSI hardware. The ultimate goal of this research is to spur technology innovation to lower the cost and improve the quality of home-care services.

Using a frame-based temporal difference image sensor, a bio-inspired feedforward feature extraction approach was proposed. Each motion event from the sensor is sent in parallel to a battery of orientation filters based on Gabor functions, and convolution operation is performed on the fly. The responses of these filters are analogous to simple cell neurons in primary visual cortex. After convolution, neural competition is applied in order to find the maximal response among the “neurons”. Only those who reach the maximal response can survive during the competition and each surviving “neuron” represents a line feature. The detected line features are organized into vectorial line segments. After feature extraction, a modified line segment Hausdorff distance (LHD) classifier, combined with a clustering-based size and position calculation module, is used for posture categorization. The system can achieve about 90% average successful rate in the categorization of human postures, while using only a small number of training samples.

An improved event-based feedforward categorization system is further proposed for a frame-free AER vision sensor [27]. The AER motion sensor does not produce intensity images but a train of spikes. In the application of assisted living/elderly care, using AER sensor instead of conventional image sensors can protect the privacy of the person under caring. In addition, the combination of AER vision sensor and event-based processing makes the categorization system more efficient. Each address event from AER vision sensor is sent in parallel to a battery of orientation filters based on the Gabor functions, and convolution operation is performed on the fly. A leaky model is used to describe the dynamics of each neuron in the map. This feature extraction unit is inspired by a recent hierarchical model of object categorization in the primate visual cortex [37, 38]. Each neuron competes with other neurons located within its receptive field, and it can only survive in higher layer if it wins a maximum operation (MAX)-like computation [39–41]. In addition, an asynchronous time-domain motion symbol detector was proposed to activate another stage of spike generation. The dynamics of the aforementioned survival neurons, which represent the strength of “features”, go through a small set of neurons that work in Time-to-First Spike (TFS) mode. The generated spikes, again in the form of spatiotemporal pattern of pulses, are fed to a spiking neural network (SNN), namely tempotron [42]. The major contribution of this system resides in two areas: 1) an efficient time-domain clustering algorithm to capture “motion symbols” and 2) a fully event based architecture that can emulate biology’s use of asynchronous, sparse, event-driven signaling.

1.2 Report Organization

The organization of the report is as follows.

Chapter 2 gives a brief review of the related work in the literature. Based on different components of a recognition system, the review is divided into four sections:

data acquisition device, preprocessing, feature extraction and classification. For data acquisition device, wearable sensors (such as accelerometers and gyroscopes) and vision-based sensors (like standard cameras, depth sensors and smart image sensors) are covered and compared. For preprocessing, background subtraction, a technique commonly used in video based object tracking and recognition, is illustrated. Different kinds of features and representations are discussed in feature extraction section; and the last classification section summaries several widely used classifiers.

Chapter 3 illustrates the proposed recognition system for a frame-based temporal difference image sensor. The system consists of a custom designed temporal difference image sensor, a bio-inspired feedforward line feature extraction unit and a size-and-position-invariant classification framework. The proposed recognition algorithm is evaluated on several posture datasets, and comparison has been made with two previous bio-inspired approaches.

Chapter 4 introduces the improved recognition system for a frame-free AER vision sensor. The system takes data from a temporal contrast AER vision sensor. The proposed system extracts bio-inspired cortex-like features and discriminates different patterns using AER based tempotron classifier (a network of leaky integrate-and-fire (LIF) spiking neurons). One appealing character of the proposed system is the event-driven processing. The input and the features are both in the form of address events (spikes). Experimental results on two datasets have proved the efficacy of the proposed system.

Chapter 5 concludes this thesis and lists the future work.

In addition, A detailed implementation of the modified Line-segment Hausdorff distance is given as Appendix.

Chapter 2

Literature Review

This chapter provides a systematic review of the related works in the literature. Without loss of generality, a recognition system can be divided into four units: data acquisition, preprocessing, feature extraction and classification. The organization of this chapter is according to this division. For data acquisition device, a variety of wearable sensors (such as accelerometers) and vision-based sensors (like standard cameras and smart image sensors) are discussed and compared. After that, background subtraction, a technique commonly used in video based object tracking and recognition, is reviewed. Then, different features and representations are discussed in the feature extraction section. Finally, the classifiers section summarizes several widely used classifiers.

2.1 Data Acquisition

For human activity recognition, existing systems in the literature adopt two kinds of sensors as data acquisition devices: wearable sensors and vision-based sensors. Wearable sensors refer to the sensors such as accelerometers and gyroscopes which must be attached to the target person. Vision-based sensors stand for all kinds of image or video acquisition devices. Sensors in this category include widely used video cameras and new emerging smart vision sensors [17, 43, 44].

2.1.1 Wearable Sensors

Sensors such as accelerometers and gyroscopes are widely used in today's handheld devices (mobile phones, portable gaming devices, tablets, etc.), due to advances in MEMS (microelectromechanical system) technology. These sensors can provide 3-axis acceleration (or angular acceleration), bringing about various applications and friendly user experience.

In addition, these sensors are also adopted by researchers to perform physical activity monitoring. Sensor Mote, which is an embedded platform that usually consists of 3-axis accelerometer, microprocessor and wireless transceiver, is required to be attached to the target person. Fig. 2.1 shows several sensor motes designed by different researchers. Commonly, the sensor mote is attached to the chest only [45], waist only [46–48] or several main body parts [49] (see Fig. 2.2). A review of accelerometry-based wearable motion detectors is given in [50]. Most studies in the literature adopted waist placement of a single sensor mote to classify a range of basic daily activities, such as walking, postures and activity transitions. This is based on the fact that the waist is close to the center of mass of the human body and therefore major human motion can be better represented by accelerations measured at this location.

Figure 2.1: Sensor motes used in the literature. They are captured from [48], [45] and [46], respectively.

Basically, wearable sensors should be fixed and attached to the human body. Rotations and other relative motion between the sensors and body should be prevented

Figure 2.2: Placement of sensor motes. Captured from [48], [45] and [49], respectively.

in order to reduce the noise and increase the recognition performance. Nevertheless, there do exist such work that does not have special requirement on how to wear the sensor except for the location [47]. However, in this case, features available are narrowed down and final recognition rate is thus relatively inferior.

2.1.2 Vision-based Sensors

Wearable sensors can provide very rich information about accelerations of the human body or its body parts. However, one major issue about these sensors is that they must be attached to the target person. Sometimes, the person may not be cooperative or may forget to wear the sensors. Also, in some applications, it is not possible to attach sensors to the targets (e.g. security surveillance). So, despite providing rich information, wearable sensors are constrained by the “wearing” requirement.

Compared to wearable sensors, vision-based sensors are more prevalently used in security surveillance and daily activity analysis nowadays. These sensors provide rich image data or video sequences in a non-intrusive way, without requiring any attachments to the targets. Various computer vision and pattern recognition algorithms have been proposed and applied to human activity recognition, based on static images or video sequences captured by different kinds of vision sensors. Generally speaking,

these sensors can be divided into four categories: conventional gray level or color camera, thermal infrared sensor, depth sensor and smart vision sensor.

2.1.2.1 Conventional Camera

In the literature, most studies adopted a conventional camera as data acquisition device [51–62]. In fact, image processing, pattern recognition and computer vision are essentially developed based on binary or gray level images, color images and video sequences. Reviews on video-based human activity recognition are given in [1, 2]. Usually, background subtraction techniques are first used to obtain the silhouette or contour of the human body. Various feature extraction and representation approaches are then adopted to describe human posture in a single frame or human actions in spatial-temporal volume. Different classifiers are thereafter employed to do the classification.

A human action recognition system based on a single camera was reported in [52]. The authors used chromaticity and gradient based background subtraction to extract human silhouettes and further human contours from video sequences. Human posture was represented by star skeleton and classified by Support Vector Machine (SVM). String matching technique was further used to perform higher-level action recognition. In [57], human postures were presented by a chain-code signature of human contours. While in [53], location, speed information together with Hu Moment Invariants -based shape representations were combined in a hierarchical action decision tree. Moments descriptors were also used in [58] to approximate an ellipse for the human silhouette. Parameters of the ellipse and coefficients of Discrete Fourier Transform (DFT) were used to represent human postures.

Efros et al. performed human action recognition at a distance based on broadcast videos of football games [60]. They adopted an optical flow based motion descriptor

and a nearest neighbor classifier. A general method for human activity recognition in video was introduced in [51]. It also used an optical flow based local motion descriptor but further considered position and velocity information as additional features. High-level activity recognition was achieved by using Hidden Markov Models (HMM). Another optical flow based representation was proposed in [56], with each video clip described by a bag of models of kinematic features derived from optical flow.

2.1.2.2 Thermal Infrared Sensor

Conventional cameras can only capture images or videos in the visible spectrum. To obtain good recognition results, the lighting illumination and visibility of the air must be well controlled. For example, when in a low-illumination environment or in a space full of smoke, the image captured by normal cameras will be either very dark or unclear. To realize security surveillance or human activity monitoring in such extreme conditions, one has to resort to special image sensors, such as thermal infrared sensors. An infrared camera collects electromagnetic radiation that is emitted by objects in the scene and presents it as a thermal image, in which pixel values means temperature but not brightness or chromaticity. An uncooled infrared sensor was used in [63] to bring robustness to hard visibility conditions (Fig. 2.3). People in an environment filled by smoke cannot be detected by a conventional camera. However, they can be clearly seen and easily segmented from background using thermal infrared sensors. Han et al. [64] performed human activity recognition in thermal infrared imagery. An efficient spatio-temporal representation, Gait Energy Image (GEI), was proposed. Silhouette images of motion sequences were described by a single GEI image and a statistical approach was used to extract features from GEI for activity recognition. Generally speaking, the introduction of thermal infrared imagery into computer vision has helped to extract human silhouettes from background and allowed various existing algorithms to work well regardless of lighting and visibility conditions.

Figure 2.3: Influence of smoke in both color and thermal images [63]. First row: no smoke, second row: area filled with smoke.

2.1.2.3 Depth Sensor

Another kind of new emerging camera is depth sensor (also called time-of-flight camera). Depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which provides a depth map of the scene through time of flight principle. Each pixel in the depth map represents the distance from that point to the camera. As active vision sensor, depth sensor can work under any ambient light conditions. It has higher accuracy compared to stereo vision. Available depth sensors in the market include MESA SwissRanger SR4000 [65] (4000+ USD) and Microsoft Kinect for Xbox 360 [66] (150 USD). Diraco et al. [67] used a wall-mounted time-of-flight camera (MESA SwissRanger SR3000) for fall detection and geodesic distance map-based posture recognition for elderly homecare applications. The same sensor was adopted by Zhu et al. in [68] to perform human pose estimation by means of a model-based, Cartesian control theoretic approach. Ganapathi et al. [69] reported a hybrid, Graphics Processing Unit (GPU)-accelerated filtering approach and made a large step toward real-time human motion capture. Amoretti et al. [70] introduced time-of-flight cameras

into wireless sensor networks and studied data fusion for user activity monitoring. In [71], Shotton et al. proposed a real-time human pose recognition system based on single depth images capture by Kinect depth sensor. They mapped the difficult pose estimation problem into a simpler per-pixel classification problem. By using randomized decision forests trained with a large and highly varied training dataset, 31 body parts could be recognized and joint positions could be thereafter obtained. A very high recognition rate was achieved and the hardware system could run in real time.

2.1.2.4 Smart Vision Sensor

Conventional cameras usually have little or no computing capability. Massive quantities of primitive, redundant image data have to be transmitted and processed before the features of interest are obtained. These data put heavy pressure on the transmission bandwidth and the following processing unit. Smart image sensors, which exhibit certain computation capabilities, provide a possible way to alleviate these problems. This kind of sensor [17, 43, 44, 72, 73] usually integrates some image processing on the focal plane and only outputs useful information (such as moving objects or spatial contrast) in the scene, largely reducing bandwidth requirements.

Lichtsteiner et al. [17] designed a 128x128 asynchronous temporal contrast vision sensor. The logarithmic transformation in each pixel allows for very high dynamic range operation (120dB), which makes it appropriate for uncontrolled light conditions. Pixels that have a value change larger than user-defined thresholds will generate ON/OFF events. The events are rendered in a so-called address event representation (AER). Based on this asynchronous transient vision sensor, real-time mean-shift-like object tracking [74], highway vehicle speed estimation [75] and a fast sensory motor control goalie have been reported. In addition, the CAVIAR project [76], an AER hardware sensory-processing-learning-actuating system, has successfully demonstrated the computing power of the spike-based neuromorphic processing framework.

Chen et al. [44] proposed a 64×64 temporal difference image sensor with ultra-wide band (UWB) wireless transceiver. The sensor features sequential scan readout strategy, high speed (160 frames per second (FPS)) and low power consumption (sensor 0.9mW, UWB 15mW), making it an appealing candidate for wireless sensor node.

2.1.3 Summary

A brief review of data acquisition devices for human activity recognition has been given. Various devices (such as wearable sensors) and cameras were covered. Wearable sensors can provide rich motion information of the human body or its body parts. However, the sensors must be attached to the subject. Vision-based recognition constitutes a more popular and non-intrusive approach. Most studies in the literature are based on images or videos captured by conventional gray level or color cameras. Conventional cameras usually have little or no computing capability. Massive quantities of raw, redundant image data have to be transmitted and processed before the features of interest can be obtained. Things will be even worse in the case of wireless sensor networks (WSN). Smart vision sensors, which have certain computation capability, provide only the useful information, greatly saving the bandwidth and making themselves promising in WSN. In addition, since smart vision sensors capture only the motion but not raw images, the privacy of the target person (e.g. in the case of elderly care) can be protected. Moreover, smart vision sensors usually have high frame rate. This property makes it suitable for high-speed object tracking and fast action detection such as people falling down. Smart image sensors still have some limitations. Like conventional cameras, they also require adequate lighting conditions and cannot work in extreme circumstances such as dark or smoky environments. In these cases, thermal infrared or depth sensors could be employed.

2.2 Preprocessing

For video based human posture recognition and many other computer vision applications, a preprocessing step is required, during which the moving objects are to be segmented from the background (as shown in Fig. 2.4). The most commonly used segmentation technique for video sequences captured by static cameras is background subtraction. The principle of this technique is detecting the moving objects from the difference of current frame and a background model. A pixel is classified as foreground if this difference is greater than a threshold. Based on the representation of background model, existing approaches can be divided into three categories: basic (such as frame difference, running average and temporal median filtering), parametric (e.g. Mixture of Gaussians) and non-parametric (e.g. kernel density estimation). Among all these techniques, frame difference is probably the simplest modeling approach of updated background, with the background model being just the previous frame. This approach is very sensitive to the threshold, however, it does adapt rapidly to the abrupt illumination changes. As for parametric modeling methods, Mixture of Gaussians is a typical one. It represents each pixel's distribution with a mixture of several Gaussians and several parameters have to be dynamically updated. While Kernel Density Estimation (KDE) gives a non-parametric modeling, it estimates the probability density function (PDF) of each background pixel by averaging a number of kernels centered in the most recent N background values. A pixel is classified as foreground if its probability of belonging to the modeled distribution is smaller than a threshold. Several reviews and comparisons of background subtraction techniques are given in [77–79]. In addition, a color characteristics based background modeling approach is proposed in [80] to deal with shadows and highlights.

Figure 2.4: Segmentation of moving objects in video. Modified from [79].

2.3 Feature Extraction

Feature extraction is a crucial step in the framework of object recognition. It extracts certain features from the original rich content of images. On one hand, feature extraction reduces the amount of information that is required to represent the original pattern. On the other hand, it increases the accuracy of the classification. Various kinds of features and a large number of representations have been proposed and exploited by researchers. Given that it is difficult to cover all of them, a brief review of the most influential and relevant works reported in the literature will be given in this section.

2.3.1 Color, Texture and Shape

Human vision can recognize and discriminate objects by their color, texture and shape. All these features have also been exploited in computer vision. Compared to color and texture, shape features are mostly adopted in object recognition area. Biederman [81] claims this is due to the reason that the object's class identity is more intrinsically related with its volumetric description (shape) rather than surface characteristics such as color and texture. However, this does not mean color and texture are useless. There are many examples in nature and also artificial environments where color (or texture) correlates well with class identity.

Swain and Ballard [82] illustrated the use of color histogram and histogram intersection for efficient indexing of large databases. Given a color space (e.g., red-green-blue (RGB)), a color histogram makes statistics of the frequency of each color (or each range of colors) occurs in the whole image. Fig. 2.5 shows an example of color histogram (generated by a Java program Color Inspector 3D [83]). The RGB color 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 colors. Color histogram is an efficient representation of color images. It is widely used in content based image retrieval (CBIR). Wang [84] proposed a robust CBIR approach based on local color histogram. The image was first divided into several blocks and a color histogram was calculated in each block. Similar technique was adopted in [85], where Wang et al. computed color 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 color histogram, and demonstrated its superior retrieval performance than conventional color histogram. Huang et al. [87] introduced spatial correlation of colors into color histogram and defined a new image feature called color correlogram. Color correlogram-based object tracking was illustrated in [88].

Texture features are also widely used in pattern recognition, especially in the analysis of remote sensed imagery, medical imagery and so on. Typical texture features include gray-level co-occurrence matrix (GLCM), Law's texture energy and Gabor filter banks analysis. GLCM technique was first proposed by Haralick in 1973 [89] and it has now become the prevalent technique for texture analysis. This technique first constructs a co-occurrence matrix which involves the spatial relationship of pairwise pixels (usually neighbors) and then GLCM is used to compute a set of scalar quantities to represent texture characteristics. Laws' texture energy [90] is another effective texture representation. The image is first convolved with a variety of kernels and then texture

Figure 2.5: An example of color histogram. Captured from [83]. The right figure shows the sample image's color histogram in the RGB space. The size of the ball represents the frequency of the corresponding range of colors.

energy is calculated on each map. A feature vector can be constructed using these texture energy measures. In addition, Gabor filters are also adopted in texture analysis [91, 92]. Manjunath and Ma [91] applied a bank of Gabor filters (R orientations and S scales, see Fig. 2.6 for example) to an image, resulting $R \times S$ filtered images. The feature vector was then formed by computing the mean and standard deviation of these filtered images. Gabor filter bank-based texture features were used in [93, 94] for optical character recognition.

Color and texture features are appealing in terms of their low complexity and high efficiency in circumstances where images contains apparent unique colors or textures. However, in general, there are many cases where colors or textures are not available or not obviously discriminative for different objects. Therefore, shape-based features and representations are prevalently used in recent research. Shapes such as edges or lines, corners, contours or silhouettes have been widely adopted in object tracking, pattern recognition, human activity analysis and so on. Various shape representations will be discussed in the following subsection.

Figure 2.6: A bank of Gabor filters with three scales and five orientations used for texture analysis [95].

2.3.2 Shape-based Representations

A number of shape representations have been proposed in the literature for the description of shape features [96]. Generally speaking, these methods can be classified into global shape representation, where shape is described as a whole, and local shape representation, where shape is described in parts or in local regions. In addition, from another point of view, shape representations can also be divided into contour-based methods and region-based methods. Furthermore, according to the domain upon which the algorithms are based, dividing the approaches into space domain and transform domain can be viewed as a third classification method of shape representations.

Unlike conventional classifications, shape description methods are divided here according to their processing approaches. Approaches that will be covered include Moments (geometric moment invariants and Zernike Moments), shape transform domain methods (e.g. Fourier Descriptor, Angular Radial Transform, $\mathfrak{R}$ transform), spatial cor-

relation representations (chain code, star skeleton and shape context) and statistical methods (projection histogram, zoning, principal component analysis (PCA)).

2.3.2.1 Moments

Region-based (or in other words, silhouette-based) Moments are widely used in object recognition. Hu [97] proposed a set of invariant moments in 1962. Seven moment invariants are defined as follows:

$$\begin{aligned}
 I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
 I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
 I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
 I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
 &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
 I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})] \\
 I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
 &\quad + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
 \end{aligned} \tag{Eq. 2.1}$$

where $\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}}$ for $(p + q \geq 2)$, and μ_{pq} is the central moment of a digital image $f(x, y)$ (more specifically, binary silhouette image). Hu moment invariants are invariant to translation, scale and rotation. They are widely used in optical character recognition (OCR) [98], object recognition and human action analysis [53].

Another type of moments is Zernike moments, which are derived from a set of complex orthogonal Zernike polynomials defined in the polar coordinate system inside a unit circle:

$$V_{nm}(x, y) = V_{nm}(r \cos \theta, r \sin \theta) = R_{nm}(r)e^{jm\theta}, \quad x^2 + y^2 \leq 1 \tag{Eq. 2.2}$$

where $R_{nm}(r)$ is the orthogonal radial polynomials:

$$R_{nm}(r) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s! \times \left(\frac{n-2s+|m|}{2}\right)! \left(\frac{n-2s-|m|}{2}\right)!} r^{n-2s} \tag{Eq. 2.3}$$

$n = 0, 1, 2, \dots$; $0 \leq |m| \leq n$; and $n - |m|$ is even.

Zernike moments are given by [99]:

$$Z_{nm} = \frac{n+1}{\pi} \sum_r \sum_{\theta} f(r, \theta) V_{nm}^*(r, \theta) \Delta r \Delta \theta \quad (\text{Eq. 2.4})$$

Moments descriptors like Hu Moment Invariants and Zernike Moments have appealing properties such as rotation invariance and multilevel representation. Lower order moments represent the global shape of pattern and higher order moments represent the detail. Zernike moments are more robust to noise, while Hu Moment Invariants have relatively lower computational complexity.

2.3.2.2 Shape Transform Domain Methods

Shape descriptors in transform domain include Fourier Descriptors, Angular Radial Transform (ART), $\mathfrak{R}$ transform, to name a few.

Fourier descriptors

Fourier descriptor is an old technique proposed nearly 40 years ago. It is still considered as an effective description tool for shape features. Generally speaking, Fourier descriptor is obtained by applying Fourier transform on the shape contour points and taking the normalized Fourier transform coefficients as the representation of the shape [100,101]. The points on the shape contour are first organized into a set in clockwise (or counter-clockwise) manner. A shape signature is then used to describe this set of points. The shape signature can be based on complex coordinates, centroid distances, and so on [102].

For the complex coordinate function, each shape contour point is described as a complex form:

$$z(t) = [x(t) - x_c] + i[y(t) - y_c], \quad t = 0, 1, \dots, L \quad (\text{Eq. 2.5})$$

where (x_c, y_c) is the centroid of the shape (in other words, average of the boundary coordinates), and L is the total number of points in the contour.

The centroid distance function describes the shape contour points by using their distances from the centroid of the shape:

$$r(t) = ([x(t) - x_c]^2 + [y(t) - y_c]^2)^{1/2}, \quad t = 0, 1, \dots, L \quad (\text{Eq. 2.6})$$

Assuming that the shape contour points are normalized to N points in the sampling stage, the discrete Fourier transform (DFT) of the shape singature $s(t)$ (e.g., complex coordinates $z(t)$, centroid distances $r(t)$) is given by

$$u_n = \frac{1}{N} \sum_{t=0}^{N-1} s(t) \exp\left(\frac{-j2\pi nt}{N}\right), \quad n = 0, 1, \dots, N-1 \quad (\text{Eq. 2.7})$$

The coefficients u_n are called Fourier descriptors of the shape, denoted as FD_n , $n = 0, 1, \dots, N-1$. All the N descriptors except the first one are needed to form the feature vector. The first one (DC component) is only related to the position of the shape and thus discarded. Position invariance is already provided in the shape signature by subtracting centroid from the shape contour point coordinates. Rotation invariance is obtained by using only the magnitude values of FDs. Finally, scale invariance can be achieved through the normalization of FDs' magnitude values. The invariant feature vector is then given by:

$$f = \left[\frac{|FD_2|}{|FD_1|}, \frac{|FD_3|}{|FD_1|}, \dots, \frac{|FD_{N-1}|}{|FD_1|} \right] \quad (\text{Eq. 2.8})$$

A comparative study on shape retrieval using Fourier descriptors was given in [102]. Zhang et al. analyzed the descriptiveness of Fourier descriptors on different shape signatures. Their results show that FDs on centroid distances signature have significantly better performance than FDs derived from other signatures. Hu et al. [103] used FDs on body contour for human posture recognition.

Angular Radial Transform

Angular radial transform (ART) is a region-based shape descriptor adopted in MPEG-7. Similar with Zernike moments, ART is also defined on a unit disk in polar coordinates. It can be described as follows [104]:

$$F_{nm} = \int_0^{2\pi} \int_0^1 V_{nm}(\rho, \theta) f(\rho, \theta) d\rho d\theta \quad (\text{Eq. 2.9})$$

where $f(\rho, \theta)$ represents an image in polar coordinates and $V_{nm}(\rho, \theta)$ is ART basis function. The ART descriptors are defined as normalized magnitudes of a set of ART coefficients. Shapes described by ART descriptors can be easily compared using L1 norm [104]. Richard et al. [104, 105] generalized the use of ART, made it robust to rotations and deformations, and applied it to color images and 3D shapes.

$\Re$ transform

$\Re$ transform is an improved representation of Radon transform [106]. Let $f(x, y)$ be an image, Radon transform is defined by:

$$T_{Rf}(\rho, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - \rho) dx dy \quad (\text{Eq. 2.10})$$

where $\theta \in [0, \pi]$, $\rho \in [-\infty, \infty]$ and $\delta(\cdot)$ is Dirac delta function.

The $\Re$ transform is defined by [106]:

$$\Re_f(\theta) = \int_{-\infty}^{\infty} T_{Rf}^2(\rho, \theta) d\rho \quad (\text{Eq. 2.11})$$

Compared to Radon transform, $\Re$ transform is more robust to geometry transformation. It is translation invariant, and has a shift of θ_0 if the original image rotates θ_0 in angle. The scaling of the shape would not change the shape of the $\Re$ transform, but only would cause amplitude scaling. Wang et al. [107] employed $\Re$ transform to represent human silhouettes. He used a set of Hidden Markov Models (HMMs) for categorization of several human activities, such as “bend”, “carry”, “jump”, “rush” and “walk”. $\Re$ transform outperformed a set of Moments-based shape descriptors in their experiments.

2.3.2.3 Spatial Correlation Representations

Unlike the above-mentioned transform-based methods, representations covered in this part are all based on spatial correlation of the shape. More specifically, shape contour can be described by using chain code, star skeleton, shape context, etc.

Chain code is a contour-based shape descriptor which can be found in classic image processing textbooks. It describes the movement along the shape contour pixels. The direction of each movement is encoded as an integer number in the range of $[0,3]$ (for 4-connectivity) or $[0,7]$ (for 8-connectivity) [96], see Fig. 2.7. Basic chain code description is translation invariant but it is very sensitive to noise. This can be addressed by first re-sampling the boundary onto a coarser grid and then computing the chain code. Singh et al. [57] derived chain code-based directional vectors from human contours and adopted vector space analysis for human activities clustering and recognition.

Figure 2.7: Basic chain code directions. (a) chain code in eight directions (8-connectivity), (b) chain code in four directions (4-connectivity). Captured from [96].

Star skeleton is another contour-based shape representation. It was first proposed by Fujiyoshi et al. [108] in 1998. Several extreme points were first found on the human contour and then connected to the centroid, giving a “star” skeleton of the human shape. Usually, the local maxima of the human contour reside in head, two arms

Figure 2.8: Examples of star skeleton. Captured from [52].

and two legs. Fig. 2.8 shows some examples of star skeleton. Yuan et al. [52] used star skeleton as feature vector to represent human postures, then performed posture recognition using SVM and further action recognition by means of string matching. Chuang et al. [109] adopted constrained Delaunay triangulation to divide human contour into triangular meshes. A graph searching scheme was then employed to extract the skeleton. Yu et al. [110] proposed variable star skeleton representation (VSS). Multi stars on the medial axis of human contour were studied. They showed superior performance compared to single star.

Shape context is also a powerful shape contour descriptor. It was first introduced by Belongie et al. in [111]. As a local shape descriptor, each point on the contour is associated with a shape context descriptor, which describes the coarse distribution of other contour points with respect to this point. For a point on the shape contour, its shape context is a log-polar histogram of other points' coordinates measured by using this point as origin. Fig. 2.9 shows some examples of shape context. Shape context is invariant to translation and insensitive to small perturbations of parts of the shape. Guo et al. [112] generalized shape context and applied it to human silhouettes. They proposed a region-based shape context technique and used it for categorizations of six static postures.

Figure 2.9: Examples of Shape context. (a) and (b) Sampled contour points of two shapes; (c) diagram of log-polar histogram bins used in computing the shape contexts; (d), (e) and (f) shape contexts for reference sample points marked by $\circ$, $\diamond$ and $\triangle$ in (a) and (b), respectively. Modified from [111].

2.3.2.4 Statistical Methods

Another type of shape descriptor is based on statistical methods, such as projection histogram, zoning, PCA, etc.

Projection histogram was first introduced in 1956 by Galuberman [113], who used it for a hardware optical character recognition (OCR) system. This technique is now widely used in character segmentation and text lines detection. There are also some studies in which projection histogram is employed for recognizing human postures [58, 61, 114]. The horizontal and vertical projection histogram of a shape silhouette is obtained by counting the number of pixels in each row and each column, respectively. Since projection histograms are location sensitive, a normalization step is usually required for the silhouette (e.g. re-scaling the silhouette to a fixed length [61], or using DFT [58]). Guo et al. [114] and Nasution et al. [61] directly used projection histograms

of foreground silhouette as feature vector for human posture recognition. Projection histograms do not provide rich description, instead, important information about the shape seems to be lost. Therefore, using projection histograms alone may not be a good representation. Foroughi et al. [58] combined the approximated ellipse of human body with projection histograms for posture recognition, and achieved promising results.

Zoning is another feature extraction approach widely used in optical character recognition [98, 115–118]. The segmented character image is first divided into several equal zones, and statistical information is obtained from each zone. Zoning description can preserve more detail information than projection histograms. Vamvakas et al. [115] divided the binary character image into 25 equal zones (grids). For each zone they calculated the density of the character pixels. Several character profiles were also obtained by using projection histogram and distance to boundary. Blumenstein et al. [116] marked direction information onto each character pixel through searching in the thinned character image. He then made statistics of directional lines for each of 9 equal-sized zones. Bo et al. [118] partitioned the character image into 16 partially overlapped zones of equal size. For each zone, they evaluated the density of active pixels and the extent to which the sub-matrix shape matches six main directions. An analog VLSI chip was designed to do the feature extraction.

Principal component analysis (PCA) is an orthogonal linear transformation. It transforms the high-dimensional correlated data to a low-dimensional uncorrelated coordinate system. The greatest variance of the data would finally lie on the first coordinate (called the first principle component), the second greatest variance on the second coordinate, and so on. PCA is commonly used for dimensionality reduction in feature extraction [62, 119]. Turk and Pentland [120, 121] applied PCA on the training set of face images and used the first M principle components (called eigenfaces) to define

the face space. An image was projected onto this face space and a set of weights were obtained and used as feature vector for face matching. Fig. 2.10 shows some examples of the eigenfaces.

Figure 2.10: Examples of eigenfaces [120]. Here the plot shows the first ten eigenfaces built from 400 face images.

2.3.3 Motion-based Representations

Several shape representations have been discussed in the previous section. In image-based object recognition, shape features and representations are mostly used. While for video-based human action/activity analysis, besides shape feature, one can also employ another kind of feature: motion feature. This section focuses on the motion-based representations, such as motion history image (MHI), motion energy image (MEI), and optical flow-based motion descriptors.

Motion history image (MHI) and motion energy image (MEI) were proposed by Bobick and Davis [122] in 2001. In a given video sequence, the foreground human silhouette in each frame is accumulated and displayed in one binary motion image, called motion-energy image (MEI). MEI describes *where* there is motion in the spatial pattern. To further describe *how* the motion is moving, motion-history image (MHI)

was proposed. MHI gives decaying weights to the images in the sequence. Newer frames tend to be brighter while older frames tend to be darker. MEI and MHI have sufficient discriminating ability for several simple actions such as “sitting”, “bending” and “crouching” [122]. However, it seems that MEI and MHI would lose discriminative power for complex activities due to the overwriting of the motion history.

Optical flow constitutes the basis of another type of motion descriptors. It shows the distribution of apparent velocities of movement in an image [123, 124]. Efros et al. [60] proposed a local motion descriptor based on half-wave rectified optical flow, and used it for far-field action recognition. Yang et al. [125] adopted the same motion descriptor for patch based human action recognition from a single clip per action. Robertson and Reid [51] extended Efros’ motion descriptor by introducing temporal context. They concatenated the motion-descriptor data from 5 consecutive frames and achieved better matching results. Ali et al. [56] extracted kinematic features based on optical flow and further built up a bag of kinematic modes for each video sequence. The coordinates of the video in the kinematic-mode-based feature space were used for classification using the nearest neighbor algorithm.

2.3.4 Handcrafted vs. Biologically Plausible

Till now, a brief review of several kinds of features (color, texture, shape and motion) and their various representations have been given. Note that most of these representations are based on either sophisticated designs (such as star skeleton and shape context) or complex mathematical models (e.g. moments, all kinds of transforms, PCA, optical flow). These comprehensive methods are widely adopted by today’s computer vision community.

Handcrafted features may have sufficient discriminative ability for recognizing different objects. However, is it how brains perform recognition tasks? Human brains

consist of billions of simple units (called neurons). Each of them has limited computation ability, but a network of neurons connected together can perform sophisticated tasks such as recognition, much better than advanced computer vision algorithms.

Since human vision has superior performance on object recognition, developing a system that is biologically plausible constitutes a promising research area.

2.3.5 Biologically Inspired Models

This section focuses on biologically plausible feature models proposed by computational neuroscientists to account for visual processing in the ventral stream of visual cortex. It starts from the illustration of the architecture of the visual cortex, then moves on to the model of simple and complex cells proposed by Hubel and Wiesel. After that, it is a brief review of the hierarchical feedforward models of increasingly sophisticated representations (with the emphasis on the “Hierarchical Model and X” (HMAX) of Riesenhuber and Poggio, and a modified version proposed by Serre, et al.).

2.3.5.1 Architecture of Visual Cortex

Anatomically, primate cerebral cortex can be divided into four regions, namely occipital lobe, parietal lobe, frontal lobe and temporal lobe [126]. The part that is related to visual information processing, so-called visual cortex, is mostly located in the occipital lobe, in the back of the brain [127]. Visual cortex consists of striate cortex (also called primary visual cortex or V1) and extrastriate cortical areas (such as V2, V3, V4 and V5). It is well known that two visual processing streams exist in the cortex: dorsal stream and ventral stream (see Fig. 2.11). The dorsal stream (also known as the “where” pathway), which starts from V1, goes through V2, V3 and Middle temporal area (MT, also called V5) to inferior parietal cortex, is considered as related to motion, representation of object locations and control of eyes and arms [128]. While the ventral visual stream, which begins from V1, goes through extrastriate cortical areas V2

Figure 2.11: Ventral stream and dorsal stream of information processing in primate cortex. Modified from [127] and [133]. Ventral stream, also called “what” pathway, mediates the sophisticated task of object recognition; while dorsal stream, or called “where” pathway, is related to motion, object localization and control of eyes and arms. Along the ventral visual pathway, neuron cells show tuning properties to increasingly complex stimuli. e.g. A V1 cell responses best to an edge-bar of particular orientation [129, 130], V4 cell responses to patterns composed by different orientations, while TE cells of the inferotemporal cortex show strong responses to complex visual stimuli, such as faces [131, 132].

and V4 to the inferotemporal cortex (IT, further divided into TEO and TE), is thought to be associated with object recognition, and thus it is also called “what” pathway [128]. Along the ventral visual pathway, neuron cells show tuning properties to increasingly complex stimuli. For instance, a V1 cell responses best to an edge-bar of particular orientation [129, 130], V4 cell responses to patterns composed by different orientations, while TE cells of the inferotemporal cortex show strong responses to complex visual stimuli, such as faces [131, 132] (see Fig. 2.11).

Figure 2.12: The Hubel and Wiesel hierarchical model of building complex cells from simple cells. Modified from [129].

2.3.5.2 Simple and Complex cells

The studies on hierarchical processing in visual cortex began with the groundbreaking work of Hubel and Wiesel. Through the investigation on primary visual cortex of cat [129] and macaque [130], they proposed a hierarchical conceptual model of simple to complex cells. Simple cells have small receptive field and show tuning property to (i.e. respond best to) bar-like stimuli at a particular orientation and position. On the other hand, complex cells have larger receptive field and respond best also to bars at a particular orientation, however, they show strong response to the bar anywhere within its receptive field (i.e. position invariance). Hubel and Wiesel suggested a hierarchical model of the visual cortex, in which complex cells can be built by integrating convergent inputs from simple cells. As illustrated in Fig. 2.12, complex cells' position invariance can be obtained by pooling over simple cells with the same preferred orientation but at different positions.

2.3.5.3 Hierarchical Feedforward View-based Models

Hubel and Wiesel conceptually modeled the primary visual cortex (V1) as a feedforward hierarchy of simple cells and complex cells. This idea is commonly accepted. However, for the visual processing in the whole ventral stream (V1-V2-V4-IT), there has been controversy between object-centered models and view-based models, also between feedforward processing and feedback connections [134]. Object-centered models, such as “Recognition-by-Component” (RBC) theory of Biederman [81], try to directly extract a view-invariant structural description of the object for matching with stored object descriptions in the library. On the contrary, view-based (or image-based) models represent objects as collections of view-specific features (view-tuned units) [134]. The view-based models can be further divided into two major groups: one employing purely feedforward processing and one utilizing feedback connections for the feature processing (excluding learning) in the recognition process.

Neurophysiological data from monkeys point to view-based theory. Cells in inferotemporal cortex (IT) of macaques show a tight tuning property to views of complex objects [131, 135]. Logothetis et al. [135] recorded response of a single IT cell after training the monkey to recognize paperclip-like objects. They found that the cell responded selectively to one view of a paperclip and showed strong invariance to scaling and translation but little invariance to rotation in depth. This supports the view-based models.

In addition, the studies on processing speed in the visual cortex have constrained the “immediate” object recognition to be mainly feedforward. Thorpe et al. [136] performed rapid categorization experiments on human subjects. The subjects were shown with previously unseen images flashed on for just 20ms and were asked to decide whether an animal existed or not in the image. Event-related potentials (ERP)

Figure 2.13: Neuron latencies at different cortical stages in rapid categorization. Captured from [137].

analysis revealed that humans could perform rapid categorization tasks in natural image approximately 150ms after stimulus onset. Thus the visual processing from V1 to IT (in human cortex) must be within 150ms. The corresponding time for monkeys is about 100ms, which is shorter probably due to their smaller brain size [137]. Fig. 2.13 shows typical neuron latencies at different cortical stages of monkeys involved in rapid categorization. The short processing time along ventral visual stream (V1-V2-V4-PIT-AIT, note that PIT and AIT are subareas of IT, roughly corresponding to another type of division: TEO and TE [132]) strongly suggest that the flow of information is mostly feedforward [6].

Based on simple and complex cells in Hubel and Wiesel model, Fukushima developed a computer vision system [138], called “Neocognitron”, to perform position invariant recognition in unsupervised manner. Simple and complex layers were alternatively constructed. C-cells in one layer are excited by pooling over local neighbor-

hood of afferent S-cells; while S-cells respond to conjunctions of C-cells in previous layer. In late 1980s, by introducing back-propagation multi-layer perceptron into a series of simple-complex layers, another feedforward system called convolutional network was proposed by LeCun et al. [139]. Such a system showed good performance in handwritten digit recognition [139, 140] and more recently in the domain of generic object recognition and face detection [141]. However, systems like Neocognitron and Convolutional network, although biologically inspired to some extent, lack a direct correspondence with cortical stages (not neurophysiologically plausible) and thus cannot be used to make predictions for physiologists [6].

As for neurophysiologically plausible models, HMAX proposed by Riesenhuber and Poggio [37] was probably one of the most successful feedforward theories. HMAX extends the Hubel and Wiesel classical models of complex cells built from simple cells. It summarizes the basic facts about the ventral visual stream (V1-V2-V4-IT). The neurophysiological plausibility of HMAX lies in that it accounts for the experimental data of Logothetis et al. [135] on the tuning properties of macaque's IT neurons. As mentioned earlier, Logothetis et al. found that cells in inferotemporal cortex (IT) show significant invariance to scaling and translation but little invariance to rotation in depth. These cells, which responded selectively to one view of complex objects (e.g. a paperclip), were modeled as view-tuned units (VTU) at the top of HMAX. As shown in Fig. 2.14, HMAX consists of a hierarchy of "S" layers and "C" layers ("S" and "C" follow the notation of Fukushima [138]). The "S" layer cells increase feature complexity by linear weighted summation of inputs; while "C" layer cells increase invariance through the nonlinear MAX operation, which selected the maximum input as the response of the cell. The new pooling mechanism, nonlinear MAX operation, plays a key role in obtaining the VTU's invariance properties [37].

Figure 2.14: HMAX model proposed by Riesenhuber and Poggio. It consists of five layers, from the simple-cell like *S1* layer to the VTU level with shape tuning and invariance properties like the view-tuned cells in macaque inferotemporal cortex. The “S” layer cells increase feature complexity by linear weighted summation of inputs; while “C” layer cells increase invariance through the nonlinear MAX operation. Captured from [37].

Based on HMAX, Riesenhuber and Poggio also suggested a conceptual feedforward view-based model [134] for the whole process of object recognition. The view-tuned units (VTU) on top of HMAX only have tolerance to scaling and translation but little invariance to rotation in depth. By interpolating between several view-tuned units which are tuned to different views of the same object, view-invariant (or object-tuned) units can be created (i.e. invariance to rotation in depth can be increased). These view-invariant units, together with view-tuned units, can then serve as input to task-related units which performs recognition tasks such as identification and categorization [134].

HMAX model was further extended by Serre et al. [38, 142]. The whole feedforward architecture remained the same ($S1-C1-S2-C2$). $S1$ and $C1$ layer correspond to simple and complex cells in primary visual cortex (V1), while $S2$ and $C2$ are roughly related to V2 and V4, respectively. The first two layers of Serre's model are mostly consistent with the old model HMAX (differences exist in the adoption of Gabor filters rather than difference of Gaussians). First, a bank of Gabor filters with different sizes and orientations are used to increase the selectivity, and then MAX operations are carried out to achieve slightly scale and position invariance. The last two layers ($S2$ and $C2$) are where Serre et al. has made significant modifications. Learning is introduced at stage $S2$ [142]. He first randomly extracts a number of patches from $C1$ maps of training images and uses these patches as RBF centers. For each image, Gaussian radial basis function is then applied to the distance between $C1$ -layer map and patches. This is followed by a MAX operation to generate the shift- and scale- invariant $C2$ features. Fig. 2.15 illustrates the approach of building $C2$ features in the Serre model. For a detailed description of algorithms and parameters, one could refer to [142] and [38]. Serre demonstrated the neurophysiological plausibility of his model in [6]. In addition, promising results comparable to state-of-the-art computer vision systems have been obtained in object recognition tasks using natural images [38].

2.4 Classification

After the review of various kinds of feature extraction approaches (including biological-like and non-biological methods), it is time to move on to the classification part. In this section, several popular classifiers, such as k-nearest neighbor (k-NN), support vector machine (SVM), artificial neural network (ANN) and so on, will be discussed.

k-Nearest Neighbor (k-NN): The k-nearest neighbor algorithm (k-NN) is a simple classification scheme that classifies an unknown example based on the consensus

Figure 2.15: The extended HMAX model proposed by Serre et al. Captured from [38].

of its neighboring training samples [143]. The feature vectors of training samples are stored and labeled in the training phase. In the classification phase, a testing example is classified by assigning the label that most frequently appears in the k nearest training samples (simply summarized as “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 neighbor algorithm, in which the unknown example is classified by directly assigning to it the label of its nearest neighbor.

Support Vector Machine (SVM): SVM is another widely used technique in data

classification. It first maps the data into a different (usually higher dimensional) space and then finds a linear separating hyperplane that maximizes the margin between two classes [144]. The mapping function is called “kernel”. Common kernels include linear, polynomial, RBF (radial basis function) and sigmoid functions. SVM is a binary classifier which only separates two classes. To deal with multi-class classification problem, multiple SVMs have to be employed. There are two common approaches to break the multiclass problem into multiple binary classification problems: one-versus-all (OVA) and one-versus-one (OVO). For a N categories classification problem, One-versus-all requires N SVMs, each one distinguishes the data in a single class from the data in all remaining classes. The classification of new instances is done by a winner-take-all strategy, in which the classifier with highest output makes the final decision. While one-versus-one (OVO) involves $N(N - 1)/2$ SVMs, each one corresponds to a two-class pair, and the classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one class. Finally, the class with most votes is selected as the label. Usually, OVO SVMs have better classification performance than OVA SVMs.

Artificial Neural Network (ANN): An artificial neural network (ANN) is a computational model of nervous system. It consists of an interconnected group of nodes, mimicking the vast network of neurons in the human brain. Based on the topology of the network, ANN can be divided into feedforward neural network (FNN) and recurrent neural network (RNN) which has feedback connections [145]. The feedforward neural network is commonly used in data classification and pattern recognition. Typical FNNs include Multilayer perceptron (MLP) and Radial Basis Function (RBF) neural network. MLP network consists of several layers of artificial neurons with fully feed-forward connections. It usually has an input layer (where the number of nodes is equal to the dimension of feature vectors), one or more hidden layers, and an output layer

(in which each output node corresponds to a class). The output of each neuron is obtained by applying an activation function (such as sigmoid) on the weighted summation of previous-layer neurons. The weights of the MLP network are determined by the training process, in which the back-propagation algorithm is usually employed (such a network is also called back-propagation neural network (BPNN)). Another kind of feedforward neural network, the RBF network, is composed of an input layer (which receives the feature vector), a hidden layer (in which a kernel such as Gaussian is applied on the distance between the input and each RBF center) and an linear output layer, in which each node is a weighted summation of hidden layer neurons and corresponds to one class. The number of hidden neurons in RBF network is the same as the number of centers, which are usually determined by clustering techniques such as k-means. The training of RBF network only involves the determination of the output weights which can be done by linear least square estimation (LSE). It is thus very fast and does not suffer from problems such as local minima, which plague MLP training techniques [146]. However, RBF typically requires more training data and the selection of centers is also a major issue.

Besides the three kinds of classifiers mentioned above, there are still many other classification tools, such as Naive Bayes and decision tree. Naive Bayes classifier is a simple probabilistic classifier based on applying Bayesian decision theory with strong independence assumptions. 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 decision tree is a hierarchical classifier, which classifies an instance in a top-down fashion from the branches to the leaf nodes according to some category-dependent variables [147]. In the training phase, histograms are built for each leaf node. In the testing phase, classification is done by first tracing to a leaf

node and then checking its histogram, in which the class with the largest probability of occurrence is assigned as the label of the unknown instance. The hierarchical essence ensures that the decision tree has a good testing performance. However, the training of the large number of parameters is computationally expensive.

2.5 Summary

In this chapter, an in-depth review of various human posture categorization systems have been given. The system can be divided into four parts: data acquisition, preprocessing, feature extraction and classification.

For the data acquisition, various devices such as wearable sensors and different kinds of cameras were covered. Wearable sensors can provide rich motion information of the human body and its body parts. However, they must be attached to the subject. Vision-based recognition is a more popular and non-intrusive approach. Most studies in the literature are based on images or videos captured by conventional gray level or color cameras. However, for the conventional camera, a massive amount of redundant information has to be transmitted, requiring a large bandwidth, and things will be even worse in the case of wireless sensor networks (WSN). Smart vision sensors, which integrate computation capability, could greatly relax bandwidth requirements. In addition, since smart vision sensors capture only the motion rather than raw images, the privacy of the target person (e.g. in the case of elderly care) can be protected. In some extreme conditions such as dark or smoky environments, infrared thermal or depth sensors would need to be employed.

The preprocessing step mainly covered background subtraction techniques in video-based recognition. The purpose of this step is to obtain the human silhouette for further processing.

In the feature extraction part, various features and representations (handcrafted and biologically plausible) were illustrated. Color, texture, motion and shape features were reported, with emphasis on several kinds of shape-based representations, such as moments, transform domain descriptors, spatial correlation approaches and statistical methods. Although these handcrafted features may be efficiently discriminative for object recognition, they lack biological plausibility. It is well known that human visual system performs the object recognition tasks extremely well, much better than state-of-the-art computer vision systems. Therefore, developing a biologically inspired system to mimic the visual processing in cortex is a promising research direction. This part started from the illustration of the architecture of visual cortex, then moved on to Hubel and Wiesel model of simple and complex cells, and finally ended with a brief review of several hierarchical feedforward models of visual processing in the cortex ventral stream, with the focus on HMAX (of Riesenhuber and Poggio) and the extended version proposed by Serre et al..

Last but not least, several commonly used classifiers (e.g. k-NN, SVM, ANN and so on) were discussed. k-NN is a simple example-based classifier which involves searching in the training samples. It works efficiently for small training datasets. However, it would be very slow once the training library becomes quite large. In the latter case, learning-based classifiers (such as SVM, BPNN and RBF neural networks, which optimize the model parameters in the training phase) would be much more efficient in the classification.

Chapter 3

Bio-inspired Human Posture Recognition using Temporal Difference Image Sensor

This chapter describes the proposed human posture recognition system for a frame-based temporal difference image sensor. This system is an innovative combination of a motion detection vision sensor and a bio-inspired processing architecture. A frame-based temporal difference image sensor is used as the input device. The motion events from the sensor are processed by the bio-inspired feature extraction unit, where an image is represented as a set of vectorial line segments. Then a nearest neighbor matching scheme based on Line-segment Hausdorff distance (LHD) metric is adopted to perform the recognition of several postures. The organization of this chapter is as follows: first, the system architecture is illustrated, followed by a brief introduction of the adopted temporal difference image sensor; then the bio-inspired feature extraction algorithm and the LHD-based classification are depicted respectively. The system performance evaluation on several posture datasets and further discussion of the system are given in the last two sections.

3.1 System Architecture

Fig. 3.1 illustrates the architecture of the overall human posture recognition system, which includes a customized image sensor, bio-inspired feature extraction unit and a classifier. A known set of posture library is used for evaluating the recognition performance. A frame-based temporal difference image sensor [44] is used as input device. This sensor can filter the background information and alleviate the recognition processor from background subtraction computation. For each frame, the output of the image sensor is a stream of binary motion events. Each event is sent to a network of oriented Gabor filters, and convolution operation is performed on the fly. The responses of the filters are analogous to the lowest-level “neurons” ($S1$) in visual cortex. After that, a MAX-like operation is applied in order to find the maximal response among the feature maps. Each surviving “neuron” after the competition represents a contour line segment in the image. In this way, the original motion object (or human posture) is translated into a set of vectorial line segments, which are then fed to a nearest neighbor classifier based on modified line-segment Hausdorff-distance scheme [148].

Figure 3.1: Block diagram of the overall human posture recognition system.

3.2 Temporal Difference Image Sensor

A temporal difference image sensor (designed by Chen et al. [44]) is used to capture human motion. The sensor can automatically remove the still background and output

only the contour of moving object (as a stream of binary motion events). It avoids the transmission of the massive raw data, largely saving the channel bandwidth and thus making itself a promising candidate of wireless sensor node. In addition, since this sensor only outputs binary images, the privacy of the target person can be protected. This property is appealing especially in the application of elderly care. On one hand, the elderly people who live alone do need a vision-based care system that can detect their accidents such as falling down and further inform their family members or medical care personnel. On the other hand, they may not want their privacy exposed to others. Indeed, being monitored by a conventional gray-level or color camera makes people feel uncomfortable. Now, by using this kind of temporal difference image sensor, people can still monitor the daily activities of the elderly, while protecting their privacy.

The working principle of this sensor is as follows. Inside the sensor, each pixel is equipped with an analog memory (capacitor) and the whole array is hence capable of storing the current frame as a reference image. The rows are first sequentially selected for reset. Later at another round of scanning, the rows of pixels are selected for readout sequentially. Each pixel will output both the new integration voltage on its photodiode and the previous voltage stored on its capacitor. The two voltages are fed into an global event generator circuit which is composed of a global amplifier with a temporal-difference computation circuit based on dual comparison. The event generator computes the difference between the two voltages, and compares it to a positive and negative threshold. An effective event (1'b1) is generated if this difference exceeds the thresholds; otherwise, 1'b0 is used as the output of that pixel. In this way, the still background is removed and the human motion is captured in a stream of binary motion events.

Fig. 3.2 shows some example images captured by this sensor. The sensor has a resolution of 64×64 . It features low power consumption and high frame rate (only 0.9 mW @ 160 FPS). More characteristics of this sensor are summarized in Table 3.1.

Figure 3.2: Temporal difference image sensor adopted in the proposed system. (a) The sensor covered by a lens and attached to a PCB. (b) Die microphotograph of the image sensor. (c) Some sample images. Images on the right side show the binary motion output of the corresponding environments on the left. Note that the person and the hand are both moving.

Table 3.1: Characteristics of the temporal difference image sensor

Process Technology	AMIS $0.5 \mu\text{m}$ 3M2P CMOS
Die Size	$3 \times 3 \text{ mm}^2$
Pixel Array	64×64
Pixel Size	$33 \times 33 \mu\text{m}^2$
Fill Factor	11.5%
FPN	0.2%
Readout Strategy	sequential scan
Max Frame Rate	160 FPS
Data Rate	640k events/s (1 bit/event)
Supply Voltage	3.3 V
Power Consumption	0.9 mW @ 160 FPS

3.3 Bio-inspired Line Feature Extraction

The motion events generated by the temporal difference image sensor are sent to the bio-inspired feature extraction unit, where each image is represented as a set of vectorial line segments.

The feature extraction approach used here is inspired by the feedforward models of visual cortex, like the standard HMAX model of Riesenhuber and Poggio [37] and the extended version of Serre et al. [38, 142] (see Section 2.3.5.3). Inside these biological-like feedforward models, layers related to feature extraction include $S1$, $C1$, $S2$ and $C2$. The $S1$ layer and $C1$ layer model the simple cells and complex cells found in primary visual cortex (V1), while $S2$ and $C2$ layers roughly correspond to V2 and V4 of the ventral stream. Using the V4-like features (i.e. $C2$ units), promising results comparable to state-of-the-art computer vision systems have been obtained in object recognition tasks on natural images [38].

On one hand, the V4-like $C2$ features show excellent discriminative ability in the classification experiments using SVM [38]. On the other hand, however, calculating the $C2$ features is very memory intensive. A huge number of patches are to be extracted on the $C1$ maps, and the storage of these patches requires large amount of memory. This algorithm is thus constrained to powerful computers. Since the aim is to design a real-time posture recognition system for light-weight embedded platforms, a trade-off between the performance of the algorithm and its complexity has to be made. Therefore, instead of using the V4-like $C2$ features, the proposed algorithm adopts the V1-like $C1$ level features, with the $S2$ and $C2$ layers abandoned. Note that, an interesting modification has been made on the calculation of $C1$ units from $S1$ units. A new MAX operation is also proposed. The proposed algorithm is named as MAX-like convolutional network because of the convolutions on $S1$ layer and the MAX-like operations on $C1$ layer.

3.3.1 Proposed MAX-like Convolutional Network For Line Extraction

The organization of the convolutional network is shown in Fig. 3.3. The overall data flow can be summarized as *motion events* $\rightarrow$ *S1 maps* $\rightarrow$ *C1 maps* $\rightarrow$ *line segments*. The input events are convolved with a network of Gabor filters to get a set of *S1* maps, in which each pixel is considered as a “neuron”. The *S1* “neurons” compete with others in the process of MAX operation, and only those who best match the edge features can survive in the *C1* layer.

Figure 3.3: MAX-like convolutional network for edge feature extraction. The input events are convolved with a network of Gabor filters to get a set of *S1* maps, in which each pixel is considered as a “neuron”. The *S1* “neurons” compete with others in the process of MAX operation, and only those who best match the edge features can survive in the *C1* layer.

Simple cells ($S1$) are used to build object-selectivity. This is done by convolving the motion events with a network of Gabor filters. For the sake of hardware friendly implementation, the network is set to be 4 scales (ranging from 3 to 9, with a step length of 2) and 4 orientations (0° , 45° , 90° , 135°). The function of Gabor filter can be described as:

$$G(x, y) = \exp\left(-\frac{X^2 + \gamma^2 Y^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda} X\right) \quad (\text{Eq. 3.1})$$

where $X = x \cos \theta + y \sin \theta$ and $Y = -x \sin \theta + y \cos \theta$. The filter parameters (orientation θ , aspect ratio γ , effective width σ and wavelength λ) have been well tuned in pioneering work [6, 38], and here a similar set of these parameters is adopted. Because of hardware implementation consideration, the floating-point filter values are normalized into integer numbers.

After convolving the original image with 16 Gabor filters, 16 $S1$ feature maps are obtained. Each pixel in the $S1$ maps is called a “neuron”. For a certain feature (say a bar) in the input image, each “neuron” in each of the 16 maps gives a response. Due to the property of the Gabor filters, a “neuron” reaches the maximum response only when its size, position and orientation matches the feature. For example, for a test image with a horizontal bar of length 7 as shown in Fig. 3.4, some of its $S1$ feature maps are illustrated in Fig. 3.5. From Fig. 3.5(a), one can see that, among the $S1$ feature maps with respect to scale-7 and all 4 orientations, the horizontal one has major response while the others are all negligible. This shows the orientation selectivity of Gabor filters. In addition, from Fig. 3.5(b), one can further notice that, among the $S1$ feature maps related to orientation- 0° and all 4 scales, scale-7 shows the largest response and the peak happens at the location that corresponds to the center of the feature (horizontal bar of length 7) in the source image. This demonstrates the scale and position selectivity of Gabor filters.

Figure 3.4: A test image with a horizontal bar of length 7.

Figure 3.5: Some of the $S1$ feature maps for a test image as shown in Figure 3.4. (a) shows the $S1$ features maps with respect to scale-7 and all 4 orientations. (b) depicts the $S1$ features maps corresponding to orientation-0° and all 4 scales. The size of the image in this example is reduced for simplicity. Pixel (0,0) is on the top left corner in these images.

Based on these observations, a two-step MAX operation is performed to extract the feature. The first step, MAX operation is performed **across local neighborhood** (Fig. 3.6) to find the position of the feature. “Neurons” located in different-scale $S1$ maps have different receptive fields, such as 3×3 , 5×5 , 7×7 and 9×9 . Each “neuron” competes with its local neighbors located within its receptive field and it will survive only when itself is the MAX in this area. Then, the processing moves to the successive

“neuron”, which will compete and strive for existence in the same way as last “neuron” did. The second step, MAX operation is performed **over orientations and scales** (Fig. 3.7). The “neuron” that won the first MAX operation will compete against other “neurons” of the same orientation but different scales (vertically, as shown in Fig. 3.7), to find the size of the feature. At the same time, comparison is also performed across the counterparts of the same scale but different orientations (horizontally, as shown in Fig. 3.7), to detect the direction of the feature.

Figure 3.6: MAX over local neighborhood. “Neurons” located in different-scale $S1$ maps have different receptive fields, such as 3×3 , 5×5 , 7×7 and 9×9 . Each “neuron” competes with all the other “neurons” located within its receptive field. It can survive only when itself is the MAX in this area.

After the two-phase MAX operation, each surviving “neuron” at the $C1$ layer represents a feature, i.e. a line segment with certain size and orientation at that position. For instance, for the test image in figure 3.4, after convolution and the two-step MAX operation, only the neuron, which is located in the scale-7 orientation- 0° $S1$ map and at the position corresponding to the center of the horizontal bar in the source image, will finally survive. Therefore, from the surviving neuron, it can be inferred that a horizontal line with length 7 exists at the position of that neuron. The size, orientation and position are put together into a vector to describe this line segment. In this manner, an input image can be translated into a set of vectorial line segments.

Figure 3.7: MAX over orientations and scales. Here, only the competition for a scale-9 orientation-135° “neuron” (the dot in the bottom right map) is shown. This “neuron” competes with all the “neurons” that reside in the 6 shadowed 9×9 regions, and can only survive if it is the MAX among these neurons.

3.3.2 Simulation Results

Fig. 3.8 shows the line segments extraction results. Compared to previous work [149], the new feature extraction algorithm brings forth enhancement in the form of less noise together with reduced number of edges. This would further lead to improved classification performance (shorter computation time and higher recognition rate) as indicated in Table 3.2, which summarizes the performance of the proposed algorithm (this work) and previous work on a posture dataset built from a temporal difference image sensor. The dataset includes 6 sets of postures (in each set, 10 out of 500 for training), namely “bend”, “hand1”, “hand2”, “squat”, “stand” and “swing”. The tests are run on a desktop computer with Intel Core 2 Quad 2.66GHz central processing unit (CPU) and 3GB random-access memory (RAM). The employed Line-segment Hausdorff distance (LHD) classifier will be illustrated in Section 3.4.

	bend	hand1	hand2	squat	stand	swing
Raw img
previous [149]
Proposed

Figure 3.8: Edge feature extraction results

Table 3.2: Comparison of the recognition rates and computation time

	Recognition Rate	Time/test (s)
[149] (with LHD classifier)	0.82	10.6
this work (with LHD classifier)	0.90	2.22

3.3.3 Discussion

Compared to the model of Serre et al. [38], the proposed approach differs in the way the MAX operations are performed. In Serre's model, $C1$ cells are obtained by performing max-like operation over simple $S1$ units with the same preferred orientation, but slightly different positions, in order to gain position tolerance. Each neuron compares its response to its surroundings and will **copy** the maximum response within its neighborhood as its own response. Therefore, the final resolved feature becomes wider, and it is harder to reduce this to a line with single pixel width, as desired. This effect is illustrated in Fig. 3.9. One can note that, compared to the $S1$ feature map of Fig. 3.9(b), the feature in Serre's $C1$ map is widened by a much larger number of neurons, as shown Fig. 3.9(d). Larger neuron populations and wider maximal re-

sponses reduce the precision of the line extraction algorithm. As a matter of fact, with the results in Fig. 3.9(d) the exact position and size information of the feature is lost. In short, Serre's model widens the maximum response in $S1$ maps to achieve blurred general features; while the proposed approach suppresses non-maximum responses to obtain thinned line features.

Figure 3.9: Comparison of MAX operation in the proposed algorithm and that in Serre's model [38]. (a) The processed image, same with Fig. 3.4. (b) $S1$ feature map corresponding to scale-7 orientation- 0° . (c) Corresponding $C1$ map in the proposed algorithm, obtained by taking a new MAX operation on $S1$ map. (d) Corresponding $C1$ map in Serre's model, obtained by taking Serre's MAX operation on $S1$ maps.

3.4 Classification Using Line-segment Hausdorff Distance

The previous section described the proposed methodology for extracting line features from input images or frames. This line extraction technique is applied to all input images or frames. A subset of inputs is used to generate a tagged library of line feature maps. Subsequent input images or frames are compared to the ones in the library by means of a nearest neighbor classifier based on Line-segment Hausdorff distance (LHD) metric. The nearest neighbor classifier is adopted because it is more suitable for small training set. This section will describe the classifier used in the proposed algorithm and illustrate how it achieves invariance to size and position.

3.4.1 Line-segment Hausdorff Distance

In computer vision, the line-segment Hausdorff distance has been used for face recognition [148]. The idea to measure distance between shapes goes back to F. Hausdorff [150], see Fig. 3.10. The original Hausdorff distance measures the distance between two sets of points, while line-segment Hausdorff distance (LHD) extends the idea to two sets of line segments. The LHD naturally fits the proposed feature extraction algorithm. Based on the definition given in [148], a modified version of LHD is implemented. The implementation details are provided in the appendix A.

LHD measures the similarity between two sets of line segments. The smaller the LHD, the larger the similarity. During feature extraction, each image is represented as a set of line segments. Then, at the classification phase, a test image will be matched with each model image in the library. The test image is identified with the library image that yields the minimal distance. In other words, the test image's nearest neighbor (best match) in the library determines which class the test image belongs to.

Figure 3.10: The classical Hausdorff distance between two geometrical shapes, which measures how close the two shapes are to each other [150]. The solid arrow is the distance from the solid figure to the dashed figure. The dashed arrow is the distance from the dashed figure to the solid figure. The classical Hausdorff distance is the maximum between the two. In the proposed approach, a similar idea is used to compare two sets of line segments by means of the line-segment Hausdorff distance.

3.4.2 Size and Position Variance

Size and position variances exist in the source images due to the different distances and locations of the target with respect to the camera. For robust recognition, the classification approach should be size- and position- invariant. The next subsection will illustrate how the size and position invariance is achieved in the proposed posture recognition framework.

3.4.3 Size- and Position- Invariant Recognition

After the bio-inspired line feature extraction, the input image is tested for similarity with each image in the posture library. To achieve invariance to position (translation), the two images need to be aligned before computing their distance. For example, face recognition algorithms operating in modern digital cameras align a face template on the location of the eyes found in the input image [148]. To achieve this for human postures or even generic objects, it is proposed to align two objects using their center position. In addition, for size-invariant recognition, one also needs to scale the two objects to make them have the same size.

Assume that the size and position information of the two objects for matching has been obtained (note that the calculation of size and position will be illustrated in following section), then both position- and size- invariant recognition can be achieved as follows. In the proposed classification framework, as shown in Fig. 3.11, the test image I_t is aligned and resized with respect to the library image I_l . The alignment of the centers is followed by a resizing operation to make them have the same size. Both alignment and re-scaling preserve the angles between the line segments and hence are consistent with the representation.

Note that the proposed approach demonstrates a great implementation efficiency of the image scaling. For instance, resizing the centered image by a ratio of α can

Figure 3.11: Size- and position- invariant recognition. After the bio-inspired feature extraction, each image is represented as a set of vectorial line segments, also called a line feature map. To achieve size-and position- invariant recognition, the test line feature map is first aligned with the library one using their center positions, and then it is scaled to make them have the same size. After the alignment and scaling, matching is performed using Line-segment Hausdorff distance (LHD).

be achieved by simply multiplying α with the coordinates of the line segments. This is a built-in advantage of the vectorial feature representation. While in conventional approaches, scaling an image involves complex operations such as nearest-neighbor interpolations, supersampling and resolution synthesis.

3.4.4 Clustering-based Size and Position Calculation

In the previous section, in order to achieve size- and position- invariant recognition, alignment and scaling are performed on the test image (more precisely, on the test line feature map) to make it at the same position, and with the size, as the library image. However, the alignment and scaling operation requires the size and position information of the two images. This section will illustrate how to obtain these information.

A clustering-based object localization algorithm is proposed for the size and position calculation. This approach processes the motion events on the fly, in other words,

each motion event from the sensor is immediately processed by the algorithm. The size and position information is available after the last event of one frame is received and processed. In addition, the clustering-based size and position calculation module can run in parallel with the line feature extraction unit. It is highly efficient and only involves simple computation.

In this clustering-based approach, the key processing element is a so-called “cluster”, which is a block of pixels belonging to the same object. As illustrated in Fig. 3.12, a cluster is visually represented as a rectangular shape and is uniquely defined by its boundary coordinates $(r1, c1), (r2, c2)$. Besides these boundary coordinates, parameters of a cluster also include number of events (NoE). By trading off between performance and implementation complexity (explained in Section 3.4.4.2), three clusters are employed and the chip is therefore able to track up to three objects at the same time.

3.4.4.1 Clustering of Motion Events

The algorithm is implemented in an on-the-fly fashion. It continuously monitors the incoming pixel data (motion event), in which ‘1’ stands for a pixel on a motion object (active motion event) and ‘0’ for a still background pixel (inactive motion event). For the sake of clarity, the “event” mentioned in the following part always refers to the active motion event.

At the beginning of each frame, all clusters are reset (i.e., number of events of each cluster is cleared to 0). A cluster is initiated upon the arrival of the first active motion event. Later on, the algorithm examines the distance of a new event to the existing clusters (non-empty) based on the following criterion:

$$d_x < Th_x \text{ and } d_y < Th_y \quad (\text{Eq. 3.2})$$

where d_x and d_y are the row distance and column distance between the event and the cluster center, respectively. Th_x and Th_y define a *search range* (the largest rectangular area in Fig .3.12) with respect to the center of the existing cluster. Th_x and Th_y are not constants; they instead grow with the cluster. Let w and h be the width and height of the cluster, then Th_x and Th_y can be described as

$$\begin{aligned} Th_x &= Th0 + h/2 \\ Th_y &= Th0 + w/2 \end{aligned} \quad (\text{Eq. 3.3})$$

where $Th0$ is a user-defined *threshold*, which stands for the distance from the boundary of the cluster to the boundary of search range.

The clusters will then be updated according to the following procedure:

Figure 3.12: Updating an existing cluster upon the arrival of a new active motion event (pixel value equals to ‘1’). $(r1, c1)$ and $(r2, c2)$ refer to the cluster’s original boundaries, w and h are its width and height, and Th_x and Th_y define the search range of the cluster. d_x and d_y show the row distance and column distance between the cluster center (red cross) and the incoming event, respectively. If the new event resides within the search range of the cluster, then it is defined as belonging to that cluster, and the cluster’s information will be updated correspondingly.

- If the event falls within the boundary of an existing cluster, the cluster simply increases its number of events by one. If the event falls out of the cluster boundary but is still within the search range, it is still considered to be part of this cluster, and the cluster therefore grows its boundary to enclose the event. This procedure is illustrated in Fig. 3.12.
- If the incoming event is out of the search range, a new cluster, centered at the address of the event, needs to be initiated.
- In case the event belongs to more than one cluster at the same time, these clusters are merged into a single larger cluster. As illustrated in Fig. 3.13, when the search ranges of cluster 1 and cluster 2 overlap and an event occurs in the common region, the two clusters are merged into a single one and the updated information is stored into cluster 1. This strategy is named as “instant merging”. In this way, the resource of cluster 2 can be reused, effectively reducing the required total number of clusters.

Figure 3.13: Illustration of instant merging. If the incoming event belongs to more than one cluster at the same time, then all corresponding clusters and current event will be merged together.

- If the event belongs to none of the existing clusters, a new cluster needs to be initiated. Due to limited resources, there is a chance that all clusters are deployed. In this case, a discarding strategy is adopted. The cluster containing the least number of events is considered a noise object and discarded. At the same time, it is re-initiated according to the address of the event.

Fig. 3.14 shows the intermediate clusters during the sequential scanning of a binary image, which models the output data stream from the image sensor. It illustrates the above-mentioned clustering process, including initiation of a new cluster, cluster growing, merging, and discarding.

3.4.4.2 Simulation and Discussion

In order to evaluate the proposed algorithm and select optimal parameters (search range threshold $Th0$ and number of clusters), a quantitative criterion is defined. As illustrated in Fig. 3.15(a), for a test image with a resolution of 64×64 , the boundary of the largest object in the scene is manually marked as the reference ground truth, and is compared to the boundary computed by the algorithm (at certain $Th0$ and number of clusters). The localization error is defined as follows:

$$err = N_d / N_t \quad (\text{Eq. 3.4})$$

where N_d is the number of pixels in the non-overlapping area of the two bounding boxes, and N_t is the number of pixels in the ground truth box.

A library of 120 motion images have been built, including various scenarios such as road traffic, pedestrians, and laboratory activities. A few examples are shown in Fig. 3.15(b). Based on the test image library, the choice of search range threshold $Th0$ is first evaluated. Given a fixed number of clusters (e.g. $1, 2, 3, \dots$), for a certain threshold, $Th0 = k$, $1 \leq k < 64$, each image of the test sequence produces an error

Figure 3.14: Evolution of the intermediate clusters during the sequential scanning of a temporal difference image. (a) shows the test image which models the output data stream from the sensor. (b) The first motion event initiates cluster 1 (solid box). (c) As more events are received and processed, cluster 1 is enlarged, and cluster 2 is initiated (dashed box). (d) Instant merging: A new event belongs to both cluster 1 and 2, and the two clusters are therefore considered to be parts of one object and merged. (e)-(f) show the discarding strategy. A new event belongs to none of the existing clusters and all cluster resources are taken, then the cluster containing the least number of events (cluster 3, in this example, shown as a dotted box) is considered a noise object and discarded. At the same time, this cluster is re-initiated at the address of the event.

(err_i , $i = 1, 2, \dots, N$; $N = 120$). The root mean square error (RMSE) is calculated based on the following performance metric:

$$RMSE_k = \sqrt{\frac{1}{N} \sum_{i=1}^N err_i^2} \quad (\text{Eq. 3.5})$$

On one hand, a smaller threshold (corresponding to a conservative boundary expansion rate) leads to better noise rejection and allows production of a “cleaner”

Figure 3.15: (a) shows the figure that is used to illustrate the definition of localization error. Dotted regions represent the non-overlapping area. (b) lists some sample test images. (c) and (d) present corresponding ground truth and simulation results (at certain $Th0$ and number of clusters), respectively.

bounding box for an object. On the other hand, a larger threshold (corresponding to an aggressive boundary expansion rate) is required to merge discontinuous regions of one object, which are commonly found in temporal difference images due to the non-uniform motion speed of parts belonging to one object. With the given library, different numbers of clusters (1 to 5) were tried. For each number, the search range threshold $Th0$ is swept to find an optimal value, based on the minimum RMSE rule. When using only one cluster, the optimal threshold $Th0$ was found to be 10; while for two to five clusters, the optimal thresholds are all coincidentally equal to 3. Fig. 3.16 shows the simulation result for three clusters. The best clustering performance (i.e. the minimum RMSE) for each cluster number is reported in Fig. 3.17. More clusters undoubtedly improve performance, however the improvement almost saturates when the cluster number is more than three. In addition, a larger number of clusters will involve more operations such as distance measurement and comparison, and hence result in increased hardware resources. Because of the above considerations, three clusters are employed in the system.

Figure 3.16: RMSE versus $Th0$ for a test image sequence (given three clusters). The optimal $Th0$ is 3.

With the obtained parameters (i.e., 3 clusters and $Th0 = 3$), the algorithm is executed on a short motion video sequence. For each frame, the computed bounding box is compared with a manually marked ground truth. Simulation results are shown in Fig. 3.18.

3.4.4.3 Performance Under Different Noise Conditions

In order to quantitatively analyze the influence of noise on the algorithm's performance, salt and pepper noise were added to all the library images. Different noise densities were tried, ranging from 0.01 to 0.10, with a step length of 0.01. The RMSE for each noise density was calculated and shown in Fig. 3.19. As expected, the RMSE gradually increases with the increase of noise density. Fig. 3.20 shows samples of localization results under different noise conditions. It demonstrates the robustness of the proposed algorithm in the presence of low to medium level noise.

Figure 3.17: RMSE versus number of clusters (@ corresponding best threshold Th_0). Three clusters are enough for the given test image library.

3.5 System Performance Evaluation

In order to evaluate the system performance, libraries were built by choosing a number of representative images for each human posture. Standard libraries with the event-based temporal-difference format were not found in the literature or online, so one specific library was built for testing the proposed algorithm. The line segments of every such image were extracted and stored as library of features. Next, each image in the test database was compared to each image in the library. The number of successful matches (successfully categorized postures) that the algorithm yielded were recorded.

Three sets of live images have been captured. During the data acquisition, the person stands in front of the sensor with a distance ranging from 2 meters to 5 meters. As long as the person's main body is enclosed in the field of view (FOV), the proposed algorithm can effectively localize it and perform categorization. As for the viewpoint,

Figure 3.18: Comparison of the width and height of the computed bounding box to the ground truth box for the test library.

the person shows his lateral profile for the posture “bend”, and shows his frontal or rear profile for postures “hand1” “hand2” “squat” and “swing”. These postures can have a tilt angle of up to $\pm 30^\circ$; while for “stand” the view angle can be anywhere. The first test set consists of 6 groups of samples using a web camera (with a scaled resolution of 64×64), approximately 1700 images. A training set of only 30 images taken from the larger test set was used. As reported in Table 3.3, the average success rate is 84%. The second set consists of 6 groups of postures, each group has 200 images (with a resolution of 64×64), among which 100 are used for test and 30 of the others are taken for training. The success rate is 90% (see Table 3.4). The third set was obtained from another type of image sensor, which is not based on differencing full frames, but on focal plane pixel light intensity temporal derivative computation and normalization with respect to ambient light [43]. When the change of light in a pixel passes a threshold, an event is triggered. The corresponding pixel address is transmitted and at the receiver

Figure 3.19: RMSE versus noise density. (Algorithm parameters: three clusters; search range threshold $Th0$ equals to 3.)

Figure 3.20: Localization results of an example image under different noise conditions. The first one is the result before adding noise. The others are results under different levels of salt and pepper noise interference (from left to right, noise densities are 0.01, 0.02, 0.05, and 0.10, respectively).

side, the silhouette of a moving object can be reconstructed [26]. Based on a set of recorded data, four groups of postures were built. Each group contains 100 binary images (at a resolution of 128×128), among which 50 were used for testing and 30 of the others were taken for training. The average success rate is 81% (see Table 3.5).

For purpose of comparison, a fourth data set has been extracted from the Yann LeCun and Fu Jie Huang's library small NORB object dataset, V1.0 [151], available online [152]. The original images are in gray scale, and were simply thresholded to obtain a binary image compatible with the temporal difference inputs. 60 test images were used in this case, with a training set of only 15 images taken from the larger test

	Bend	Hand1	Hand2	Stand	Squat	Swing Hand	Total

No. Images	317	262	236	283	327	288	1713
Categorized	289	178	205	268	288	210	1438
Success Rate	91%	68%	87%	95%	88%	73%	84%

Table 3.3: Experimental results for images taken by a web camera.

	Bend	Hand1	Hand2	Stand	Squat	Swing Hand	Total

No. Images	100	100	100	100	100	100	600
Categorized	98	83	93	99	95	74	542
Success Rate	98%	83%	93%	99%	95%	74%	90%

Table 3.4: Experimental results for images taken by a temporal difference image sensor (use 30 training images for each group).

set. The success rate is 87% (see Table 3.6).

The proposed algorithm has been compared to the original HMAX scheme [37] and the model by T. Serre et. al [38]. Matlab implementations of the two approaches

	Hand2	Stand	Squat	Swing Hand	Total

No. Images	50	50	50	50	200
Categorized	48	35	42	37	162
Success Rate	96%	70%	84%	74%	81%

Table 3.5: Experimental results for images taken by asynchronous motion detection image sensor [43] (use 30 training images for each group).

	Human	Plane	Truck	Total

No. Images	20	20	20	60
Categorized	20	15	17	52
Success Rate	100%	75%	85%	87%

Table 3.6: Experimental results for images from Yann LeCun and Fu Jie Huang's library small NORB object dataset, V1.0 [151]. The original images were converted into the binary format first.

can be found on the Internet. Both the two approaches use Support Vector Machine (SVM) as classifier. To perform multi-class categorization on the groups of postures, one-versus-one (OVO) SVM scheme is employed. For c classes, $c \times (c - 1)/2$ times OVO SVMs are needed. For [38], each image is described with a 150-dimension $C2$ feature vector, and 15 OVO SVMs are used for this 6-group categorization problem.

Fig. 3.21 show the simulation results of the three methods, namely the original HMAX+SVM, Serre's model+SVM, and the proposed algorithm, using the second data set which is obtained from a temporal difference image sensor. The simulations were performed on a laptop computer equipped with Intel Core I5-540M CPU and 4GB RAM. Categorization success rate and CPU time are measured with respect to different numbers of training images. The proposed algorithm gives the highest success rates and consumes moderate CPU time. This simulation also showed the tradeoff between the size of training image set and system performance (success rate and run time). Larger size of training set leads to higher success rate but at the expense of more execution time. One can note that the proposed algorithm does not require a large training set. 10-30 training images per group can have a pretty good result and at the same time achieve more than 50% saving in CPU time compared to Serre's

Figure 3.21: Comparison between the proposed algorithm, original HMAX [37] and T. Serre's Model [38] using the second data set which is obtained from a temporal difference image sensor. Better categorization results are obtained when more images are used as training images, but at the expense of more execution time. (a) average success rate vs. number of training images per group. (b) CPU time vs. number of training images per group.

model.

3.6 Computational Cost of the System

3.6.1 Computational Cost Of Line Segment Extraction

For a given image with dimensions $n \times n$, the number of computations of the algorithm is estimated as follows. In the proposed system, 16 Gabor filters F_i of 4 sizes $|F_i| \in \{3, 5, 7, 9\}$ and 4 orientations were used. Convolution of the image with each of the filters was performed. However, the convolutions are computed only when an active event occurs (i.e., when the sensor's binary output is "1"). Since only the pixel's neighbors are affected through the convolution (as explained in Section 3.3), the number of operations (additions and subtractions) per filter is $|F_i|^2$, where $i = 1 \dots 16$. Note that in the adopted temporal difference image sensor, the number of active events per

image (n') is significantly smaller than the total number of pixels n^2 . It was found empirically that n' is at most 25% of n^2 . Thus, the total number of operations per image is:

$$\sum_{i=1}^{16} n' \times |F_i|^2 = 4n'(3^2 + 5^2 + 7^2 + 9^2) = 656n'. \quad (\text{Eq. 3.6})$$

Within the MAX operation, only the surviving neurons whose number is n'' were taken into account. It was found empirically that n'' is about 11% of n^2 . For a single neuron, the number of operations (comparisons) is $|F_i|^2$ per filter. Hence, the total number of operations on the MAX stage of the algorithm sums up to:

$$\sum_{i=1}^{16} n'' \times |F_i|^2 = 656n''. \quad (\text{Eq. 3.7})$$

In addition, at the last stage (the line segment extraction), the number of operations performed is equal to the number of the line segments e where $e \ll n^2$ (in the simulation, the size of the image is 64×64 and the typical number of line segments is 30).

In the proposed system, the overall number of operations is:

$$656n' + 656n'' + e = 656 \times 36\% \times n^2 + e \approx 9.7 \times 10^5 \quad (\text{Eq. 3.8})$$

3.6.2 Computational Cost Of Classifier

Suppose the library consists of k images (referred to as library images). Let e be the average number of line segments representing an image. To classify the given image I_{test} , its distance to each library image I_{lib} has to be calculated. For each library image I_{lib} , both $\vec{D}(I_{test}, I_{lib})$ and $\vec{D}(I_{lib}, I_{test})$ need to be computed, costing the same number of operations on average. Thus, the total cost is $k \times 2T(e)$, where $T(e)$ is the cost of evaluating Eq. A.5.

The element $\left(\min_{e_t \in I_{test}} d(e_l, e_t) \right)$ in Eq. A.5 is first examined. For all line segments in the test image $e_t \in I_{test}$, the distance between each e_t and a fixed line segment in the library image e_l is computed. The minimum distance is then selected. Note that the squared distance $d(e_l, e_t)^2$ (as shown in Eq. A.9) is actually used to avoid the square root operation in $d(e_l, e_t)$ (see Eq. A.1) during the searching process. The square root operation will be taken after finding the minimum squared distance. The cost of computing each $d(e_l, e_t)^2$ is estimated as follows. There are 3 cases:

- e_l and e_t are perpendicular: 0 operations (the distance is infinite). This occurs in 1/4 of all cases.
- e_l and e_t are parallel. 5 additions and 3 multiplications are needed to compute the first component of the distance in Eq. A.9. In case it is below the current minimum, the second component of Eq. A.9 needs to be further computed (8 multiplications and 6 additions). Since the second component is not required in approximately 1/2 of the cases, the total number is 8 additions and 7 multiplications on average. This occurs in 1/4 of all cases.
- e_l and e_t intersect at angle $\pm 45^\circ$. The number of operations is the same as in the parallel case plus 2 additions and 2 multiplications. This occurs in 1/2 of all cases.

After computing $\left(\min_{e_t \in I_{test}} d(e_l, e_t)^2 \right)$ for each line segment e_l of I_{lib} , e square root operations are taken to obtain $\left(\min_{e_t \in I_{test}} d(e_l, e_t) \right)$. The other computations in Eq. A.5 then further require e multiplications and $2e$ additions. Also, the line segments of I_{test} have to be shifted and rescaled to achieve position and size invariance ($2e$ operations for each lib image), as explained in Section 3.4.3.

Overall the total cost for classifying a test image (including alignment, scaling and the distance calculations) is:

$$\begin{aligned}
 k \times 2e + k \times 2T(e) = \\
 2k \times (e + (4e + e \times \text{cost}(d(e_t, e_l))^2) = \\
 2k \times (5e + e \times (e/4 \times 0 + 15 \times e/4 + 19 \times e/2)) \approx \\
 30ke^2 + 10ke.
 \end{aligned} \tag{Eq. 3.9}$$

The value of k for the posture library is about 30, while the typical value of e for a 64×64 image is about 30. It results in $\approx 8.2 \times 10^5$ operations.

3.7 Hardware Implementation of The Frame-based Recognition System

This section illustrates the VLSI design of the proposed human posture recognition system.

3.7.1 Bio-inspired Line Feature Extraction

For real-time applications, the proposed biologically inspired feature extraction needs to be implemented into hardware. In the method, the input image is convolved with a bank of Gabor filters (4 scales and 4 orientations), obtaining 16 $S1$ feature maps which correspond to simple cells in primary visual cortex. Two-step MAX-like operation is further performed on $S1$ maps to find the position, size and orientation of the line features. This section will illustrate the hardware design of the bio-inspired feature extraction unit.

3.7.1.1 Hardware Implementation Considerations

The MAX-like convolutional network used to extract line features involves vast amount of memory access and comparison, therefore special care is required to efficiently implement it into hardware.

Firstly, the event stream produced by the motion image sensor is to be processed in parallel by a battery of $S1$ convolutional filters. Conventional implementation requires the whole frame of pixels first being stored in one set of memory, then each $n \times n$ ($n = 3, 5, 7, 9$) pixels undergo convolution with a kernel, and finally the response has to be written into another set of memory. Thanks to the nature of event-based representation of the image sensor, the on-the-fly convolution scheme [153] can be adopted. Within each “neuron” map, the “neurons” are updated on the fly, reducing both memory cost and computation time.

Secondly, during MAX operation a large number of “neurons” are to be read out from the memory and get compared in parallel. For the largest size “neuron”, each MAX operation involves $9^2 \times 16 = 1296$ operands, producing a big challenge to the memory throughput. On top of this, the number of computation units (i.e. comparators) is another design trade-off. Without any optimization, the number of comparison can reach up to

$$N_c = \sum_{i=1}^4 4(n_i^2 - 1) + \sum_{i=1}^4 [16(n_i^2 - 1) + 24] = 3296 \quad (\text{Eq. 3.10})$$

where $n_i = (3, 5, 7, 9)$. Full parallel implementation will cost large amount of hardware resources.

3.7.1.2 Hardware Architecture

Fig. 3.22 shows the block diagram of the feature extraction unit. Each motion event is sent in parallel to 16 sets of “on-the-fly-Gabor + memory”. Each memory has 4096

words (the sensor features a resolution of 64×64) and 11-bit width, among which 10 bits represent $S1$ response value and the other 1-bit stores a survival flag. Upon the arrival of the last motion event, the two-step MAX operation is initiated. As discussed earlier, this procedure involves a large amount of comparisons in parallel. In order to provide high throughput, a memory interface is designed to wrap the memory array, allowing each SRAM to output 32 words concurrently by running internally at a $32 \times$ clock. The central controller produces SRAM read/write addresses and delivers $S1$ values to an array of comparators. In the first phase MAX operation, when a “neuron” fails the competition, the controller will toggle its corresponding survival flag and write it back into the memory. In the second phase competition, each surviving “neuron” represents a line segment with certain size and orientation.

Figure 3.22: Block Diagram of the feature extraction unit

3.7.1.3 Hardware reuse methodology

To alleviate the heavy requirement on hardware resources, a pipelined technique was proposed to reuse both the computation circuits and memory data. The principle is illustrated in Fig. 3.23. For the sake of clarity, Fig. 3.23(a) shows two 5×5 consecutive

“neurons” during the 1st stage MAX operation. One can note that when the MAX operation moves to a new “neuron” (denoted as “current neuron” in the figure), it shares a large number of operands (shadowed) with the previous “neuron”. A straightforward way to optimize the memory access is to “cache” the overlapped data and each time only read in a new column. To further reuse the computation results, the comparison is performed in a column-wise manner by finding the maximum value within each column. The centering “neuron” only needs to compete with 5 column-max values, of which the rightmost 4 will be “cached” and reused by a new “neuron”. In this way, the number of “cache”, memory access and comparison is largely reduced.

Figure 3.23: Hardware reuse methodology for MAX operation. (a) shows the case of MAX over local neighborhood (taking the “neurons” with receptive field of 5×5 as example). The current “neuron” shares a large number of operands (the 4 shadowed columns) with the last “neuron”. MAX is computed column by column. The rightmost four column-max in last neuron’s computation are stored and reused for current neuron’s processing. (b) illustrates the reuse technique in MAX over orientations and scales. The maximum value of the 3×3 block can be reused in computing the maximum of 5×5 region, which can then be reused in finding the MAX of 7×7 , and similar rule applies to 9×9 . Each time the “neuron” size increases, only the new peripheral data need to be read in and compared.

A similar technique is applied to the 2nd phase MAX operation. As introduced earlier (Fig. 3.7), competition is performed over all orientations and scales. The procedure

appears fully parallel and requires huge hardware resources. To reduce the hardware cost, a bottom-up approach was proposed, with the processing being carried out from the smallest size to the largest (i.e., $3 \rightarrow 5 \rightarrow 7 \rightarrow 9$). The comparison is performed in an array-wise manner (as shown in Fig. 3.23(b)). For the smallest size 3, the maximum value of each 3×3 block is computed. The centering “neuron” only needs to compete with other 6 block-max values (over orientations and scales, not shown). Next, the “neuron” size grows to 5. One can note that the maximum value of the overlapping 3×3 block can be reused. Therefore, each time the “neuron” size increases, computation is only limited to finding out the maximum value among the new “peripheral” data. The most computation-intensive scenario happens when the “neuron” size grows from 7 to 9, where the maximum value among $9^2 - 7^2 = 32$ numbers will be calculated.

In summary, with the above hardware reuse methodology, the maximum parallel comparison is only $(9^2 - 7^2)/2 \times 16 = 256$. A dramatic reduction of 92% has thus been achieved when comparing to 3296 for a direct implementation).

3.7.1.4 Implementation Results

The proposed edge feature extraction algorithm has been implemented using UMC $0.18\mu\text{m}$ CMOS technology. Table 3.7 summarizes the implementation results. The design can operate at a maximum clock frequency of 14.8 MHz. The feature extraction for each frame, which involves on-the-fly convolution and 2-step MAX operation, totally needs about 286720 clock cycles. Therefore, a maximum frame rate of 51 can be attained. This means the design can support most real-time applications. In addition, the total hardware savings enabled by the proposed reuse technique was also estimated. It was shown that the number of logic gates drops from 402k to 170k, a reduction of 58% has thus been achieved.

The achieved specifications are further compared with other hardware implementations of HMAX-like feature extractors, as shown in Table 3.8.

Table 3.7: Hardware implementation results of the feature extraction unit

Technology	UMC 0.18 μm CMOS
gate count	170k
MAX freq.	14.8 MHz
processing capability	51 FPS
silicon area	1.5 mm $\times$ 1.5 mm
power consumption	90 mW @ 30 FPS

Table 3.8: Comparison of hardware implementations for HMAX-like feature extraction

	this work	Folowosele et al. [154]	Sabarad et al. [155]
Technology	0.18 μm CMOS	0.5 μm CMOS	Xilinx Virtex-6 SX475T FPGA
Processing layers	$S1, C1$	$S1, C1, S2, C2$	$S2$
input size	64×64	24×24	256×256
processing capability	51 FPS	1/4.8 FPS	22.92 FPS
silicon area	1.5 mm $\times$ 1.5 mm	3 mm $\times$ 3 mm	Not applicable
power consumption	90 mW @30 FPS	645 μW	65 W

3.7.2 Clustering-based Size and Position Calculation

This section illustrates the hardware design of the clustering-based size and position calculation module (see Section 3.4.4).

3.7.2.1 VLSI architecture

Fig. 3.24 shows the block diagram of the object localization unit. Major building blocks include three sets of registers for storing clusters' information and a batch of computational logics, including Distance Measurement Unit ("belong2clu1", "belong2clu2" and "belong2clu3"), "Clustering Flags Generator", and "Clusters Update Logic".

One interesting property of the proposed VLSI architecture is the on-the-fly clustering of each motion event. As shown in Fig. 3.24, the motion event and its address are sent in parallel to three blocks (namely "belong2clu1", "belong2clu2", and "be-

long2clu3”) to determine whether this event belongs to any of them, leading to three bits of comparison results. A variety of clustering flags are further generated from these comparison results. The “Clusters Update Logic” uses these clustering flags to update the values of cluster registers. At each clock cycle, cluster registers are updated. And at the end of one frame, the clusters’ information is fixed and outputted.

Figure 3.24: Block diagram of the object localization unit.

3.7.2.2 Distance Measurement

The motion event and its address are sent in parallel to the distance measurement unit. Based on the criterion described in expression (Eq. 3.2), three bits of comparison results are produced, namely “belong2clu1”, “belong2clu2” and “belong2clu3”, respectively.

For instance, signal “belong2clu1” indicates whether the new event falls within the search range of cluster 1. Let (x, y) , (x_c, y_c) , $(r1, c1)$ and $(r2, c2)$ denote the addresses

Figure 3.25: Arithmetic operations for $(a < b)$ and $abs(a - b)$. (a) $(a < b)$. (b) $abs(a - b)$.

of the motion event, cluster-1's center, top left corner and bottom right corner, respectively. The distance between the event and the cluster center, the search range, and the final comparison result are then computed by:

$$\begin{aligned}
 h &= r2 - r1 \\
 w &= c2 - c1 \\
 Th_x &= Th0 + h/2 \\
 Th_y &= Th0 + w/2 \\
 x_c &= (r1 + r2)/2 \\
 y_c &= (c1 + c2)/2 \\
 d_x &= abs(x - x_c) \\
 d_y &= abs(y - y_c) \\
 belong2clu1 &= (d_x < Th_x) \& \& (d_y < Th_y)
 \end{aligned}
 \tag{Eq. 3.11}$$

The arithmetic operations involved here are addition, subtraction, “divided by 2”, comparison $(a < b)$, and absolute difference computation $(abs(a - b))$. The addition and subtraction are implemented as 7-bit adders. “Divided by 2” is just to shift the operand to the right by 1 bit. The comparison $(a < b)$ operation is essentially a subtraction as shown in Fig. 3.25(a). The absolute difference operation $(abs(a - b))$ is about making a selection between $(a - b)$ and its two's complement based on the value of the most significant bit (MSB) of $(a - b)$. As shown in Fig. 3.25(b), if the MSB is 0, the former is selected; otherwise, the latter is selected.

3.7.2.3 Clustering Flags Generator

The three bits of comparison results (generated by blocks “belong2clu1”, “belong2clu2”, and “belong2clu3”), together with the information on number of events (NoE) in each cluster, are fed to “Clustering Flags Generator” to generate a variety of clustering flags.

For instance, the input combination of 'b100 (“belong2clu1”=1, “belong2clu2”=0, “belong2clu3”=0) indicates that the motion event only belongs to cluster-1 and hence cluster-1 should be enlarged. As shown in Fig. 3.26, this can be simply implemented in hardware by a “AND3” gate, which outputs a flag signal, denoted as “flag_enlarge_1”. Another case of input combination 'b111 asserts “flag_merge_1~3”; this corresponds to a merge operation of all three clusters on condition that the motion event belongs to all of them. In addition, for the input 'b000, which means the event belongs to none of the clusters, one of three flags (“flag_init_1”, “flag_init_2” and “flag_init_3”) will be asserted based on which cluster has the smallest NoE . This actually corresponds to two kinds of operations in the proposed algorithm: 1) the discarding strategy (if the smallest NoE is not 0, i.e., all clusters resources have been occupied), and 2) initiating a new cluster (if the smallest NoE is 0, i.e., there is at least one empty cluster available). Note that the “discarding” and “re-initiating” processes in the discarding strategy (see Section 3.4.4.1) are simplified as “initiating”, since the “discarding” is only a temporary state and can be omitted without influencing the result.

3.7.2.4 Clusters Update Logic

The aforementioned flags will determine the clusters' update operations, i.e, initiation, growing, merging, etc. In hardware (shown in Fig. 3.27), a batch of dedicated arithmetic blocks first compute every possible future cluster information (i.e., boundary address and number of events). The clustering flags then control a series of tri-state buffers to select one out of the future values.

Figure 3.26: Generation of the clustering flags. The 3 bits of comparison results (“belong2clu1”, “belong2clu2”, and “belong2clu3”), together with the information of number of events (`NoE`) in each cluster, generate ten clustering flags.

The boundary update logics shown in Fig. 3.27 perform the functions of initiating cluster-1, enlarging cluster-1, merging cluster 1&2, merging cluster 1&3, and merging all three clusters, respectively. The transmission of their outputs is controlled by several clustering flags, among which only one flag is active at any time. For instance, when “flag_init_1” is active, the results of operation “initiating cluster-1” are sent to cluster-1 registers. Similarly, when “flag_merge_1&2” is active, the results of operation “merging cluster 1&2” are instead selected.

The computational core in each arithmetic block only consists of adders. For instance, enlarging cluster-1 involves the following arithmetic operations:

$$\begin{aligned}
 r1' &= \min(r1, x) \\
 r2' &= \max(r2, x) \\
 c1' &= \min(c1, y) \\
 c2' &= \max(c2, y) \\
 NoE1' &= NoE1 + 1
 \end{aligned}
 \tag{Eq. 3.12}$$

Figure 3.27: Update of cluster-1 registers. Clustering flags control a series of tri-state buffers to choose the right boundary update results.

where $(r1, c1)$, $(r2, c2)$ and $NoE1$ denote the original boundary and number of events, and $(r1', c1')$, $(r2', c2')$ and $NoE1'$ represent the updated boundary and number of events. The computation of $NoE1'$ is simply implemented as a 13-bit adder (bitwidth 13 is selected because the number of events is within the range of $[0, 4096]$). The other two operations, $\min(a, b)$ and $\max(a, b)$, are just to make a selection between a and b based on the value of $(a < b)$, which, as mentioned earlier, is implemented as a 7-bit adder as shown in Fig. 3.25(a).

3.7.2.5 Complexity With Respect To Scaled Number of Clusters

The implementation complexity of the localization block with respect to a scaled number of clusters is examined:

- Storage of the clusters (i.e., flip-flops) will linearly increase. Each cluster records

its two corner addresses $(r1, c1)$ and $(r2, c2)$, and number of events (NoE). This needs 41 bits of flip-flops in total.

- Since a new event has to check whether it belongs to each cluster, the complexity of the distance measurement unit (i.e., $+$, $-$, and $<$ operations) will linearly increase.
- More inter-cluster operations are needed. For instance, when there are three clusters, an incoming event will check whether it belongs to only one of them or simultaneously to two of them, or even to three of them. These operations are translated to “AND” gates (as shown in Fig. 3.26). For N clusters, the total number of “AND” operations is $C(N, 0) + C(N, 1) + C(N, 2) + \dots + C(N, N) = 2^N$. Moreover, in order to find out which cluster has the least number of events, another $2N$ “AND” gates and $C(N, 2)$ comparators are required.

In summary, building larger number of clusters on-chip allows to track more objects and offers better noise rejection performance, however this is at the expense of more hardware resources.

3.7.2.6 Implementation Results

The clustering-based size and position module has been implemented using UMC $0.18\mu\text{m}$ CMOS technology. Table 3.9 summarizes the implementation results. The design occupies a small silicon area of $600\mu\text{m} \times 220\mu\text{m}$, with gate count of approximately 8.6k. It can operate at a maximum clock frequency of 75.7 MHz. This module shares the same clock with the image sensor. In each clock cycle, one event is generated from the sensor and sent to this module for clustering. Therefore, the maximum processing speed of this module is about 75.7 M events/s. Note that the image sensor has a resolution of 64×64 , this means that each frame takes 4k clock cycles, thus a

frame rate of 100 FPS requires a clock of only 400kHz. The design can easily follow the speed of the sensor ($400k \ll 75.7M$). The power consumption at 400kHz is about $8.63 \mu W$.

Table 3.9: Hardware implementation results of the clustering-based size and position module

Technology	UMC 0.18 μm CMOS
gate count	8.6k
MAX freq.	75.7 MHz
processing capability	75.7 M events/s
silicon area	$600\mu m \times 220\mu m$
power consumption	$8.63 \mu W$ @ 400kHz (about 100 FPS for 64×64 resolution)

The clustering-based size and position calculation unit has been verified in a prototype smart image sensor. It was integrated with a temporal difference pixel array, leading to a CMOS image sensor with motion detection and object localization functionality. This sensor was fabricated using UMC 0.18 μm CMOS process. The chip has a die size of $1.5mm \times 1.5mm$ (including pads). Fig. 3.28 shows the chip microphotograph with main building blocks highlighted. Each pixel features an area of $14 \times 14 \mu m^2$ with a fill-factor of 32%. The object localization unit was implemented with standard cells and occupies a relatively small footprint of $600 \times 220 \mu m^2$. Guard rings were extensively used to limit substrate coupling and shield the pixels from the outer-array digital circuitry. Power and ground buses were routed using top layer metal. The chip characteristics are also summarized in Table 3.10.

3.8 Discussion

As seen in Section 3.5, the combination of a temporal difference image sensor and the proposed feedforward characterization algorithm performs well with both objects and human postures. This system is not completely free of problems:

Figure 3.28: Chip microphotograph of the image sensor with motion detection and object localization functionality.

Table 3.10: Chip Characteristics of the image sensor with motion detection and object localization functionality

Process Technology	UMC 0.18 μm 1P6M CMOS
Die Size	$1.5 \times 1.5 \text{ mm}^2$
Pixel Array	64 x 64
Pixel Size	$14 \times 14 \mu\text{m}^2$
Number of trans/pixel	10
Fill Factor	32%
Readout Strategy	sequential scan
Frame Rate	100fps
Supply Voltage	1.8 V
Power Consumption	pixels array + motion detection 0.4 mW, object localization 8.63 μW (@100fps)

One typical problem is when multiple objects are moving back and forth in the scene, or the background is moving. In this case, categorization fails because the algorithm loses the person-of-interest. When monitoring a single person in a room, like in assisted living applications, this is not an issue. However, for real-world applications, an object tracking stage should be added to the system. At present, the event-based clustering algorithm can locate the size and position of one human in the scene and reject small disturbing moving objects in the background, such as a cat [156]. A further challenge emerges when multiple objects run into one and then separate. More

advanced object tracking algorithm or facility is to be employed.

Another concern is the system robustness against viewpoint variance and field of view full coverage. In present experimental setup, the person should show his lateral profile for the posture “bend”, and show his frontal or rear profile for posture “hand1” “hand2” “squat” and “swing”. These postures can have a tilt angle of up to $\pm 30^\circ$. For practical usage, multiple camera nodes should be used and at this point, the proposed system is superior. Due to its high computation efficiency, it allows making a compact, small footprint embedded system that can be easily installed. Since no raw video data is involved, patients’ privacy is protected when they are monitored.

One more problem is that the classification speed for each testing image is related to the size of training set. The classifier used in the frame-based posture recognition system is in fact a nearest neighborhood classifier based on LHD distance. Each testing image needs to be compared with every training image in the library. More training images will lead to more computations and thus slower classification speed, as can be seen from Fig. 3.21(b). Learning-based classifiers such as the artificial neural network can address this problem. During the training process, the features of all training images are learned and “stored in the synaptic weights of the network. The testing phase, which mainly involves weighted summation, will then be very fast and independent from the size of the training set.

Chapter 4

Feedforward Human Postures Recognition on AER vision events

This chapter describes the improved human posture recognition system for a frame-free AER vision sensor.

4.1 Introduction

Recent years have witnessed increasing efforts in event-based neuromorphic systems. Neuromorphic engineers aim to build electronic systems that have the same efficiency of brains by mimicking the biological use of the asynchronous, sparse, spike-based representation and computation [7].

AER sensors naturally provide a way to incorporate event-driven computation. These sensors generally have an output-by-demand nature. Hardware-based pixel-level computation is performed on chip to reduce output redundancy. For example, AER temporal contrast vision sensors [17, 19, 27] allow pixel-parallel detection of temporal relative changes in light intensity at the focal plane. The output of an AER sensor is an asynchronous stream of digital address events. Each event has an address and a timestamp. The address indicates which pixel the event is from, and the timestamp represents the event's time of occurrence.

In this chapter, an event based feedforward categorization system is introduced. The system processes the address events from an AER temporal contrast vision sensor [27]. Bio-inspired, cortex-like, spike-based features are obtained through event-driven convolution and neural competition. The extracted spike feature patterns are then classified by a network of leaky integrate-and-fire (LIF) spiking neurons, in which the weights are trained using tempotron learning rule.

The aim of this work is to develop a real-time human posture categorization system using AER motion sensor, which does not produce intensity images but a train of spikes. In the application of assisted living, due to concern of privacy reservation, the elderly may be reluctant to be monitored by conventional image sensors. The major contribution resides in two areas: 1) an efficient time-domain clustering algorithm to capture “motion symbols” and 2) a fully event based architecture that can emulate the biological use of asynchronous, sparse, spike-based signaling.

The rest of this chapter is organized as follows. First, the processing issues of address events are discussed in Section 4.2. Then, Section 4.3 describes the system architecture. Sections 4.4 - 4.6 illustrate the building modules of the system. Experimental results are reported in Section 4.7 and Section 4.8 illustrates the hardware implementation of the AER recognition system.

4.2 About Address Event Processing

Due to the silicon area limitation of the AER “silicon retina”, limited feature extraction can be done at the pixel level [23–25]. Therefore, the output address events usually cannot be directly fed to classification algorithms. Additional processing, such as segmentation, resizing, repositioning and even more complicated high level feature extraction, is still needed. However, most of the existing algorithms are designed for

conventional frame-based image sensors. In order to adopt these algorithms, a common practice (jAERViewer [26], for example) is to divide events into fixed time slices (20 ms, for example) and accumulate them into pseudo-frames.

One example of accumulated frame in jAERViewer is shown in Fig. 4.1. The event stream is divided into slices in time domain. Each slice has a time length of 20ms (user-specified variable, can be set to any other numbers). For instance, slice 1 contains the events that happen within $[0, 20)ms$, slice 2 contains the events that happen within $[20, 40)ms$, and so on. A frame/image can be accumulated from each slice of events. The resolution (size) of the frame is the same as that of the AER vision sensor. For the example frame shown in Fig. 4.1, the events are from a 128×128 AER vision sensor and thus the accumulated (also known as “reconstructed”) frame has a resolution of 128×128 . The frame is reconstructed in the following way. At the beginning of each slice, the pixel values of the frame/image are reset to zero. Then for each event within this slice, the corresponding pixel (specified by the address event) will increase its value by 1 or -1. If the input event has a positive polarity (also known as ON event), the increment is 1; otherwise for the OFF event, the increment is -1. After this accumulation process, all the pixel values of the frame are obtained. Some pixels have positive values (because of ON events), some pixels have negative values (due to OFF events), and some pixels remain zero (background). The frame can be shown after normalization. The background is shown in gray color, ON events in white and OFF events in black.

This frame reconstruction can also be simplified as follows. The frame is reset to zero at the beginning of the slice. Then, for each incoming event within the slice, the corresponding pixel in the reconstructed frame will be lightened up (i.e. pixel value is changed from 0 to 1). For example, let (x, y) denote the row address and column address of an input event, this event will change the pixel value at position (x, y) from

Figure 4.1: One example of jAERViewer.

0 to 1. This simplified reconstruction method will generate a binary image for each slice (black background and white foreground). In addition, all events from the sensor, whether ON events or OFF events, are treated in the same way in this simplified method. In other words, the event polarity is discarded. This simplifies subsequent processing.

Fig. 4.2 shows a space-time scatter plot of one piece of address events captured by an AER vision sensor [27]. A person is doing stand-up and sit-down actions in this recording. The lower part of this figure shows several selected frames reconstructed from the address events. By inspecting these reconstructed frames, one can easily find that the human silhouette in some frames is incomplete or totally missing. The

main difficulty arises from the asynchronous nature of “motion” with respect to the time “slice”. A motion may fall into two time slices and neither of the pseudo frames would be sufficient. In fact, this is a common problem when using frame-based image sensor for motion processing.

Figure 4.2: Example of one piece of address events captured by an AER vision sensor. The lower part shows some reconstructed frames by dividing events into time slices and applying accumulation in each slice. One can find that the human silhouette in some frames is incomplete or totally missing.

In order to fully utilize the power of AER sensors, event-based processing should be used. As the following sections will show, an event-based MAX-like convolutional network is used to extract features. Since it is event-based processing, the features are dynamically changing for each input event. Classification at such a high frequency is not only almost impossible, but also inefficient and inaccurate. (This is because the same incomplete “frame” problem may be faced, given that the features at some timings maybe incomplete). Therefore, some good timings to perform classification have to be selected, and the selection shall be adaptive to input address events. As the following sections will illustrate, a “Motion Symbol Detector” is proposed to fulfill this task.

4.3 System Architecture

Fig. 4.3 shows the architecture of the proposed system. One appealing characteristic of the proposed system is its fully event-driven processing. Similar to most categorization systems, it can be divided into two parts, namely feature extraction and classification. The adopted classifier is a spiking neural network constructed with tempotron neurons, which can learn and discriminate spatiotemporal spike patterns efficiently. The flow of information processing is as follows.

- (i) **Feature Map Construction and Neuron Competition:** Each address event from the AER vision sensor will be projected onto a group of simple filters $S1$. Each filter models a neuron cell with certain size of feature map and responses best to basic feature at a certain orientation. The response of each $S1$ neuron is changing dynamically due to on-the-fly convolution as well as a forgetting mechanism. Leakage is introduced to eliminate the impact of very old motion events on current response. $S1$ neurons compete with local neighbors through MAX operation to strive for survival in $C1$ layer. Each surviving $C1$ neuron, i.e. the local peak response, stands for a bar feature of the same size and orientation as that neuron cell [39, 40].
- (ii) **Motion Symbol Detection and Feature Spike Generation:** Note that $S1$ and $C1$ maps are updated for each incoming AER motion event. In order to avoid doing classification on the feature maps all the time, a “Motion Symbol Detector” module is introduced in the system. This module consists of a leaky integrate type neuron and a peak detection unit. Each input event initiates a postsynaptic potential to this neuron. The total potential is monitored all the time by the peak detection unit. When a peak is detected, a pulse will be triggered to turn *ON* the

Figure 4.3: Architecture of the categorization system. The system consists of several building blocks, namely convolution and competition, feature spike conversion, motion symbol detector and tempotron classifier. Each input event is projected onto a group of dynamic $S1$ feature maps through on-the-fly convolution with a forgetting mechanism. $S1$ neurons compete with local neighbors through MAX operation to strive for survival in $C1$ layer. Surviving $C1$ neurons represent some salient bar features. The “Motion Symbol Detector” can detect a burst of events in a short time and then take a snapshot of the dynamic $C1$ feature maps. Surviving $C1$ neurons at that snapshot are going through a small set of “TFS neurons” to be converted into spikes, which are further fed to a network of tempotron neurons for classification. The address of feature spike is used to fetch corresponding weight from the “weights lookup table”. The final categorization decision is made according to the output of tempotron neurons. The lower right part of the figure illustrates the concept of the proposed feature extraction. One $S1$ map is shown in the bottom. Corresponding $C1$ map in the middle has only one surviving neuron due to MAX competition. The neuron’s position is the same as that of $S1$ peak. The surviving $C1$ neuron represents a bar feature of a certain size and orientation at that position.

switches in Fig. 4.3. At that particular moment, $C1$ feature maps are fed to a set of “TFS neurons” to encode $C1$ responses into time domain spikes.

- (iii) **Categorization by Spiking Neuron Network:** The adopted classifier is a network of tempotron neurons. In principle, we need all the $C1$ responses for classification. The number of inputs of the tempotron network is the same as the number of $C1$ responses. Thanks to the beautiful nature of spatiotemporal AER spikes and the MAX operation, we can achieve a virtually fully connected system by physically activating only a very small subset of the network. Only a small portion of neurons survive in $C1$ feature maps after the MAX operation. In other words, most $C1$ neurons are killed during the competition, and the zero values of these “dead” neurons have no impact on the final output. Therefore we only need to build a few “TFS neurons” for the response-to-spike conversion. Each feature spike is associated with an address, which can be used to access to a lookup table (LUT) and fetch the corresponding weight.

Note that in the proposed system, spikes are used in all the processing stages. This is driven by a few design criteria: 1) to avoid falling back to frame-based processing; 2) to avoid processing the dynamic responses all the time; 3) to reduce resource requirement for hardware implementation.

4.4 Feature Map Construction and Neuron Competition

4.4.1 Proposed Cortex-like Feature Extraction

Inspired by the feedforward models of cortical information processing (HMAX [37] and Serre Model [38]), a convolution-based network is proposed to extract features from motion events. For the purpose of simplicity, a hierarchy of two layers ($S1$ and $C1$)

is adopted. Note that in proposed model, a different MAX operation is used in $C1$ layer, and on-the-fly convolution with a forgetting mechanism is introduced in $S1$ layer for continuous event-based processing. The overall data flow can be summarized as $motion\ events \rightarrow S1\ maps \rightarrow C1\ maps$.

Simple cells ($S1$) are used to build feature selectivity. This is done by convolving the input event with a network of Gabor filters. Each filter models a neuron cell that has a certain size of receptive field and responses best to a basic feature of a certain orientation. Considering both the coverage of various sizes and orientations and the complexity of implementing the algorithm into hardware, we trade-off the network to 4 scales (ranging from 3 to 9, with a step length of 2) and 4 orientations (0° , 45° , 90° and 135°). The function of Gabor filter can be described as:

$$G(x, y) = \exp\left(-\frac{X^2 + \gamma^2 Y^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda} X\right) \quad (\text{Eq. 4.1})$$

where $X = x \cos \theta + y \sin \theta$ and $Y = -x \sin \theta + y \cos \theta$. The filter parameters (orientation θ , aspect ratio γ , effective width σ and wavelength λ) have been well tuned in pioneering work [6, 38], and here a similar set of these parameters are adopted.

The on-the-fly convolution is illustrated in Fig. 4.4. When an input address event comes in, the convolution kernel is overlaid onto the response map at the position specified by the input event's address. Each element of the convolution kernel is then added to the corresponding original response. The response map is therefore updated. In addition, in order to eliminate the impact of very old events on current response map, a forgetting mechanism is adopted. Each pixel in the response map will decrease (or increase) toward the resting potential (usually set as 0) as time goes by. For implementation simplicity, we use a constant linear leakage.

In this way, 16 $S1$ convolution maps are obtained. For a certain object (say a bar), each "neuron" in the 16 maps gives a response, which represents the strength of the

Figure 4.4: On-the-fly convolution with a forgetting mechanism. (a) The input event comes in. (b) The convolution kernel is overlaid onto the response map at the position specified by the event address. (c) shows the updated response map after adding the convolution kernel to the map. (d) shows the decayed response map after a while.

Figure 4.5: MAX over local neighborhood. “Neurons” located in different-scale $S1$ maps have different receptive fields, such as 3×3 , 5×5 , 7×7 and 9×9 . Each “neuron” competes with all the other “neurons” located within its receptive field. It can survive in $C1$ layer only when it is the MAX in this area. The right 3D figure shows an example of one $S1$ map, in which “neuron” A will survive in $C1$ layer but B will not.

“bar” feature. $C1$ cells are obtained by performing MAX-like operation over simple $S1$ units. The MAX operation is performed across local neighborhood to find the center of the feature. As illustrated in Fig. 4.5, “Neurons” located in different-scale $S1$ maps have different receptive fields, such as 3×3 , 5×5 , 7×7 and 9×9 . Each “neuron” competes with all the other “neurons” located within its receptive field. It can survive only when it is the MAX in this area. After MAX operation, an additional thresholding procedure can be applied to further suppress those local maxima that have very small values.

After the MAX operation, each surviving “neuron” in the $C1$ maps represents a feature, i.e, a line segment with a certain size and orientation (refer to the lower right part of Fig. 4.3).

4.4.2 Simulation Results

Figure 4.6: one example piece of address events used for simulation. The upper row shows the scatter plot of address events. The lower row shows two reconstructed frames at different timings. The reconstructed frame at time t_1 shows a complete pattern “0”, while the one at time t_2 contains an incomplete pattern.

Fig. 4.6 shows one example piece of address events used for simulation. The reconstructed frame at time t_1 shows a complete pattern “0”, while the reconstructed frame at another time t_2 has an incomplete pattern. Thus, time t_1 is depicted as a

Figure 4.7: S1 maps and C1 maps at the selected “good” timing. The “good” timing means that the reconstructed image at that particular time shows a complete pattern “0”. For the input address events in Fig. 4.6, through the on-the-fly convolution and MAX operation, 16 dynamic S1 maps and 16 dynamic C1 maps are obtained. The snapshots of S1 and C1 maps at the selected “good” timing are as shown in (a) and (b), respectively.

Figure 4.8: Bars reconstructed from surviving C1 neurons at the selected “good” timing. For each surviving C1 neuron in Fig. 4.7(b), a bar (with corresponding size, orientation, and position) can be reconstructed. For instance, a surviving neuron in scale-7 orientation- 45° C1 map leads to a 45° bar of length 7 at corresponding position. (a) shows the bars reconstructed from each surviving C1 neuron. (b) shows the combined bar map.

“good” timing and time t_2 as a “bad” timing. As mentioned earlier, the S1 maps (and C1 maps) are updated for each input event. They are changing dynamically, so here only show the snapshots at a particular timing. For example, at the selected “good” timing t_1 , the corresponding S1 maps and C1 maps are shown in Fig. 4.7. Note that only a few number of neurons are surviving in C1 layer. In this AER recognition system, the surviving C1 responses (add corresponding addresses) are actually used as the features for following computation. However, to illustrate the meaning of these surviving C1 neurons, some bar maps are reconstructed. As shown in Fig. 4.8 (a), a surviving neuron in scale-7 orientation- 45° C1 map leads to a 45° bar of length 7 at corresponding position. Combining all these reconstructed bars together, the final reconstructed bar map can be obtained (see Fig. 4.8(b)). It should be emphasized

once again that this bar map is only to illustrate the meaning of the features, and that the actual features used for subsequent computation are the responses and addresses of surviving C1 neurons.

For another selected time t_2 which corresponds to an incomplete reconstructed pattern, Fig. 4.9 (a), Fig. 4.9(b), Fig. 4.10(a), and Fig. 4.10(b) illustrate the S1 maps, C1 maps, bars reconstructed from each C1 neuron, and the final reconstructed bar map, respectively.

4.4.3 Discussion

The feature extraction unit in this AER recognition system adopts the similar idea as the one in the frame-based recognition version (section 3.3). Both versions are based on convolution and MAX competition. The differences made in this AER version are as follows: 1) The event-based on-the-fly convolution with a forgetting mechanism is used. Convolution maps are updated for each input address event. The forgetting mechanism is used to forget the impact of very old input events. While in the frame-based version in Section 3.3, convolution maps are reset for each frame and thus no leakage is needed. 2) Instead of using two-step MAX operation as in frame-based version (i.e. MAX over local neighborhood, MAX over orientations and scales), here only MAX over local neighborhood is adopted. The reason is that there will be too few surviving neurons if two-step MAX operation were chosen. The corresponding recognition rate using tempotron classifier would also be relatively low. Therefore, only MAX over local neighborhood is performed in this AER recognition system. This also reduces the number of computations needed.

4.4.4 Summary

Each input address event from the AER vision sensor is projected onto a group of dynamic S1 feature maps through on-the-fly convolution with a forgetting mechanism.

Figure 4.9: S1 maps and C1 maps at the selected “bad” timing. The “bad” timing means that the reconstructed image at that particular time shows an incomplete pattern “0”. For the input address events in Fig. 4.6, through the on-the-fly convolution and MAX operation, 16 dynamic S1 maps and 16 dynamic C1 maps are obtained. The snapshots of S1 and C1 maps at the selected “bad” timing are as shown in (a) and (b), respectively.

Figure 4.10: Bars reconstructed from surviving C1 neurons at the selected “bad” timing. For each surviving C1 neuron in Fig. 4.9(b), a bar (with corresponding size, orientation, and position) can be reconstructed. For instance, a surviving neuron in scale-7 orientation-45° C1 map leads to a 45° bar of length 7 at corresponding position. (a) shows the bars reconstructed from each surviving C1 neuron. (b) shows the combined bar map.

S1 neurons compete with local neighbors through MAX operation to strive for survival in *C1* layer. After the on-the-fly convolution and MAX competition, only a very small number of neurons are surviving in *C1* layer. The features used for subsequent computations are: the responses and addresses of surviving C1 neurons.

4.5 Motion Symbol Detection and Feature Spike Generation

4.5.1 Motion Symbol Detection

As mentioned earlier, in frame-based sensors, a motion may be wrongly segmented into different frames due to the asynchronous nature of “motion” with respect to time “slice”. On the other hand, in the event-based system the *C1* feature map is updated

with each input event and every time step. So, when is the good time for classification? Note that the time interval between two consecutive events from the AER motion sensor can be very small, 100 nanosecond or less, depending on the handshaking speed of the sensor. To avoid doing classification all the time, a time domain clustering algorithm is proposed and a “Motion Symbol Detector” module is introduced to the system.

The word “symbol” is borrowed from terms in speech recognition. The AER motion sensor only outputs a few noise events when capturing a static scene. However, it generates a burst of output events when presented with moving objects. Here, the word “symbol” is used to denote one slice from such a burst of output events. The “Symbol Detector” module consists of a leaky integrate type neuron and a peak detection unit. As illustrated in Fig. 4.11(a), each input event will contribute a postsynaptic potential (PSP) to this neuron. For an input event received at time t_i , the normalized PSP kernel K is defined as:

$$K(t - t_i) = V_0 \times (\exp(\frac{-(t - t_i)}{\tau_m}) - \exp(\frac{-(t - t_i)}{\tau_s})) \quad (\text{Eq. 4.2})$$

where τ_m and τ_s denote decay time constants of membrane integration and synaptic currents. For simplicity, τ_s is set to be $\tau_m/4$. V_0 normalizes PSP so that the maximum value of the kernel is 1.

The neuron’s total potential is then obtained by superposition.

$$V(t) = \sum_{t_i} K(t - t_i) + V_{rest} \quad (\text{Eq. 4.3})$$

where V_{rest} is the rest potential of this neuron, which is typically set to be 0. A peak detection unit is thereafter applied on the neuron’s total potential to locate temporal peaks. The principle of peak detection is as follows. For a certain timing t_0 , the potential at that timing is considered as a peak if the following criterion is met:

$$V(t_0) \geq V(t); \forall t \in [t_0 - t_{SR}/2, t_0 + t_{SR}/2] \quad (\text{Eq. 4.4})$$

(a)

(b)

Figure 4.11: Symbol detector. (a) Each input event generates a postsynaptic potential. The integration neuron's total potential can then be obtained by superposition. (b) Peak detection on the total potential. The potential at a certain time compares with other potentials in its temporal search range. If it is maximum in search range, it is considered as a peak, otherwise it is not.

where t_{SR} denotes the time span of the search range. This means potential at time t_0 is compared with all the potentials within its search range $[t_0 - t_{SR}/2, t_0 + t_{SR}/2]$, if this potential is the maximum, then it is considered as a peak. Let $t_0 + t_{SR}/2$ be denoted as t_c (current timing), then potential at timing $t_0 = t_c - t_{SR}/2$ is considered as a peak if

$$V(t_c - t_{SR}/2) \geq V(t); \forall t \in [t_c - t_{SR}, t_c] \quad (\text{Eq. 4.5})$$

Fig. 4.11(b) illustrates two examples of peak detection. The upper one shows that $V(t_1)$ is not a peak since it is not the maximum among its search range $[t_1 - t_{SR}, t_1]$; while the lower one $V(t_2)$ is considered as a peak since (Eq. 4.5) is met.

A pulse is triggered when a peak is identified. This pulse is then used to turn *ON* the switches in Fig. 4.3. At that particular moment $C1$ feature maps are fed to the following processing stages. Note that, in order to avoid detection of very small peaks (for instance, when there is only one event occurring during some time interval), a threshold should also be applied. In addition, a refractory time can also be added to limit the frequency of output pulse, i.e., the motion detector will remain halted for a while after a pulse has been generated.

4.5.2 Feature Spike Generation

$C1$ features maps at a certain moment selected by the symbol detector will be fed forward to a set of “TFS neurons”. Each TFS neuron is in charge of the conversion of one response. All TFS neurons work in parallel and should be triggered simultaneously. As stated in its name, each TFS neuron generates only one spike. The higher the response, the shorter the time to first spike. Let m and n denote the resolution of AER vision sensor. After convolution and MAX operation, each $C1$ feature map has the same size as input resolution, thus the number of all responses in $C1$ layer is $4 \times 4 \times m \times n$ (4×4 filters are used). Fully parallel response-to-spike conversion

would require $4 \times 4 \times m \times n$ TFS neurons, which would lead to huge hardware resource occupation. Fortunately, due to the MAX surviving operation, only a small amount (refer to Section 4.7.1 for the detailed analysis) of neurons survive in $C1$ layer (i.e., most $C1$ responses equal to zero). Instead of using all $C1$ responses, only the surviving neurons' responses (non-zeros ones) together with their unique addresses (positions within 16 $C1$ maps) are fedforward. After conversion, the addresses of original responses should be preserved and fed forward together with corresponding spikes. In this way, the features are encoded into AER spikes (also called spatiotemporal spikes), a combination of timestamp and address. The timestamp is inversely proportional to the strength of surviving $C1$ responses, and the address indicates the unique position within $C1$ feature maps.

4.5.3 Summary

The “Motion Symbol Detector” selects some “good” timings. The surviving $C1$ neurons at a selected timing then go through a set of “TFS neurons” to convert $C1$ responses into spikes. The larger the $C1$ response, the smaller the spiking timing. At this step, the features are converted into AER events/spikes again. Each feature spike has a timestamp and address. The timestamp encodes the feature response (inversely proportional), and the address indicates the unique position within $C1$ feature maps.

4.6 Categorization by Spiking Neural Network

At the last step, the AER feature spikes, also known as spatiotemporal spike patterns, are obtained. A spiking neural network can be used to classify different spike patterns. The adopted classifier is a network of tempotron spiking neurons. In principle, all the $C1$ feature spikes are needed for classification and each tempotron neuron needs as many weights as the number of $C1$ responses. Thanks to the beautiful nature of

spatiotemporal AER spikes and the MAX operation, a virtually fully connected system can be achieved by physically activating only a very small subset of the network. The several subsections below will illustrate how the tempotron spiking neural network is used to perform classification on the extracted feature spikes.

4.6.1 About Spiking Neural Network

Various models have been proposed in the literature to describe the dynamics of single spiking neuron, such as leaky integrate-and-fire (LIF) model [157], Hodgkin-Huxley model [158], and Izhikevich model [159]. Among these models, LIF has the simplest structure and thus has been widely used. By combining multiple spiking neurons and storing weight information in synapses, one can construct a spiking neural network to learn and discriminate spatiotemporal spike patterns. Experimental studies in neuroscience have revealed a phenomenon namely spike-timing-dependent plasticity (STDP). The synaptic strength will be regulated by the relative timing of presynaptic and postsynaptic spike. It has been observed that the synaptic strength undergoes a long term potentiation (LTP) when a presynaptic neuron fires shortly before a postsynaptic neuron, and a long term depression (LTD) when the presynaptic neuron fires shortly after [160]. STDP-based rules have been studied in [160–162] for unsupervised learning of spike patterns. In addition to unsupervised STDP rules, supervised learning schemes such as tempotron [42] and ReSuMe [163] have also been widely exploited. Compared to ReSuMe, which specifies a desired firing time, tempotron learning rule only needs to label the status of firing or not, and thus it is more suitable for the real-world stimuli categorization.

Tempotron is a model of supervised temporal learning that allows a spiking neuron to efficiently discriminate spatiotemporal spike patterns. It utilizes spike timing information and integrates postsynaptic potentials from afferent spikes with different

addresses. These properties make tempotron by nature a perfect match for the extracted AER feature spikes.

4.6.2 Tempotron Learning Rule

Tempotron uses the LIF neuron model. Each input spike will initiate a dynamic change of postsynaptic potential (PSP) which has an exponential decaying shape (see Fig. 4.12 (a)). The neuron's membrane potential is the weighted summation of PSP from all input spikes:

$$V(t) = \sum_i \omega_i \sum_{t_i} K(t - t_i) + V_{rest} \quad (\text{Eq. 4.6})$$

where ω_i and t_i are the synaptic efficacy and the firing time of the i^{th} afferent, respectively. V_{rest} is the resting potential of the neuron. K denotes the normalized PSP kernel as defined in (Eq. 4.2).

If the neuron's potential is higher than a specified threshold, the neuron will fire an output spike then reset its potential to the resting level. Fig. 4.12 (b) and (c) illustrate the dynamics and learning rule using two example spike patterns. In Fig. 4.12 (b), the neuron fires as the membrane potential caused by *pattern1* exceeds the threshold. After firing, the neuron shunts all the following input spikes and the potential decreases to the resting level. In other words, the spikes arrive after the firing time have no impact on the postsynaptic potential any more. In Fig. 4.12 (c), the neuron does not fire as the membrane potential fails to cross the threshold.

The tempotron learning rule aims to train the weights of afferent synapses so that the output neuron can fire or not according to its class label. If the neuron is supposed to fire (or not fire, on the other hand) but it actually fails to do so (or does fire, vice versa), then the weights of afferent synapses should be modified in the following way: first, the peak potential during the effective period is found and the corresponding

Figure 4.12: The dynamics and learning rule of tempotron neuron. (a) shows the PSP kernel. (b) and (c) illustrate the operations of tempotron using two spatiotemporal patterns. The vertical thick bars stand for spikes, and the dash curve beside each bar denotes the PSP kernel generated by corresponding spike. For *pattern1* in (b), the total potential crosses the threshold, which means the neuron would fire for this input. If this is an error (the neuron should not fire for this input), then all the spikes before t_{max} are found and the weights of corresponding afferents are decreased. *Pattern2* in (c) does not make the neuron fire, if this is an error, the weights of those afferents which have spikes before t_{max} will be increased. Note that the curve of weight change is just the mirror of PSP kernel.

timestamp is labeled as t_{max} ; second, the weights are updated using this equation:

$$\Delta\omega_i = \lambda \sum_{t_i < t_{max}} K(t_{max} - t_i) \quad (\text{Eq. 4.7})$$

where $\Delta\omega_i$ stands for the weight increment and λ is a constant coefficient. For example, in Fig. 4.12 (c), the neuron fails to fire. If this is an error, one needs to increase the weights of those afferents with spikes arriving before t_{max} .

4.6.3 Virtually Fully Connected Tempotron Network

In principle, the spikes from all $C1$ neurons are needed to perform classification. Each tempotron neuron needs as many weights as the number of $C1$ responses. Fig. 4.13(a) shows the full tempotron network which uses the spikes from all $C1$ neurons (both surviving and dead). Assume the resolution of AER vision sensor is $m \times n$ pixels, the number of responses in $C1$ layer will be $4 \times 4 \times m \times n = 16mn$ (as 4×4 filters are used). Therefore, each tempotron neuron should have $16mn$ weights. In a N -class categorization task, N tempotron neurons are needed (one for each category), the total number of weights are $16mn \times N$. The full tempotron network has $16mn$ inputs and it needs a lot of computation to calculate the results.

Thanks to the beautiful nature of spatiotemporal AER spikes and the MAX operation (where only a few neurons survive after competition), a virtually fully connected system can be achieved by physically activating only a very small subset of the network. Fig. 4.13(b) illustrates the proposed virtually fully connected tempotron network. only the feature spikes from surviving $C1$ neurons are fedforward to the tempotron network (since the spikes from dead $C1$ neurons locate outside the time window and thus have no impact on tempotron potential, see Fig. 4.13(a)). A lookup table (LUT) is used to store all the $16mn \times N$ weights. Each feature spike is associated with an address, which can be used to visit the lookup table and fetch a corresponding weight. The

Figure 4.13: Full Tempotron Network and Virtually Fully Connected Tempotron Network. (a) shows the full tempotron network which uses the spikes from all C1 neurons (both surviving and dead). The 16 C1 maps have totally $16mn$ C1 responses. In this $16mn$ -input network, each tempotron neuron has $16mn$ weights. A N -class categorization task needs totally $16mn \times N$ weights. The spikes from dead C1 neurons (blue bars) locate outside the time window; they actually have no impact on the potential of tempotron neuron. Therefore, a virtually fully connected tempotron network is proposed as shown in (b) to reduce the size of network and to save computation. Only the feature spikes from M surviving C1 neurons are used. The address of each feature spike is used to visit a lookup table, which store all the weights, and fetch corresponding weight. The computation of the virtually fully connected tempotron network is the same as an M -input tempotron network. Since M is far less than $16mn$, a lot of resources can be saved for the tempotron network.

virtually fully connected tempotron network still has $16mn \times N$ weights in total, but it can be considered as an M-input network, the number weights involved in computation is only $M \times N$, which is far less than $16mn \times N$.

During the training process, for a N -class categorization task, N tempotron neurons are needed. They are labeled using one-hot coding scheme. If a pattern belongs to the first class, then the first tempotron neuron's output is labeled to be 1 (which means it should fire), and all other neurons' outputs are labeled to be 0 (not fire). During testing, the decision making for each input pattern is easy. One just needs to check which neuron has fired. In order to further improve the performance, multiple neurons can be used for each category [164]. Since the initial weights are set randomly, these neurons will have different weights after training. A majority voting scheme is used to make the final decision: to check which category has the largest number of firing neurons.

4.7 Experimental Results

4.7.1 Using AER Motion Sensor as input

The performance of the proposed algorithm has been evaluated on real AER motion events. Three human actions were captured, namely bending to pick something up (*BEND*), sitting down and standing up (*SITSTAND*), and walking back and forth (*WALK*). Fig. 4.14 shows a few reconstructed sample images. Each row is corresponding to one action; images are reconstructed from the AER motion events, using the aforementioned fixed time slice approach with a frame interval of 20 ms.

These address events data can be downloaded from the web [165]. They can be visualized by jAERViewer [26]. For example, Fig. 4.15 shows some examples of the AER data viewed in jAERViewer. Note that these raw data from an AER temporal

Figure 4.14: Some reconstructed frames from the posture dataset. There are three kinds of human actions, each row shows an action.

contrast vision sensor have a spatial resolution of 64×64 . The resolution is reduced to 32×32 before feeding the data to the proposed AER recognition system.

Figure 4.15: Some examples of the AER data viewed in jAERViewer.

Note that in the proposed system, we only focus on the detection and recognition of abrupt action transitions. We do not care about the movements that happen in a constant speed since they can be inferred from the last action transition. The system performs recognition only when abrupt changes of body movement happen, such as suddenly bending down, sitting down, and suddenly changing the walking direction. Compared to constant movements, abrupt changes tend to generate more events in the sensor output, causing a burst effect. In our system, we use a Motion Symbol

Detector to detect such a burst of events (i.e. a motion symbol) generated by abrupt changes, and then trigger the classification at those moments.

4.7.1.1 On-the-fly Event-based Centroid Computation

The human's position may vary in the field of view, especially for the *WALK* action. In this case, position invariance is necessary to the algorithm. This can be achieved by aligning the human posture silhouette to the center of the scene using centroid information. Then alignment process is used to simply offset the address of each incoming motion event before feeding it to the *S1* map. An alternative way is to align the address of *C1* feature spikes. The latter method involves less computation since the number of surviving neurons is reduced in *C1* map.

Figure 4.16: On-the-fly centroid calculation. Each incoming event initiates a PSP kernel for corresponding neuron. The vertical bars represent events, the fast-rising and exponential-decay curves depict PSP kernels. Using PSP potential as the weight of each address, the centroid can be easily calculated using the equation shown.

Fig. 4.16 illustrates the on-the-fly centroid calculation. Similar to feature extraction, a map of leaky integrate neurons are built. Each incoming event initiates a PSP kernel

in the neuron specified by the event's address, as defined in equation (Eq. 4.2). Let k_i denote the PSP kernel of the neuron with address x_i , and let n denote the number of neurons. Using PSP potential as the weight of each address, one can easily calculate the centroid address x_c using the following equation:

$$x_c = \frac{\sum_{i=1}^n x_i k_i}{\sum_{i=1}^n k_i} \quad (\text{Eq. 4.8})$$

This process can be visualized by the simulation results in Fig. 4.17. A person is walking to the left and then back to the right side. The middle and lower figures illustrate the integration potential curve of symbol detector neuron and on-the-fly centroid address, respectively. The upper row shows images reconstructed at several selected timing points, the green dot highlights the centroid and the arrow indicates the moving direction. The centroid follows the human's action quite well, in particular at the moment when symbol detector reaches peak. Note that in this experiment, it is assumed there is only one person in the scene. If more people exist, one could resort to event-based clustering algorithms (eg. methods presented in [28, 29]) to obtain the position of each cluster (person).

4.7.1.2 Parameter Selection

Though very few neurons survive after MAX operation in $C1$, the number does vary depending on input scenarios. For future hardware implementation consideration, this number needs to be fixed. The following rule is defined. First, the statistics of surviving neurons in $C1$ layer (e.g. mean μ , standard deviation σ) are obtained, and then the number of feature spikes (as well the number of "TFS neurons") is determined by:

$$M \geq \mu + 3 \times \sigma \quad (\text{Eq. 4.9})$$

The statistics of the three posture groups are shown in Table 4.1:

Figure 4.17: Simulation results of on-the-fly centroid calculation on a stream of address events. A human is first walking to the left and then back to the right side. From bottom to top, the three rows respectively show the centroid curves, the potential curve of the integration neuron in “Motion Symbol Detector”, and some images reconstructed at selected timing points. The green dot depicts the calculated centroid and the arrow means the moving direction. One can see that the centroid curves do match well with the human action.

Table 4.1: number of surviving $C1$ neurons

	Three Groups of postures		
	BEND	SITSTAND	WALK
mean (μ)	40	45	65
std (σ)	10	7	9

One can see that the numbers of surviving $C1$ neurons are very small. According to (Eq. 4.9), $M \geq 65 + 3 \times 9 = 92$. Therefore, the use of 100 TFS neurons is sufficient.

There are several key parameters in the proposed algorithm that need to be tuned according to the specific application. When using the AER temporal contrast vision sensor to observe walking humans, a minimum time of about 10-20 ms is needed to reconstruct a human-like silhouette. The time constant is therefore set as $\tau_m = 20ms$. In addition, transition actions like bending and sitting-down last no more than one second during the data collection, thus the search range parameter in symbol detector is set to be $t_{SR} = 1s$. The leakage rate for on-the-fly convolution is determined by the time constant and finally set to be $\theta = 1/\tau_m = 50s^{-1}$. Moreover, multiple tempotron neurons are used for each category to improve the performance as discussed in Section 4.6, and here this parameter is set to be 10. The selection of these key parameters are summarized in Table 4.2.

Table 4.2: Parameters for posture dataset

time constant τ_m	20 ms
search range t_{SR}	1 s
leakage rate	$50 s^{-1}$
tempotron neurons per category	10

4.7.1.3 Performance

The posture dataset consists of 191 *BEND*, 175 *SITSTAND* and 118 *WALK* actions. Among these actions, 80% were randomly picked for training and the remaining 20% were used for testing. By repeating this evaluation process 10 times, the average performance was obtained. For the training set, a correct rate of 100% was achieved, while the rate for the testing set was 99.48% on average, with a standard deviation of 0.35%. The algorithm was then run on a continuous event stream which is combined

from all testing actions, the result is shown in Fig. 4.18. The blue line represents the ground truth of classification, and the red circles denote the decisions made by the proposed algorithm. One can see that the decisions match very well with the ground truth.

Figure 4.18: The performance of the proposed algorithm on the posture dataset. All testing actions are connected one by one into a continuous event stream and then fed to the system for evaluation. One can see that the decisions made by the proposed algorithm (red circles) match very well with the ground truth (blue line).

The proposed approach was also compared with two popular biologically inspired algorithms: the original HMAX scheme [37] and the model by Serre et. al [38]. MATLAB implementations of the two approaches can be found on the Internet. For the original HMAX, there are 256 C2 features. For the Serre model, there are 1000 C2 features and the patches are randomly extracted. In short, we keep all the default settings and use the codes without changing either settings or mechanisms. Both the HMAX and the Serre model use linear Support Vector Machine (SVM) for classification. To perform multi-class categorization on the three-class posture dataset, the One-Versus-All (OVA) SVM scheme is employed. Since these two models are designed for recognition of 2D frames/images instead of events, our AER posture data can not be directly used. We use the Motion Symbol Detector in our system to select

motion symbols. Each motion symbol is a piece of events that happened before a peak timing found by the Motion Symbol Detector. We reconstruct each motion symbol into an image. The reconstructed images are then fed to the HMAX and the Serre model for performance comparison. We randomly pick out 80% of these images for training and the others for testing. The testing accuracy (averaged from 10 runs) of these two models are listed in Table 4.3, where they are compared to the performance of the proposed algorithm. We can see that the proposed algorithm has a performance that is comparable to the Serre model and much better than the original HMAX on our posture dataset. Note that the Serre model has computationally intensive S2 and C2 layers, it is thus reasonable that it has a slightly better performance than the proposed algorithm.

Table 4.3: Performance Comparison on the AER posture Dataset

Models	mean accuracy	standard deviation
HMAX [37]	78.65%	3.37%
Serre et.al [38]	99.84%	0.25%
this work	99.48%	0.35%

4.7.2 Using Standard Dataset as Input

The proposed algorithm has been further evaluated on a standard hand-written digit dataset MNIST [140, 166], which has ten digits (0-9) and about 70,000 images in total. Fig. 4.19 shows some sample images of this dataset.

The proposed algorithm works on AER events instead of images, therefore these pictures have to be first converted into event streams. A basic thresholding method is used to convert gray level MNIST images into binary ones, with black pixels standing for background and white ones for foreground. Address events are then generated from all foreground pixels, assuming that pixels fire at the same time and events are

Figure 4.19: Some sample images from MNIST hand-written digits dataset.

driven out following a random priority. Each foreground pixel generates one event (note that for the proposed algorithm, one event per pixel is enough, but multiple events per pixel as in rate coding also work fine), and each image generates about 200 events. The average length of converted event stream is about $20 \mu s$, with a mean interspike interval of $100 ns$.

The time constant τ_m and search range t_{SR} are set to be $20 \mu s$, and the leakage rate is $1/\tau_m = 5 \times 10^4 s^{-1}$. For the classifier, ten tempotron neurons are used for each category. These parameters are summarized in Table 4.4.

Table 4.4: Parameters for MNIST dataset

time constant τ_m	$20 \mu s$
search range t_{SR}	$20 \mu s$
leakage rate	$5 \times 10^4 s^{-1}$
tempotron neurons per category	10

The MNIST dataset has 60000 images in the training set and 10000 images in the testing set. Our algorithm can achieve a success rate of about 99.68% for the training set and 90.65% for the testing set, respectively.

Note that the proposed event-driven categorization system is designed mainly for processing the motion events from the AER temporal contrast vision sensor. The purpose of testing the proposed system on the MNIST dataset is not to compete with state-of-the-art algorithms but to demonstrate that the proposed system can work on not only raw AER data but also images (with a preprocessing step to convert images into AER events). Since the algorithm is not designed for recognition of images, it inevitably has a relatively lower performance than other highly optimized frame-based algorithms. As a matter of fact, the state-of-the-art algorithm can achieve an error rate of as low as 0.23%, as listed on the webpage of the MNIST dataset [166].

4.8 Hardware Implementation of The Frame-free AER Recognition System

This section will illustrate the hardware design of the proposed AER recognition system.

4.8.1 AER Recognition System Architecture

Fig. 4.20 shows the hardware architecture of the frame-free AER recognition system. Each event from the AER vision sensor is time-stamped by “Sensor Ctrl” module, which handles the hand-shaking with AER sensor and contains a counter for time-stamping of events. The time-stamped event then goes through feature extraction and classification (as shown as dashed boxes in Fig. 4.20). As mentioned earlier in the algorithm description part, feature extraction can be divided into several modules, namely “Symbol Detector”, “Convolution”, “Max Competition”, and “Spike Generator”; and the classification includes a weights lookup table and a tempotron neural network.

Figure 4.20: Architecture of the frame-free AER Recognition System.

4.8.2 Memory and Computation Cost of the AER Recognition System

In this section, the memory and computation cost of the AER recognition system are illustrated. The resolution of the input AER vision sensor is $n \times n$ ($n=32$).

4.8.2.1 Motion Symbol Detector

As described in the algorithm description section, The “Motion Symbol Detector” consists of one leaky integrate neuron and a peak detection unit.

Each input event initiates a PSP to the leaky integrate neuron. The PSP has an exponential decay shape. From the kernel shape (see Fig. 4.12 (a)), one can see that it nearly reaches zero after time $5\tau_m$. This can be interpreted as that one input event will have little contribution to the total potential after $5\tau_m$ from the time it is received. Therefore, only the events happened within the time range $[t - 5\tau_m, t]$ need to be cached, where t denotes the current time. Let R denote the average input event rate, then the number of cached events are $5\tau_m R$. For the posture dataset, $\tau_m = 20ms$,

and the event rate of these postures R is smaller than $30K$ events per second (eps). Thus, for the time range of $5\tau_m = 100ms$, there will be about $30Keps \times 0.1s = 3000$ events. This means at most 3000 timestamps need to be cached (the addresses of these events are not required for symbol detector). For the AER vision sensor, the timestamp of each event is generated by a 32-bit counter (with $1 \mu s$ time resolution) inside the sensor. Therefore, the memory requirement for event caching is $3000 \times 32 = 96Kbit$.

The PSP kernel is simply implemented as a lookup table, using the time difference between current time and event timestamp as entry index. In order to get the total potential of the leaky integrate neuron, all the PSP kernels generated by the events that happened in the time range of $[t - 5\tau_m, t]$ need to be accumulated (up to 3000 events as derived above). The leaky integrate neuron's total potential doesn't need to be updated in a very high frequency, it is updated every 1 ms ($dt = 1ms$). The number of entries in the PSP kernel is about $5\tau_m/dt = 100ms/1ms = 100$, the value of each entry is represented as 16-bit floating point number (1 sign bit, 5 bits for exponent, and 10 bits for mantissa), thus the memory requirement for PSP kernel is $100 \times 16bit = 1.6Kbit$.

The Symbol Detector module has a peak detection unit to locate temporal peaks of the leaky integrate neuron's total potential. As illustrated in Fig. 4.11(b) and Equation Eq. 4.5, the potential at time $t - t_{SR}/2$ is compared with other potentials within its temporal search range $[t - t_{SR}, t]$. If this potential is the maximum, then it is considered as a peak. This peak detection unit requires to store all the potentials within $[t - t_{SR}, t]$. For human posture recognition, this search range t_{SR} is usually no more than $1s$, thus $t_{SR}/dt = 1s/1ms = 1000$ potential values need to be cached in a RAM. Each potential is represented as a 16-bit floating point number, therefore the memory requirements for the peak detection unit are $16bit \times 1000 = 16Kbit$.

For each time step, the computation of “Motion Symbol Detector” involves the following: (1) calculating the time difference between current time and each event timestamp (32-bit integer Subtraction, denoted as $Int32Sub$), (2) accumulating each PSP value to the total potential (16-bit floating point Addition, $FP16Add$), and (3) checking whether the potential is a temporal MAX or not (16-bit floating point comparison, $FP16Comp$). The total number of computations for “Motion symbol detector” is $3000events \times (Int32Sub + FP16Add) + 1000potentials \times (FP16Comp)$.

4.8.2.2 Convolution with Leakage

As mentioned earlier in the algorithm description, on-the-fly convolution (with leakage) is applied to the address events from the AER vision sensor. The on-the-fly convolution is a simple addition of kernel onto the region specified by the address of incoming event. It also has a linear leakage mechanism to forget the impact of very old events.

In the proposed system, Gabor filters F_i of 4 sizes $|F_i| \in 3, 5, 7, 9$ and 4 orientations are used, leading to 16 convolution modules in parallel. The generated 16 S1 feature maps have the same resolution with the input ($n \times n$, i.e. 32×32). If 48 bits are used to represent the information of each S1 (16 bits of S1 response, 32 bits of last update time for leakage calculation), then for each S1 map $32 \times 32 \times 48bit = 48Kbit$ storage is needed.

For each S1 map, the on-the-fly convolution only affects a neighborhood region, the size of which is specified by the corresponding filter and the center by the input address event. The operations involved are leakage calculation, leakage subtraction, and adding the kernel to the old S1 response. The new S1 ($S1'$ value as well as the event time) will be written back to S1 map.

$$S1' = S1 + Kernel - R_{leak} \times (t_e - t_l) \quad (\text{Eq. 4.10})$$

where R_{leak} means the leakage rate, t_e and t_l denote the input event time and the last update time of this S1 neuron, respectively. For on-the-fly convolution, the number of pixels involved in each S1 map is $|F_i|^2$, where $i = 1, 2, \dots, 16$. Let $Int32Sub$ denote 32-bit integer subtraction, let $FP16AddSub$ denote 16-bit floating point addition/subtraction and let $FP16MULT$ denote 16-bit floating point multiplication, the total number of operations per input event is:

$$\begin{aligned} & \sum_{i=1}^{16} |F_i|^2 \times (Int32Sub + FP16MULT + FP16AddSub \times 2) \\ & = 656 \times (Int32Sub + FP16MULT + FP16AddSub \times 2) \end{aligned} \quad (Eq. 4.11)$$

4.8.2.3 MAX Competition and Feature Spike Generation

Once a temporal peak is detected by the “Motion Symbol Detector”, MAX competition will be conducted over local neighborhood in each S1 map to find the features. In order to do the MAX competition over local neighborhood, $|F_i| \times |F_i|$ S1 responses ($|F_i|$ denotes the kernel size, i.e., 3,5,7,9) need to be read out. The maximum of these responses needs to be computed and finally the center is judged whether it is a MAX or not. In order to save hardware resources and expedite comparison, similar hardware reuse methodology as illustrated in Fig. 3.23(a) is again adopted. Taking a scale-5 neuron as example, the “current” neuron have 4 overlapped columns with “previous” neuron. The max is calculated column wise and the computed column MAXs are stored for later reuse. For “current” neuron, only 5 new data (of the new column) need to be read out and used to compute a column max, then the MAX of 5×5 local neighborhood is computed from 5 column max numbers, and finally the “current” neuron compares itself with the local MAX to decide to survive or die. Note that for the left boundary neurons, all 5 columns of S1 responses still need to be read in. However, for the following ones (non left boundary), only one new column need to

be read in. The number of operations ($FP16Comp$, 16-bit floating point comparison) involved in MAX competition are

$$\begin{aligned}
 & \sum_{i=1}^{16} [n|F_i|^2 + n(n-1)|F_i|] \\
 &= 4 \times [32 \times (3^2 + 5^2 + 7^2 + 9^2) + 32 \times 31 \times (3 + 5 + 7 + 9)] \\
 &= 116224
 \end{aligned} \tag{Eq. 4.12}$$

The neurons that win the competition will be converted to spikes in a Time-to-First-Spike manner. The larger the response, the smaller the spike timing. $y = a - bx$, where y and x denote the spike timing and the original response respectively, a and b are two constant parameters. The generated spikes are normalized and rounded into the time range of $[0, 256]$. Each spike also inherits the address of the corresponding winner neuron. As mentioned in the algorithm description part, the surviving neurons are very sparse and 100 TFS neurons are sufficient. Each spike needs 32 bits (16-bit address and 16-bit time), therefore the generated spikes require $32bit \times 100 = 3.2Kbit$ storage. The number of operations for feature spike generation is $100 \times (FP16MULT + FP16Sub)$.

4.8.2.4 Tempotron Neural Network

The generated spikes are first sent to “Weights LUT” to fetch their corresponding weights (using their addresses as entry index), and then the spike timings and the obtained weights are fed to a network of tempotron neurons to get the final classification results.

The number of weights in “Weights LUT” is N times (N denotes the number of categories, e.g. 3 for the posture dataset) of the number of all S1 neurons, thus the number of weights equals to $N \times (32 \times 32 \times 16) = 48K$. Each weight is represented as a 16-bit floating point number, so $48K \times 16bit = 768Kbit$ memory is needed to store all the weights.

Similarly as in Symbol Detector, the PSP kernel for tempotron is also implemented as a lookup table, using the time difference between current time and spike time as entry index. Each spike's PSP is multiplied with its corresponding weight and then accumulated onto the total potential of a tempotron neuron. Each tempotron neuron's total potential needs to be traced and whether the neuron fires or not needs to be decided. As mentioned earlier, the generated spikes' timings are normalized into the range of $[0, 256]$. The time span for potential tracing is $T = 256$. The step length for potential tracing and the time constant in PSP kernel are respectively set to be $dt = 1$ and $\tau_m = 20$. The 3-group posture dataset requires three tempotron neurons (one hot coding), and thus three sets of Multiply Accumulator (MAC) are needed. The total number of MAC operations for the tempotron network is $100afferent \times 256steps \times 3sets = 76800$.

4.8.3 FPGA-based Software-Hardware Co-processing

The proposed AER recognition system is implemented using a field programmable gate array (FPGA)-based software-hardware co-processing architecture.

4.8.3.1 Why Software-Hardware Co-processing

The proposed AER recognition system involves a lot of parallel computation, making its execution as a software in a personal computer (PC) very slow. To achieve real-time posture recognition, the system has to be accelerated by implementing the algorithm into hardware, such as FPGA or application specific integrated circuits (ASIC). Compared to ASIC, FPGA is more flexible and convenient for rapid prototyping development. FPGA can provide many parallel resources to expedite the computational speed, however it is more difficult for complicated control as compared to software. To sum up, software (i.e. processing by general purpose CPU) is easy for complicated control but slow for parallel computation, while hardware (i.e. FPGA or ASIC)

is difficult to implement complicated control procedures but it can provide fast parallel computation speed. Therefore, by combining the advantages of both software and hardware, the proposed recognition system is implemented using a FPGA-based software-hardware co-processing architecture, using hardware to handle computation intensive tasks and using software to control the procedure.

4.8.3.2 Software-Hardware Co-processing Architecture

Inside Xilinx FPGA, an embedded system can be built using the MicroBlaze soft processor provided by Xilinx. MicroBlaze is a 32-bit reduced instruction set computing (RISC) soft processor IP core. As a soft processor, it can be configured by user to choose or trim optional features and implemented entirely in general purpose memory and logic resource of FPGAs. Fig. 4.21 shows the block diagram of the MicroBlaze core. MicroBlaze has separate instruction and data buses (Harvard architecture). It has bus interfaces to two buses, namely Local Memory Bus (LMB) and Processor Local Bus (PLB). The dedicated memory accessing bus LMB reduces loading on PLB. Various types of peripherals can be attached to PLB to build an embedded system. Fig. 4.22 shows an example of MicroBlaze-based embedded system. The MicroBlaze core accesses the Block RAM-based instruction and data memory through LMB bus, and it can have various peripherals connected through the PLB bus, such as universal asynchronous receiver/transmitter (UART), inter-integrated circuit (I²C), Ethernet, general purpose input/output (GPIO), etc.

Fig. 4.23 shows the FPGA-based Software-Hardware Co-processing architecture for the proposed AER recognition system. A MicroBlaze soft processor is instantiated inside the FPGA, and several user logic modules are attached to the peripheral bus of MicroBlaze processor. The FPGA is connected with a full custom designed AER temporal contrast vision sensor to perform sensor control and AER data acquisition.

Figure 4.21: Block Diagram of Xilinx MicroBlaze soft processor.

Figure 4.22: MicroBlaze embedded system.

Memory devices outside (or inside) the FPGA are used to store the events and intermediate data during processing. The FPGA is connected to PC through universal serial bus (USB) microcontroller and high speed USB 2.0 cable.

Figure 4.23: FPGA-based Software-Hardware Co-processing architecture.

4.8.3.3 Opal Kelly USB-FPGA board

The FPGA board that is adopted is Opal Kelly USB-FPGA module XEM3050. The FPGA board 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. It also has a Cypress USB controller to handle the USB protocol. The USB data transfer between PC and FPGA has been made transparent, which is very convenient for users.

4.8.3.4 Implementation Results

The AER recognition system was implemented in Opal Kelly XEM3050 (XC3S4000). Xilinx Spartan 3 XC3S4000 model has totally 96 dual-port Block RAMs (BRAM), each BRAM is 18Kbit. The 16 convolution maps take *768Kbit* storage (48 BRAMs in FPGA). In addition, the 16 convolution kernels can be stored in 8 BRAMs. For “Motion Symbol Detector”, the events cache *96Kbit*, the potential cache *16Kbit*, and the PSP kernel

1.6Kbit occupy 6 BRAMs, 1 BRAM, and 1 BRAM, respectively. For the MAX competition and feature spikes generation, 1 BRAM is needed to store the generated feature spikes (3.2Kbit). Moreover, the tempotron network uses on-board SSRAM to store 768Kbit weights and 1 BRAM for PSP kernel (1.6Kbit).

These implementation results are summarized in Table 4.5 and 4.6. The final system can run up to a maximum frequency of 126 MHz. It can achieve a event processing speed of up to 1Meps, since the convolution for each input event takes about $1\mu s$. The delay from symbol detection to final output is about 0.17ms.

Table 4.5: Resource Summary of AER Recognition System

	memory occupied	FFs	LUTs	Remarks
Convolution	56 BRAMs	8,754	9,426	48 BRAMs for S1 maps, 8 BRAMs for convolution kernels
Symbol Detector	8 BRAMs	984	596	Event cache 6 BRAMs, potential cache 1 BRAM, PSP kernel 1 BRAM
MAX competition	1 BRAM	2,080	5,640	
Tempotron network	SSRAM + 1 BRAM	1,514	1,746	SSRAM for weights LUT, 1 BRAM for PSP kernel
MicroBlaze	16 BRAMs	1,557	2,381	Soft processor
Total	SSRAM + 82 BRAMs	16,377	21,558	Some logics not listed, like sensor ctrl, USB comm.

Table 4.6: Performance of AER Recognition System using XC3S4000

Max Frequency	126 MHz
MicroBlaze clock	67 MHz
Event Processing speed	1 Meps
Delay from symbol to final output	0.17ms
Power Consumption	1.85 W

4.9 Summary

This chapter has presented a feedforward categorization system to process data from AER motion sensors. In order to fully utilize the power of asynchronous, sparse, event-driven representation of the sensor data, the concept of “event”-based processing was adopted at every stage. Feature is extracted by hierarchical maps of leaky integrate type neurons, inspired by a model of object categorization in the primate visual cortex. Due to the use of MAX operation, features are encoded into a limited number of spikes. A virtually connected neuron network efficiently discriminates the spatiotemporal spike patterns. Two types of on-the-fly co-processing are also explored, namely “motion symbol detector” and centroid calculation. The overall system has been evaluated by extensive simulation. Promising results have been obtained.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In the past two decades, CMOS image sensors have played a major role in the market of solid-state image sensing devices. An increasingly large number of consumer electronic devices have built-in CMOS image sensors. Examples include cell phones, cameras, fax machines, scanners, just to name a few. The selling point behind the success of CMOS image sensors lies in the usage of the well-established standard CMOS process in semiconductor industry, which results in reduced development and fabrication costs for CMOS image sensors. These cheap, commercially available image sensors are excellent for the purpose of collecting visual memories, and their proliferation has led to advanced research in intelligent image processing such as sensory adaptation, image compression, motion detection, image identification and object recognition for a variety of applications. These cameras, coupled with modern personal computers (PCs), have generated impressive results but with low computational efficiency. On the other hand, small and lightweight wireless platforms, such as ultra-mobile PCs or smart cellular phones, are currently unable to perform image interpretation and recognition of objects for real-time applications, due to power and computational constraints. The lack of efficiency in image interpretation is an intrinsic limitation imposed by the architecture of most commercial image sensors, i.e., the

inability to extract relevant and timely information from pixels. Moreover, there is a growing division between the latest computer based vision algorithms and what is actually implementable in low-complexity hardware. On the other hand, primates vision is extremely accurate and efficient in the categorization of objects. The current theory of the cortical mechanism responsible for object categorization has been pointing to a hierarchical and mainly feedforward organization, where short-range feedback is believed to play a secondary role. This organization can provide hierarchical features of increasing complexity and invariance to size and position, making object categorization a multi-layered and tractable problem. However, there are huge differences between the computation method of existing computer and that of the brain. Conventional computing paradigm based on binary logic and Von Neumann architecture becomes increasingly inefficient as the complexity of computation increases. For some tasks such as real-time image data analysis in ever-changing and unstructured environment, the computation time scales exponentially with the input size, making it difficult to perform such tasks with conventional computers and computation methods.

Hence, new computational paradigms and architectures are being explored to extend the capabilities of information technology beyond digital logic. Neuromorphic research is a growing branch of engineering that takes inspiration from biological neural systems in order to optimize engineered systems. One of the key areas of interest in neuromorphic research is in neuromorphic systems design and sensor fusion, i.e. the combination of neuromorphic vision sensory hardware and hardware-oriented neural networks. This thesis work sets itself in the middle of this effort as it investigates the design of biologically inspired human posture recognition system, using smart vision sensors. Smart image sensors utilize novel focal-plane signal processing to improve the computation efficiency, when compared to conventional discrete sensor-processor systems. Two system architectures were proposed, one based on a frame-based smart image sensor and the other one using frame-free AER sensor.

- (i) The frame-based recognition system includes a custom designed temporal difference image sensor, a bio-inspired feedforward line feature extraction unit and a size-and-position-invariant classification framework.

The adopted frame-based temporal difference image sensor can automatically remove still background in the scene through focal plane processing; it captures only the motion contour of the moving object and generates a stream of binary motion events as output, in which “1” stands for a pixel on a motion object and “0” represents a still background pixel. This sensor avoids the transmission of massive raw data, largely saving the channel bandwidth. In addition, since the outputs of this sensor are only some binary motion events, the privacy of the person can thus be protected.

The motion events generated by the temporal difference image sensor are sent to the bio-inspired feature extraction unit, where each image is represented as a set of vectorial line segments. The proposed feature extraction approach is inspired by several feedforward models of primate visual cortex, such as HMAX model and Serre’s model. A bank of Gabor filters (4 scales and 4 orientations) are used to model the $S1$ units which correspond to simplex cells in primary visual cortex. Two-step MAX operation is then performed on $S1$ layer maps to generate $C1$ maps, where the position, size and orientation of the line features can be easily extracted. One major difference between the proposed approach and previous feedforward models is the MAX operation. Previous models widen the maximum response in $S1$ maps to achieve blurred general features, while the proposed approach suppresses non-maximum responses to obtain thinned line features.

After the feature extraction, each image is represented as a set of vectorial line segments. Modified line-segment Hausdorff distance (LHD) is then employed to

measure the similarity between two images. The smaller the LHD, the larger the similarity. The test image is matched with each one in the model library, and the classification decision is made based on its nearest neighbor (best match). In addition, to achieve invariance to size and position, alignment and re-scaling are performed before matching. To efficiently find the size and position of a human posture, a clustering-based algorithm was proposed. This algorithm processes the motion events on the fly and provides the size and position information immediately after the last event of one frame.

The proposed recognition algorithm has been evaluated on several posture datasets built from a temporal difference image sensor, a web camera and an asynchronous address-event vision sensor, respectively. The proposed algorithm was compared with HMAX model and Serre's model. Experimental results show that the proposed algorithm has better recognition rates and faster simulation speed. In addition, the proposed algorithm does not require a large training set since example-based matching scheme is used. 10-30 training images per group can already produce good results and at the same time achieve more than 50% saving in CPU time compared to Serre's model.

- (ii) An improved event-based AER recognition system is further proposed for frame-free AER vision sensor. The AER motion sensor does not produce intensity images but a train of asynchronous spikes. In the envisaged application of assisted living/elderly care, using AER sensor instead of conventional image sensors can protect the privacy of the person. In order to fully utilize the power of asynchronous, sparse, event-driven representation of the sensor data, the concept of event-based processing is adopted at every stage. The combination of frame-free AER spikes and event-based processing is shown to make the recognition system more efficient.

The sensor is equipped with direct difference hardware in the pixel and outputs an event if a threshold is reached. The output data is a stream of address events. Each event has an address and a timestamp; the address indicates which pixel the event is from, and the timestamp represents the happening time of event.

Each stage of the system uses the concept of event-based processing. Each address event from AER vision sensor is sent in parallel to a battery of orientation filters based on the Gabor functions, and convolution operation is performed on the fly. A leaky model is used to describe the dynamics of each neuron in the map. Each neuron competes with other neurons located within its receptive field, and it can only survive in higher layer if it wins a MAX-like operation. In addition, an asynchronous time-domain motion symbol detector was proposed to activate another stage of spike generation. The dynamics of the aforementioned surviving neurons, which represent the strength of “features”, go through a small set of neurons that work in Time-to-First Spike (TFS) mode. The generated spikes, again in the form of spatiotemporal pattern of pulses, are fed to a tempotron spiking neural network.

The AER recognition system was evaluated on a posture dataset captured from an AER temporal contrast vision sensor, which consists of 191 *BEND*, 175 *SITSTAND* and 118 *WALK* actions. Among these actions, 80% were randomly selected for training and the remaining 20% were used for testing. This random dataset split and evaluation process was repeated 10 times. For the training set, a correct rate of 100% is obtained; while for the testing set, the rate was 99.48% on average, with a standard deviation of 0.35%.

In order to build a real-time human posture recognition system, the proposed recognition algorithm has to be implemented into hardware for acceleration. An

FPGA-based software-hardware co-processing architecture was adopted. A MicroBlaze soft processor is instantiated in Xilinx FPGA and used for algorithm control, other dedicated user hardware modules are designed in FPGA and used to handle computation intensive tasks. The Opal Kelly XEM3050 (XC3S4000) FPGA board is used to build the whole AER recognition system. The final AER recognition system can achieve a processing speed of up to 1 Meps. The delay from symbol detector to final output is just about 0.17ms.

5.2 Future Work

Human posture recognition is a challenging task, since human activity is fairly complex and highly diverse. From different view angles, the same action can be interpreted differently by human being with possible misunderstanding as well. The proposed system performs well with both objects and a number of human postures. However, the system is not completely free of problems: one typical problem is when multiple objects are moving back and forth in the scene, or the background is moving. In this case categorization fails because the algorithm loses the person-of-interest. When monitoring a single person in a room, like in assisted living applications, this is not an issue. However, for real world application, an object tracking stage should be added to the system. Further challenge emerges when multiple objects run into one and then separate. More advanced object tracking algorithms or techniques need to be employed. One possible way is to have the target person carry a detectable tag or marker. Another concern is the system robustness against viewpoint variance and field of view full coverage. The current AER recognition system can achieve translation invariance by simple alignment of the feature maps. The MAX operation used in feature extraction also contributes slightly to size invariance. However, it does not cover the view

invariance. When watching from different angles of view, the shape captured by silicon retina can be significantly different. To achieve view invariance, one may resort to stereo vision by combining two or more AER retinas. A further problem is adaptation to lighting conditions. Even indoor, light intensities can vary by ten times or more, making it difficult for the image sensor to always extract complete contours of objects and human postures. Resolving such issues will undoubtedly lead to a very promising new generation of categorization systems.

Appendices

Appendix A

Line-Segment Hausdorff Distance

The goal is to compare two images I_{test} (the test image) and I_{lib} (an image from the library, i.e., a training image). How much I_{test} is similar to I_{lib} needs to be determined. The extracted line segments of both images are passed to the classifier. The two sets of line segments are then compared using line-segment Hausdorff distance (LHD). Three parameters need to be taken into account:

- Difference in locations;
- Difference in sizes;
- Difference in orientations.

Moreover, from observing the constructed line segments, one can note that along with the main features of the image, line segments that could be referred to as “noise” are also obtained. This happens because the source image contained random noise. Note that when comparing two images, the difference between the main features is more important than the difference between the “noisy” line segments.

The distance between two line segments e_1 and e_2 is defined (similarly, but not equivalently to [148]) as:

$$d(e_1, e_2) = \sqrt{d_{||}^2 + d_{\perp}^2 + d_{\theta}^2} \quad (\text{Eq. A.1})$$

where:

- $d_{\perp}$ is the Euclidean distance between the lines containing e_1 and e_2
- $d_{||} = \max(d_{||1}, d_{||2})$
- $d_{\theta} = \text{penalty} \times |\tan(\theta)|$

This is illustrated in Fig. A.1 using two horizontal line segments. For parallel non-horizontal line segments, the definition extends naturally by rotating the coordinates. For non-parallel line segments, distance components $d_{||}$ and $d_{\perp}$ are computed by first rotating the smaller line segment around its center to make it colinear with the larger edge.

Figure A.1: Definitions of the distances $d_{\perp}$, $d_{||1}$ and $d_{||2}$ for two parallel line segments.

Denote the size of line segment e by $|e|$. First consider $d_{||}$. In the notation of Fig. A.1:

$$d_{||1} = |x_1 - x_2| = \left| (x_{1cnt} - x_{2cnt}) - \frac{(|e_1| - |e_2|)}{2} \right| \quad (\text{Eq. A.2})$$

$$d_{||2} = |x_3 - x_4| = \left| (x_{1cnt} - x_{2cnt}) + \frac{(|e_1| - |e_2|)}{2} \right| \quad (\text{Eq. A.3})$$

The maximum occurs when both summands inside the modulus are of the same sign, hence:

$$d_{||} = |x_{1cnt} - x_{2cnt}| + \frac{||e_1| - |e_2||}{2} \quad (\text{Eq. A.4})$$

The component d_{θ} reflects the difference in orientations of two line segments. In the system, two line segments can:

- Be parallel;
- Be perpendicular;
- Intersect at the angle of $\pm 45^\circ$.

In the first case, the line segments e_1, e_2 have the same orientation. Only the difference in locations and sizes of the two line segments are taken into account, since $\tan(\theta) = 0$ and $d_\theta = 0$. In the second case, e_1 and e_2 are perpendicular. These two line segments are viewed as completely different, $d_\theta = \infty$, and moreover $d(e_1, e_2) = \infty$. In the third case, e_1 and e_2 intersect at $\pm 45^\circ$, so $|\tan(\theta)| = 1$ and $d_\theta = \text{penalty}$. Therefore, the significance of the orientation difference is reflected by the parameter *penalty*, which has no obvious a priori value and is found experimentally.

The directed modified Line Segment Hausdorff Distance from I_{lib} to I_{test} is defined by:

$$\vec{D}(I_{lib}, I_{test}) = \frac{\sum_{e_l \in I_{lib}} \left(\min_{e_t \in I_{test}} d(e_l, e_t) \right) \times |e_l|}{\sum_{e_l \in I_{lib}} |e_l|} \quad (\text{Eq. A.5})$$

The weighted mean is taken to emphasize that the large line segments are more important than the small line segments. Finally the distance between I_{test} and I_{lib} is defined to be:

$$D = \max(\vec{D}(I_{lib}, I_{test}), \vec{D}(I_{test}, I_{lib})) \quad (\text{Eq. A.6})$$

See Fig. A.2 for an example of the distance computation between two sets of line segments.

To find the impact of the *penalty* value on the distance calculations, numerical experiments were conducted. First of all, the greater the value of the *penalty* is, the

Figure A.2: Example of modified line-segment Hausdorff distance calculation. (a) The two sets of line segments to be compared are 'uv' (solid) and 'abc' (dotted). The 'uv' object contains line segments of the following sizes (in pixels): $u = 15$, $v = 9$. The 'abc' object's line segments and its sizes are: $a = 9$, $b = 5$, $c = 5$. Here $penalty = 10$. (b) The highlighted line segment is the line segment u in the 'uv' object. The numbers indicate the distance from u to each line segment of the 'abc' object. The solid line (the line segment a) in the 'abc' object is the edge having the minimal distance from u . (c) The highlighted line segment is the line segment v in the 'uv' object. The line segment with the minimum distance in this case is the line segment b (solid line) in the 'abc' object. (d) The highlighted line segment is the line segment a in the 'abc' object. Here the line segment in the 'uv' object with the minimal distance from a is shown as dotted line. (e) The highlighted line segment is the line segment b in the 'abc' object. The line segment with minimal distance from b is v (dotted line). (f) The highlighted line segment is the line segment c in the 'abc' object. The minimal line segment is u (dotted). The directed distance from the 'uv' object to 'abc' object is $\vec{D}('uv', 'abc') = (4.5 \times 15 + 5.4 \times 9) / (15 + 9) \approx 4.84$, and the distance from 'abc' to 'uv' is $\vec{D}('abc', 'uv') = (4.5 \times 9 + 5.4 \times 5 + 13.7 \times 5) / (9 + 5 + 5) \approx 7.16$. Overall the distance between 'uv' and 'abc' is $D = 7.16$.

more line segments of the same type will be chosen as minimum in equation Eq. A.5, this can be easily seen from the equation Eq. A.5 itself. For example on Fig. A.3, the line segment with minimum distance (among b and c) from a is looked for. If $penalty \rightarrow \infty$, b would be chosen (even though b is much further from a). However, if $penalty \rightarrow 0$, c would be chosen (picking the closest line segment disregarding the difference in direction). The ratio:

$$\frac{\# \text{ line segments of the same type picked as minimum}}{\# \text{ all line segments picked as minimum}} \rightarrow 1 \quad (\text{Eq. A.7})$$

as $penalty \rightarrow \infty$.

A first series of MATLAB simulations were performed to find the value of $penalty$ for which the ratio:

$$\frac{\# \text{ line segments of the same type picked as minimum}}{\# \text{ all line segments picked as minimum}} = 1 \quad (\text{Eq. A.8})$$

which occurs for $penalty = 80$ (see Fig. A.4(a)).

Figure A.3: Example shows the meaning of different values of $penalty$ from the perspective of choosing the line segment with minimal distance. In this figure, one aims to look for the line segment (among b and c , red dashed) with the minimal distance to a (black).

After this, another series of simulations were run to find the optimal value of $penalty$, which is defined by the maximal number of successful matches of the algorithm (see Fig. A.4(b)). The optimal value is found to be $penalty = 10$.

Computational efficiency is achieved through the following observations. First, in the search of the closest line segment the perpendicular line segments are skipped

(the line segments are naturally sorted by orientation during the construction, which makes this task easy). Thus, the orthogonal line segments, the distance between which is ∞ anyway due to d_θ , are not compared at all. Second, for the non-perpendicular line segments equation Eq. A.1 simplifies to:

$$d(e_1, e_2)^2 = \left(\Delta c^2 + \frac{\Delta s^2}{4} \right) + \Delta p \Delta s + \text{penalty}^2 \times \begin{cases} 0 & e_1, e_2 \text{ are parallel,} \\ 1 & \text{otherwise,} \end{cases} \quad (\text{Eq. A.9})$$

(a)

(b)

Figure A.4: Results of MATLAB simulations conducted to find the optimal value for *penalty*. The algorithm was run on 370 images with different values of *penalty*. (a) Each time equation Eq. A.5 is computed, the minimum can occur for either *different type line segments* or *same type line segments*. The figure shows the proportion of same type minima in equation Eq. A.5 as a function of *penalty*. (b) The number of correct matches of the images that the algorithm yielded as a function of *penalty*.

where Δc is the Euclidean distance between the centers, Δs is the absolute size difference and Δp is the projected distance between the centers, depending on the

orientation of the larger line segment only (see Fig. A.5). The terms in brackets is computed in any case, and only if it does not exceed the current minimum the next terms are computed. The square root is taken only on the stage of computing equation Eq. A.5, when the minimum for each line segment has already been found.

Figure A.5: Definitions of Δp and Δc for two line segments for the simplified distance in equation Eq. A.9. The projection to compute Δp is always done onto the larger line segment (below), regardless of the orientation of the smaller edge.

Author's Publications

Journal Papers

- (i) **Bo Zhao**, Shoushun Chen, Bernabe Linares-Barranco and Huajin Tang, "Feed-forward categorization on AER motion events using cortex-like features and spiking neural network," Minor Revision at *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*
- (ii) J. A. Perez-Carrasco, **Bo Zhao**, C. Serrano, B. Acha, T. Serrano-Gotarredona, Shoushun Chen and B. Linares-Barranco, "Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate-Coding and Coincidence Processing. Application to Feedforward ConvNets," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 35, no. 11, pp. 2706-2719, Nov. 2013.
- (iii) Hualiang Zhuang, **Bo Zhao**, Shoushun Chen and Kay Soon Low, "A Bio-inspired Pulse Neural Network and its Application to Human Posture Categorization Using a Range Sensor," submitted. Under review.
- (iv) **Bo Zhao**, Xiangyu Zhang, Shoushun Chen, Kay Soon Low and Hualiang Zhuang, "A 64×64 COMS Image Sensor with On-Chip Moving Objects Detection and Localization," *IEEE Transactions on Circuits and Systems on Video Technology (TCSVT)*, vol. 22, no. 4, pp. 581-588, April 2012.
- (v) Shoushun Chen, Polina Akselrod, **Bo Zhao**, Jose Antonio Perez Carrasco, Bernabe Linares-Barranco and Eugenio Culurciello, "Efficient feedforward categorization of objects and human postures with address-event image sensors," *IEEE*

Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 34, no. 2, pp. 302-314, Feb. 2012.

Conference Papers

- (i) **Bo Zhao**, Qiang Yu, Hang Yu, Shoushun Chen and Huajin Tang, "A Bio-inspired Feedforward System for Categorization of AER Motion Events," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Rotterdam, Netherlands, Oct. 2013.
- (ii) **Bo Zhao** and Shounshun Chen, "Live Demonstration: A Real-Time Moving Object Localization and Extraction System," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Seoul, South Korea, May 2012.
- (iii) Hualiang Zhuang, **Bo Zhao**, Zohair Ahmad, Shoushun Chen and Kay Soon Low, "3D depth camera based human posture detection and recognition Using PCNN circuits and learning- based hierarchical classifier," in *2012 International Joint Conference on Neural Networks (IJCNN)*, Brisbane, Australia, June 2012.
- (iv) Xinyuan Qian, Hang Yu, **Bo Zhao**, Shoushun Chen and Kay Soon Low, "Design of a Radiation Tolerant CMOS Image Sensor," in *2011 IEEE International Symposium on Integrated Circuits (ISIC)*, Singapore, Dec. 2011.
- (v) **Bo Zhao**, Xiangyu Zhang and Shoushun Chen, "A CMOS Image Sensor with On-chip Motion Detection and Object Localization," in *IEEE Custom Integrated Circuits Conference (CICC)*, San Jose, USA, Sep. 2011.
- (vi) **Bo Zhao** and Shounshun Chen, "Realtime feature extraction using MAX-like convolutional network for human posture recognition," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Rio de Janeiro, Brazil, May 2011.

Bibliography

- [1] P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1473–1488, nov. 2008.
- [2] M. Ahad, J. Tan, H. Kim, and S. Ishikawa, "Human activity recognition: Various paradigms," in *International Conference on Control, Automation and Systems. ICCAS 2008.*, pp. 1896–1901, oct. 2008.
- [3] E. Duthie, "Falls," in *Medical Clinics of North America*, pp. 1321–1335, April-June 1989.
- [4] A. Fish, L. Sudakov-Boresha, and O. Yadid-Pecht, "Low-power tracking image sensor based on biological models of attention," *International Journal on Information Theory and Applications*, vol. 14, no. 2, pp. 103–114, 2006.
- [5] J. Choi, S.-W. Han, S.-J. Kim, S.-I. Chang, and E. Yoon, "A spatial-temporal multiresolution CMOS image sensor with adaptive frame rates for tracking the moving objects in region-of-interest and suppressing motion blur," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2978–2989, 2007.
- [6] T. Serre, *Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans and Machines*. PhD thesis, MIT, Apr. 2006.
- [7] G. Indiveri, S.-C. Liu, T. Delbruck, and R. Douglas, "Neuromorphic systems," *Encyclopedia of Neuroscience*, pp. 521–528, 2009.
- [8] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, pp. 288–295, 2010.
- [9] E. Chicca, A. M. Whatley, P. Lichtsteiner, V. Dante, T. Delbruck, P. Del Giudice, R. J. Douglas, and G. Indiveri, "A multi-chip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity," *IEEE Transac-*

- tions on Circuits and Systems I, Regular Papers*, vol. 54, pp. 981–993, May 2007.
- [10] R. Vogelstein, U. Mallik, E. Culurciello, G. Cauwengberghs, and R. Etienne-Cummings, “A multi-chip neuromorphic system for spike-based visual information processing,” *Neural Computation*, 2007.
- [11] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Haliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, “Neuromorphic silicon neuron circuits,” *Frontiers in Neuroscience*, vol. 5, no. 73, 2011.
- [12] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, “STDP and STDP variations with memristors for spiking neuromorphic learning systems,” *Frontiers in Neuroscience*, vol. 7, no. 2, 2013.
- [13] C. Zamarreno-Ramos, L. A. Camunas-Mesa, J. A. Perez-Carrasco¹, T. Masquelier, T. Serrano-Gotarredona¹, and B. Linares-Barranco, “On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex,” *Frontiers in Neuroscience*, vol. 5, no. 26, 2011.
- [14] T. Hamilton, C. Jin, A. van Schaik, and J. Tapson, “An active 2-d silicon cochlea,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, pp. 30–43, Mar. 2008.
- [15] V. Chan, S.-C. Liu, and A. van Schaik, “AER EAR: A matched silicon cochlea pair with address event representation interface,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, 2007.
- [16] A. Cassidy, T. Murray, A. Andreou, and J. Georgiou, “Evaluating on-chip interconnects for low operating frequency silicon neuron arrays,” in *2011 IEEE International Symposium on Circuits and Systems*, pp. 2437–2440, May 2011.
- [17] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128×128 120 dB 15 μ s latency asynchronous temporal contrast vision sensor,” *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.
- [18] J. Lenero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, “A 3.6 μ s latency asynchronous frame-free event-driven dynamic-vision-sensor,” *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 1443–1455, Jun. 2011.

- [19] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128×128 1.5% contrast sensitivity 0.9% fpn 3 μ s latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, pp. 827–838, Mar. 2013.
- [20] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.
- [21] K. Boahen, "A burst-mode word-serial address-event link-I: transmitter design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 7, pp. 1269–1280, 2004.
- [22] M. T. L. Aung, A. T. Do, S. Chen, and K. S. Yeo, "Adaptive priority toggle asynchronous tree arbiter for AER-based image sensor," in *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, pp. 66–71, Oct. 2011.
- [23] X. Zhang and S. Chen, "A hybrid-readout and dynamic-resolution motion detection image sensor for object tracking," in *2012 IEEE International Symposium on Circuits and Systems*, pp. 1628–1631, May 2012.
- [24] B. Zhao, X. Zhang, S. Chen, K. S. Low, and H. Zhuang, "A 64x64 CMOS image sensor with on-chip moving object detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 581–588, 2012.
- [25] S. Chen, W. Tang, X. Zhang, and E. Culurciello, "A 64x64 pixels UWB wireless temporal-difference digital image sensor," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20, pp. 2232–2240, 2012.
- [26] "jAER." <http://jaer.wiki.sourceforge.net>.
- [27] X. Zhang and S. Chen, "Live demonstration: A high-speed-pass asynchronous motion detection sensor," in *2013 IEEE International Symposium on Circuits and Systems*, (Beijing, China), p. 671, May 2013.
- [28] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, B. K. P. Schon, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *12th Signal Processing Education Workshop*, pp. 173–178, Sept. 2006.

- [29] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *IEEE International Symposium on Circuits and Systems*, pp. 845–848, May 2007.
- [30] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Trans. Circuits and Systems, Part-I: Regular Papers*, vol. 53, pp. 2548–2566, Dec. 2006.
- [31] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, C. Serrano-Gotarredona, J. Perez-Carrasco, B. Linares-Barranco, A. Linares-Barranco, G. Jimenez-Moreno, and A. Civit-Ballcells, "On real-time AER 2D convolutions hardware for neuromorphic spike based cortical processing," *IEEE Trans. Neural Networks*, vol. 19, pp. 1196–1219, July 2008.
- [32] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Ballcells, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory processing learning actuating system for high-speed visual object recognition and tracking," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1417–1438, Sept. 2009.
- [33] J. A. Perez-Carrasco, B. Acha, C. Serrano, L. Camunas-Mesa, T. Serrano-Gotarredona, and B. Linares-Barranco, "Fast vision through frame-less event-based sensing and convolutional processing. Application to texture recognition," *IEEE Transactions on Neural Networks*, vol. 21, pp. 609–620, Apr. 2010.
- [34] C. Farabet, R. Paz, J. Perez-Carrasco, C. Zamarreno-Ramos, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing," *Frontiers in Neuroscience*, vol. 6, no. 32, 2012.
- [35] J. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free

- event-driven vision systems by low-rate rate-coding and coincidence processing. Application to feedforward ConvNets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2706–2719, Nov. 2013.
- [36] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” in *2010 IEEE International Symposium on Circuits and Systems*, pp. 253–256, 2010.
- [37] M. Riesenhuber and T. Poggio, “Hierarchical models of object recognition in cortex,” *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.
- [38] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.
- [39] B. Zhao and S. Chen, “Realtime feature extraction using max-like convolutional network for human posture recognition,” in *2011 IEEE International Symposium on Circuits and Systems*, pp. 2673–2676, May 2011.
- [40] S. Chen, P. Akselrod, B. Zhao, J. Perez-Carrasco, B. Linares-Barranco, and E. Culurciello, “Efficient feedforward categorization of objects and human postures with address-event image sensors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 302–314, Feb. 2012.
- [41] S. Chen, P. Akselrod, and E. Culurciello, “A biologically inspired system for human posture recognition,” in *2009 IEEE Biomedical Circuits and Systems Conference*, pp. 113–116, Nov. 2009.
- [42] R. Gutig and H. Sompolinsky, “The tempotron: a neuron that learns spike timing-based decisions,” *Nature Neuroscience*, vol. 9, pp. 420–428, Feb. 2006.
- [43] J. A. Lenero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, “A 3.6 μ s asynchronous frame-free event-driven dynamic-vision-sensor,” *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 1443–1455, June 2011.
- [44] S. Chen, W. Tang, and E. Culurciello, “A 64 $\times$ 64 pixels UWB wireless temporal-difference digital image sensor,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1404–1407, 2010.
- [45] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim, “A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierar-

- chical recognizer,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, pp. 1166–1172, Sept. 2010.
- [46] R. Jafari, W. Li, R. Bajcsy, S. Glaser, and S. Sastry, “Physical activity monitoring for assisted living at home,” in *Proceedings of 4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN2007)*, vol. 13, pp. 213–219, 2007.
- [47] X. Long, B. Yin, and R. Aarts, “Single-accelerometer-based daily physical activity classification,” in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2009)*, pp. 6107–6110, sept. 2009.
- [48] D.-U. Jeong, S.-J. Kim, and W.-Y. Chung, “Classification of posture and movement using a 3-axis accelerometer,” in *International Conference on Convergence Information Technology*, pp. 837–844, nov. 2007.
- [49] L. Dong, J. Wu, and X. Chen, “Real-time physical activity monitoring by data fusion in body sensor networks,” in *2007 10th International Conference on Information Fusion*, pp. 1–7, july 2007.
- [50] C.-C. Yang and Y.-L. Hsu, “A review of accelerometry-based wearable motion detectors for physical activity monitoring,” *Sensors*, vol. 10, pp. 7772–7788, 2010.
- [51] N. Robertson and I. Reid, “A general method for human activity recognition in video,” *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 232–248, 2006. Special Issue on Modeling People: Vision-based understanding of a person’s shape, appearance, movement and behaviour.
- [52] X. Yuan and X. Yang, “A robust human action recognition system using single camera,” in *International Conference on Computational Intelligence and Software Engineering. CiSE 2009.*, pp. 1–4, dec. 2009.
- [53] Z. Zhou, X. Chen, Y.-C. Chung, Z. He, T. Han, and J. Keller, “Activity analysis, summarization, and visualization for indoor human activity monitoring,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1489–1498, nov. 2008.
- [54] X. Ji and H. Liu, “Advances in view-invariant human motion analysis: A review,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, pp. 13–24, jan. 2010.

- [55] M. Yang, F. Lv, W. Xu, K. Yu, and Y. Gong, "Human action detection by boosting efficient motion features," in *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 522–529, oct. 2009.
- [56] S. Ali and M. Shah, "Human action recognition in videos using kinematic features and multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 288–303, feb. 2010.
- [57] M. Singh, A. Basu, and M. Mandal, "Human activity recognition based on silhouette directionality," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1280–1292, sept. 2008.
- [58] H. Foroughi, B. Aski, and H. Pourreza, "Intelligent video surveillance for monitoring fall detection of elderly in home environments," in *11th International Conference on Computer and Information Technology (ICCIT 2008)*, pp. 219–224, dec. 2008.
- [59] N. Kiryati, T. Raviv, Y. Ivanchenko, and S. Rochel, "Real-time abnormal motion detection in surveillance video," in *19th International Conference on Pattern Recognition (ICPR 2008)*, pp. 1–4, dec. 2008.
- [60] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *IEEE International Conference on Computer Vision (ICCV 2003)*, vol. 2, pp. 726–733, oct. 2003.
- [61] A. Nasution, P. Zhang, and S. Emmanuel, "Video surveillance for elderly monitoring and safety," in *2009 IEEE Region 10 Conference (TENCON 2009)*, pp. 1–6, jan. 2009.
- [62] E. Herrero-Jaraba, C. Orrite-Urunuela, F. Monzon, and D. Buldain, "Video-based human posture recognition," in *Proceedings of the 2004 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety (CIHSPS 2004)*, pp. 19–22, july 2004.
- [63] Q.-C. Pham, L. Gond, J. Begard, N. Allezard, and P. Sayd, "Real-time posture analysis in a crowd using thermal imaging," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–8, june 2007.
- [64] J. Han and B. Bhanu, "Human activity recognition in thermal infrared imagery," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops. CVPR Workshops 2005.*, p. 17, june 2005.

- [65] M. Imaging, "Swissranger sr4000." <http://www.mesa-imaging.ch/>.
- [66] M. C. R. WA., "Kinect for xbox 360."
- [67] G. Diraco, A. Leone, and P. Siciliano, "An active vision system for fall detection and posture recognition in elderly healthcare," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pp. 1536–1541, 2010.
- [68] Y. Zhu, B. Dariush, and K. Fujimura, "Controlled human pose estimation from depth image streams," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. CVPRW '08.*, pp. 1 –8, june 2008.
- [69] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," in *CVPR '10*, pp. 755–762, 2010.
- [70] M. Amoretti, F. Wientapper, F. Furfari, S. Lenzi, and S. Chessa, "Sensor data fusion for activity monitoring in ambient assisted living environments," in *First International ICST Conference on Sensor Systems and Software, S-CUBE 2009.*, sept. 2009.
- [71] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *CVPR '11*, june 2011.
- [72] S. Mizuno, K. Fujita, H. Yamamoto, N. Mukozaka, and H. Toyoda, "A 256×256 compact CMOS image sensor with on-chip motion detection function," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 1072–1075, 2003.
- [73] M. Gottardi, N. Massari, and S. Jawed, "A $100 \mu\text{W}$ 128×64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 1582–1592, May 2009.
- [74] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schon, B. Kohn, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Digital Signal Processing Workshop*, pp. 173–178, 2006.
- [75] M. Litzenberger, A. Belbachir, N. Donath, G. Gritsch, H. Garn, B. Kohn, C. Posch, and S. Schraml, "Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 653–658, 2006.

- [76] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. Ballcells, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "Caviar: A 45k neuron, 5m synapse, 12g connects/s aer hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1417 –1438, sept. 2009.
- [77] M. Piccardi, "Background subtraction techniques: a review," in *2004 IEEE International Conference on Systems, Man and Cybernetics (SMC '04)*, vol. 4, pp. 3099 – 3104, oct. 2004.
- [78] S.-C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proc. Video Communications and Image Processing*, Jan. 2004.
- [79] S. Herrero and J. Bescós, "Background subtraction techniques: Systematic evaluation and comparative analysis," in *Proceedings of the 11th International Conference on Advanced Concepts for Intelligent Vision Systems, ACIVS '09*, pp. 33–42, 2009.
- [80] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *Proceedings of the 7th IEEE International Conference on Computer Vision, ICCV '99*, pp. 1–19, Sept. 1999.
- [81] I. Biederman, "Human image understanding: Recent research and a theory," *Computer Vision, Graphics, and Image Processing*, pp. 29–73, 1985.
- [82] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.
- [83] B. G. Kai Uwe Barthel, Internationale Medieninformatik, "Color histogram 3d." <http://rsbweb.nih.gov/ij/plugins/color-inspector.html>.
- [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] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, pp. 610 –621, nov. 1973.
- [90] K. Laws, "Rapid texture identification," in *Proceedings of the Seminar on Image Processing for missile guidance*, pp. 376–380, 1980.
- [91] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 837 –842, aug 1996.
- [92] R. S. Choras, "Image feature extraction techniques and their applications for CBIR and biometrics systems," *International Journal of Biology and Biomedical Engineering*, vol. 7, pp. 6–16, 2007.
- [93] C.-L. Liu, M. Koga, and H. Fujisawa, "Gabor feature extraction for character recognition: comparison with gradient feature," in *Proceedings. Eighth International Conference on Document Analysis and Recognition (ICDAR '05)*, pp. 121 – 125 Vol. 1, 2005.
- [94] R. Ramanathan, A. Nair, L. Thaneshwaran, S. Ponmathavan, N. Valliappan, and K. Soman, "Robust feature extraction technique for optical character recognition," in *International Conference on Advances in Computing, Control, Telecommunication Technologies*, pp. 573 –575, dec. 2009.
- [95] S. Newsam and C. Kamath, "Comparing shape and texture features for pattern recognition in simulation data," in *SPIE's Annual Symposium on Electronic Imaging*, (San Jose, CA), Jan 2005.

- [96] Y. Mingqiang, K. Kidiyo, and R. Joseph, "A survey of shape feature extraction techniques," in *Pattern Recognition Techniques, Technology and Applications*, pp. 43–90, 2008.
- [97] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, february 1962.
- [98] O. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition - a survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641–662, 1996.
- [99] M. R. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America (1917-1983)*, vol. 70, pp. 920–930, Aug. 1980.
- [100] E. Persoon and K.-S. Fu, "Shape discrimination using fourier descriptors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, pp. 170–179, march 1977.
- [101] E. Persoon and K.-S. Fu, "Shape discrimination using fourier descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 388–397, may 1986.
- [102] D. Zhang and G. Lu, "A comparative study on shape retrieval using fourier descriptors with different shape signatures," *Journal of Visual Communication and Image Representation*, no. 14 (1), pp. 41–60, 2003.
- [103] J.-S. Hu, T.-M. Su, and P.-C. Lin, "3-d human posture recognition system using 2-d shape features," in *2007 IEEE International Conference on Robotics and Automation*, pp. 3933–3938, april 2007.
- [104] J. Ricard, D. Coeurjolly, and A. Baskurt, "Generalization of angular radial transform," in *International Conference on Image Processing*, pp. 2211–2214, 2004.
- [105] J. Ricard, D. Coeurjolly, and A. Baskurt, "Generalizations of angular radial transform for 2d and 3d shape retrieval," *Pattern Recogn. Lett.*, vol. 26, pp. 2174–2186, October 2005.
- [106] S. Tabbone, L. Wendling, and J. P. Salmon, "A new shape descriptor defined on the radon transform," *Comput. Vis. Image Underst.*, vol. 102, pp. 42–51, April 2006.
- [107] Y. Wang, K. Huang, and T. Tan, "Human activity recognition based on r transform," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–8, june 2007.

- [108] H. Fujiyoshi and A. Lipton, "Real-time human motion analysis by image skeletonization," in *Proceedings., Fourth IEEE Workshop on Applications of Computer Vision*, pp. 15–21, oct 1998.
- [109] C.-H. Chuang, J.-W. Hsieh, L.-W. Tsai, and K.-C. Fan, "Human action recognition using star templates and delaunay triangulation," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP '08)*, pp. 179–182, aug. 2008.
- [110] E. Yu and J. Aggarwal, "Human action recognition with extremities as semantic posture representation," in *CVPR Workshops 2009*, pp. 1–8, june 2009.
- [111] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 509–522, apr 2002.
- [112] P. Guo and Z. Miao, "A home environment posture and behavior recognition system," in *International Conference on Convergence Information Technology (ICCIT '07)*, pp. 175–180, nov. 2007.
- [113] M. H. Galuberman, "Character recognition for business machines," *Electornics*, pp. 132–136, Feb. 1956.
- [114] P. Guo and Z. Miao, "Projection histogram based human posture recognition," in *2006 8th International Conference on Signal Processing*, vol. 2, pp. 16–20, 2006.
- [115] G. Vamvakas, B. Gatos, S. Petridis, and N. Stamatopoulos, "An efficient feature extraction and dimensionality reduction scheme for isolated greek handwritten character recognition," in *Ninth International Conference on Document Analysis and Recognition (ICDAR '07)*, vol. 2, pp. 1073–1077, sept. 2007.
- [116] M. Blumenstein, B. Verma, and H. Basli, "A novel feature extraction technique for the recognition of segmented handwritten characters," in *Proceedings of Seventh International Conference on Document Analysis and Recognition*, vol. 1 of *ICDAR'03*, pp. 137–141, aug. 2003.
- [117] S. Rajashekararadhya and P. Vanaja Ranjan, "Zone-based hybrid feature extraction algorithm for handwritten numeral recognition of two popular indian scripts," in *2009 World Congress on Nature Biologically Inspired Computing (NaBIC '09)*, pp. 526–530, dec. 2009.

- [118] G. Bo, D. Caviglia, and M. Valle, "An analog vlsi implementation of a feature extractor for real time optical character recognition," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 556–564, apr 1998.
- [119] F. Niu and M. Abdel-Mottaleb, "View-invariant human activity recognition based on shape and motion features," in *Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering*, ISMSE '04, pp. 546–556, 2004.
- [120] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '91)*, pp. 586–591, jun 1991.
- [121] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
- [122] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 257–267, March 2001.
- [123] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.
- [124] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt, "Performance of optical flow techniques," in *1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pp. 236 –242, jun 1992.
- [125] W. Yang, Y. Wang, and G. Mori, "Human action recognition from a single clip per action," in *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops '09)*, pp. 482 –489, Oct 2009.
- [126] "Cerebral cortex, wikipedia." http://en.wikipedia.org/wiki/Cerebral_cortex, 2011.
- [127] "Visual cortex, wikipedia." http://en.wikipedia.org/wiki/Visual_cortex, 2011.
- [128] L. G. Ungerleider and J. V. Haxby, "'what' and 'where' in the human brain," *Current Opinion in Neurobiology*, pp. 157–65, 1994.
- [129] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architectures in cats visual cortex," *Journal of Physiology*, vol. 160, pp. 106–154, 1962.

- [130] D. H. Hubel and T. N. Wiesel, "Sequence regularity and geometry of orientation columns in the monkey striate cortex," *J. Comp. Neur.*, vol. 158, pp. 267–294, 1974.
- [131] C. G. Gross, C. E. Rocha-Miranda, and D. B. Bender, "Visual properties of neurons in the inferotemporal cortex of the macaque," 1972.
- [132] K. Tanaka, "Inferotemporal cortex and object vision," 1996.
- [133] "Object recognition - hierarchical models of object recognition, slides online." [URL] http://www.tu-chemnitz.de/informatik/KI/scripts/ws0910/Networks_ObjectRecognition.pdf, 2011.
- [134] M. Riesenhuber and T. Poggio, "Computational models of object recognition in cortex: A review," tech. rep., Massachusetts Institute of Technology, sept. 2003.
- [135] N. K. Logothetis, J. Pauls, and T. Poggio, "Shape representation in the inferior temporal cortex of monkeys," *Current Biology*, vol. 5, pp. 552–563, 1995.
- [136] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, pp. 520–522, june 1996.
- [137] S. J. Thorpe and M. Fabre-thorpe, "Seeking categories in the brain," *Science*, vol. 291, pp. 260–263, 2001.
- [138] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
- [139] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comp.*, vol. 1, no. 4, pp. 541–551, 1989.
- [140] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, nov. 1998.
- [141] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*, pp. 253 –256, june 2010.
- [142] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'05)*, pp. 994–1000, june 2005.

- [143] P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.
- [144] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, Sept. 1998.
- [145] L. Fausett, *Fundamentals of Neural Networks*. New York: Prentice Hall, 1994.
- [146] StatSoft, "Neural networks." <http://www.statsoft.com/textbook/neural-networks/>.
- [147] StatSoft, "Classification trees." <http://www.statsoft.com/textbook/classification-trees/>.
- [148] Y. Gao and M. Leung, "Face recognition using line edge map," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 764–779, June 2002.
- [149] S. Chen, P. Akselrod, and E. Culurciello, "A biologically inspired system for human posture recognition," in *IEEE BIOCAS*, pp. 113–116, 2009.
- [150] M. Barnsley, *Fractals Everywhere*. Academic Press, 1988.
- [151] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.
- [152] F. J. Huang and Y. LeCun, "NORB object database." URL: <http://www.cs.nyu.edu/~ylclab/data/norb-v1.0-small/>, 2009.
- [153] A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, A. Jimenez, M. Rivas, G. Jimenez, and A. Civit, "On the AER convolution processors for FPGA," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 4237–4240, June 2010.
- [154] F. Folowosele, R. J. Vogelstein, and R. Etienne-Cummings, "Towards a cortical prosthesis: Implementing A spike-based HMAX model of visual object recognition in silico," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 4, pp. 516–525, 2011.
- [155] J. Sabarad, S. Kestur, M. S. Park, D. Dantara, V. Narayanan, Y. Chen, and D. Khosla, "A reconfigurable accelerator for neuromorphic object recognition," in *2012 17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 813–818, 2012.
- [156] Z. Fu, T. Delbruck, P. Lichsteiner, and E. Culurciello, "An address-event fall detector for assisted living applications," *IEEE Transactions on Biomedical Circuits and Systems TBCAS*, vol. 2, pp. 88–96, 2008.

- [157] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 1 ed., Aug. 2002.
- [158] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, p. 500C544, Aug. 1952.
- [159] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, p. 1569C1572, Nov. 2003.
- [160] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PLoS ONE*, vol. 3, no. 1, 2008.
- [161] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput Biol*, vol. 3, p. e31, Feb. 2007.
- [162] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Competitive STDP-based spike pattern learning," *Neural computation*, vol. 21, pp. 1259–1276, May 2009.
- [163] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting," *Neural Computation*, vol. 22, pp. 467–510, Feb. 2010.
- [164] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Transactions on Neural Networks and Learning Systems*, to be published. 2013.
- [165] <http://www3.ntu.edu.sg/home2009/zhao0130/AERpostures.htm>.
- [166] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits." URL: <http://yann.lecun.com/exdb/mnist/>.