Space-time event clouds-based event processing

Wang, Qinyi

2021

Wang, Q. (2021). Space-time event clouds-based event processing. Doctoral thesis,
Nanyang Technological University, Singapore. <https://hdl.handle.net/10356/151982>

<https://hdl.handle.net/10356/151982>

<https://doi.org/10.32657/10356/151982>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0
International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 17:44:57 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**SPACE-TIME EVENT CLOUDS-
BASED EVENT PROCESSING**

WANG QINYI

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2021

SPACE-TIME EVENT CLOUDS-BASED EVENT PROCESSING

WANG Qinyi

School of Electrical & Electronic Engineering

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

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

25 May 2021

.....

Date

.....

WANG Qinyi

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

28 May 2021

.....

Date

Professor LU Yilong

Authorship Attribution Statement

This thesis contains material from one paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as **Wang Qinyi**, Zhang Yexin, Yuan Junsong, and Lu Yilong, “Space-time Event Clouds for Gesture Recognition: from RGB Cameras to Event Cameras”, in Proceedings of IEEE Winter Conference on Applications and Computer Vision (WACV’19), 2019

The contributions of the co-authors are as follows:

- Associate Professor Yuan Junsong provided the initial idea.
- Zhang Yexin and I co-developed the theory framework.
- I designed the implementation framework and evaluation process.
- Zhang Yexin designed a rolling buffer mechanism to improve real-time performance.
- Professor Lu Yilong provided overall guidance to the research.
- Zhang Yexin and I prepared manuscript together. The manuscript was revised by Professor Lu Yilong and Associate Professor Yuan Junsong.

25 May 2021

.....

Date

.....

WANG Qinyi

Acknowledgements

The last five years have passed in a flash. Throughout my PhD study, there were many unexpected twists and turns. Without the amazing support and encouragement from everyone I have met along the way, I could not have possibly made it to this point.

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Lu Yilong. Without his assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. As a part-time PhD student, I would like to thank Professor Lu very much for his support and understanding for both my PhD study and my work.

I would like to express my sincere gratitude to Associate Professor Chen Shoushun being my supervisor in the first two years of my PhD study from August 2015 to July 2017. He continually and convincingly conveyed a spirit of adventure in frontiers exploration and excitement regarding solving real-world problems. In the first year and a half, I designed a CMOS image sensor with on-chip data compression and achieved a successful tape-out under the guidance of Associate Professor Chen. In July 2017, it was unexpected but understandable that Associate Professor Chen decided to suspend his academic career and became an entrepreneur. At that time, I was working on my second image sensor chip. However, considering that there was no other suitable professor to fund my following research, and after an in-depth discussion with Associate Professor Chen, I decided to give up my second chip design and changed my research topic from circuits design to computer vision. Although it is a pity that the work I have done in the first two years' PhD study is not included in this thesis, the knowledge and skills I have gained are

beneficial to lifelong learning. I truly appreciated that Associate Professor Chen was continuously offering support and help on both my PhD study and my career.

I would like to express my sincere gratitude to Associate Professor Yuan Junsong being my second supervisor from August 2017 to January 2018. Associate Professor Yuan introduced me to the field of computer vision and was very patient with my knowledge gaps in the area. Associate Professor Yuan initialized the idea of this research work and helped to shape the thesis. Much of the analysis presented in Section 3 is owed to the guidance from Associate Professor Yuan. Associate Professor Yuan moved to the U.S. in January 2018 and Professor Lu Yilong became my third supervisor since February 2018. It was a short time but an unforgettable and extraordinary experience to work with Associate Professor Yuan. I truly appreciated that Associate Professor Yuan was continuously offering support and help to the best of his ability afterward until I finished my thesis.

Getting through my thesis required more than academic support. I would like to express my special thanks to my fiance Mr Zhang Yexin for his constant support and love in my life. Thanks for his company in those deadline eves. Thanks for his inspiration and encouragement when I hit a bottleneck. Thank him for curing my fatigue with superb cooking skills and delicious food. Most importantly, none of this could have happened without my parents, who offered their encouragement through phone calls every week. They are the source of my determination and perseverance in pursuing all my endeavors.

WANG Qinyi, 5 October 2020

Abstract

Conventional cameras are designed with a human audience in mind. Pixels in the sensor array generate an image that resembles the final scene we see in our minds. Although modern deep learning pipelines have been shown their capability to extracting progressively higher levels of visual understanding, one might pose the question, is this the optimal starting point for an artificial visual understanding pipeline? Bio-based event cameras provide possible positive responses to this question. Instead of integrating photons over fixed exposure times, event cameras aim to closely mimic the ganglion cells in human visual systems, which respond to changes in illumination. Event cameras output an asynchronous sequence of events, i.e., an event stream, where individual event (x, y, t) corresponds to the space-time location when a pixel sensor captures an intensity change. Compared with conventional cameras, event cameras are frameless but can sense and track extremely fast motions with low latency and high dynamic range. These advantages make event cameras extremely suitable to solve the bottlenecks faced by fast motion-related vision tasks (e.g., action recognition, gesture recognition).

The benefits of event cameras are primarily attributed to their asynchronous reaction to lighting changes in the scenes. Ideally, these benefits should only be utilized if the asynchronous event streams running on a completely asynchronous processor are completely implemented in a fully asynchronous algorithm. Neuromorphic processors are asynchronous by nature and have the potential to enable asynchronous processing directly from event cameras in real-time. Neuromorphic computing, however, is going to take at least another decade to become mainstream. The challenge we are facing so far is to develop algorithms which take advantage of the asynchronous nature of event data, while also utilizing the benefits provided

by current parallel computing. Thus, an efficient representation of event data is required to fully utilize the sparse and asynchronous features of event data. In addition, efficient algorithms need to be explored to analyze the event data and solve the challenges of event-based processing on synchronous processors.

In this thesis, the sparse and asynchronous features of event data are first studied, and the concept of Space-time Event Cloud (STEC) is proposed. Under the STEC perspective, the classification of an action becomes the classification of the geometric distribution of the space-time event cloud generated by that action. A subspace feature sampling scheme is proposed to deal with the classification of noisy space-time event clouds. To the best of our knowledge, this is the first time to interpret event streams as space-time event clouds for gesture recognition problems. Experimental results validate the proposed space-time event cloud concept as an effective representation to characterize the event streams and prove that the space-time event cloud is end-to-end learnable. Secondly, the concept of STEC is re-defined as a sequence of temporally ordered space-time events. The Space-time Event Net (ST-EVNet) with hierarchical space-time kernels is proposed to analyze STECs. Proposed hierarchical space-time kernels learn the local spatiotemporal features in a unified way and aggregate the learned spatiotemporal features with great efficiency. A two-channel architecture is proposed to deal with single-hand and dual-hand gestures under a unified system. Finally, a light-weighted event-based Human-Computer Interaction system for real-time applications is developed. This is a fully event-driven detector-tracker-classifier full-stack solution, which demonstrates the possibility of building a fully event-based HCI system on a synchronous processor while keeping advantages of the asynchronous event-based sensing. It is believed that these contributions will provide the basis for the further use of event cameras to overcome the challenges of computer vision and allow researchers to make use of their advantages to enhance the stability, robustness and performance of these solutions.

Contents

Acknowledgements	iv
Abstract	vi
List of Figures	xi
List of Tables	xvii
Acronyms	xix
1 Introduction	1
1.1 Background	1
1.1.1 Neuromorphic Sensing	1
1.1.2 Bio-inspired Event Cameras	2
1.1.3 Representations of Event Data	5
1.1.4 Advantages of Event Cameras	9
1.1.5 Challenges of Event Cameras	9
1.2 Objectives of This Study	12
1.3 Major Contributions of This Study	13
1.4 Organization of the Thesis	14
2 Literature Review	16
2.1 Overview of Event-based Processing Using ANNs	16
2.2 Processing Events as Event Frames	17
2.2.1 The Four-Channel Event Image	17
2.2.2 The Spatio-Temporal Voxel Grid Based Event Frame	18
2.2.3 The SBT Frames and SBE Frames	21
2.3 Processing Events as Neuron Spikes	21
2.3.1 Event-based Classification Using SNNs	23
2.3.2 Processing Event Data on Neuromorphic Processors	25
2.4 Processing Event Data as It Is	27
2.4.1 RG-CNN: Graph-based Spatial-temporal Feature Learning	28
2.4.1.1 The RG-CNN Architecture	28
2.4.1.2 Performance	29

2.4.2	EventNet: Asynchronous Recursive Event Processing	31
2.4.2.1	Performance	36
3	Solving Space-time Entanglements with Space-time Event Clouds	38
3.1	Introduction	38
3.2	Solving Space-time Entanglements with Space-time Event Clouds	41
3.3	Event Bagging: A Global Subspace Sampling for Noisy Event Clouds	45
3.3.1	Curse of Dimensionality	45
3.3.2	Low Dimensional Feature Projection	46
3.3.3	Event Bagging: A Ensemble-Centric Perspective	49
3.4	Classifiers for Space-time Event Clouds	50
3.4.1	Adapted PointNet for 3D Space-time Event Clouds Classification	50
3.4.2	Hierarchical PointNet	52
3.5	Online Real-time Gesture Recognition Framework	53
3.6	Experiments	54
3.6.1	Dataset	55
3.6.2	Classifiers Training	55
3.6.3	Classification ability of different classifiers	57
3.6.4	Online gesture recognition system	59
3.7	Discussion	60
3.7.1	Robustness Against Noise	60
3.7.2	Efficiency of Proposed Method	60
3.8	Concluding Remarks	61
4	ST-EVNet: Hierarchical Space-time Feature Learning on Event Clouds	63
4.1	Introduction	63
4.2	Related Works	66
4.3	Proposed Method	67
4.3.1	Properties of Space-time Event Clouds	67
4.3.2	Hierarchical Local Spatiotemporal Feature Learning	70
4.3.3	Dual-channel Architecture	74
4.4	Experiments	75
4.4.1	Datasets	76
4.4.2	Effectiveness of Temporally Ordered STECs	76
4.4.3	Importance of Dual-channel Architecture	77
4.4.4	Network Efficiency Analysis	79
4.4.5	Comparison with EventNet	81
4.5	Concluding Remarks	82
5	A Portable Event-based HCI System for Real-time Applications	84
5.1	Introduction	84
5.2	Vision-based HCI Systems	86

5.3	A Portable Event-based HCI System: Design Criteria and Considerations	88
5.4	The Event-based Hand Detector	90
5.5	The Event-based Hand Tracker	93
5.5.1	The Event-based Asynchronized Tracking	93
5.5.2	Search Region Offsetting for Fast Hand Tracking	95
5.6	Event-based Classifier	96
5.6.1	Representation of Gestures	96
5.6.2	Constructing STEC Using a Dynamic Buffer	97
5.6.3	Classification of STECs	98
5.7	Experiments	99
5.7.1	Detector Robustness	99
5.7.2	Tracker Robustness	103
5.7.3	Classifier Accuracy	105
5.7.4	System Evaluation	108
5.8	Concluding Remarks	110
6	Conclusions and Future Work	112
6.1	Conclusions	112
6.2	Discussion and Future Work	114
	List of Publications	116
	Bibliography	117

List of Figures

1.1.1 Neuromorphic sensing and computation promises better AI	2
1.1.2 In the human vision system, light signals are propagated through the bipolar cells to the ganglion cells, which fire asynchronous spikes to the visual cortex at a rate proportional to differences in light intensity arriving at each rod and cone. However, conventional vision sensors are designed with a human audience in mind. They bypass most of the early layers of visual understanding in the brain and present the final full frames in a synchronized manner. Instead, to mimic more closely the human visual system, event-based vision sensors monitor the intensity change in each pixel and report these intensity changes caused by the scene as an asynchronous sequence of events.	3
1.1.3 Working principle of event cameras. Ideally, an event $e_i \doteq (\mathbf{x}_i, t_i)$ is triggered at pixel location $\mathbf{x}_i \doteq (x_i, y_i)^T$ once the log intensity L change reaches a predefined contrast sensitivity threshold at time t_i	4
1.1.4 Events in a spatiotemporal neighborhood are converted into an image. Event count frames are obtained by counting the total number of events generated in each pixel location within a time interval. Event binary frames consider pixel locations generating at least one event as the same value.	7
1.1.5 Time surfaces. (a)The time surface in 3D view of a pen moving from left to right. The time surface is constructed by the timestamp of the latest event at each pixel location. (b)The time surface in 2D view of (a). (c)The time surface in 3D view of a hand moving from left to right. (d)The time surface in 2D view of (c).	7
1.1.6 In SNN-based approach, an individual event is treated as a neuron spike.	8
1.1.7 Denoising of events has very high computation cost.	12
2.2.1 Event timestamp images	18

2.2.2	The network architecture of EV-FlowNet proposed in [1].	
	In the forward path, the four-channel event image is passed through four strided convolution layers, with output channels doubling each time. The resulting activations are passed through two residual blocks, and then four upsample convolution layers, where the activations are upsampled using nearest neighbor resampling and then convolved, to obtain a final flow estimate.	19
2.2.3	The workflow and network architecture of EVGait proposed in [2].	
	The network is divided into two major components: convolutional layers with Residual Block layers are responsible for feature extraction and fully-connected layers with softmax associate the features to different identities.	19
2.2.4	(a)The spatio-temporal voxel grid based event frames proposed in [3] is illustrated. The event stream (depicted as red/blue dots on the time axis) is split into windows E_k containing multiple events. Each window is converted into an event tensor $\mathbf{F}_k$ and passed through the network. (b)The network proposed in [3] consuming the spatio-temporal voxel grid event frames for gray-scale image reconstruction.	20
2.2.5	(a)The event stream and construction of stacks by SBT and SBE proposed in [4]. The color pairs Red(+)/ Blue(-) and Green(+)/Cyan(-) indicate the event polarity plus/minus. (b)The generator and discriminator networks proposed in [4] consuming the stack of SBT and SBE for grayscale image reconstruction.	22
2.3.1	(a)System setup. The DVS128 camera is connected to an NS1e development board using a USB cable.(b)TrueNorth architecture. Neurosynaptic cores connect inputs to neurons using a synaptic crossbar. An asynchronous communication fabric routes events between cores.	25
2.4.1	The RG-CNN Architecture. Firstly, S graphs are constructed from the event stream. Each graph is passed through the spatial feature learning module. For object classification, the output of the spatial feature learning module is directly sent to fully connected layers. For action recognition, the coarse temporal dependencies are modeled over multiple graphs by converting to a grid representation via the Graph2Grid module and perform temporal feature learning with a conventional 3D-CNN.	29

2.4.2 The comparison of EventNet with PointNet. EventNet has a novel temporal coding layer to compute the sequence of events in a recursive manner. To improve computation efficiency at inference, the most computationally significant parts, mlp1 and mlp2, are realized using a lookup table (LUT) based on the trained parameters, which is significantly faster than an MLP. As a result, EventNet processes streams of events efficiently in an event-driven manner. It computes per-event feature by a LUT, applies the temporal code to the global feature, and aggregate global feature by max-pooling of the two vectors, which is repeated recursively as it receives a new event. Similar to PointNet, EventNet has a variant of architecture that can output on a per-event basis, which is realized with mlp4 by a concatenation of local feature and global feature(blue line).	33
3.1.1 (a)Counter-clockwise circling recorded by frames. (b)Hand waving recorded by frame. (c)Counter-clockwise circling recorded by events. The hand movement generates a large number of events, while the stationary background will not generate events. (d)Counter-clockwise circling recorded by events. The hand movement generates a large number of events, while the stationary background will not generate events.	40
3.2.1 (a)Conventional approaches disentangle the spatial and temporal features of event streams . (b)The proposed method solves the space-time entanglement of event streams by model them as space-time event clouds.	43
3.2.2 A sliding window is applied to select a space-time event cloud for each classification.	44
3.3.1 The number of events in each event cloud is huge and it varies case by case. The effect of noise is significant in raw event clouds.	45
3.3.2 Unlike standard bagging approach where data points are randomly sampled from data distribution, global subspace sampling randomly select a subset of features for all data samples each time and form a subset of all data points with subsampled features.	47
3.3.3 The proposed event bagging scheme is applied on-the-fly during the training phase. For each event stream containing a gesture with label y_i , training samples are selected based on the pre-defined sliding window size and step size. The entire training set is represented as $S = \{(s_1, y_1), (s_2, y_2), \dots, (s_m, y_m)\}$. In each epoch j , for each s_i in set S , randomly pick (without replacement) N events to create a event subset $F_i^j \in \mathbf{X}^N$ and obtain new subspace dataset $S^j = \{(F_i^j, y_1), (F_2^j, y_2), \dots, (F_m^j, y_m)\}$. Train the model using the new subspace dataset S^j	48
3.4.1 (a)PointNet consumes point clouds for object classification. (b)In this work, PointNet is adapted to consume space-time event clouds for gesture recognition in event-camera based classification system. .	51

3.4.2 A PointNet-based hierarchic feature learning architecture, PointNet++, is trained to learn local and global features of event clouds generated by hand gestures	53
3.5.1 Rolling buffer framework for real time gesture recognition	54
3.6.1 Two-second snippets of 10 classes in DVS128 Gesture Dataset.	56
3.6.2 Confusion matrices of different classifiers on the same test setup, where 512 events are randomly down-sampled from a 0.5s sliding window. Only non zero percentages are shown. (a)LSTM, 10 classes (b)PointNet, 10 classes (c)PointNet, 10 classes (d)PointNet++, 11 classes	58
3.7.1 Efficiency of proposed framework (a)CNN-based approach takes in 16k input (b)event clouds-based approach takes in 1k input	61
4.1.1 Event-based gesture recognition solution. (a) The top shows a typical video-based gesture recognition solution. When using RGB cameras to shot a <i>Grab</i> gesture. The dynamic grabbing process is recorded as discrete frames. To obtain a more reliable prediction, different features (e.g. RGB, depth, optical flow) are fused at different stages, which increases the complexity of the whole solution. (b) The bottom shows the proposed gesture recognition solution using event cameras. Event cameras record the <i>Grab</i> gesture as a continuous event stream in the space-time continuum. In this work, we design a simple and unitary ST-EVNet to learn rich spatiotemporal information inherent in event data itself for efficient gesture recognition.	64
4.3.1 Temporally ordered space-time event clouds. In [5], the event stream of a gesture is modeled as an unordered space-time event cloud (left, all events are in the same color), where local features embed in its geometric local region. In this work, we model the event stream of a gesture as a temporally ordered space-time event cloud (right, events are colored according to temporal order), where local features can be extracted from the temporal local region without extra computation to construct a local region.	68
4.3.2 Rationality of downsampling on STECs. Top shows the raw STECs of the same gesture and bottom shows the corresponding downsampled STECs. The downsampled STECs preserve the long-term behaviors of the gesture.	70
4.3.3 Hierarchical Space-time kernels. The left shows temporal feature learning via temporal convolution. The right shows 3D geometric feature summarizing via point-wise MLPs and max-pooling. Bottom is the proposed ST-EVNet with hierarchical space-time kernels, which takes advantage of the previous two architectures to aggregate local spatiotemporal features with great efficiency.	71

4.3.4 Dual-channel ST-EVNet architecture. The dual-channel network takes N events from each STEC as input. Local spatiotemporal features are aggregated by hierarchical space-time kernels, and are summarized by max pooling. The numbers in bracket are feature vector sizes of each layer. The output is classification scores for m classes. For the classification net, numbers in bracket are layer sizes of the multi-layer perceptron network.	72
4.3.5 Constructing dual-channel training data. We select six gestures from DVS128 Gesture Dataset to illustrate how we prepare the dual-channel data. We show the STEC of each gesture in a one second snippet. Corresponding frame-based gestures are presented on the left. For gestures generating two separate STECs (right), we arrange them into two channels. Otherwise, we sample the unified STEC (left and middle) into one channel and pad the other channel with zeros.	74
5.3.1 The proposed HCI system includes an event-based hand detector, and event-based tracker and an event-based classifier.	90
5.4.1 (a)The detection framework. (b)Working principle of the mass layer and lifetime layer.	91
5.5.1 (a) The search region and extra search regions; (b) The tracking framework.	94
5.6.1 (a)As the tracking center is updated under a event-based perspective, a very smooth bounding volume is generated. (b)The space-time event cloud within the dynamic bounding volume will be normalized to a $1 \times 1 \times 1$ cube. (c)The ST-EVNet is applied for gesture classification	98
5.7.1 The mass layer's and the lifetime layer's response signals strength change under normal light condition are shown in (a)-(e). The hand location is detected at T=2.9s. The 2D event frame is the accumulation of events in last 100ms.	101
5.7.2 The mass layer's and the lifetime layer's response signals strength change under dim light condition are shown in (a)-(e). The hand location is detected at T=4.1s. The 2D event frame is the accumulation of events in last 100ms.	102
5.7.3 The mass layer's and the lifetime layer's response signals strength at the moment when the hand locations are detected. The 2D event frame is the accumulation of events in last 100ms.	103
5.7.4 Tracker search region and extended search region settings.	105
5.7.5 Visualizing tracking results with groundtruth over different speed. (a)horizontal direction (b)vertical direction	106
5.7.6 Event clouds of gestures used in the experiments and its equivalent shot using RGB cameras. (a)grab. (b)release. (c)up-down. (d)left-right. (e)clockwise. (f)counter-clockwise.	107

5.7.7 In the TV control application:(a)the user waves hand up and down to enter a program. (b)the user waves hand left and right to quite a program. (c)the user waves hand clock-wisely to awake a volume bar. (d)the user moves hand to adjust the volume.(e)the user waves hand counter clock-wisely to awake a hidden menu.	109
---	-----

List of Tables

2.3.1 Benchmark Classification Results on N-MNIST and DVS128 Gestures Datasets	25
2.3.2 The event-based gesture recognition system performance on Truenorth Chip.	27
2.4.1 Quantitative evaluation of RG-CNN on the DVS128 Gestures Dataset [6].	30
2.4.2 Computational complexity of RG-CNN [6].	30
2.4.3 Quantitative evaluation of EventNet [7].	36
2.4.4 Computational complexity of EventNet [7].	36
3.6.1 Classifiers test results. Accuracy is reported in percentage.	57
3.6.2 System test results compared with state-of-the-art results. Accuracy is reported in percentage.	59
4.4.1 Quantitative evaluation on different datasets. Metric is global classification accuracy on test trails. For gesture recognition, the sliding window length is 0.5s. For action recognition, the sliding window length is 1s. A number of N (e.g. 256) events are downsampled from the total events within the sliding window. The x,y and t are normalized to (0, 1) accordingly before feeding into the network.	77
4.4.2 Quantitative evaluation on dual-channel architecture. The configuration of dual-channel ST-EVNet is similar to basic ST-EVNet, except the input size changes to (N, 2, 3). The results of configuring ST-EVNet with different kernel length are also presented. Metric is overall classification accuracy on DVS128 Gesture dataset test trails.	78
4.4.3 Efficiency for real-time gesture recognition.	80
4.4.4 Computational complexity of two efficient event processing techniques.	81
4.4.5 Using timestamp as a feature dimension. EventNet uses timestamp as a temporal decay factor, while the concept of space-time event clouds considers timestamp as an identical input feature dimension as x and y . The concept of STEC is a better description of motion with deformation.	82
5.7.1 Different sliding rail speeds used in experiments. The motion motor settings and corresponding speeds reflected on the image plane are estimated.	104

5.7.2 Stability of proposed tracker against different speeds.	106
5.7.3 Test accuracy of designed gestures.	108
5.7.4 Comparison with existing hand gesture recognition systems.	110

Acronyms

ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
DART	Distribution Aware Retinal
DVS	Dynamic Vision Sensors
GRU	Gated Recurrent Unit
HCI	Human-Computer Interaction
LSTM	Long Short-Term Memory
LUT	Lookup Table
MEP	Mega Events per Second
MLP	Multilayer Perceptron
N-MNIST	The Neuromorphic-MNIST dataset
RG-CNN	Residual-Graph Convolutional Neural Networks
RNN	Recurrent Neural Networks
SBE	Stacking Based on Events
SBT	Stacking Based on Time
SKIM	Skimming Digits
SLAYER	Spike Layer Error Reassignment
SNN	Spiking Neural Networks
STEC	Space-time Event Clouds
ST-EVNet	Space-time Event Net
TCN	Temporal Convolutional Networks

Chapter 1

Introduction

1.1 Background

1.1.1 Neuromorphic Sensing

The development of deep learning algorithms has made significant achievements in the past decade. Most current deep learning implementation techniques are based on Von Neumann computing. The demand for high-performance computing chips is increasing tremendously. As Moore's Law has been slowing down lately, many in the industry believe it will not be able to sustain deep learning progress if it continues to be implemented the way it is today [8]. From another aspect, the current data loading is increasing even faster than Moore's Law, while our traditional sensing systems are not optimized to sense only the most useful information causing significant data redundancy. Moreover, the exponential increase in computing power requirements has created a heat wall for each application.

Although we have made extraordinary progress in Artificial Intelligence, most of our sensing systems and computation hardware are not built in a brain-like way. Thus, the current technology paradigm may not be able to ensure continuous progress in AI, while the neuromorphic-based solutions appear to be a more practical path

FIGURE 1.1.1: Neuromorphic sensing and computation promises better AI

(Figure 1.1.1). The concept of neuromorphic engineering was first developed by Carver Mead in the late 1980s. It refers to using of very-large-scale integration (VLSI) systems containing electronic analog circuits to mimic neuro-biological architectures present in the nervous system [9]. Understanding how biology functions will undoubtedly provide profound insights into new paradigms of sensing and processing in a data-efficient and computation-efficient way.

Vision is one of the most useful sensory functions, but the real-time processing of the continuous, high-dimensional input signals provided by vision sensors is a major challenge. Conventional digital vision sensors tend to have excessive power consumption, size, and cost for useful applications. Neuromorphic vision sensory is an alternative strategy that are based on biological vision systems. In these sensors, specialized sensory processing functions inspired by biological systems such as fly eyes are integrated in parallel, asynchronous circuits that respond in real time. These sensors offer significant advantages over conventional vision sensors.

1.1.2 Bio-inspired Event Cameras

In the human vision system, the incoming light is focused onto a layer of cells on the retina, in the rear of the eye. As shown in figure 1.1.2, this light passes

through a set of rods and cones, whose membrane potentials alter with the amount of light arriving at each cell. These cells are connected to bipolar cells, which in turn relay information to ganglion cells. The ganglion cells play the role of firing action potentials down the optic nerve to the visual cortex, where high-level visual understanding occurs. It is worth mentioning that ganglion cells do not necessarily relay direct intensity information, such as the images we see in our minds. Instead, the firing rate from each ganglion cell correlates with changes in light intensity. That is, a ganglion cell's firing rate is maximized when the amount of light arriving at its corresponding photoreceptor cells changes while minimized when it is static. This change information, encoded as a firing rate, is sent to the primary visual cortex in our brain.

FIGURE 1.1.2: In the human vision system, light signals are propagated through the bipolar cells to the ganglion cells, which fire asynchronous spikes to the visual cortex at a rate proportional to differences in light intensity arriving at each rod and cone. However, conventional vision sensors are designed with a human audience in mind. They bypass most of the early layers of visual understanding in the brain and present the final full frames in a synchronized manner. Instead, to mimic more closely the human visual system, event-based vision sensors monitor the intensity change in each pixel and report these intensity changes caused by the scene as an asynchronous sequence of events.

Conventional cameras are designed with a human audience in mind, they bypass most of the early layers of visual understanding in the brain. Pixels in the sensor array accumulate the number of photons arriving at that pixel in a fixed time interval, generating an image that resembles the final product we see in our minds. Due to the intrinsic benefits of parallelization for electronic devices, most cameras operate synchronously. That is, photons are integrated for exposure times which are fixed for every pixel on the sensor. Synchronizing the pixels allows for an entire image to be read out at once for global shutter cameras, and entire rows for rolling shutter cameras. This departs from the human visual model, where each ganglion cell operates independently and fires asynchronously as light arrives at the retina. Although modern deep learning pipelines have been shown their capability to extracting progressively higher levels of visual understanding, one might pose the question, is this the optimal starting point for an artificial visual understanding pipeline? In other words, can we design a camera that is more suitable for vision understanding than the traditional ones originally designed for human visualization?

FIGURE 1.1.3: **Working principle of event cameras.** Ideally, an event $e_i \doteq (\mathbf{x}_i, t_i)$ is triggered at pixel location $\mathbf{x}_i \doteq (x_i, y_i)^T$ once the log intensity L change reaches a predefined contrast sensitivity threshold at time t_i .

The bio-inspired event cameras provide a possible positive response to this question. Instead of integrating photons over fixed exposure times, event cameras aim to closely mimic the ganglion cells in the human visual system, which respond to changes in illumination. Specifically, in the event camera, each pixel independently

responds to changes in the log intensity $L \doteq \log(I)$. As shown in Figure 1.1.3, in a ideal scenario, and an event $e_i \doteq (\mathbf{x}_i, t_i)$ is triggered at pixel location $\mathbf{x}_i \doteq (x_i, y_i)^T$ and at time t_i as soon as the log intensity L changes since the last event triggered at this pixel reaches a predefined contrast sensitivity threshold C :

$$\Delta L(\mathbf{x}_i, t_i) \doteq L(\mathbf{x}_i, t_i) - L(\mathbf{x}_i, t_i - \Delta t) \quad (1.1.1)$$

$$\Delta L(\mathbf{x}_i, t_i) = p_i C \quad (1.1.2)$$

where $C > 0$ and Δt is the time elapsed since the last event at the same pixel. The contrast sensitivity threshold C determines how fast the pixel responds to the log intensity change. As event cameras respond to illumination changes, they can still capture useful information in low light conditions. The Δt in event camera is usually within a few microseconds, thus event data has a much higher temporal resolution. Compared with conventional cameras with 30~100fps, event cameras are extremely sensitive to motions. The polarity $p_i \in \{+1, -1\}$ indicating the sign of log intensity change. The polarity information plays an import role when using to events to reconstruct image intensities [10] and to solve computer vision problems, where relative image intensity is necessary. However, in the thesis, we do not take polarity information into consideration, as we interpret events as 3D points and the relative intensity information is not involved.

1.1.3 Representations of Event Data

An individual event is denoted as $e_i = (\mathbf{x}_i, t_i)$. The raw event data are continuous streams of events $e_0, e_1, e_2, \dots$, where the data format is much different from the conventional visual processing. Raw event streams are often transformed to

alternative representations. The event data representation is task driven and to facilitate the extraction of useful features.

Event Packets

An event packet is a fragment of events in the event stream. An event packet is often denoted as $E \doteq \{\mathbf{e}_i\}_{k=1}^{N_e}$, where N_e is the number of events in the packet. Events in a packet can be fixed number or from a fixed time interval, which depends on the assumptions in applied algorithms [11–14]. In this kind of representation, the precise timestamp information is preserved and the spatial-temporal neighboring events are processed together to produce valid output.

Event Frames

As shown in Figure 1.1.4, the events in a spatiotemporal neighborhood are converted into an event frame that can be fed to image-based computer vision algorithms. Some algorithms may work despite the different statistics of event frames and natural images. However, this practice is not ideal in the event-based paradigm because it quantizes event timestamps, discards sparsity, and the resulting images are highly sensitive to the number of events used. Nevertheless, the high impact of event frames in the literature [15–19], is clear because

- They are a simple way to convert an unfamiliar event stream into a familiar 2D representation that contains spatial information about scene edges, which are the most informative regions in natural images.
- They inform not only about the presence of events but also about their absence, which is informative.
- They have an intuitive interpretation (e.g., an edge map, a brightness increment image)
- They are the data structure compatible with conventional computer vision.

Time Surfaces

A time surface is a 2D map where each pixel stores a single time value. The time

FIGURE 1.1.4: Events in a spatiotemporal neighborhood are converted into an image. Event count frames are obtained by counting the total number of events generated in each pixel location within a time interval. Event binary frames consider pixel locations generating at least one event as the same value.

value can be the timestamp of the last/first/event or average timestamp of all events at that pixel. Thus events are converted into an image whose “intensity” is a function of the motion history at that location with brighter values corresponding to a more recent motion. The time surfaces are called *Motion History Images* in classical computer vision. They explicitly expose the rich temporal information of the events and can be updated asynchronously. Normalization can be applied to achieve invariance to movement speed. Comparing to other grid-like representations of events, time surface highly compresses information as they only keep one timestamp per pixel, thus their effectiveness degrades on textured scenes, in which pixels spike frequently. To make the time surface less sensitive to noise, each pixel value may be computed by filtering the events in a space-time window.

FIGURE 1.1.5: **Time surfaces.**(a)The time surface in 3D view of a pen moving from left to right. The time surface is constructed by the timestamp of the latest event at each pixel location. (b)The time surface in 2D view of (a). (c)The time surface in 3D view of a hand moving from left to right. (d)The time surface in 2D view of (c).

Neuron Spikes

In neuromorphic based algorithms, i.e, Spiking Neural Networks (SNNs), an event e_i fired at pixel location $\mathbf{x}_i$ is treated as a neuron spike. As shown in Figure 1.1.6, a sequence of events fired at pixel location $\mathbf{x}_i$ is treated as a spike train. In SNN-based approaches, additional information is built up from past events or given by additional knowledge, that is fused with the incoming event asynchronously to produce an output [13, 20, 21].

FIGURE 1.1.6: In SNN-based approach, an individual event is treated as a neuron spike.

Bag of Events

In the concept of Bag of Events [54], event streams are divided into multiple segments and each segment is regarded as a bag. Let $S = \{s_1, s_2, \dots, s_n\}$ be a collection of segments and $E = \{e_1, e_2, \dots, e_m\}$ be a set of distinct events contained in S , where n and m denote the number of segments and the number of pixels, respectively. For each segment s_j , the joint probability distribution of E is used to represent s_j . By assuming the occurrences of the events in segments are statistically independent, s_j can be represented as $[f_{1j}, f_{2j}, \dots, f_{mj}]$, where f_{ij} is the frequency of e_i within s_j . Bag of Events represents the event streams using the joint probability distribution of multiple events instead of lines, corners, or other visual features, which has the advantage of good interpretability in mathematics.

1.1.4 Advantages of Event Cameras

Event cameras offer numerous potential advantages over standard cameras.

High Temporal Resolution

In analog circuitry, monitoring of brightness changes is quick, and the read-out of the events is digital, with a 1MHz clock, i.e. events are detected and microsecond resolution time stamped. Event cameras can also catch very fast movements without suffering from the motion blur typical of frame-based cameras.

Low Latency

In event cameras, each pixel functions independently and there is no need to wait for a global frame exposure time. It is transmitted as soon as the intensity change is identified. Event cameras therefore have low latency of sub-milliseconds in the real world.

High Dynamic Range (HDR)

In particular, the very high dynamic range of event cameras (>120 dB) exceeds the 60dB of high-quality, frame-based cameras, allowing them to collect data from moonlight to daylight. The fact is that the pixel photoreceptors operate on a logarithmic scale and each pixel operates independently, not waiting for a global shutter. DVS pixels will, like biological retinas, respond to very dark as well as very bright stimuli.

1.1.5 Challenges of Event Cameras

Event cameras are gaining more and more attention in recent years due to their benefits mentioned in previous section and their great potential of solving computer vision bottleneck problems. However, event cameras still haven't been widely adopted in computer vision and robotics community. To be more practical, there are a number of technical challenges for event cameras to overcome.

Space-time Entanglement

The nature of event data is asynchronous and space-time entangled, while frame-based data is synchronous and space-time disentangled. By having every pixel report its intensity value at the same time, each image gives us a snapshot in time. Thus, we can separate spatial reasoning (e.g. finding edges in an image) from temporal reasoning (e.g. finding the motion of an object). For event cameras, the two dimensions are interwoven in the event stream. Given an event stream without any prior information, it is impossible to determine whether a new event corresponds to a new point in the image, or an older point that has already generated an event, moving to a new pixel location. This can be represented as a data association problem, where each new event must be associated with either a new point in the image, or an older event.

Synchronous Processing of Asynchronous Data

The advantages of the event cameras mainly come from their asynchronous response to illumination changes in the scenes. Ideally, these advantages can only be utilized if we apply a fully asynchronous algorithm perfectly response to the asynchronous event streams operating on a fully asynchronous processor. While practically, the design of event-based algorithms is influenced by the choice of event data representation and hardware platform available.

An ideal asynchronous algorithm would have some internal state representing of its inference of the scene and updates this inference as every event arrives, which achieves the least processing latency. Existing asynchronous approaches includes event-by-event processing based on probabilistic filters and SNNs. These asynchronous algorithms have successfully showed their capability of extracting information from the events, promising extremely fast reaction times [7]. Intrinsically, asynchronous processing is a sequential task. Event cameras are able to generate up to millions asynchronous events per second. Although we can implement asynchronous algorithms on synchronous processors (e.g. CPUs), it is very hard to achieve asynchronous response according to each coming event. It requires the synchronous processor to be able to produce corresponding output faster than the

maximum event generation rate. However, most modern computing systems were designed to operate in synchronous manners (e.g. CPUs) and can be accelerated by parallel batch processing (e.g. GPUs). Unfortunately, these asynchronous algorithms can not be utilized using parallel batch processing.

In recent years, a new generation of commercial neuromorphic processors has arisen as a possible solution to this problem, such as the SpiNNaker [22, 23], the IBM TrueNorth [24] and the Intel Loihi [25]. Such processors are asynchronous by nature and have the ability to enable asynchronous processing directly from the camera in real-time. But it will take at least another decade for neuromorphic computing to become mainstream. The challenge we are facing so far is to develop algorithms which take advantage of the asynchronous nature of event data, while also utilizing the benefits provided by current parallel computing.

Photometric Loss

One of the fundamental principles behind modern computer vision was the presumption of photoconsistency. In this presumption, a point in the world should reflect the same intensity, regardless of the point of view. This presumption helps one to apply a photometric loss to solve data correlations between points in different images. Unfortunately, events no longer contain any intensity information explicitly, though some studies have shown that a grayscale picture can be reproduced for a given set of events. Because each event reflects just a difference in the signal, variations between very low intensities and very high intensities will produce the same events. The lack of photometric loss is a major barrier to the direct transition of conventional computer vision techniques to events.

Noise and Dynamic Effect

All vision sensors are noisy because of the inherent shot noise in photons and from transistor circuit noise, and they also have nonidealities. This situation is especially true for event cameras, where the process of quantizing temporal contrast is complex and has not been completely characterized. From another aspect, as shown in Figure 1.1.7, the noise removal of the event data in spatial-temporal

domain has very high computation cost and efficient noise removal algorithms need to be explored.

FIGURE 1.1.7: Denoising of events has very high computation cost.

Lack of Training Data

This is less of an intrinsic property that is unique to the camera itself, but more a standard problem with any new sensor. As event cameras are less than 15 years old, there are obviously not many event data sets. Good data is potentially one of the most critical aspects of building a state-of-the-art learning infrastructure for the latest surge of deep learning in computer vision.

1.2 Objectives of This Study

The objective of this research is to utilize the advantages of event cameras to gesture recognition tasks from a completely new perspective. This thesis aims to:

- develop an efficient representation of event data, which fully utilizes the sparse and asynchronous features of event data.
- develop efficient algorithms to analyze the event data and solve the challenges of event-based processing (e.g. space-time entanglement of event data, synchronous processing of asynchronous event data, noise and dynamic effect).

- develop a portable and fully event-based Human-Computer Interaction system for real-time applications, which is light-weighted for edge-computation with low latency.

1.3 Major Contributions of This Study

There are three major contributions from this PhD study.

1) For the first contribution, we successfully design an efficient representation of event data, which fully utilizes the sparse and asynchronous features of event data:

- The concept of Space-time Event Cloud is proposed to enable synchronous event processing without space-time disentanglement.
- The PointNet is leveraged to verify that the spatiotemporal features can be extracted simultaneously from the geometric distribution of space-time event clouds.
- The effectiveness of proposed subspace feature sampling-based classification against the noisy space-time event clouds is evaluated and presented.

2) In the second contribution, we successfully develop an efficient algorithms to analyze the event data and solving the challenges of event-based processing (e.g. space-time entanglement of event data, synchronous processing of asynchronous event data, noise and dynamic effect).

- We re-define the Space-time Event Cloud as a sequence of temporally ordered space-time events rather than a set of unordered space-time events.
- The Space-time Event Net with hierarchical space-time kernels is proposed to analyze Space-time Event Clouds with great efficiency.

- A two-channel architecture to deal with single-hand and dual-hand gestures under a unified system is developed. This architecture achieves state-of-the-art performance on event-based gesture recognition tasks.

3) In the third contribution of the research work, we successfully develop a portable and fully event-based Human-Computer Interaction system for real-time applications, which is light-weighted for edge-computation with low latency.

- A fully event-driven gesture recognition system using a single event camera as sensory input has been presented.
- The proposed system is a detector-tracker-classifier full-stack solution, which is extensible to many types of interaction, such as menu selection, text entry and gaming.
- The complete system has been implemented and tested on an embedded platform with an ARM Cortex-A8 CPU and achieves a throughput of 100Hz.

1.4 Organization of the Thesis

The thesis is organized as follows.

Chapter 1 introduces the background knowledge and the motivation of this thesis. First, the prospect of neuromorphic sensing is introduced. This thesis focuses on algorithm development of event cameras, which is one of the emerging neuromorphic sensors. Second, the technical background of event cameras is introduced, including the working principles of event cameras, the representation of event data, as well as the advantages and challenges of event cameras. Third, the objectives of the thesis and major contributions are clearly stated. Finally, the outline of the thesis is provided.

Chapter 2 reviews existing event-based processing algorithms based on different representations of event data. In Section 3.2, CNN-based synchronous processing

methods by converting event streams to event frames are reviewed. In Section 3.3, SNN-based processing methods by converting event streams to neuron spikes are reviewed. In Section 3.4, EventNet with recurrent processing of sparse event sequence is reviewed.

Chapter 3 illustrates the proposed Space-time Event Cloud concept. The proposed subspace feature sampling-based classification to deal with the noisy space-time event clouds is presented. The adoption of PointNet to learn the geometric distribution of space-time event clouds is described. In the experiment part, we evaluate the classification ability of the proposed classifier against different classifiers and the effectiveness of the proposed subsampling scheme.

Chapter 4 describes the proposed Space-time Event Net architecture. First, we redefine the Space-time Event Cloud concept as a sequence of temporally ordered space-time events rather than a set of unordered space-time events. Second, the Space-time Event Net architecture with hierarchical space-time kernels is illustrated. In the experiment part, the effectiveness of modeling event data as temporally ordered space-time clouds on different datasets is demonstrated. The significant performance improvement when applying the dual-channel architecture of ST-EVNet on gesture recognition tasks is presented.

Chapter 5 introduces the proposed event-driven gesture recognition system. First, system considerations are addressed. Second, the proposed event-based hand detector, the event-based hand tracker, and the event-based gesture classifier are described. In the experiment part, the robustness of individual modules and the entire system efficiency is evaluated.

Chapter 6 concludes the thesis by summarizing our contributions and provides some discussion on recommendations for future works.

Chapter 2

Literature Review

2.1 Overview of Event-based Processing Using ANNs

An event-based processing system mainly contains three stages: preprocessing of input event streams, major processing of event-based algorithms and post-processing of output. As introduced in Chapter 1, the methods used to process event data largely depend on the representation of event streams and the computation platforms. It is intuitive to use dense representations by constructing event frames and design algorithms that can be efficiently executed on standard processors (e.g., CPUs or GPUs). From another aspect, it is also natural to treat event streams as spike trains and process events using SNNs. Neuromorphic hardware is required in this approach in search for more efficient and low-latency solutions. Recently, new neural network architectures are proposed to make the best of the sparseness of the event data. These approaches are able to be processed efficiently on standard processors without converting sparse event data to dense frames. In Section 3.2, CNN-based synchronous processing methods by converting event streams to event frames are reviewed. In Section 3.3, SNN-based processing methods by converting event streams to neuron spikes are reviewed. In Section 3.4, we review two

new neural network architectures, which aim to preserve the sparsity of events and process event data as it is.

2.2 Processing Events as Event Frames

Considering the asynchronous nature of the events, it is to be determined what representation of the events should be used in the standard convolutional neural network architecture. CNN-based architectures expect array-like dense inputs, with a fixed, relatively low, number of channels and spatial correlations between neighboring pixels. Therefore, a good representation is key to fully take advantage of existing networks while summarizing the necessary information from the event stream.

2.2.1 The Four-Channel Event Image

Zhu *et al.* [1] proposed to represent event streams as a four-channel image. The first two channels are the positive event count image and negative event count image, encoding the number of positive and negative events fired at each pixel respectively. However, constructing frames with only event count loses the motion information encoding in the precise timestamps. It is not straightforward to encode timestamps in image form. Constructing k channels to stack all incoming timestamps, where k is the maximum number of events that can be accumulated in the frame, is one potential solution. However, this approach will cause the dimensionality of the input to increase greatly. Instead, Zhu *et al.* [1] proposed to encode the pixels in the last two channels as the timestamp of the latest positive and negative event at that pixel respectively. From Figure 2.2.1, it can be observed that the motion flow is evident by following the gradient in the image, especially for faster moving objects. A potential problem of this representation is that areas with very dense events and large motion will have all pixels overridden by very recent events with very similar timestamps. Constructing timestamp frames with small time interval

will solve this problem. As a result, magnitude of the motion is reduced. Before feeding into the network, the positive and negative timestamp images area are normalized to $[0, 1]$, ensuring that fast motions with a small time window and slow motions with a large time window that generate similar displacements have similar inputs to the network.

In [1], the four-channel event image is fed into the EV-FlowNet for optical flow estimation. The architecture of EV-FlowNet is shown in Figure 2.2.2, which very closely resembles the encoder-decoder networks like the U-Net. The $\tanh$ function is used as the activation function. The total loss is defined as the weighted sum of the photometric and smoothness losses. In order to minimize the difference in amplitude between the distorted second image and the first image, the photometric loss is used while the smoothness loss is used to regularize the display flow by minimizing the flow gap between adjacent pixels. In [2], the four-channel event image is fed into the EV-Gait network for gait recognition. The network architecture is shown in Figure 2.2.3.

FIGURE 2.2.1: Event timestamp images

2.2.2 The Spatio-Temporal Voxel Grid Based Event Frame

Rebecq *et al.* [3] proposed to convert event stream E_k into a fixed size tensor representation $\mathbf{F}_k$, where the events are encoded in a spatio-temporal voxel grid shown in Figure 2.2.4(a). The duration $\Delta T = t_{N-1}^k - t_0^k$ spanned by the events in E_k is discretized into B temporal bins. Every event distributes its polarity p_i to

FIGURE 2.2.2: The network architecture of EV-FlowNet proposed in [1]. In the forward path, the four-channel event image is passed through four strided convolution layers, with output channels doubling each time. The resulting activations are passed through two residual blocks, and then four upsample convolution layers, where the activations are upsampled using nearest neighbor resampling and then convolved, to obtain a final flow estimate.

FIGURE 2.2.3: The workflow and network architecture of EVGait proposed in [2]. The network is divided into two major components: convolutional layers with Residual Block layers are responsible for feature extraction and fully-connected layers with softmax associate the features to different identities.

the two closest spatio-temporal voxels as follows:

$$\mathbf{F}(x_l, y_m, t_n) = \sum_{x_i=x_l, y_i=y_m} p_i \max(0, 1 - |t_n - t_i^*|) \quad (2.2.1)$$

The network proposed in [3] to learn to reconstruct intensity images from event streams directly is shown in Figure 2.2.4(b), which is a recurrent, fully convolutional network inspired by the U-Net [26] architecture. A perceptual loss was introduced to encourage reconstructions to follow natural image statistics.

FIGURE 2.2.4: (a)The spatio-temporal voxel grid based event frames proposed in [3] is illustrated. The event stream (depicted as red/blue dots on the time axis) is split into windows E_k containing multiple events. Each window is converted into an event tensor $\mathbf{F}_k$ and passed through the network. (b)The network proposed in [3] consuming the spatio-temporal voxel grid event frames for gray-scale image reconstruction.

2.2.3 The SBT Frames and SBE Frames

Wang *et al.* [4] proposed to use the combination of Stacking Based on Time (SBT) images and Stacking Based on the number of Events (SBE) images. In SBT, the event stream within a time window is divided equally into n portions to construct n frames. The n frames, $\mathbf{F}_p^i(x, y), i = 1, 2, \dots, n$ are formed by merging the events in each time interval $\left[\frac{(i-1)\Delta t}{n}, \frac{i\Delta t}{n}\right]$. $\mathbf{F}_p^i(x, y)$ is the sum of polarity p values at (x, y) . These n stack $\mathbf{F}_p(x, y, i) = \mathbf{F}_p^i(x, y)$. However, SBT loses the precise time information of events within the target time window and larger n can keep more temporal information. In SBE, a frame is formed by merging the events based on the number of incoming events as shown in Figure 2.2.5(a). The first N events are merged into frame 1 and the next N events into frame 2, and this is repeated up to frame n to create one stack of n frames. Depending on the N value, this approach guarantees rich event data to reconstruct images. $\mathbf{F}_E, \mathbf{F}_F, \mathbf{F}_G$, and $\mathbf{F}_H$ shown in Figure 2.2.5(a)(right) are frames corresponding to different numbers of events, $N, 2N, 3N, 4N$, respectively.

It is proved that both SBT frames and SBE frames are robust enough to reproduce the grayscale images based on the spatiotemporal intensity changes using the network architecture shown in Figure 2.2.5(b). The generator is a U-net [26] like architecture with skip connections. The discriminator network is adopted from PatchGAN [27] and takes the original gray-scale image and the image generated by the generator from events. The discriminator first concatenates the condition of feature maps from the last layer of the generator and discriminates whether the generated image respects the condition of domain transfer from event to intensity.

2.3 Processing Events as Neuron Spikes

Different from conventional CNNs, SNNs are more biologically plausible since it incorporates spike times into computational models which mimic the information processing in the brain [28]. CNNs process dense data, thus the sparse events have

2.3.1 Event-based Classification Using SNNs

Artificial neurons, such as Leaky-Integrate and Fire or Adaptive Exponential, are neuron-inspired computational models found in the visual cortex of mammals. They are the fundamental building blocks of artificial SNNs. A neuron receives input spikes ('events') from a receptive field, which, when the state reaches a threshold, modifies its internal state (membrane potential) and produces an output spike (action potential). In a hierarchical way, neurons are connected, creating an SNN. Spikes can be either generated by pixels of the event camera or by SNN neurons. Information travels along the hierarchy, from the event camera pixels to the first layers of the SNN, and then through to higher (deeper) layers. The receptive fields become increasingly more complex as the information travels deeper into the network. The work carried out by internal layers in CNNs is approximated as a convolution. In artificial SNNs, one typical method is to assume that if it has not received any input spikes from the previous SNN layer, a neuron would not create any output spikes.

Spiking Neuron Model

Each event is considered as a neuron spike and event streams are treated as spike trains. A spike train input to a spiking neuron is denoted as

$$s_k(t) = \sum_f \delta(t - t_k^i) \quad (2.3.1)$$

where t_k^i is timestamp of the i^{th} spike (event) of k th input. In a SNN neuron, each incoming spike is converted into a spike response signal $a_k(t)$ by convolving $s_k(t)$ with a spike response kernel $\varepsilon(\cdot)$

$$a_k(t) = (\varepsilon * s_k)(t) \quad (2.3.2)$$

The $v(\cdot)$ is the refractory kernel and refractory response of a neuron is represented as $(v * s)(t)$. A SNN neuron receiving spike trains from multiple inputs. The weighted sum of the input spike response signals produces the internal state of a

SNN neuron (membrane potential) $u(t)$ as

$$u(t) = \sum_k \omega_k (\varepsilon * s_k)(t) + (\nu * s)(t) = \omega^T a(t) + (\nu * s)(t) \quad (2.3.3)$$

An output spike is generated whenever $u(t)$ reaches a predefined threshold ϑ . The spike function f is defined as

$$f_s(u) : u \rightarrow s, s(t) := s(t) + \delta(t - t^{i+1}) \quad (2.3.4)$$

where $t^{i+1} = \min(t : u(t) = \vartheta, t > t^i)$.

SNN Model

A general formulation to described a feedforward spiking neural network architecture is provided as follows. It can be used to describe fully connected layers, convolutional layers, as well as pooling layers. Consider a layer l with N_l neurons and weights $\mathbf{W}^{(l)}$, the network forward propagation is described as

$$a^{(l)}(t) = (\varepsilon_d * s^{(l)})(t) \quad (2.3.5)$$

$$u^{(l+1)}(t) = \mathbf{W}^{(l)} a^{(l)}(t) + (\nu * s^{(l+1)})(t) \quad (2.3.6)$$

$$s^{(l+1)}(t) = f_s(u^{(l+1)}(t)) \quad (2.3.7)$$

Performace

The performance of SNNs with different configurations on different event-based datasets are shown in Table 2.3.1. By converting events to neuron spikes and using SNNs to process event data is capable to achieve supreme results. However, the spiking-based computation is only efficient on spiking-based neuromorphic processors.

TABLE 2.3.1: Benchmark Classification Results on N-MNIST and DVS128 Gestures Datasets

Dataset	Method	Architecture	Accuracy
N-MNIST	Lee [34]	34x34x2-800-10	98.66%
	SKIM [35]	34x34x2-10000-10	92.87%
	DART [36]	DART feature descriptor	97.95%
	SLAYER	34x34x2-500-500-10	98.89 $\pm$ 0.06%
	SLAYER	34x34x2-12c5-2a-64c5-2a-10o	99.20 $\pm$ 0.02%
DVS128 Gestures	TrueNorth [33]	SNN (16 layers)	91.77%
	SLAYER	SNN (8 layers)	93.64 $\pm$ 0.49%

2.3.2 Processing Event Data on Neuromorphic Processors

Amir *et al.* [33] presented the first event-based hand gesture recognition (hand detection and tracking are not considered) system processing on a neuromorphic chip - TrueNorth. The TrueNorth [24] is an asynchronous, low-power, event-based neuromorphic processor which achieves significant power and speed advantages over mobile GPUs. The system setup is shown in Figure 2.3.1(a).

FIGURE 2.3.1: (a)System setup. The DVS128 camera is connected to an NS1e development board using a USB cable.(b)TrueNorth architecture. Neurosynaptic cores connect inputs to neurons using a synaptic crossbar. An asynchronous communication fabric routes events between cores.

The TrueNorth chip

The TrueNorth chip from IBM is a reconfigurable, non-von Neumann processor. It contains 1 million spiking neurons and 256 million synapses distributed across

4096 parallel, event-driven, neurosynaptic core [24]. The 4096 cores are tiled in a 64×64 array, embedded in a fully asynchronous network-on-chip. The chip consumes only 70mW under normal workloads with a 1ms computation tick. Each neurosynaptic core has 256 inputs and the 256 inputs are connected to 256 neurons using a crossbar of 256×256 synapses with 2 bits of weight precision. The membrane potential is implemented as a neuron state variable, integrating synaptically weighted input events with an optional leak decay. Neurons can be configured to generate an output event deterministically, whenever the membrane potential exceeds a threshold. Alternatively, neurons can also be configured to generate an output event stochastically, with a pseudorandom probability related to the difference between the membrane potential and its threshold.

Via the synaptic crossbar, the output events of a certain neuron within one core can be delivered to any or more of the core's neurons. As seen in Figure 2.3.1(b), before releasing them into the crossbar, every core input has a delay buffer and can cache incoming events for up to 15 ticks. Algorithms enforced on TrueNorth must fulfill the architecture's constraints. Neurons contain only binary events, not constant values; synapses have low precision, not high precision; core-to-core, not all-to-all, connectivity. In the Corelet Programming Language, a hierarchical, compositional, object-oriented language implemented in MATLAB, TrueNorth programs are written.

Performance

In [33], a CNN-like architecture is mapped to TrueNorth using the Energy-efficient deep networks (Eedn) algorithm. The Eedn [37] algorithm satisfies the TrueNorth architecture constraints by restricting network precision to binary neuron output and trinary weights $\{-1, 0, 1\}$, and by limiting neuron fan-in and fan-out. Instead of the standard rectified linear unit (ReLU), Eedn uses a binary step function as the activation function and it can be implemented by TrueNorth neurons operating as threshold logic units. The derivative of a step function is approximated with a triangle function during backpropagation. Trinary weights used during the forward and backpropagation passes are updated by rounding with hysteresis their

corresponding shadow weights. The system performance with classification accuracy and power consumption is shown in Table 2.3.2. By integrating a DVS camera with a TrueNorth processor, the end-to-end event-based gesture recognition system operates at 1000 Hz with 96.49% accuracy on learned gestures with a low power consumption of 178.8mW.

TABLE 2.3.2: The event-based gesture recognition system performance on Truenorth Chip.

Experiment	Input $W \times H$	Cascade filters	Filter duration (ticks)	Augment (pixels)	CNN train acc.	CNN test acc.	System acc. 11 cat.	System acc. 10 cat.	Neuro- synaptic cores	TN leak mW	TN active mW	TN total mW
E1	64×64	6	32	8	93.06	91.77	94.59	96.49	3838	134.3	44.5	178.8
E2	64×64	6	32	0	98.53	90.29	92.23	93.53	3838	134.3	44.5	178.8
E3	32×32	6	32	4	86.16	88.07	90.78	92.59	3459	122.5	46.5	169.0
E4	64×64	4	32	8	89.98	90.69	93.44	95.25	3625	127.0	45.0	172.1
E5	42×42	6	32	5	90.01	84.59	90.59	91.46	3968	140.2	55.6	195.8
E6	64×64	6	64	8	94.58	88.29	90.57	92.17	3951	138.7	44.2	182.9
E7	42×42	6	64	5	85.16	83.74	86.49	87.62	1936	68.8	19.7	88.5
E8	64×64	6	85	5	92.95	88.49	90.76	92.24	3951	139.0	37.2	176.2
E9	64×64	6	128	8	94.55	91.10	93.15	94.81	3951	142.8	49.7	192.5

2.4 Processing Event Data as It Is

In this section, we focus on analyzing the Residual-Graph Convolutional Neural Networks (RG-CNN) [6] and the EventNet [7]. The philosophy presented in these two works is similar in spirit to our work. Both works believe in preserving the sparseness of event data as much as possible when developing the processing methods, as sparsity is one of the biggest advantages of event data. However, the two works are quite different in the final implementation.

2.4.1 RG-CNN: Graph-based Spatial-temporal Feature Learning

2.4.1.1 The RG-CNN Architecture

Non-Uniform Grid Sampling

Firstly, non-uniform grid sampling is applied to sample a subset of N representative events $\{\mathbf{e}_i\}_N \in \{\mathbf{e}_i\}_M$, where $N \ll M$. Only one event is randomly sampled within a defined space-time volume.

Constructing the Graph

The downsampled events $\{\mathbf{e}_i\}_N$ are mapped on directed graph $G = \{v, \epsilon, U\}$. The v represents the set of vertices and the ϵ represents the set of edges. U is the coordinates of the nodes that locally define the spatial relations of the nodes. Each event $\mathbf{e}_i$ is a node $v_i : x_i, y_i, t_i$ in the graph, such that $v_i \in v$. The connectivity of nodes in the graph is defined based on the radius-neighborhood-graph strategy. The distance $d_{i,j}$ between $\mathbf{e}_i$ and $\mathbf{e}_j$ is defined as the weighted spatio-temporal distance:

$$d_{i,j} = \sqrt{\alpha (|x_i - x_j|^2 + |y_i - y_j|^2) + \beta |t_i - t_j|^2} \quad (2.4.1)$$

The nodes v_i and v_j are connected with an edge only if their weighted $d_{i,j}$ is less than radius distance R . The node i with the connected node j is defined as

$$u(i, j) = [|x_i - x_j|, |y_i - y_j|] \in U \quad (2.4.2)$$

The architecture of RG-CNN is shown in Figure 2.4.1. The parameter S is the number of graphs constructed from one sample. For tasks like object classification, typically only requires a short temporal extent and S is set to 1. A time window T is randomly trimmed from the event streams to construct the graph. For action

recognition, a time window T from the event stream is equally divided into S volumes. where events within each volume are downsampled to conduct a component graph G_k . As a result, a set of graphs $\{G_k\}_{k=1}^S$ is obtained.

FIGURE 2.4.1: **The RG-CNN Architecture.** Firstly, S graphs are constructed from the event stream. Each graph is passed through the spatial feature learning module. For object classification, the output of the spatial feature learning module is directly sent to fully connected layers. For action recognition, the coarse temporal dependencies are modeled over multiple graphs by converting to a grid representation via the Graph2Grid module and perform temporal feature learning with a conventional 3D-CNN.

For motion-based tasks, the constructed graphs $\{G_k\}_{k=1}^S$ are first fed into the graph convolution-based spatial feature learning module to learn appearance-based features. The output spatial feature learning map for the l th output feature map of the i th node is denoted as $f_l^{spatial}(i)$. To deal with motion-based tasks, a Graph2Grid module is introduced to transform the learned graphs from our spatial feature learning module to a grid representation and performs stacking over the temporal dimension.

2.4.1.2 Performance

The quantitative evaluation of RG-CNN on the DVS128G gestures dataset compared with recent state-of-the-art methods are shown in Table 2.4.1. In the experiments for all recent methods, event recording durations are set to 0.25 and

TABLE 2.4.1: Quantitative evaluation of RG-CNN on the DVS128 Gestures Dataset [6].

Method	Duration(0.25s)	Duration(0.5s)
LSTM [38]	0.882	0.865
PointNet [5]	0.887	0.902
PointNet++ [5]	0.923	0.941
ResNet34 [39]	0.943	0.955
I3D [40]	0.951	0.965
RG-CNN + Plain 3D	0.954	0.968
RG-CNN + Incep. 3D	0.957	0.968
RG-CNN + Res. 3D	0.961	0.972

TABLE 2.4.2: Computational complexity of RG-CNN [6].

Method	GFLOPs	Parameters($\times 10^6$)
PointNet [5]	0.079	0.55
PointNet++ [5]	0.542	1.47
ResNet34 [39]	11.64	63.70
I3D [40]	30.11	12.37
RG-CNN + Plain 3D	12.46	6.95
RG-CNN + Incep. 3D	12.39	3.86
RG-CNN + Res. 3D	13.72	12.43

0.5 seconds. According to Table 2.4.1, LSTM-based methods [38] is to be outperformed by others, considering that LSTM processes event streams as pure temporal sequences. It only learns the temporal features from events while the spatial interactions are not considered. PointNet-based methods shown in Table 2.4.1 are presented in Chapter 3 of this thesis. The PointNet-based methods, which consider inputs as 3D event clouds to learn their geometric features, are more accurate. With regards to reference networks, although I3D [40] and 3D ResNet-34 [39] perform spatiotemporal feature learning, there is no explicit modeling of event dependencies as events are directly grouped into frames. The RG-CNN [6] achieves comparable results with the ST-EVNet proposed in Chapter 4 on this dataset.

2.4.2 EventNet: Asynchronous Recursive Event Processing

The EventNet Architecture

To process event streams recursively, Sekikawa's work [7] modeled this problem as

$$y_j = f(E_j) \quad (2.4.3)$$

where y_i is the target output given an event stream $E_j = \{\mathbf{e}_i, \mathbf{e}_{i+1}, \dots, \mathbf{e}_j\}$ and the function f is approximated by neural networks to estimate y_j . It is worth mentioning that the events are generated as non-uniform rates and it varies from hundreds to millions per second. Thus, the target network is required to be capable of absorbing variable-rate event data in an event-wise manner. The EventNet proposed in [7] is designed to achieve the following goals:

- *Event-Wise Processing*

Considering that events' sparseness in the spatial domain and non-uniformity in the temporal domain, the input signal E_j needs to be analyzed in an event-wise manner without converting to dense format to avoid unnecessary computation.

- *Recursive Processing*

The event rate is non-uniform rates and varies from hundreds to millions per second. Thus, the number of events in E_j could be tens of thousands depending on the event rate and sliding window T . It is counter-intuitive to process such a large variable-length non-uniform event-stream E_j in a batch manner at a very high event rate. Thus, an efficient algorithm that can recursively process the stream is required.

- *Local Permutation Invariance*

On the one hand, the function f is required to be invariable against events permutations in a very short time window in microseconds). On the other

hand, the function f needs to be able to capture behaviors of event sequence in a longer time window in milliseconds. The temporal order of event data generated in a short time window can be generally thought of as stochastic due to the limited temporal resolution and the noise of timestamps. The order of incoming events varies even if the same scene is observed with the same camera motion. Thus, it is essential for the function f to be invariance against a temporally local permutation of events. However, the long-time temporal behavior of an event sequence E_j needs to be modeled to capture motion in a scene.

Symmetric Function to Cope with Permutation

EventNet co-opted the idea of PointNet to cope with short-term events permutation. The PointNet architecture [41] is designed to analyze the unordered set of 3D points. One of the key features of PointNet is that the network architecture is totally invariant to input points permutations. The architecture of PointNet is shown in Figure 2.4.2(a), point-wise features are first learned by multilayer perceptron (MLPs) and a symmetric function $\max$ is introduced to ensure that the output does not change under permutation of Points. PointNet architecture demonstrates the universal approximation ability of continuous set functions. A Hausdorff continuous symmetric function $f: 2^x \rightarrow \mathbb{R}$ can be arbitrarily approximated by PointNet as

$$|f(s) - \gamma(\max_{p_i \in s} \{h(p_i)\})| < \epsilon \quad (2.4.4)$$

where $p_1, \dots, p_n$ are 3D points in s ordered arbitrarily, h and γ are continuous functions approximating multi-layer perception networks, and $\max$ is an element-wise max pooling operator.

PointNet originally processes 3D point clouds in a batch manner. Modifications have to be made to process event streams in a recursive manner. When a new event $\mathbf{e}_{j+1}$ arrives, the events within the time window change from $E_j = \{\mathbf{e}_i, \mathbf{e}_{i+1}, \dots, \mathbf{e}_j\}$ to $E_{j+1} = \{\mathbf{e}_{i+1}, \mathbf{e}_{i+2}, \dots, \mathbf{e}_{j+1}\}$. Comparing $f(E_j)$ with $f(E_{j+1})$, most of the events

are overlap and have already been processed by h . Also, we need to compute $\max$ within all high-dimensional feature vectors learned by h . A single cycle of these computations are themselves intensive for the number of events in E_j may be thousands or tens of thousands with a time window. Furthermore, these two computations should run on the event rate, which could be more than 1 mega events per second (MEPS). These issues make it impossible to use PointNet to process event streams in real time.

FIGURE 2.4.2: **The comparison of EventNet with PointNet.** EventNet has a novel temporal coding layer to compute the sequence of events in a recursive manner. To improve computation efficiency at inference, the most computationally significant parts, mlp1 and mlp2, are realized using a lookup table (LUT) based on the trained parameters, which is significantly faster than an MLP. As a result, EventNet processes streams of events efficiently in an event-driven manner. It computes per-event feature by a LUT, applies the temporal code to the global feature, and aggregate global feature by max-pooling of the two vectors, which is repeated recursively as it receives a new event. Similar to PointNet, EventNet has a variant of architecture that can output on a per-event basis, which is realized with mlp4 by a concatenation of local feature and global feature (blue line).

Temporal Coding

To overcome the above-mentioned difficulties in efficient processing of event data, [7] proposed a temporal coding scheme to process the sparse event signal recursively rather than processing large numbers of events in a batch manner. $f(E_j)$ is

originally computed as:

$$f(E_j) \approx \gamma(\max\{h(\mathbf{e}_i^-, \Delta t_{j,i}), h(\mathbf{e}_{i+1}^-, \Delta t_{j,i+1}), \dots, h(\mathbf{e}_j^-, 0)\}) \quad (2.4.5)$$

where $\mathbf{e}_i^- \doteq \{x_i, y_i, p_i\}$. And $f(E_{j+1})$ is computed as:

$$f(E_{j+1}) \approx \gamma(\max\{h(\mathbf{e}_{i+1}^-, \Delta t_{j+1,i+1}), h(\mathbf{e}_{i+2}^-, \Delta t_{j+1,i+2}), \dots, h(\mathbf{e}_{j+1}^-, 0)\}) \quad (2.4.6)$$

To avoid the multiple computations of h for the same events, a temporal coding function g is introduced to remove the dependence on Δt from h and encode the information of Δt as

$$h(\mathbf{e}_i) = g(z_i, \Delta t_{j,i}) \quad (2.4.7)$$

where $z_i = h(\mathbf{e}_i^-)$. Then Equation 2.4.3 now becomes

$$f(E_j) \approx \gamma(\max\{g(z_i, \Delta t_{j,i}), \dots, g(z_j, 0)\}) \quad (2.4.8)$$

According to this formulation, the h is only needed to compute once for each observed event. So far function g and $\max$ still need to be computed for all events in the time window every time a new event arrives.

Recursive Processing

To process $\mathbf{e}_j$ and $\mathbf{e}_{j+1}$ sequentially, the computation of g and $\max$ need to be recursive. To achieve the recursive computation of g and $\max$, the temporal coding function g is defined as

$$g(z_i, \Delta t_{j,i}) = \left[|z_i| - \frac{\Delta t_{j,i}}{\tau} \right]^+ \exp \left(-i \frac{2\pi \Delta t_{j,i}}{\tau} \right) \quad (2.4.9)$$

The first term decays the input linearly to the elapsed time within the fixed time window. The second term encodes the temporal information by complex rotation. When a new event $\mathbf{e}_{j+1}$ arrives, complex rotation of each feature represents elapsed time by sequentially rotating the element by $\frac{2\pi \delta t_{j+1}}{\tau}$, where $\delta t_{j+1} := t_{j+1} - t_j$. As a

result, the recursive processing of g and max is archived as

$$max\{g(z_{i+1}, \Delta t_{j+1,i+1}), \dots, g(z_{j+1}, 0)\} = max(g(s_j, \delta t_j), z_{j+1}) \quad (2.4.10)$$

where s_j is the result of max when event $\mathbf{e}_j$ comes. The final recursive implementation of function f is written as

$$f(E_{j+1}) = \gamma(max\{g(s_j, \delta t_j), h(\mathbf{e}_{j+1}^-)\}) \quad (2.4.11)$$

where s_j is the global feature at time t_j and s_{j+1} is the global feature at time t_{j+1} . s_{j+1} is updated recursively by $max\{g(s_j, \delta t_j), h(\mathbf{e}_{j+1}^-)\}$. As a result,

- $h(\mathbf{e}_j^-)$ is calculated once when the event $\mathbf{e}_j$ arrives and decays correspondingly when consecutive events arrive.
- g is computed only for s , not for all feature vectors from mlp2.
- max is computed between $g(s_j, \delta t_j)$ and $h(\mathbf{e}_{j+1}^-)$ instead of computing among all feature vectors from mlp2.

Asymmetric Training/Inference

The network structures of EventNet in both training phase and inference are shown in Figure 2.4.2(b). In theory, the recurrent structure of EventNet does not allow for the computation of batch statistics but can be trained recursively without batch normalization. In this approach, batch structure during the training phase in order to perform batch normalization during training and to parallelize the computation of MLPs for efficiency. Thus, the structure of EventNet in the training phase is similar to PointNet [41] except for the temporal coding layer. In the inference phase, there is a huge structural change. On the one hand, the mlp1 and mlp2 are realized as a lookup table (LUT), enabling a significant increase in computation efficiency. On the other hand, the temporal coding is only applied to the previous global feature s_j .

TABLE 2.4.3: Quantitative evaluation of EventNet [7].

	ETHTED+			MVSEC	Real-time processing at 1 MEPS
	Semantic segmentation		Object-motion	Ego-motion	
	GA [%]	mIoU [%]	error [pix/ τ]	error [deg/sec]	
PointNet	98.9	97.4(0.13)	3.14(0.08)	4.55	NO
EventNet	99.2	97.5(0.22)	3.11(0.28)	4.29	YES
Ablation	w/o TD	99.4	98.8 (0.16)	—	NO
	w/o TR	98.1	97.9(0.11)		YES
	w/o ALL	98.3	97.1(0.25)		NO

TABLE 2.4.4: Computational complexity of EventNet [7].

	#input mlp1	#input max	mlp1/2	max pool(+t-code)	total
PointNet	$n(j)$	$n(j)$	936.9×10^3	16.47×10^3	953.3×10^3
EventNet	1	2	0.65 (29.27)	0.36	1.01

2.4.2.1 Performance

EventNet is able to achieve comparable performance to PointNet while the computation complexity is much lower. Table 2.4.3 presents the quantitative evaluations of EventNet and PointNet in both ETHTED+ [42] and MVSEC [43] datasets. For the ETHTED+ dataset, EventNet achieved a global accuracy (GA) of 99.2% and a mean intersection of union (mIoU) of 97.5% for semantic segmentation. For the MVSEC dataset, EventNet achieved an error of 4.29 degree/second in ego-motion estimation. For both datasets, EventNet achieves comparable performance to PointNet. Table 2.4.4 compared the computation complexity of EventNet with PointNet. The computational times (μs) is provided for processing a single event for both EventNet and PointNet. By realizing the mlp1 and mlp2 using a LUT in inference, EventNet achieves much higher computation efficiency compared with PointNet, which is able to process events at 1 MEPS in real-time.

To achieve efficient recursive processing on a large number of raw events, EventNet replaces the event-wise feature extraction containing a limited combination of x and y by a LUT mechanism in inference. Although events are generated asynchronously at the pixel level, raw events are packaged for efficient transmission according to

standard protocols. The program does not receive an event immediately when it generates. What the program actually receives is a bundle of events each time. Thus, efficient batch processing of events is a more practical implementation on conventional processors(e.g. CPUs). EventNet extracts 2D features and decays the features according to time. It separates the spatial and temporal information. For ego-motion estimation of cameras, it is intuitive to learn the 2D distribution of events on the image plane and decay it temporally to regress the camera pose at each timestamp. However, in action and gesture recognition tasks, considering the spatial and temporal information as a whole is a better description of a gesture or an action with deformation during motion. Thus, considering timestamp as a feature dimension identical to (x, y) and extracting local features among the spatiotemporal distribution of space-time event clouds are significant for event-based action recognition tasks.

Chapter 3

Solving Space-time Entanglements with Space-time Event Clouds

3.1 Introduction

Hand gestures are widely used in human-machine interaction [44–48], sign-language recognition [49] and gaming [50, 51]. As a result, gesture recognition becomes a hot topic in recent years. There are two main challenges in conventional RGB cameras systems to conquer gesture recognition tasks. The first challenge is the localization and tracking of hand gestures from the surrounding environment against different backgrounds and light conditions [52]. The other one is to capture hands with greater motion clarity against rapid gesture movements [53] for spatial feature extraction and temporal feature association. Depth cameras are introduced to help segment hands from the background. A straightforward solution to capture the exact and perfect hand movements is to increase the frame rate. Optical flow is often introduced to assist temporal feature matching. As a result, multimodal and multimodality are essential to provide reliable gesture recognition results. However, these frameworks are computationally expensive to be geared up in resource-constrained real-time HCI applications.

Event cameras have shown great potential to solve gesture recognition tasks in a new perspective. As shown in Figure 3.1.1, different from the frame-based vision system, the event-based vision system is a motion-focus attention system by nature. Getting rid of redundant backgrounds, event cameras drive detection and tracking of hands motion towards an efficient eye-like way. The extremely high temporal resolution helps capture the detailed motion trajectory and deformation process. To make the use of event data with a specific format, practitioners normally use synchronous processing to realize batch operations. Synchronous event processing is to synchronize asynchronous event data into event frames and then apply existing frame-based algorithms. The essence of this process is space-time disentanglement. Spatial features are extracted based on event frames. Then, temporal associations are extracted among these frames. However, this approach weakens the advantage of event cameras in providing rich temporal information. Meanwhile, it magnifies the disadvantage of event cameras in a lack of intensity information.

In this chapter, we propose a novel concept of *Space-time Event Clouds* to interpret an event stream as a 3D point cloud. Each event becomes a point in a three-dimensional space-time continuum represented as (x, y, t) . Each gesture generates a distinctive event cloud in the same space-time continuum. The representation is straightforward, but the implications deserve careful consideration. Under the space-time event cloud perspective, the intrinsic space-time entanglement in event data are preserved. The spatiotemporal features are extracted directly and simultaneously from the geometric distribution of event clouds. The asynchronous event data are processed synchronously without explicit space-time disentanglement. Our framework keeps the exclusive characteristics of event cameras well and take the advantage of existing parallel computing architectures. The proposed classification framework of space-time event clouds is conducted under a subspace sampling perspective to deal with the noisy raw event data in a super efficient way.

The key contributions of this chapter are as follows:

- We propose the concept of Space-time Event Cloud to enable synchronous event processing without space-time disentanglement.
- We leverage PointNet to verify that the spatiotemporal features can be extracted simultaneously from the geometric distribution of space-time event clouds.
- We provide an intuitive explanation and strict experimental prove on the effectiveness of proposed subspace sampling-based classification against the noisy space-time event clouds.
- We design a rolling-buffer mechanism to enhance the performance. The proposed framework achieves 97.08% accuracy in the IBM DVS128 Gesture dataset and 118ms latency.

FIGURE 3.1.1: (a)Counter-clockwise circling recorded by frames. (b)Hand waving recorded by frame. (c)Counter-clockwise circling recorded by events. The hand movement generates a large number of events, while the stationary background will not generate events. (d)Counter-clockwise circling recorded by events. The hand movement generates a large number of events, while the stationary background will not generate events.

3.2 Solving Space-time Entanglements with Space-time Event Clouds

In the conventional vision system, intensity information of all pixels is captured at each time. The temporal change captured is determined by the frame rate varying from tens to thousands of frames per second. Thus, conventional video streams are dense in the spatial domain but sparse in the temporal domain. When investigating a frame generated at a time instant, the comprehensive spatial feature can be extracted and learned. The temporal information is retrieved among consecutive frames. In the event camera-based vision system, the event camera outputs an event whenever the intensity change of a certain pixel exceeds the threshold. Each event carries only the spatial coordinates (x, y) of a pixel reporting a change and a timestamp t indicating when that change happened [33]. Event streams are asynchronous and have a much higher temporal resolution from microseconds to nanoseconds level. Thus, event streams are sparse in the spatial domain while dense in the temporal domain. Different from conventional cameras, at a certain time instant, an event camera can only generate one or very few data points (events). These events are too sparse to contain enough spatial information. To retrieve meaningful information from events, we need to observe them in a considerably long duration and view them in a spatial-temporal perspective.

In the human vision system, we see the world as three dimensions of space plus one dimension of time. Similarly, camera vision systems see the world as two dimensions of image space plus one dimension of time. In the Newtonian view of space and time, time is a measurement separated from the space dimensions and it is an independent variable flowing on its own. In conventional video analysis, algorithms are intuitively developed under the Newtonian idea of time, where time is considered as a measurement of duration, sequential order or frequency of motions. As a result, video processing in the conventional vision system usually consists of two stages, spatial feature extraction within each frame and temporal feature extraction among these frames. Although there are many different approaches to extract

features from the unique type of asynchronous event streams, they subconsciously treat the event streams as temporal sequences. The concept of temporal sequences and matrix of pixels inherited from conventional video analysis of frames is applied to event streams in a similar way. Under this perspective, an event e_i is denoted as

$$e_i \doteq (\mathbf{x}_i, t_i), \quad \mathbf{x}_i \doteq (x_i, y_i)^T \quad (3.2.1)$$

The event streams are time-series data recording the positions of intensity change in image space in chronological order. Different from the highly synchronized frames in conventional cameras, event data are asynchronous responses of event cameras to the intensity changes. To apply existing techniques on event data, the first step is to synchronize the events into a matrix of pixels. Although there are many different synchronization approaches, they all share the merit of space-time disentanglements essentially. However, space-time disentanglements do not fully utilize the spatial sparsity of event streams. Meanwhile, the dense temporal information captured is diluted. In this work, we think out of the conventional concept of temporal sequences and view the event streams in a three-dimensional space continuum. In conventional 3D space, a point is denoted as (x, y, z) while in the world of event cameras, a point (event) in 3D space is denoted as

$$e_i \doteq (x_i, y_i, t_i) \quad (3.2.2)$$

The continuous event stream forms a cloud of points (events) in 3D space, thus we call it *space-time event clouds*. With space-time event clouds, we aim to extract spatio-temporal features directly from the entangled states.

In the conventional event camera-based classification system, event streams are divided into multiple segments for feature extraction. Temporal segmentation divides events by fixed time intervals or a fixed number of events, while soft segmentation adaptively obtains segments according to some certain predefined rules [54]. In a selected time interval T , assume a total number of d events are generated by an

FIGURE 3.2.1: (a) Conventional approaches disentangle the spatial and temporal features of event streams. (b) The proposed method solves the space-time entanglement of event streams by modeling them as space-time event clouds.

event camera with a $N \times M$ resolution. When treating event streams as temporal sequences, a set of events within time interval T is expressed as:

$$s_{temporal}^T = \{e_i \doteq (x_i, y_i) | t_i \in T, i = 1, 2, \dots, d\} \quad (3.2.3)$$

When treating event streams as 3D space-time event clouds, a set of events within time interval T is expressed as:

$$s_{3D}^T = \{e_i \doteq (x_i, y_i, t_i) | t_i \in T, i = 1, 2, \dots, d\} \quad (3.2.4)$$

In conventional time sequence approach, no matter modeling events as postsynaptic potentials for neural spiking pattern classification [55] or interpreting event streams as a bag of events for joint probability distribution classification [54], feature extraction is conducted on set $s_{temporal}^T$ and events are accumulated over the time axis on the image plane. Thus, an $N \times M$ array is required to store the spiking patterns or probability patterns as shown in Figure 3.2.1(a). Processing of event

sequences becomes the processing of dense arrays as what we do in conventional video analysis. The event streams are extremely sparse in the spatial domain. In DVS128 Gesture Dataset [33], only around 10% of the array are occupied by valid data. The advantage of spatial sparsity is largely diluted and memory cost is increased. The accumulation of events over a short time interval also dilutes the dense temporal information captured by event cameras. The temporal orders among events within that interval are lost during integration.

In the proposed approach, event streams are treated as space-time event clouds, which is a set of points (events) in 3D space. Spatial features and temporal features are fused in the 3D space-time continuum. Space-time event clouds are point clouds with infinity extension on the z-axis (time axis). In the proposed gesture recognition framework, a fixed time window of events is selected to form a bounded space-time event cloud for each classification as shown in Figure 3.2.2. Processing of event streams becomes a set operation over s_{3D}^T . It fully utilizes the spatial sparsity and achieves an effective usage of memories. As each event is a point in 3D space denoted with coordinates (x, y, t) , there is no integration over the time axis anymore. Thus, the dense time information is also well preserved and is transformed into geometric information.

FIGURE 3.2.2: A sliding window is applied to select a space-time event cloud for each classification.

3.3 Event Bagging: A Global Subspace Sampling for Noisy Event Clouds

There are two main challenges of the recognition on the raw space-time event cloud as illustrated in Figure 3.3.1. On the one hand, the number of events in each event cloud is huge and it varies case by case. On another hand, the effect of noise is significant in event clouds. Directly employing the raw event clouds as training data will cause overfitting of the classifier. Inspired by the feature bagging algorithm [56] for outlier detection, in this work, we propose an event bagging scheme under a global subspace sampling perspective to project the high dimensional features with various dimensionality to lower-dimensional features with unified dimensionality. Subsets formed by low dimensional projections are employed for model training to reduce variance and representational bias of the classifier against the noise.

FIGURE 3.3.1: The number of events in each event cloud is huge and it varies case by case. The effect of noise is significant in raw event clouds.

3.3.1 Curse of Dimensionality

For each gesture, it generates an event cloud consisting of a very large number of events. A gesture is described by an event cloud $s^d = \{e_1, e_2, \dots, e_d\}$ of d events. We consider each event as a feature point and each event cloud of a gesture is described by a feature vector F with d features. The number of events (the value of d) is usually very large and varies case by case. In the DVS128 gesture dataset, there are 20k to 80k events for each gesture in 0.5s time interval. Thus, the value of d varies from 20k to 80k. As a result, the feature vector representation of an event cloud is in a very high dimensional space and its dimensionality varies.

A straightforward approach is to down-sample/up-sample a fixed number of events from raw event clouds on the original order of magnitude of raw events, e.g. 30k events. However, the original number of events is relatively large and the features are projected into a relatively high dimensional space. Many classification algorithms cannot work effectively in high-dimensional space due to a well-known phenomenon, *curse of dimensionality* [57]. The reason for the curse of dimensionality is data sparsity in high-dimensional space. For example, distance-based methods will measure distances on all dimensions. The discrimination ability deteriorates due to noisy dimensions [56, 58, 59].

Different algorithms have different manifestations of the curse of dimensionality. For example, we use the nearest-neighbor framework to illustrate this problem. We pre-determine that a fraction r of all samples is used as neighbors for target input. It is assumed that all samples are randomly distributed in a unit hypercube. Therefore, these neighbors correspond to a fraction r of the unit volume aggregately. The expected edge length in each dimension should be $l_d(r) = r^{1/p}$. If $p = 10000$ and $r = 0.1\%$, $l_{10000}(0.001) = 0.9993$. These neighbors almost cover the whole length of each dimension. They are no longer local. In other words, it is difficult to discriminate neighbors or non-neighbors.

3.3.2 Low Dimensional Feature Projection

It has been shown recently that by examining the behavior of the data in subspaces, it is possible to develop a more effective algorithm for cluster discovery and similarity search in high dimensional spaces [56, 60–62]. In addition, when a significant number of events in the raw event cloud are considered noisy, taking all events into consideration does not result in an effective classification of event clouds.

To deal with the above-mentioned challenges, a novel event bagging framework under a global subspace sampling perspective is proposed in this work to randomly select a fixed N number of features (events) from raw event cloud with a various

FIGURE 3.3.2: Unlike standard bagging approach where data points are randomly sampled from data distribution, global subspace sampling randomly select a subset of features for all data samples each time and form a subset of all data points with subsampled features.

number of features (events). As illustrated in Figure 3.3.2, unlike the standard bagging approach where the classifiers use randomly sampled data distribution, in this approach, proposed event bagging randomly select a subset of features (events) for all data samples each time and form a subset of all data points with subsampled features (events). A single classifier is trained by a newly sampled subset in each epoch. The detailed algorithm is shown in Figure 3.3.3 and illustrated as follows:

- Given: Set $S = \{(s_1, y_1), (s_2, y_2), \dots, (s_m, y_m)\}$, s_i represents a space-time event cloud of a gesture and $s_i \in \mathbf{X}^{d_i}$, with label $y_i \in Y = 0, 1, \dots, k-1$, where k is the number of classes, and d_i corresponds to the dimensionality (number of events) of vector $\mathbf{X}$. For the raw event cloud, the dimensionality d_i for each s_i varies case by case.
- Determine $N (N \ll d_i)$, which is the number of events to be sampled.
- In each epoch j , for each s_i in set S , randomly pick (without replacement) N events to create a event subset $F_i^j \in \mathbf{X}^N$ and obtain new subspace dataset $S^j = \{(F_i^j, y_1), (F_2^j, y_2), \dots, (F_m^j, y_m)\}$.

FIGURE 3.3.3: The proposed event bagging scheme is applied on-the-fly during the training phase. For each event stream containing a gesture with label y_i , training samples are selected based on the pre-defined sliding window size and step size. The entire training set is represented as $S = \{(s_1, y_1), (s_2, y_2), \dots, (s_m, y_m)\}$. In each epoch j , for each s_i in set S , randomly pick (without replacement) N events to create a event subset $F_i^j \in X^N$ and obtain new subspace dataset $S^j = \{(F_i^j, y_1), (F_2^j, y_2), \dots, (F_m^j, y_m)\}$. Train the model using the new subspace dataset S^j .

- Train the model using new subspace dataset S^j .

3.3.3 Event Bagging: A Ensemble-Centric Perspective

Here, we provide an ensemble-centric view of the proposed scheme. The event bagging method samples different subsets of feature dimensions. The basic idea is to select N dimensions of features (events) randomly from the original feature dimension. The classifier is trained using these subsets of lower-dimensional feature projections.

Although event bagging might seem like a data-centric ensemble method, it is better explained using a model-centric view of the bias-variance trade-off. Event bagging is a method that reduces model variance in a model-centric view. Event bagging with a particular subset of dimensions has a data-centric bias that depends on the selected dimensions. However, if one views the step of randomly selecting the subset of dimensions as a part of the component classifier, then each such (randomized) classifier has exactly the same model-centric bias, and the aforementioned variability in the bias across different dimension-specific instantiations now becomes a part of this (randomized) classifier's model-centric variance. Note that the random process for quantifying the bias-variance trade-off, in this case, is over the different randomized base classifiers, which are applied to a fixed data set S . In such cases, using a model ensemble is able to achieve variance reduction. The smaller the subset of dimensions that are selected, the greater the variance reduction that is achieved. This is because the underlying classifier tends to be relatively uncorrelated if few overlapping dimensions are selected by different classifiers. In experiments, 256/512/1024 events are randomly sampled from each raw event cloud with more than 10k events.

3.4 Classifiers for Space-time Event Clouds

In the proposed approach, the recognition of a hand gesture becomes the recognition of geometric distribution of the 3D space-time event clouds generated by that gesture. As conventional neural network models are no longer suitable to process sparse set data, a network architecture to learn geometric distribution features of event clouds in 3D space is needed.

3.4.1 Adapted PointNet for 3D Space-time Event Clouds Classification

PointNet [41] is a neural network architecture that directly takes point clouds as input and it learns to summarize the geometric features of the input point clouds. A point cloud is denoted as a set of 3D points $s = \{p_i | i = 1, \dots, n\}$, where each point p_i is a vector of its coordinates (x, y, z) . The architecture of PointNet is shown in Figure 3.4.1(a). Each input point p_i is processed by a shared multi-layer perceptron network and it is trained to capture different properties of the input set. As points in the set are unordered, the input sequence of points should not affect the output. A single symmetric function, max pooling, is applied. The network learns a set of optimization functions to select informative points of the input set of points. The fully connected layers aggregate these learnt optimal values into a global descriptor of the entire shape of point clouds for classification. Although simple, PointNet architecture demonstrates the universal approximation ability of continuous set functions. A Hausdorff continuous symmetric function $f: 2^x \rightarrow \mathbb{R}$ can be arbitrarily approximated by PointNet as

$$|f(s) - \gamma(\text{MAX}_{p_i \in S} \{h(p_i)\})| < \epsilon \quad (3.4.1)$$

where $p_1, \dots, p_n$ are elements in s ordered arbitrarily, h and γ are continuous functions approximating multi-layer perceptron networks, and MAX is an element-wise max pooling operator.

FIGURE 3.4.1: (a) PointNet consumes point clouds for object classification. (b) In this work, PointNet is adapted to consume space-time event clouds for gesture recognition in event-camera based classification system.

Considering that PointNet is highly efficient, and it demonstrates strong performance in point clouds classification, PointNet is adapted and trained for space-time event clouds classification in this work. As shown in Figure 3.4.1(b), our input is a set of event clouds within time interval T denotes as $s^T = \{e_i | i = 1, \dots, n\}$, where each event e_i is a vector of its coordinates (x, y, t) . The universal approximation becomes

$$|f(s^T) - \gamma(\text{MAX}_{e_i \in s^T} \{h(e_i)\})| < \epsilon \quad (3.4.2)$$

where $e_1, \dots, e_n$ is elements in s^T ordered arbitrarily, h and γ are continuous functions approximated multi-layer perception networks, and MAX is an element-wise max pooling operator. PointNet architecture models set functions that directly take a set of events s^T as input and output a global feature of the input event cloud.

3.4.2 Hierarchical PointNet

PointNet learns to summarize a global feature of the input event clouds. It is not able to capture local structures induced by the metric space points live in, limiting its ability to recognize fine-grained patterns or nonuniform points distributions. The event clouds generated by movements are not uniform and largely depends on the speed of movements. The event clouds generated by hand gestures involving two hands are more complex. These two challenges make the global feature learned by PointNet less discriminative. PointNet++ [63] is an advanced version of PointNet that applies PointNet recursively on a nested partitioning of the input set. In this work, PointNet++ architecture is trained to aggregate local and global features of event clouds. A hierarchical feature extraction architecture is shown in Figure 3.4.2. Given the input event cloud of time interval T denoted as $s^T = \{e_1, e_2, \dots, e_n\}$, farthest point sampling (FPS) is used to select N_1 number of events as the central point of N_1 sub-regions. Ball query finds out all the neighboring events within radius r_1 and a fixed K_1 number of events is sampled. Thus, the input event cloud is portioned into N_1 sub event clouds and each sub event cloud contains K_1 number of events. A basic PointNet network is trained to learn local features of each sub event clouds. The learned feature summarizes the geometric distribution of events within each sub-region and each learned feature becomes a point in higher dimension M_1 . They form a new event cloud in metric space which is again partitioned into smaller sub event clouds. A basic PointNet network learns the local feature of each newly selected sub-region. The local features are learned and aggregated layer by layer. The feature in the final layer contains both local and global features of the input event cloud and the fully connected layers are applied to classify the entire distribution of the event clouds generated by movements.

FIGURE 3.4.2: A PointNet-based hierarchic feature learning architecture, PointNet++, is trained to learn local and global features of event clouds generated by hand gestures

3.5 Online Real-time Gesture Recognition Framework

Many real-time interaction tasks desire very low latency. Some researches show that a good real-time gesture interaction application needs a response time within 100ms to 200ms [64]. This means that decision must be given within 200ms after starting of the gesture. As a 100-200ms duration including processing time only contains a very short snippet of the gesture, past information is combined with current input for a better classification of a gesture in this work. As shown in Figure 3.5.1, a rolling buffer mechanism is applied to store past events of a predefined time duration T . The events rolling buffer is updated every Δt , which means that the latest Δt events will be popped into the events rolling buffer each time while the earlier Δt events will be flopped. For each classification, the latest Δt input events are fused with past events stored in the events rolling buffer to form the input event clouds to the trained network for classification and random down-sampling technique is applied. To improve the classification performance, a decision rolling buffer is used to store classification results for consecutive L windows and a majority decision filter is used to make the final classification for the input window captured $\frac{(L+1)}{2} \times \Delta t$ before. This decision filter outputs the class with maximum frequency

in decision buffer to remove unreliable classification results. The latency of this framework is $\frac{(L+1)}{2} \times \Delta t + t_c$, where t_c is the average processing time of down-sampling and classification. By this rolling buffer mechanism with a decision filter, high classification accuracy can be achieved within a response time around 100ms. The proposed event cloud-based online processing framework with a low memory requirement is suitable for many portable platforms with limited memory budget.

FIGURE 3.5.1: Rolling buffer framework for real time gesture recognition

3.6 Experiments

The experiments are divided into two parts. First, we evaluate the classification ability of the proposed classifier against different classifiers for event streams in the offline test. For comparison, an LSTM-based classifier is used to handle event streams as pure temporal sequences. Both PointNet-based and PointNet++-based classifiers are trained to demonstrate the contribution of the hierarchical feature extraction architecture. Second, we show the fast response real-time hand gesture recognition system, where a well-trained classifier is embedded in the rolling buffer

framework. We compare the proposed system with a state-of-the-art event-camera based hand gesture classification system.

3.6.1 Dataset

The DVS128 Gesture Dataset [33] from IBM is used in this paper. This dataset is captured by the iniLabs DVS128 camera with a 128×128 resolution. Thus, the spatial coordinates x, y are within the range of $[1, 128]$, where $x, y \in \mathbb{N}$. This dataset includes 1,342 instances of 11 classes of hand and arm gestures, grouped in 122 trials collected from 29 subjects under different lighting conditions including natural light, fluorescent light, and LED light. Snippets of the first ten classes are shown in Figure 3.6.1. The 11th class is labeled as random gestures excluding the first ten classes. As provided in the dataset [33], 23 subjects are used as the training set and 6 subjects are reserved for out-of-sample validation. The dataset is available at <http://research.ibm.com/dvsgesture/>.

3.6.2 Classifiers Training

Different from conventional 3D point clouds, the event clouds are continuous video streams over time. To train the classifiers with the appropriate signals, we first preprocess the dataset by segmenting the raw event streams through a fixed-size sliding window that shapes the input to classifiers. The sliding window size determines the input length of an event stream segment used for one classification, which equals to the total length of event streams stored in the event rolling buffer. In this experiment, the sliding window size is selected as a fixed time interval $T = 1s, 0.5s, 0.25s$ and each window is labeled accordingly. The step size is chosen to be half of the sliding window size when generating the training input.

As event cameras generate data on output-by-demand nature, the event rate per second is not fixed and is largely determined by the range and frequency of movements. Given a sliding window of 1s, the number of events varies from 5k to 300k in

FIGURE 3.6.1: Two-second snippets of 10 classes in DVS128 Gesture Dataset.

the dataset [33] used in this experiment. In [41], it has been proved that PointNet learns to summarize a shape by a sparse set of critical points, which means that it is not necessary to input all the events (points) generated in order to obtain a reliable prediction. In our experiment, we randomly sampled a subset of events $S_{3D,n=256,512,1024}^T$ from the original event stream S_{raw}^T within each sliding window as critical points input to the classifiers.

TABLE 3.6.1: Classifiers test results. Accuracy is reported in percentage.

Experiment (model,classes,events)	Sliding window		
	$T=0.25s$	$T=0.50s$	$T=1.00s$
LSTM,10,256	80.96	85.58	84.10
LSTM,10,512	88.17	86.55	82.28
PointNet,10,256	87.85	89.63	88.54
PointNet,10,512	88.67	90.20	89.61
PointNet,10,1024	88.77	89.68	89.92
PointNet++,10,256	95.28	95.59	95.54
PointNet++,10,512	95.39	96.34	95.61
PointNet++,10,1024	94.93	95.89	95.97
PointNet++,11,256	91.92	93.38	93.61
PointNet++,11,512	92.23	94.10	93.83
PointNet++,11,1024	91.87	91.91	92.63

As the event streams can be treated as temporal sequences, a two-hidden-layer-stacked LSTM model with 256 hidden neurons each layer is trained to consume event streams as temporal sequences. For a fair comparison, the input event sequence $S_{temporal,n}^T$ to the LSTM-based model is the same set of events in $S_{3D,n}^T$ but treating them as a temporal sequence of n events. The detailed setting of PointNet classifier is $MLP(64; 64; 64; 128; 1024) \rightarrow FC(512) \rightarrow FC(256; 0.7) \rightarrow FC(K)$. The detailed setting of PointNet++ (SSG) classifier is $SA(256; 0.2; [64; 64; 128]) \rightarrow SA(64; 0.4; [128; 128; 256]) \rightarrow SA([256; 512; 1024]) \rightarrow FC(512; 0.5) \rightarrow FC(256; 0.5) \rightarrow FC(K)$. All the classifiers are trained on Linux system with single GPU.

3.6.3 Classification ability of different classifiers

Classification ability of classifiers are evaluated with different sliding window length ($T = 0.25s, 0.5s, 1s$) and different number of randomly down-sampled events ($n = 256, 512, 1024$). The out-of-sample test results of different classifiers are shown in Table 3.6.1. LSTM-based models consume event streams as pure temporal sequences and learn the temporal features from the events. Lacking sufficient spatial information makes LSTM-based models achieve the lowest accuracy. PointNet summarizing a global feature of the input event clouds achieves an accuracy 4%–5%

FIGURE 3.6.2: Confusion matrices of different classifiers on the same test setup, where 512 events are randomly down-sampled from a 0.5s sliding window. Only non zero percentages are shown. (a)LSTM, 10 classes (b)PointNet, 10 classes (c)PointNet, 10 classes (d)PointNet++, 11 classes

higher than the LSTM-based approach. PointNet++ with a hierarchical feature learning architecture demonstrates a classification accuracy 6% – 7% higher than PointNet. When including random gestures in PointNet++ classifier, the accuracy decreases 2% – 3%.

The confusion matrices of different classifiers with the same experiment setup are shown in Figure 3.6.2. For the LSTM classifier, the error rates among class 0/7/8/9 are high. PointNet classifier mainly reduces the misclassification rate of class 7. PointNet++ classifier reduces misclassification rate significantly. When adding the interference of random gestures (class 10), it affects mostly the classification of class 8 and class 9. In summary, trained networks are more likely to be confused among

TABLE 3.6.2: System test results compared with state-of-the-art results. Accuracy is reported in percentage.

		This work					
Buffer length (sliding window)	No.of events	System acc. (10 classes)			System acc. (11 classes)		
		$L = 5$	$L = 7$	$L = 9$	$L = 5$	$L = 7$	$L = 9$
$T = 1.00s$	$n=1024$	96.96	96.69	96.81	92.71	92.83	93.17
	$n= 512$	96.42	96.42	96.61	94.12	93.96	94.27
	$n= 256$	96.53	96.53	96.58	94.28	94.81	94.59
$T = 0.50s$	$n=1024$	96.84	96.89	96.84	92.45	92.58	92.47
	$n= 512$	96.93	96.81	96.80	94.33	94.47	94.73
	$n= 256$	96.91	97.08	96.97	94.68	95.08	95.32
$T = 0.25s$	$n=1024$	95.70	95.78	95.90	92.12	91.92	92.07
	$n= 512$	96.54	96.63	96.83	92.65	93.03	93.31
	$n= 256$	96.27	96.42	96.47	93.33	93.87	94.03
Benchmark accuracy [33]		96.49			94.59		

these two-hands gestures (class 0/7/8/9). PointNet++ shows a significant increase in the classification ability of two-hands gestures due to the hierarchical feature learning and aggregation process.

3.6.4 Online gesture recognition system

We embed the trained PointNet++ based event clouds classifier into the proposed rolling buffer framework. The performance of our online real-time gesture recognition framework is evaluated with different events rolling buffer length T and different decision buffer length L . The overall results are summarized in Table 3.6.2. Given the same dataset, Amir *et al* [33] using the CNN approach achieved a system accuracy of 96.49% without the interference of random gestures and a system accuracy of 94.59% when including random gestures. Our work using the event clouds approach achieved a system accuracy of 97.08% without the interference of random gestures and 95.32 %. In [33], the average latency from t_{start} to $t_{decision}$ are 104.6ms and 120.6ms respectively. In our approach, the average latency from t_{start} to $t_{decision}$ are 93ms ($L=5$), 118ms ($L=7$) and 143ms ($L=9$). For both tests based

on 10 classes and 11 classes, we achieved better than state-of-the-art accuracy with comparable system latency.

3.7 Discussion

3.7.1 Robustness Against Noise

As shown in Figure 3.6.1, it can be found that there are many outliers in the event clouds, which are not triggered by the target movements. It can be sensor noise or intensity changes caused by the background or light condition changes. In the proposed method, both training and inference are carried out under a subspace sampling perspective, where a fixed number N events are randomly sampled from each raw event cloud sample and used as input features. Using lower-dimensional feature projections as input features largely reduce the effect of noise on the model and reduce the effect of over-fitting.

From another perspective, PointNet [41] itself is robust to extra noise points. Thus, in our approach, better than the state-of-the-art result presented in Table 3.6.3 is achieved without using extra noise removal techniques. Moreover, PointNet learns to summarize a shape by a sparse set of critical points without sacrificing the classification accuracy. It enables us to downsample a small subset of critical events from a large number of raw events as classifier input, which largely reduces the data to be processed by the classifier.

3.7.2 Efficiency of Proposed Method

Compared with the CNN-based approach, the data scale processed in the proposed framework is particularly small. As shown in Figure 3.7.1, for a region of interest with 128×128 resolution, when casting the events back to virtual frames, each frame contains 16k pixels. The data processed by the CNN-based classifier per

FIGURE 3.7.1: Efficiency of proposed framework (a)CNN-based approach takes in 16k input (b)event clouds-based approach takes in 1k input

second is $16k \times \text{frame rate}$, where the frame rate is usually greater than 30fps. In our approach, the input to the classifier is not more than 1k when randomly down-sampling to 256 events per sliding window, and the number of sliding windows is set as 40. The data processed by the proposed classifier per second is around $1k \times 40$, which is more than 10 times smaller than the CNN-based approach. It indicates that many real-time applications with strict timing and memory requirements are possible to be conquered by the space-time event clouds concept with the proposed framework.

3.8 Concluding Remarks

In real-time gesture recognition, event cameras successfully address the issues of motion blur and large scale redundant data inherent in conventional cameras by its output-by-demand nature. However, it is still challenging to think out of the conventional view of space and time when dealing with sparse and asynchronous event data in recognition tasks. To the best of our knowledge, this is the first time to interpret event streams as space-time event clouds for gesture recognition

problems. We leverage PointNet++ in 3D object recognition to analyze space-time event clouds. Our method achieves the best ever accuracy of 97.08% on IBM DVS128 Gesture dataset [33]. Our results show that the proposed event clouds concept is an effective representation to characterize the event streams from event cameras. It preserves both spatial and temporal information to analyze the event streams and is end-to-end learnable. As verified in our experiments, event clouds can benefit from down-sampling too. We believe that space-time event clouds are promising to conquer other real-time multimedia and computer vision tasks with limited memory and computation power.

Chapter 4

ST-EVNet: Hierarchical Space-time Feature Learning on Event Clouds

4.1 Introduction

As frame-based RGB camera cannot provide enough spatiotemporal information for accurate hand gesture recognition, multimodal architectures [65–67] become prevailing in the computer vision community to fuse different types of features (e.g. RGB, depth and optical flow) as shown in Figure 4.1.1(a). To mitigate interference from backgrounds, researchers adapt detection techniques to achieve a hand-focus mechanism [66, 67]. The necessity of multimodality and hand-focus mechanism make these frameworks too complicated to be geared up in resource-constrained real-time human-computer interaction applications.

Recently developed event cameras [7, 68–72] have received extensive attention in solving different computer vision tasks. They also show great potential to solve gesture recognition tasks in a new perspective. The event-based vision system is a motion-focus attention system by nature. Stationary background inducing no

FIGURE 4.1.1: **Event-based gesture recognition solution.** (a) The top shows a typical video-based gesture recognition solution. When using RGB cameras to shot a *Grab* gesture. The dynamic grabbing process is recorded as discrete frames. To obtain a more reliable prediction, different features (e.g. RGB, depth, optical flow) are fused at different stages, which increases the complexity of the whole solution. (b) The bottom shows the proposed gesture recognition solution using event cameras. Event cameras record the *Grab* gesture as a continuous event stream in the space-time continuum. In this work, we design a simple and unitary ST-EVNet to learn rich spatiotemporal information inherent in event data itself for efficient gesture recognition.

intensity changes will not generate output events, whereas only motions inducing intensity changes are recorded as asynchronous event streams. Getting rid of the redundant background, event cameras drive detection and tracking of hands motion towards an efficient eye-like way. From another aspect, constructing spatiotemporal features [73] is the key to motion recognition. Frame-based motion recognition extracts spatial features on discrete frames and performs feature matching among them. Without frame constraints, event cameras record motion in the form of a continuous event stream, where each individual event (x, y, t) corresponds to the space-time location when a pixel captures an intensity change. Event streams of gestures embed richer spatiotemporal information than frame-based data, which make recognition of gestures can be achieved by a more elegant approach.

Taking advantage of rich spatiotemporal information in event data, Wang et al. [5] modeled event streams of a gesture as a Space-time Event Cloud (STEC). The STEC is a point cloud distributed in the 3D space-time continuum, which consists of two spatial axes (x, y) and one temporal axis t . This concept implicates that the features of a certain gesture are embedded in the geometric distribution of its corresponding STEC. PointNet-based networks [5, 41, 63] with hierarchical feature learning are proven to achieve outstanding results on analyzing STECs. However, constructing geometric local regions for unordered points is computation-intensive and it is significantly inefficient for STECs. By nature, event streams are generated in the temporal order in accordance with hand movements. Events within a short time interval form a temporal local region. In this work, we propose the first network architecture dedicated to processing STEC and we name it *Space-time Event Net* (ST-EVNet) as shown in Figure 4.1.1(b). ST-EVNet learns spatiotemporal features from temporal local regions instead of geometric local regions. The spirit of sequence modeling is adapted to aggregate temporal local regions hierarchically. The basic ST-EVNet archives a 90.52% accuracy on IBM128 Gesture Dataset with only 3.48M FLOPs/sample, which can be easily implemented in embedded platforms for real-time HCI applications. A dual-channel network is designed to consume STECs from two hands separately. Spatiotemporal features from the two STECs are extracted in parallel before the final classification. The dual-channel ST-EVNet unifies the single-hand gesture and dual-hand gesture recognition task and achieves the best ever accuracy of 97.27% on IBM128 Gesture Dataset.

The key contributions of our work are as follows:

- We studied the properties of space-time event clouds (STECs) and redefine the STEC as a sequence of space-time events rather than a set of unordered space-time events.
- We propose a Space-time Event Net with hierarchical space-time kernels to analyze Space-time Event Clouds with great efficiency. This is the first network architecture dedicated to process STECs.

- We design a two-channel architecture to deal with single-hand and dual-hand gestures under a unified system. This architecture achieves state-of-the-art performance on event-based gesture recognition tasks.

4.2 Related Works

Frame-based Gesture Recognition In frame-based gesture recognition solutions, depth [74, 75] and optical flow [76] are important information to make reliable prediction. Most existing frame-based gesture recognition frameworks fuse multiple data of different modalities (i.e. RGB data, depth data and flow data) to achieve accurate classification. With the development of 3D-CNN, it is widely used in existing gesture recognition systems [65, 77–79]. Wang et al. [66] proposed a multimodal framework to learn short term spatiotemporal features by 3D-CNN and learn long-term spatiotemporal features by ConvLSTM. Besides, additional hand level features were extracted to achieve a hand-focus purpose. Pradyumna et al. [67] proposed to make spatial channels focus on the hands to improve the recognition performance. The necessity of multimodality and hand-focus mechanism make these frameworks too tedious for resource-constrained real-time applications.

Event-based Gesture Recognition In the event camera-based classification system, there are a variety of approaches for gesture recognition and most of them target for real-time applications. In [33], they connected event camera outputs to an event-based neuromorphic processor TrueNorth and applied SNN-based architecture to achieve gesture recognition in real-time. It achieved a model accuracy of 91.77% on IBM DVS128 Gesture Dataset. Shrestha et al. [21] applied Spiking Neural Networks (SNNs) to process event streams and obtained an accuracy of 93.64%. By modeling event data as space-time event clouds, Wang et al. [5] used PointNet-based architecture to consume 512 events and achieved an accuracy of 96.34% on the same dataset.

Sequence Modelling Recently, the idea of temporal convolution [80] is proposed to be an alternative to well-known RNN-based models (e.g. LSTM and GRU) for sequence modeling. TCN convolves along the time axis to extract temporal features and uses a dilated convolution scheme to increase receptive fields. Bai et al. [81] demonstrated an empirical evaluation on TCN and different RNN models for different sequential tasks. For sequence modeling of event streams, Sekikawa et al. [7] proposed a recursive version of PointNet, i.e. EventNet, to processing event sequences. For each coming event, a 2D PointNet is applied to extract event-wise spatial features. The recursive processing is achieved by the decaying previous global feature to the current timestamp and apply max pooling to the decayed global feature and current spatial feature to obtain the current global feature. It archived impressive results on ego-motion estimation and scene segmentation of event cameras.

4.3 Proposed Method

In this chapter, we propose a new network architecture dedicated to process temporally ordered Space-time Event Clouds and we call it *Space-time Event Net*. In Section 4.3.1, we describe the properties of STECs. In Section 4.3.2, we illustrate the proposed hierarchical space-time kernels to extract spatiotemporal features from STECs with great efficiency. In Section 4.3.3, we introduce a parallel dual-channel architecture to deal with the recognition of both single-hand and dual-hand gestures.

4.3.1 Properties of Space-time Event Clouds

The concept of Space-time Event Cloud (STEC) was first proposed by Wang et al. in [5]. Under STEC, each gesture generates a distinctive cloud of events in (x, y, t) coordinate system, in which a space-time event is denoted as $\mathbf{e}_i = (x_i, y_i, t_i)$. A gesture recorded within time interval T is a set of space-time events, which is

FIGURE 4.3.1: **Temporally ordered space-time event clouds.** In [5], the event stream of a gesture is modeled as an unordered space-time event cloud (left, all events are in the same color), where local features embed in its geometric local region. In this work, we model the event stream of a gesture as a temporally ordered space-time event cloud (right, events are colored according to temporal order), where local features can be extracted from the temporal local region without extra computation to construct a local region.

represented as:

$$S^T = \{\mathbf{e}_i | t_i \in T, i = 1, 2, \dots, N\} \quad (4.3.1)$$

Spatiotemporal features of a gesture are encoded in the Euclidean geometry of its corresponding STEC. In this perspective, to recognize a gesture is equivalent to recognize the geometric distribution of a STEC. This principle is in spirit similar to recognizing an object by its 3D point cloud [41]. In other words, the time axis in STEC becomes a numerical measurement instead of sequential information.

A general approach to extract spatiotemporal features from STECs has two steps, i.e., a common function (e.g. a multi-layer perceptron network) to extract event-wise features and a global function (e.g. max-pooling) to summarize event-wise features into a global feature vector. Mathematically, the recognition function is described as:

$$\mathcal{F}(S^T) \approx \gamma(\text{MAX}_{e_i \in S^T} \{h(\mathbf{e}_i)\}) \quad (4.3.2)$$

However, this two-step scheme fails to capture the local interactions among neighboring events.

Recent researches [63, 82, 83] show that learning interactions among neighboring points plays a significant role in point clouds analysis. These solutions also bring performance improvement on STEC-based gesture recognition tasks [5]. The techniques designed to extract local interactions from conventional 3D point clouds have an important implication, i.e., points within a point cloud are internally unordered. In other words, these networks are designed to be invariant to permutations of the input set in the data feeding order. When applying these techniques on STECs, events within a STEC are treated as permutation invariant implicitly. The interactions among events are measured by their space-time Euclidean distances. Constructing geometric local regions and aggregating local features hierarchically for unordered points are computation-intensive, which is not efficient for resource-constrained real-time HCI applications. However, STECs are intrinsically ordered. The STEC of a gesture is generated in the temporal order in accordance with hand movements. As shown in Figure 4.3.1, the STEC of a gesture is not generated randomly (left) but generated with temporal order (right). In this perspective, a STEC within time interval T is better modeled as a sequence of space-time points G^T rather than a set of unordered space-time points S^T :

$$G^T = (\mathbf{e}_i)_{i=1}^N = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N) \quad (4.3.3)$$

The events within a short time interval form a temporal local region in nature without any extra computation to construct the local regions for feature learning.

STECs are ordered sequences of space-time points, but it does not mean they are fully permutation variant. Figure 4.3.2 shows three STECs of the same gesture (top) and their corresponding downsampled STECs (bottom). When investigating a small time interval of a STEC, the distribution of events are stochastic. If we divide the STEC of a gesture into multiple short-term cells, events within a cell cannot tell too much about the gesture. However, the inter-cell trend encodes the spatiotemporal features of a gesture. As shown in Figure 4.3.2 (bottom), a down-sampled STEC well preserves the unique characteristics of the gesture. In

FIGURE 4.3.2: **Rationality of downsampling on STECs.** Top shows the raw STECs of the same gesture and bottom shows the corresponding downsampled STECs. The downsampled STECs preserve the long-term behaviors of the gesture.

short words, events within an event cloud are short-term permutation invariant, but long-term permutation variant.

In summary, STECs have three properties:

- STECs are 3D event clouds in (x, y, t) coordinates and spatiotemporal features are encoded into the geometric distribution of STECs.
- Events within a STEC are temporally ordered.
- Events within a STEC are short-term permutation invariant, but long-term permutation variant.

4.3.2 Hierarchical Local Spatiotemporal Feature Learning

Existing techniques to process unordered spatial 3D point clouds emphasize interactions among neighboring points within a geometric local region. As illustrated in Section 4.3.1, events within a STEC are temporally ordered and form temporal local regions by nature. A STEC describes how the hand movements of a gesture

FIGURE 4.3.3: **Hierarchical Space-time kernels.** The left shows temporal feature learning via temporal convolution. The right shows 3D geometric feature summarizing via point-wise MLPs and max-pooling. Bottom is the proposed ST-EVNet with hierarchical space-time kernels, which takes advantage of the previous two architectures to aggregate local spatiotemporal features with great efficiency.

change along the time axis. It is intuitive to learn spatiotemporal features hierarchically from sequential temporal local regions rather than unordered geometric local regions from STECs of gestures.

In this chapter, we propose to use the space-time kernel to extract interactive spatiotemporal features among events within a temporal local region. A space-time kernel ϕ is a $k \times 1 \times 3 \times d$ matrix, where k is the kernel length determining how many neighboring events are considered. When $k = 2$, two sequential events form a temporal local region. The local spatiotemporal feature is learned by convolving events within the local region with a space-time kernel. The output feature $\mathbf{f}$ is

FIGURE 4.3.4: **Dual-channel ST-EVNet architecture.** The dual-channel network takes N events from each STEC as input. Local spatiotemporal features are aggregated by hierarchical space-time kernels, and are summarized by max pooling. The numbers in bracket are feature vector sizes of each layer. The output is classification scores for m classes. For the classification net, numbers in bracket are layer sizes of the multi-layer perceptron network.

computed as

$$\begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & t_1 \\ x_2 & y_2 & t_2 \end{bmatrix} \quad (4.3.4)$$

$$\mathbf{f}_1 = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} \otimes \phi_1, \quad \phi_1 \in \mathbb{R}^{2 \times 1 \times 3 \times d} \quad (4.3.5)$$

Events within a temporal local region form a mini STEC, the local spatiotemporal information of a gesture is embedded in the geometric distribution of that mini STEC. The geometric distribution is learned by the space-time kernel in a PointNet-based manner, where the kernel depth d determines the output channels of local geometric features. Considering a STEC of $G^T = (\mathbf{e}_i)_{i=1}^N$, a space-time kernel convolves with sequential temporal local regions formed by k consecutive events, which is in spirit similar to temporal convolution. Each output feature $\mathbf{f}_n^1$ of first

layer is computed as

$$\mathbf{f}_n^l = \begin{bmatrix} \mathbf{e}_{(n-1) \times k+1} \\ \mathbf{e}_{(n-1) \times k+2} \\ \dots \\ \mathbf{e}_{(n-1) \times k+k} \end{bmatrix} \otimes \phi_l, \phi_l \in \mathbb{R}^{k^l \times 1 \times 3 \times d^l} \quad (4.3.6)$$

We further arrange space-time kernels hierarchically to aggregate sequential spatiotemporal features. The new network architecture is dedicated to processing STECs, and we call it Space-time Event Net. A basic ST-EVNet is shown in Figure 4.3.3 (bottom). Following hidden layers convolve dilatedly among k^l neighboring feature vectors:

$$\mathbf{f}_n^l = \begin{bmatrix} \mathbf{f}_{(n-1) \times k+1}^{l-1} \\ \mathbf{f}_{(n-1) \times k+2}^{l-1} \\ \dots \\ \mathbf{f}_{(n-1) \times k+k}^{l-1} \end{bmatrix} \otimes \phi_l, \phi_l \in \mathbb{R}^{k^l \times 1 \times d^{l-1} \times d^l} \quad (4.3.7)$$

After a predetermined number of convolutional layers, these abstracted features undergo an element-wise max-pooling to get the global spatiotemporal feature vector for final classification:

$$\mathbf{g} = \text{MAX}\{\mathbf{f}_1^L, \mathbf{f}_2^L, \dots, \mathbf{f}_N^L\} \quad (4.3.8)$$

In summary, the SP-EVNet can be described as a function applied among a sequence of space-time events:

$$\mathcal{F}(G^T) \approx \gamma(\text{MAX}\{h((\mathbf{e}_i)_{i=1}^n)\}) \quad (4.3.9)$$

FIGURE 4.3.5: **Constructing dual-channel training data.** We select six gestures from DVS128 Gesture Dataset to illustrate how we prepare the dual-channel data. We show the STEC of each gesture in a one second snippet. Corresponding frame-based gestures are presented on the left. For gestures generating two separate STECs (right), we arrange them into two channels. Otherwise, we sample the unified STEC (left and middle) into one channel and pad the other channel with zeros.

4.3.3 Dual-channel Architecture

For a single-hand gesture (e.g. hand waving), it generates a single STEC in space-time. For gestures involving two hands, some gestures (e.g. air guitar and air drum) generate two separate STECs, while others (e.g. hand clapping and forearm roll) generate two closely interactive event streams and form a unified STEC in space-time. When dealing with gestures involving two separate STECs, it is more reasonable to learn spatiotemporal features respectively and find the interaction between them rather than simply treating them as a whole. Therefore, an end-to-end network to handle single- and dual-hand gestures is needed.

Based on the basic ST-EVNet with hierarchical space-time kernels, we design a

parallel dual-channel architecture to unify single- and dual-hand gesture recognition tasks. As shown in Figure 4.3.4, two hands are tracked¹ separately and respective STECs of two hands during the same time interval are fed into two parallel feature learning channels. The spatiotemporal features of two event clouds are extracted by the same hierarchical space-time kernels and hand level max pooling is applied to obtain a global description of each hand,

$$\begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 \end{bmatrix} = \text{MAX}\{h(\begin{bmatrix} G_1^T & G_2^T \end{bmatrix})\} \quad (4.3.10)$$

Two abstract global spatiotemporal feature vectors of two hands are concatenated for final classification.

$$\mathcal{F}(\begin{bmatrix} G_1^T & G_2^T \end{bmatrix}) \approx \gamma(\mathbf{g}_1 \widehat{\mathbf{g}}_2) \quad (4.3.11)$$

When dealing with a single-hand gesture or a closely interactive dual-hand gesture, the unified STEC is fed into first channel of the network and second channel is filled in with zeros.

4.4 Experiments

In this part, we design four sets of experiments to evaluate the performance of the proposed method. In Section 4.4.1, we evaluate the effectiveness of modeling event data as temporally ordered space-time clouds on different datasets. In Section 4.4.2, we demonstrate significant performance improvement when applying the dual-channel architecture of ST-EVNet on gesture recognition tasks. In Section 4.4.3, we compare the proposed ST-EVNet with state-of-the-art techniques on event-based gesture recognition tasks in terms of both accuracy and efficiency. In Section 4.4.4, we compare the proposed approach with recent developed EventNet [7] under the perspective of efficient event processing.

¹Hands detection and tracking techniques are not discussed in this section.

4.4.1 Datasets

DVS128 Gesture Dataset This dataset [33] includes 10 categories of defined hand/arm gestures with one open class of random gestures. To evaluate the dual-channel architecture, we further sample the 10 gestures into a dual-channel format. A detailed illustration is shown in Figure 4.3.5. For one-hand gestures (e.g. hand waving and arm rotation), we sample it as one STEC and put it into one channel. The other channel is padded with zeros. For a gesture involving two hands, if the two hands generate two separate event streams (e.g. air guitar and air drum), we sample them as two separate STECs and arrange them into two channels. If the two hands generate two interactive event streams in space-time (e.g. hand clapping and forearm roll), we sample them as one STEC and put it into one channel. For simplicity, the open class with random gestures is not included.

DVS364 Action Dataset This dataset [84] includes 10 categories of actions with totally 450 recordings. Each recording contains one action and only recording-wise label is given. The starting and ending time of each action within the recording is not provided. Thus, we further manually labeled the starting and ending time of each action for each recording. For each category, two-thirds of the data is used for training while the rest is used for testing.

4.4.2 Effectiveness of Temporally Ordered STECs

In this section, we demonstrate the effectiveness of modeling the event streams as temporally ordered space-time event clouds. We compare the results of both unordered STEC and temporally ordered STEC.

The basic PointNet architecture is used to consume unordered event clouds. A basic ST-EVNet with hierarchical space-time kernels are used to consume temporally ordered event clouds. The configuration of ST-EVNet is $layer1(k^1 = 2, d^1 = 64) \rightarrow layer2(k^2 = 2, d^2 = 128) \rightarrow layer3(k^3 = 2, d^3 = N) \rightarrow max - pooling \rightarrow$

TABLE 4.4.1: **Quantitative evaluation on different datasets.** Metric is global classification accuracy on test trails. For gesture recognition, the sliding window length is 0.5s. For action recognition, the sliding window length is 1s. A number of N (e.g. 256) events are downsampled from the total events within the sliding window. The x, y and t are normalized to $(0, 1)$ accordingly before feeding into the network.

Gesture Recognition			
PointNet		ST-EVNet (basic)	
Inputs	GA[%]	Inputs	GA[%]
256, xyt	89.42	256, xyt	90.52
512, xyt	90.91	512, xyt	91.93
1024, xyt	90.87	1024, xyt	92.91
Action Recognition			
PointNet		ST-EVNet (basic)	
Inputs	GA[%]	Inputs	GA[%]
512, xyt	75.12	512, xyt	82.54
1024, xyt	72.53	1024, xyt	88.75
2048, xyt	67.57	2048, xyt	87.58

$mlp(512, 256, 10)$. The dual-channel architecture of ST-EVNet is not included in this experiment. As shown in Table 4.4.1, for gesture recognition, the ST-EVNet has consistent 1-2% higher accuracies than PointNet under different down-sampling rate. For action recognition, ST-EVNet demonstrates much better accuracies than PointNet. These results demonstrate the importance of treating event clouds as temporally ordered input and using space-time kernels to extract local interactions among temporal neighboring events. In summary, this series of experiments demonstrate that STEC is a superior concept to model an event-based gesture/action and ST-EVNet takes the advantage of space-time kernels to capture local features of STEC for better recognition.

4.4.3 Importance of Dual-channel Architecture

In this section, we prove the rationalities of using a parallel dual-channel to unify single- and dual-gesture recognition. From the results in Table 4.4.2, it can be found that the dual-channel ST-EVNet architecture has a consistent 3-4% accuracy superiority than basic structure under different down-sampling rates N with

TABLE 4.4.2: **Quantitative evaluation on dual-channel architecture.** The configuration of dual-channel ST-EVNet is similar to basic ST-EVNet, except the input size changes to $(N, 2, 3)$. The results of configuring ST-EVNet with different kernel length are also presented. Metric is overall classification accuracy on DVS128 Gesture dataset test trails.

ST-EVNet, $k^1 = k^2 = k^3 = 2$			
Basic		Dual-channel	
Inputs	GA[%]	Inputs	GA[%]
(256,1,3)	90.52	(256,2,3)	94.05
(512,1,3)	91.94	(512,2,3)	95.55
(1024,1,3)	92.91	(1024,2,3)	97.27
ST-EVNet, $k^1 = k^2 = k^3 = 4$			
Basic		Dual-channel	
Inputs	GA[%]	Inputs	GA[%]
(256,1,3)	90.03	(256,2,3)	93.89
(512,1,3)	91.75	(512,2,3)	95.72
(1024,1,3)	92.74	(1024,2,3)	96.30
ST-EVNet, $k^1 = k^2 = k^3 = 8$			
Basic		Dual-channel	
Inputs	GA[%]	Inputs	GA[%]
(512,1,3)	88.65	(512,2,3)	91.17
(1024,1,3)	90.32	(1024,2,3)	93.93

different space-time kernel length k . These results demonstrate that if a dual-hand gesture generates two separate STECs, extracting spatiotemporal features from each STEC respectively provides a more precise description than treating them as a whole. With $k=2$ and $N=1024$, a dual-channel ST-EVNet achieves state-of-the-art accuracy of 97.27%. It proves that the dual-channel ST-EVNet is an efficient unified system to recognize both single- and dual-hand gestures. As STECs are downsampled within the same length of a time interval, the temporally neighboring k events are more likely to cover a wider receptive when N is smaller. A network with a large N and a small k learns more detailed features. On DVS128 Gesture Dataset, we achieve the best accuracies on both basic and dual-channel ST-EVNet with larger $N=1024$ and smaller $k=2$. With only 256 randomly sampled events, the dual-channel ST-EVNet can achieve an impressive accuracy of 94.05%. It further proves the events within a STEC are short-term permutation invariant and long-term permutation variant. Good classification can be achieved as long as the

STECs preserve the long-term trends of gestures.

4.4.4 Network Efficiency Analysis

In this section, we compare the performance of the proposed ST-EVNet with existing methods on event-based gesture recognition in terms of both accuracy and efficiency (Table 4.4.3). Approaches using Spiking Neural Networks (SNNs) achieve good classification results. However, these methods treating events as asynchronous sparse neuron spikes and strongly rely on neuromorphic processors. They are significantly inefficient on conventional processors (e.g. CPUs and GPUs) with synchronous dense processing. A basic ST-EVNet architecture taking only 256 events as input achieves comparable accuracy with PointNet (90.52% vs. 90.20%). In this case, the ST-EVNet model has only half parameters compared with PointNet mode (0.29M vs. 0.55M) and less than a twentieth FLOPs (3.48M vs. 78.58M). A dual-channel ST-EVNet with 256 inputs provides an impressive 4% accuracy improvement compared with the single-channel version, but the computation cost only increases linearly. A light-weighted ST-EVNet model is efficient enough to be geared up on resource-constrained embedded systems for real-time HCI applications.

A dual-channel ST-EVNet with 1024 events as input achieves state-of-the-art accuracy of 97.27%, which is slightly higher than the previous best result of 96.34% using PointNet++. Even though the number of parameters of these two networks are at the same level (1.6M vs. 1.47M), the dual-channel ST-EVNet has only one-seventh computational cost compared with PointNet++ (78.65M vs. 542.14M). A common CPU is enough to cope with a high-accuracy dual-channel ST-EVNet with 1.6M parameters and 1.47M FLOPs. In summary, ST-EVNet is an ultra-efficient dedicated network to process STECs.

TABLE 4.4.3: Efficiency for real-time gesture recognition.

Method	Architecture	GA[%]	#params	FLOPs/sample
Truenorth [33]	SNN (16 layers)	91.77	-	-
SLAYER [21]	SNN (8 layers)	93.64	-	-
PointNet [5]	event-wise mlp + maxpooling	90.20	0.55M	78.58M
PointNet++ [5]	hierarchical PointNet	96.34	1.47M	542.14M
ST-EVNet (N=256,k=2)	space-time kernels	90.52	0.29M	3.48M
ST-EVNet (N=512,k=2)	space-time kernels	91.94	0.55M	11.21M
ST-EVNet (N=1024,k=2)	space-time kernels	92.91	1.7M	39.46M
ST-EVNet (N=256,k=2)	space-time kernels + dual-channel	94.05	0.42M	6.89M
ST-EVNet (N=512,k=2)	space-time kernels + dual-channel	95.55	0.81M	22.29M
ST-EVNet (N=1024,k=2)	space-time kernels + dual-channel	97.27	1.6M	78.65M

4.4.5 Comparison with EventNet

The EventNet [7] and proposed ST-EVNet are two efficient techniques to process event streams in real-time with low latency and they do not rely on GPUs. They are designed according to different perspectives.

Recursive Processing vs. Batch Processing To achieve efficient recursive processing on a large number of raw events, EventNet replaces the event-wise feature extraction containing a limited combination of x and y by a LUT mechanism in inference. However, ST-EVNet downsamples a number of events from each raw event batch and extracts the spatial-temporal features from them in a super-efficient way. The computation complexity of EventNet and ST-EVNet is shown in Table 4.4.4. Although events are generated asynchronously at the pixel level, raw events are packaged for efficient transmission according to standard protocols. The program does not receive an event immediately when it generates. What the program actually receives is a bundle of events each time. Thus, efficient batch processing of events is a more practical implementation on conventional processors(e.g. CPUs).

TABLE 4.4.4: Computational complexity of two efficient event processing techniques.

Method	EventNet	ST-EVNet
FLOPs/events	0.8K	0.02M
random drop	No	Yes
#events processed/s	0.35M	15K
#FLOPs	0.28G	0.3G

Using Timestamp as a Feature Dimension EventNet extracts 2D features and decays the features according to time. It separates the spatial and temporal information. For ego-motion estimation of cameras, it is intuitive to learn the 2D distribution of events on the image plane and decay it temporally to regress the camera pose at each timestamp. However, in action and gesture recognition tasks, considering the spatial and temporal information as a whole is a better description

of a gesture or an action with deformation during motion. In Table 4.4.5, EventNet-based architecture with 2D spatial feature extraction and temporal decay achieves the lowest accuracy on gesture recognition. ST-EVNet taking only (x, y) as input achieves better results due to hierarchical local feature extraction. ST-EVNet taking (x, y, t) as input archives best results. Thus, considering timestamp as a feature dimension identical to (x, y) and extracting local features among the spatiotemporal distribution of space-time event clouds are significant for event-based action recognition tasks.

TABLE 4.4.5: **Using timestamp as a feature dimension.** EventNet uses timestamp as a temporal decay factor, while the concept of space-time event clouds considers timestamp as an identical input feature dimension as x and y . The concept of STEC is a better description of motion with deformation.

EventNet			
Inputs		GA[%]	
256, xy, temporal decay		70.84	
512, xy, temporal decay		65.25	
1024, xy, temporal decay		57.93	
ST-EVNet (basic)			
Inputs	GA[%]	Inputs	GA[%]
256, xy	78.21	256, xyt	90.52
512, xy	78.78	512, xyt	91.93
1024, xy	80.30	1024, xyt	92.91

4.5 Concluding Remarks

Previous work has proved *Space-time Event Clouds* is an efficient modeling of event-based gestures [5]. However, existing techniques consider STECs as unordered space-time points and are inefficient in local spatiotemporal feature extraction of STECs. With an in-depth study of the properties of STECs, we propose a *Space-time Event Net* with hierarchical space-time kernels to summarize spatial and temporal features of STECs simultaneously with great efficiency. On the IBM DVS128 Gesture dataset, the basic ST-EVNet achieves 90.52% accuracy with extremely low computation complexity of 3.48M FLOPs per classification prediction, while

dual-channel ST-EVNet achieves state-of-the-art accuracy of 97.27%. The lightweight characteristic makes ST-EVNet highly suitable for resource-constrained edge-computing scenarios. We believe that STEC modeling and ST-EVNet will contribute to many real-time computer vision tasks, including but not limited to sign-language processing and HCI applications.

Chapter 5

A Portable Event-based HCI System for Real-time Applications

5.1 Introduction

Human-Computer Interaction (HCI) is a cutting-edge topic with increasing impact on the way we interact with machines in desktop applications, smart environments, entertainment, sign language interpretation, virtual reality and gamification [85, 86]. Vision-based gesture recognition techniques are dominant research approaches in this regard. An HCI framework is an integration of a series of tasks, such as hand detection, hand tracking, and gesture classification. Therefore, an HCI application expects a well-designed system dealing with individual subproblems using suitable methodologies. From the aspect of practical applications, response time and cost-efficiency are factors that a comprehensive HCI system should take into consideration [33, 85].

With conventional RGB frames as input, there are two main challenges in gesture recognition. The first challenge is the localization of hands against different backgrounds and light conditions. The other one is to capture hands with greater

motion clarity against rapid gesture movements. Existing methods including increasing frame rates, using depth information and computing optical flow are able to deal with these challenges. Multimodal architectures [65–67] become prevailing in computer vision community to fuse different types of features (e.g. RGB, depth and optical flow). To mitigate interference from backgrounds, researchers adapt detection techniques to achieve hand-focus mechanism [66, 67]. When taking cost-efficiency into consideration, these frameworks are too complicated to be geared up in resource constrained real-time HCI applications.

Recently developed event cameras [7, 68–72] show great potential to solve gesture recognition tasks from a new perspective. An event-based vision system is a motion-focus attention system by nature. Stationary background inducing no intensity changes will not generate output events, whereas only motions inducing intensity changes are recorded as asynchronous event streams. Getting rid of redundant background, event cameras drive detection and tracking of hands motion towards an efficient eye-like way. Without frame constraints, event cameras record motion in the form of a continuous event stream, where each individual event (x, y, t) corresponds to the space-time location when a pixel captures an intensity change. Event streams of gestures embed richer spatiotemporal information than frame-based data, enabling the gesture-based HCI framework to be achieved via a more elegant approach.

With above considerations, in this work, we propose a highly efficient event-based HCI framework which is also extensible for customized purposes. This framework has a backbone consisting of a detector, a tracker and a recognizer. These three parts uses different methodologies to process event data based on characteristics of individual tasks. All the three parts make the most use of the natural features of event data. This ensures that the proposed framework reaches a global optimization of accuracy-efficiency tradeoff. Gestures proposed are dynamic and physical/deictic gestures[87]. These gestures are in line with human cognition and behavior in everyday life, which makes the HCI as natural as an interaction between humans. The experimental results are reported on real-world sensor data,

allowing the proposed system for the sensitive detection of hands, fast-tracking of hands and accurate recognition of gestures under challenging conditions with low latency. The key contributions of our work are as follows:

- A fully event-driven gesture recognition system using a single event camera as sensory input has been presented. The proposed system is a detector-tracker-classifier full-stack solution, which is extensible to many types of interaction, such menu selection, text entry and gaming.
- An event-based detector, which converts events to neuron spikes to produce initial hand location, has been proposed. An event-based tracker, which updates the hand position towards maximum event density within an asynchronizably-updated search region, has been proposed. Events generated by hand gestures are modeled as 3D space-time event clouds and a ST-EVNet has been adopted to learn the spatiotemporal features from temporal local regions.
- The robustness and accuracy of the proposed detector, tracker and classifier have been demonstrated using real-world data. The complete system has been implemented and tested on an embedded platform with an ARM Cortex-A73 processor and achieves a throughput of 100Hz.

5.2 Vision-based HCI Systems

A vision-based HCI system usually consists of three phases, detection, tracking and recognition. Researchers have put numerous efforts into techniques related to the HCI system for more than twenty years. Derpanis et al.[88] did a comprehensive review of vision-based gesture techniques around the last millennium.

Hand detection is the primary phase of an HCI system, which is used to initialize the system. It involves segmenting and localizing the hand from the background.

Researchers have made use of different features to achieve hand detection. Skin-color is an intuitively good candidate to discriminate hands from the image background. To properly model the chromaticity of hands, methods based on different color spaces have been proposed, e.g. HSV[89], YCrCb[90], and YUV[91]. Color-based hand detection suffers from changing illumination conditions[92] and background objects with similar color distribution to human skin[93]. The shape is also used to detect hand [94]. To mitigate the occlusion problem, stereo system[95] and multi-camera settings[96] are adopted. Classifiers based on appearance and texture of hands are also investigated[97]. Boosting algorithms are used to enhance classification performance. Besides the aforementioned features, researchers also make use of motion information to help improve hand detection performance[98, 99].

A classical tracking paradigm is tracking-by-detection[100] that derives from object detection. Under this paradigm, a sliding window is usually adopted to feed image patches into an online updated classifier. The patch candidate with the highest classification score is the updated target position. The performance of the online updated classifier relies on a sampler and a labeler to generate training samples on the fly. Many techniques have been applied to improve the robustness of the sampler and the labeler, e.g. robust loss function[101] and semi-supervised learning[102]. Sam[103] proposed a framework based on a structured support vector machine that improves the accuracy and the robustness of object tracking significantly. Another mainstream of object tracking is trackers based on correlation filter [104]. These trackers update the classifier online with a large number of artificial training samples generated by a circular shift. It is noteworthy that these trackers update the classifier by solving an optimization problem in close-form in the frequency domain. It is a very efficient process that enables a high framerate. Neural Networks are also applied in object tracking. Benefit from offline training, David's work[105] reaches a hundred-level FPS. Event cameras have been used for object tracking for almost a decade. The first attempt is to use event cameras to track high-speed particles in a noise-free environment[106]. In recent years, more sophisticated tracking algorithms based on event cameras have been proposed. Ramesh

proposes a pure event-based tracking-by-detection framework[107]. Mitrokhin proposes an event-based motion compensation tracking framework[108] to make use of optical flow estimators.

Vision-based gesture recognition techniques suffer from interferences from different backgrounds and lighting conditions. To learn more features from hand gestures, multimodal architectures are adopted to fuse different information sources. These sources can be RGB, depth, optical flow, even data from other sensors. Miao[65] leverages the advantages of both residual networks and Convolution 3D networks to fuse data sources and extract features better. In recent work, Wang[66] uses 3DCNN to extract short-term spatiotemporal features. Further, Convolution LSTM is used to learn long-term spatiotemporal features from short-term features. Besides multimodal fusion, Narayana adds a hand-focus mechanism[67]. In recent years, event cameras are gradually investigated for gesture recognition. Wang's work[5] demonstrates that event data has rich spatiotemporal information to complete gesture recognition tasks. In this paper, we demonstrate that the event cameras are especially suitable for motion-based detectors.

5.3 A Portable Event-based HCI System: Design Criteria and Considerations

Different from discrete frames in the conventional vision system, event cameras generate event streams driven by motion. Such event streams have an ultra-high temporal resolution. The proposed HCI system makes the most of dedicated characteristics of event cameras. Benefit from the motion-driven mechanism, we design a detector to detect a hand waving to initialize the system. Each event is treated as a neuron spike that contributes to a neuron response signal. The neuron response signal predicts a hand position when reaching a predefined threshold. Due to the high temporal resolution of event data, the tracker in the proposed system

is designed to update the position event-wisely. Conventional hand tracking relies on feature matching between frames. This methodology suffers from several challenges, e.g. motion blur, deformation, etc. However, event cameras can capture the complete trajectory of a moving hand. The proposed tracker updates the prediction to the position of maximum event density without feature matching. Based on the tracking position, events within a space-time volume are passed into ST-EVNet to do the final gesture classification.

The purpose of this work is to build up a purely event-based, real-time HCI system, applicable to a resource-constrained embedded system. The design criteria can be derived from this purpose as low latency, low computation cost, and high accuracy. The aforementioned design choices stick to the design criteria tightly. With the waving hand detector, computational hand detections among the whole frame are not required. This behavior is also consistent with human's daily habits to provide a good user experience. In conventional hand tracking systems, most computation costs are spent on complicated feature matching between frames. Taking advantages of the outstanding motion capturing capability of event cameras, the proposed tracker updates hand positions towards maximum event density. This tracker not only reduces the computational cost but also provides very accurate and smooth tracking results. The adopted ST-EVNet classifier demonstrates impressive gesture recognition performance. It also has reduced network complexity compared with other networks applied to event data. Besides block-level performance optimization, we also provide system-level configurable settings to enable performance enhancement and compatibilities to the downstream. The event data is acquired by a configurable acquisition interval, which can be used to adjust the output frequency to be compatible with downstream devices or applications. We also empirically demonstrated that down-sampling can be used within each interval to reduce computation cost without deteriorating accuracy.

FIGURE 5.3.1: The proposed HCI system includes an event-based hand detector, and event-based tracker and an event-based classifier.

5.4 The Event-based Hand Detector

Humans usually wave spontaneously to get some one's attention and start communication [87]. The proposed detector mimics this natural behavior of humans and the interaction system is designed to be initialized when the user is intentionally waving at the camera. The basic idea is to treat each event as a neuron spike. According to the generative model of event camera, the waving of hand generates a large number of events (spikes), which are converted to a neuron response signal. Once the response signal reaches a pre-defined threshold, a new neuron spike will

be generated indicating the position of hand as well as triggering a new interaction session.

FIGURE 5.4.1: (a)The detection framework. (b)Working principle of the mass layer and lifetime layer.

The detector structure is shown in Fig 5.4.1(a). The overall input image plane is divided into a grid of M by N tiles. A mass layer is introduced to detect areas bursting out a large number of events within a short time interval. When a pre-defined threshold is reached, a new spike is generated at an active area and sent to the lifetime layer. The lifetime layer records the lifetime of each active area and only the area keeping activating for a sufficient time is considered as a candidate

hand location. Events fired at pixel (x, y) is treated as a spike train

$$s_{x,y}(t) = \sum_i \delta(t - t_i) \quad (5.4.1)$$

For a neuron (m, n) in the mass layer, it receives events (spikes) from all pixels within the receptive field

$$s_{m,n}^1(t) = \sum_{x,y} s_{x,y}(t) \quad (5.4.2)$$

According to Spike Response Model (SRM)[21, 109], a spike train is converted into a spike response signal by a spike response kernel. In mass layer, the spike response signal $u_{m,n}^1(t)$ of a neuron (m, n) is obtained by convolving $s_{m,n}^1(t)$ with the mass spike response kernel $\varepsilon^1(\cdot)$, which can be written as

$$u_{m,n}^1(t) = (\varepsilon^1 * s_{m,n}^1)(t) \quad (5.4.3)$$

For each neuron, an output spike is generated whenever $u_{m,n}^1(t)$ reaches the pre-defined mass threshold V_{th}^1 . The spike function $f^1(u_{m,n}^1)$ of mass layer is defined as

$$f^1(u_{m,n}^1) : u_{m,n}^1 \rightarrow s_{m,n}^2 \quad (5.4.4)$$

$$s_{m,n}^2(t) := s_{m,n}^1(t) + \delta(t - t_i), \text{ where } u_{m,n}^1(t) \geq V_{th}^1 \quad (5.4.5)$$

The output spike train $s_{m,n}^2(t)$ of node (m, n) in mass layer is the input spike to neuron (m, n) in the lifetime layer. Similarly, the spike response signal $u_{m,n}^2(t)$ is obtained by convolving $s_{m,n}^2(t)$ with the lifetime spike response kernel $\varepsilon^2(\cdot)$.

$$u_{m,n}^2(t) = (\varepsilon^2 * s_{m,n}^2)(t) \quad (5.4.6)$$

The spike function $f^2(u_{m,n}^2)$ of lifetime layer is defined as

$$f^2(u_{m,n}^2) : u_{m,n}^2 \rightarrow s_{m,n}^3 \quad (5.4.7)$$

$$s_{m,n}^3(t) := s_{m,n}^3 + \delta(t - t_i), \text{ where } u_{m,n}^2(t) \geq V_{th}^2 \quad (5.4.8)$$

Once a spike is fired at $s_{m,n}^3$, block (m, n) is treated as a candidate hand location.

5.5 The Event-based Hand Tracker

The tracker, which is inspired by [110], aims to provide hand positions given an initial detection result obtained in previous section. The basic idea is to update the position towards maximum event density within the asynchronously-updated search region, which is established for robustness. As illustrated in Fig. 5.5.1(a), the search region is centered at current tracking position. Outside the search region, we define four areas where events are counted for hand direction estimation.

The tracking framework is shown in Figure 5.5.1(b). Once an event vector is obtained, for each event, the asynchronized tracker determines if the event can be used for computing the tracking center. An extra offsetting step, where immediate hand direction is estimated, is required for better tracking performance under fast-moving scenarios. Lastly, a Kalman filter is implemented to balance quick response and trajectory smoothness.

5.5.1 The Event-based Asynchronized Tracking

The raw events from the sensor are too many to be processed in real-time. We firstly downsample raw events at a fixed rate for consistent performance under varying lighting conditions and background scenes. Then, the events are sorted in chronological order. For each sorted event, we check the coordinates to determine

FIGURE 5.5.1: (a) The search region and extra search regions; (b) The tracking framework.

if it falls within any extra search region. Otherwise, if the event is within the search region, save the event into a list $\{e_i\} = \{x_i, y_i, t_i\}$, where i is the event index. To facilitate tracker's dynamic performance, the search region is updated at a higher frequency by averaging positions of recent events within the search region. The tracking center is computed at a lower frequency from

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{(t_i - t_0)}{\sum_{\forall i} (t_i - t_0)} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (5.5.1)$$

where t_0 denotes the timestamp of the first event in this vector. If there is not enough events in the search region, the tracker stops updating to avoid fluttering.

5.5.2 Search Region Offsetting for Fast Hand Tracking

Hand directions are defined by states

$$s = \{\text{upper left, up, } \dots, \text{lower right}\} = \{s_1, s_2, \dots, s_9\} \quad (5.5.2)$$

where s_5 denotes the unknown or holding state. Since the wrist would produce events while moving, we measure current hand direction deterministically by simply comparing the event number accumulated in extra search regions. For example, more events in the lower extra search region may result in hand direction of "up". In particular, the measured direction is denoted as m_j , corresponding to the state with the same subscript. The hand direction measurement model can be written as a conditional probability:

$$\mathbb{P}(m_j|s_i) = \begin{cases} p, & \text{if } i = j \\ \frac{1-p}{8}, & \text{if } i \neq j \end{cases} \quad (5.5.3)$$

where p is a tunable parameter that represents measurement confidence level; m_j and s_i denote the measurement and state, respectively. Since we have the gesture detection module which classifies released or holding hands, it is assumed in the tracker that s_5 is disabled, thus we require $i \neq 5$.

The Bayesian framework is used in hand direction estimation, which aims to obtain probabilities for all possible states. By assuming the discrete state transition matrix from s_i to s_j as $\mathbb{P}(s_j|s_i) \in \mathbb{R}^{8 \times 8}$, the estimator can be written as the "*prediction-correction*" convention. For each iteration, we firstly compute the current prior probabilities from last-time posterior probabilities and the transition matrix for all states:

$$\mathbb{P}(s_i) = \sum_j \mathbb{P}(s'_j) \mathbb{P}(s_i|s_j) \quad (5.5.4)$$

where $\mathbb{P}(s'_j)$ denotes the last-time posterior probability of state s_j . Then, we derive the current posterior probabilities with measurement:

$$\mathbb{P}(s_i|m_j) = \frac{\mathbb{P}(m_j|s_i)\mathbb{P}(s_i)}{\mathbb{P}(m_j)} \quad (5.5.5)$$

where $\mathbb{P}(m_j)$ is the normalization constant which satisfies

$$\sum_i \mathbb{P}(s_i|m_j) = \frac{1}{\mathbb{P}(m_j)} \sum_i \mathbb{P}(m_j|s_i)\mathbb{P}(s_i) = 1 \quad (5.5.6)$$

$$\Rightarrow \mathbb{P}(m_j) = \sum_i \mathbb{P}(m_j|s_i)\mathbb{P}(s_i) \quad (5.5.7)$$

The state with maximum probability is deemed as the estimated hand direction.

The proposed tracking framework is efficient but has little understanding of event source. Thus, the tracker may be misled into following other objects, especially the wrist, when the hand direction is similar to the direction of fast motion. In this work, we propose to offset the search region towards hand direction to avoid such tracking failures. An example can be seen in Figure 5.5.1(a), where the search region is offset upwards to include possible events from an up-moving hand.

5.6 Event-based Classifier

In the proposed gesture interaction system, once a valid gesture is detected, the system will output control signal to its host and the host user interface should react correspondingly. The events generated by hand movements within the bounding volume built up by the proposed tracker are modeled as temporally ordered space-time event clouds and ST-EV Nets are adopted for gesture recognition.

5.6.1 Representation of Gestures

Wang et al. in [5] interpreted an event-based gesture as a space-time event cloud (STEC). The events generated by a gesture is presented as a set of unordered

points in the 3D Space-time Continuum, in which the time axis in STEC becomes a numerical measurement instead of sequential information. In this perspective, to recognize a gesture is equivalent to recognize the geometric distribution of a STEC, which is in spirit similar to recognizing an object by its 3D point cloud. Considering that STEC of a gesture are generated in temporal order in accordance with hand movements. A STEC is better represented as a sequence of space-time points in temporal order rather than a set of unordered space-time points:

$$G = (\mathbf{e}_i)_{i=1}^N = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N) \quad (5.6.1)$$

As a result, the spatiotemporal feature and the local interaction among events can be better extracted.

5.6.2 Constructing STEC Using a Dynamic Buffer

The time duration covered by each input event vector is small and depends on data acquisition rate. If the data acquisition rate is set to 100Hz, each event vector contains all the events triggered during the last $T_{vector} = 10\text{ms}$. As gesture recognition usually need to observe data in a longer duration, a dynamic classification buffer illustrated in section 3.6.4 is introduced to store the space-time event clouds generated by hand within a pre-defined classification time window $T_{classification} = M \times T_{vector}$.

Within each event vector, when the event-based asynchronized tracker is updating the tracking center, the events within the classifier search region generated by that tracking center are stored to a list $s_{vector} = \{e_i\}$ simultaneously. The classifier search region and the tracker search region share the same tracking center, while they are set to different size. The list $s_{vector} = \{e_i\}$ are sampled to N_{vector} events and pushed to the classification buffer, while the oldest N_{vector} events are popped from the buffer. Thus, the dynamic buffer maintains $N_{vector} \times M$ events from a

time duration of $T_{classification} = M \times T_{vector}$. As shown in Figure 5.6.1(a), the classification buffer stores the STEC generated by the moving hand within a bounding volume dynamically updated by the event-based asynchronized tracker.

FIGURE 5.6.1: (a) As the tracking center is updated under a event-based perspective, a very smooth bounding volume is generated. (b) The space-time event cloud within the dynamic bounding volume will be normalized to a $1 \times 1 \times 1$ cube. (c) The ST-EVNet is applied for gesture classification

5.6.3 Classification of STECs

Learning interaction among local regions plays a significant role in point clouds-based analysis. In temporally ordered STEC, consecutive events form a temporal local region. Events within the temporally local region are induced by hand motion almost at the same time but they are not necessarily close in Euclidean distance. As shown Fig. 5.6.1(c), the Space-time Event Net (ST-EVNet) is a network architecture with hierarchical space-time kernels, which is used to learn spatiotemporal features of STECs. Local spatiotemporal feature is learned by convolving events within the temporal local region with the space-time kernel. The output feature f

is computed as

$$\mathbf{f} = \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_k \end{bmatrix} \otimes \phi_1, \quad \phi_1 \in \mathbb{R}^{k \times 1 \times 3 \times d} \quad (5.6.2)$$

The kernel length k determines the number of events in the temporal local region. The kernel depth d determines the output channels of local geometric features. The final abstracted features at layer L undergo an element-wise max-pooling to get the global spatiotemporal feature vector for final classification:

$$\mathbf{g} = \text{MAX}\{\mathbf{f}_1^L, \mathbf{f}_2^L, \dots, \mathbf{f}_N^L\} \quad (5.6.3)$$

5.7 Experiments

In the experiments, the robustness and accuracy of the detector, tracker and classifier are evaluated using real-world data. All the testing data are collected using a CeleX5 event camera with a resolution of 800×1280 . The entire system is developed using C++ and implemented on a embedded platform with ARM Cortex-A73 dual core processor. The proposed event-based HCI system is compared with existing solutions.

5.7.1 Detector Robustness

In the experiment, the image plane is divided into 25×40 tiles, which means both the mass layer and lifetime layer consist of 25×40 neurons. Each neuron in the mass layer receives event signals from a 32×32 pixel array. Each neuron in the lifetime layer receives a spiking signal from the corresponding neuron in the mass layer. The mass layer spike response kernel ε^1 is set as $\exp(\frac{t_{i-1}-t_i}{\tau_1})$, where τ_1 is set to 500. The lifetime layer spike response kernel ε^2 is set as $\exp(\frac{t_{i-1}-t_i}{\tau_2})$, where τ_2 is

set to 5000. The mass layer output spiking threshold V_{th}^1 is 10. The lifetime layer output spiking threshold V_{th}^2 is 20.

The robustness of the detector is evaluated under both normal and dim light conditions. The normal light condition is daytime with the light on in the indoor environment. The dim light condition is daytime with the light off and closed curtains. The interference is also considered. A person is walking around to produce interference event signals. Fig. 5.7.1 and Fig. 5.7.2 present the corresponding map layers' and lifetime layers' spike response signals strength changes before the initial hand position is detected under normal light condition and dim light condition respectively. The darker the color, the stronger the signal strength. The red dot indicates the initial hand position detected. By keeping waving hand, the neurons in the mass map receive continuous events (spikes) generated at the hand location. The continuous spike trains induce strong response signals at corresponding locations. The strong response signals generate output spikes and send to neurons at corresponding locations at lifetime layer. Once the response signal at certain location in the lifetime reaches the pre-defined threshold, a candidate hand location is detected and sent to the hand tracker. On the contrary, background interference motion also induces activate neurons in mass layer. However, these response signals will decay in quickly and they are unlikely to accumulate in the lifetime layers.

Fig. 5.7.3 presents the corresponding map layers' and lifetime layers' spike response signals strength at the moment when the hand locations are detected. As the event camera generates noisy events over the entire image plane, the mass layers have a large number of inspired neurons during a short time interval. However, the response signals inspired by noisy events will decay quickly and will not be able to reach the mass threshold to generate an output spike to the lifetime layer. The mass layers have more active neurons in dim light conditions due to the decreasing SNR of event signals when the illumination is decreasing. Overall, the detector is capable of providing an accurate hand position in different light conditions even with interference.

FIGURE 5.7.1: The mass layer's and the lifetime layer's response signals strength change under normal light condition are shown in (a)-(e). The hand location is detected at $T=2.9s$. The 2D event frame is the accumulation of events in last 100ms.

FIGURE 5.7.2: The mass layer's and the lifetime layer's response signals strength change under dim light condition are shown in (a)-(e). The hand location is detected at $T=4.1s$. The 2D event frame is the accumulation of events in last 100ms.

FIGURE 5.7.3: The mass layer's and the lifetime layer's response signals strength at the moment when the hand locations are detected. The 2D event frame is the accumulation of events in last 100ms.

5.7.2 Tracker Robustness

Experiments Setup

As the gesture classifier only takes the tracked region as classification input, a robust tracker against different hand moving speeds is needed. To evaluate the stability of proposed tracker against different speed, a test system with a mechanical sliding rail and a hand model is established. The hand model is fixed to the sliding rail and a motion motor is used to control the speed of the sliding rail. The

event camera is located three meters away from the sliding rail to simulate real application scenarios. Different sliding rail speeds used in experiments is shown in Table 5.7.1. The motion motor setting of 1000 4000 corresponds to normal hand moving speed in a HCI scenario, while the speed setting over 10000 corresponds super fast hand movements.

TABLE 5.7.1: Different sliding rail speeds used in experiments. The motion motor settings and corresponding speeds reflected on the image plane are estimated.

Motion motor param	Speed(pixels/second)
1000	112.13
2000	202.07
4000	347.08
6000	485.04
8000	607.35
10000	717.57
12000	821.56
14000	935.58

For each speed setting, the hand model travels ten round trips horizontally along the sliding rail. Under the unit test of tracker performance, the initial tracking position is given. The tracker output frequency is 60Hz. As shown in Figure 5.7.4, the groundtruth tracking position is manually labeled. The tracking search region is set as a 100 pixels $\times$ 100 pixels square centered at the tracking center, while the extended search region is set as 150 pixels $\times$ 150 pixels square centered at the tracking center. The tracking search region size is decided based on the sensor resolution(800 $\times$ 1280) and the averaging hand size captured in the sensor plane when the user is three meters away from the sensor.

Evaluation of Tracker Stability

The tracking position offsets over different speed settings are shown in Table 5.7.2. The visualization of tracker output compared with groundtruth are shown in Figure 5.7.5. Under normal hand moving speed, the tracking position offsets on both horizontal and vertical directions are around 5 pixels, while the standard deviations on both directions are around 2 pixels. In the experiments, the sliding rail

FIGURE 5.7.4: Tracker search region and extended search region settings.

moves horizontally. Thus, the tracker performance over different speeds mainly reflects in the horizontal direction. The maximum speed setting of 14000 is approximately 935.58 pixels/second, which means the hand moves from the leftmost side to the rightmost side of the image only uses one second. The averaging shift of the horizontal tracker output increases from 5.28 pixels to 27.85 pixels as the speed increases from 1000 (112.13 pixels/second) to 14000 (935.58 pixels/second). The standard deviation increases from 1.67 pixels to 6.41 pixels accordingly. The averaging shift of the vertical tracker output increases gradually from 4.42 pixels to 6.68 pixels as the speed increases, and the standard deviation remains around 2.5 pixels. The proposed tracker performance well under normal hand moving speeds. Under the maximum speed setting of 14000, according to the three-sigma rule, 99.7% of the tracker output shift is within the interval [8.62 ,47.08]. Thus, the tracker is robust and unlikely to lose track of target objects with a search region radius of 50 pixels in high-speed conditions. According to the experiment results, the proposed tracker is robust and stable over different hand moving speeds, which provides reliable input to the gesture classifier in the next stage.

5.7.3 Classifier Accuracy

Designed HCI Gestures

The proposed system is a mouse-like control system with six designed gestures.

TABLE 5.7.2: Stability of proposed tracker against different speeds.

speed	Horizontal direction error		Vertical direction error	
	Mean μ	Standard deviation σ	Mean μ	Standard deviation σ
1000	5.28	1.67	4.42	2.93
2000	3.49	2.21	5.25	2.24
4000	10.15	2.94	6.14	2.20
6000	15.88	4.13	5.09	2.57
8000	16.99	3.89	4.72	2.50
10000	19.41	4.53	5.72	2.40
12000	23.96	4.99	6.31	2.12
14000	27.85	6.41	6.68	1.62

(a)

(b)

FIGURE 5.7.5: Visualizing tracking results with groundtruth over different speed. (a)horizontal direction (b)vertical direction

FIGURE 5.7.6: Event clouds of gestures used in the experiments and its equivalent shot using RGB cameras. (a)grab. (b)release. (c)up-down. (d)left-right. (e)clockwise. (f)counter-clockwise.

The event clouds of designed gestures captured using an event camera and their corresponding shots using a RGB camera are shown in Figure 5.7.6. The *grab* and *release* are a pair of gestures to implement a *drag-and-drop* functionality. When the user moves to the target item and performs the *grab* gesture, the item will be selected. The user keeps holding the *grab* gesture and move. The selected item will move along with the user's hand. Once the desired position is reached, the user performs the *release* gesture and the item will be dropped. The *up-down* gesture implements the *confirm* functionality and the *left-right* gesture implements the *return* functionality. The *clockwise* and *counter-clockwise* gestures implement the page turning functionality.

Classifier Training

The total trained classes include the designed six gestures and one class of random gestures. Proposed gesture interaction system is targeting for embedded platforms, a simplified version of ST-EVNet is applied. The number of input events is 256. The configuration of the simplified ST-EVNet is $layer1(k^1 = 4, s^1 = 4, d^1 = 32) \rightarrow layer2(k^2 = 4, s^2 = 4, d^2 = 32) \rightarrow layer3(k^3 = 4, s^3 = 4, d^3 = 128) \rightarrow maxpooling \rightarrow mlp(64, 7)$.

TABLE 5.7.3: Test accuracy of designed gestures.

Gestures	Precision	Recall	F-score
Grab	0.872	0.902	0.886
Release	0.887	0.942	0.914
Up-down-wave	0.945	0.950	0.947
Left-right-wave	0.988	0.988	0.988
Clockwise-wave	0.982	0.990	0.986
Counterclockwise-wave	0.988	0.986	0.987
Others	0.888	0.804	0.849
Average	0.936	0.937	0.937

5.7.4 System Evaluation

In this experiment, the entire system is developed using C++ and implemented on an embedded platform with ARM Cortex-A73 dual-core processor. A single

FIGURE 5.7.7: In the TV control application:(a)the user waves hand up and down to enter a program. (b)the user waves hand left and right to quite a program. (c)the user waves hand clock-wisely to awake a volume bar. (d)the user moves hand to adjust the volume.(e)the user waves hand counter clock-wisely to awake a hidden menu.

CeleX5[111] event camera is used as the sensory input connecting to the embedded platform. The classifier model is trained using TensorFlow. To fully optimize the computation efficiency, the inference is customized using Eigen3 in C++, which is independent of TensorFlow. The comparison of the proposed system with existing solutions is shown in Table 5.7.4. Comparing with conventional methods, the proposed event-driven system achieves the best throughput with considerably good recognition accuracy. The great improvement in response time has great potential

to provide a much better interactive experience for the users.

To better demonstrate the effectiveness of the proposed system, the system is configured for a TV control application. As shown in Fig. 5.7.7, the *up-down* gesture is configured as a *select-and-confirm* command. The user moves to the target program and waves *up-down* to enter the program. The *left-right* gesture is configured as a *return* command. The user waves *left-right* to quit the program. The user waves *clock-wises* gesture to awaken a volume bar and moves freely to adjust the volume.

TABLE 5.7.4: Comparison with existing hand gesture recognition systems.

	[112]	[113]	[114]
Sensory device	Realsense 3D	Kinect V1	Single RGB
Number of gestures	6	5	6
Classification accuracy	98%	90.70%	84.07%
Motion detection	yes	yes	no
Gesture tracking	yes	no	no
Throughput	15Hz	20Hz	30Hz
Scenes	Complex	Office	Pure white
Platform	PC	Mobile Devices	PC
	[115]	[116]	This work
Sensory device	Single RGB	Single RGB	CeleX5
Number of gestures	4	4	6
Classification accuracy	97%	90%	93.60%
Motion detection	no	yes	yes
Gesture tracking	yes	yes	yes
Throughput	15Hz	25Hz	100Hz
Scenes	Pure white	Complex	Home/office
Platform	PC	Embedded	Embedded

5.8 Concluding Remarks

In this work, the first fully event-driven gesture recognition system using a single event camera as sensory input has been presented. The proposed system is a detector-tracker-classifier full-stack solution, which is extensible to many types of interaction, such as menu selection, text entry and gaming. The complete system has been implemented and tested on an embedded platform with an ARM

Cortex-A73 processor and achieves a throughput of 100Hz. Due to the motion-focus nature of event cameras, the event-driven HCI system is capable of providing ultra-fast response time and excellent accuracy with the least computation cost. The proposed system demonstrates the great potential of using event cameras in real-time human-computer interaction with edge computing.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

As a neuromorphic sensor, an event camera is able to sense and track extremely fast motions with low latency and high dynamic range. These advantages make event cameras extremely suitable to solve fast motion-related vision tasks. There are many obstacles to the development of event-based vision solutions. So far in this thesis, efficient representations of event data and corresponding algorithms are developed to analyze the event streams and to overcome the challenges of event-based processing on synchronous processors.

For a better event data representation, a novel concept of Space-time Event Clouds (STEC) is proposed to interpret an event stream as a 3D point cloud. Under the space-time event cloud perspective, the intrinsic space-time entanglement in event data is preserved. The spatiotemporal features are extracted directly and simultaneously from the geometric distribution of event clouds. On the one hand, the noise effect of event clouds is solved by the proposed subspace feature sampling-based classification scheme. On the other hand, the computation efficiency is largely increased without extra noise removal operations. To the best of our knowledge, this is the first time to interpret event streams as space-time event clouds for gesture

recognition problems. Experimental results validate the proposed space-time event cloud concept as an effective representation to characterize the event streams. It preserves both spatial and temporal information to analyze the event streams and is end-to-end learnable.

For efficient feature learning of Space-time Event Clouds (STEC), the Space-time Event Net (ST-EVNet) architecture with hierarchical space-time kernels is proposed. Experimental results reveal that the hierarchical space-time kernels learn the local spatiotemporal features in a unified way and aggregate the learned spatiotemporal features with great efficiency. A significant performance improvement is observed when applying the dual-channel architecture of ST-EVNet on gesture recognition tasks. The light-weighted characteristic makes ST-EVNet highly suitable for resource-constrained edge-computing scenarios. It is believed that STEC modeling and ST-EVNet will contribute to many real-time computer vision tasks, including but not limited to sign-language processing and HCI applications.

In the subsequent pieces of work in Chapter 5, a fully event-based HCI framework is presented, which demonstrates the possibility of building a fully event-based HCI system on a synchronous processor while keeping advantages of the asynchronous event-based sensing. The proposed detector-tracker-classifier full-stack solution is extensible to many types of interaction, such as menu selection, text entry and gaming. The complete system has been implemented and evaluated on an embedded platform. Due to the motion-focus nature of event cameras, the proposed event-driven HCI system is capable of providing ultra-fast response time and excellent accuracy with the least computation cost. The proposed system demonstrates the great potential of using event cameras in real-time human-computer interaction with edge computing.

It is believed that these contributions will provide the basis for the further use of event cameras to overcome the challenges of computer vision and allow researchers to make use of their advantages to enhance the stability, robustness and performance of these solutions. However, there are also open concerns that need to be

answered to the event camera, as well as the shortcomings of our methods, which we address in the next section.

6.2 Discussion and Future Work

This study focuses on the efficient modeling and processing of the event data. However, this work is not comprehensive enough, and there are still many aspects worth exploring.

As illustrated in Section 3.3, a global subspace sampling scheme is proposed to deal with the noise effect of raw events and to ensure an effective and efficient classification of space-time event clouds. By applying the proposed global subspace sampling scheme in the training phase, randomly sampling N events from the raw event data to form a sub-space-time event clouds is able to achieve comparable classification results. However, further explorations can be done to sample sub-space-time events according to the pointwise importance w.r.t the classification performance. In [117], Zheng *et al.* proposed a novel way of characterizing critical points and building saliency maps 3D point clouds. Every point is given a score that indicates its contribution to the recognition loss. Explicitly, the saliency map describes which points are essential to model recognition. The aggregations of highly-scored points in a point-cloud imply large subsets. In this approach, the intuition is that the recognition outcome is to be decided by all the external outer points of a point cloud, since such points represent structure details of 3D objects, while the points close the point center have no effect on the efficiency of the recognition. This approach may not be entirely applicable to clouds of space-time occurrences, but it provides at least a good starting point and demonstration of how to study this topic.

In Section 4.4, we have demonstrated the significant performance improvement when feeding the event clouds of two hands correspondingly to the dual-channel

architecture of ST-EVNet. However, the method to segment the event clouds generated two hands is undetermined. In [118], a per-event segmentation approach was proposed by Stoffregen *et al.* to split a scene into separately moving objects. This method jointly estimates the associations of events generated by different objects and the motion parameters of the objects by maximizing an objective function that builds upon recent results on event-based motion-compensation. However, distinguishing between events caused by two hands with continuous deformation over time is a much more challenging task to be studied.

With this, we conclude our discussion about the event camera. What impact this sensor has on the computer vision community remains to be seen. However, in conclusion, we agree that whether or not the event camera can have a lasting effect is a valuable area of study as it helps us to re-evaluate our approaches and determine whether the solutions we have selected are really the most optimal technologies for the problems we have selected to solve.

List of Publications

Journal Article

- Jiang Rui, **Wang Qinyi**, Shi Shunshun, Mou Xiaozheng, and Chen Shoushun, “Flow-Assisted Visual Tracking Using Event Cameras,” *CAAI Transactions on Intelligence Technology*, 2021.
- Jiang Rui, Mou Xiaozheng, Shi Shunshun, Zhou Yueyin, **Wang Qinyi**, Dong Meng, and Chen Shoushun, “Object Tracking on Event Cameras with Offline-Online Learning,” *CAAI Transactions on Intelligence Technology*, 2020.

Conference Proceedings

- **Wang Qinyi**, Zhang Yexin, Yuan Junsong, and Lu Yilong, “Space-time Event Clouds for Gesture Recognition: from RGB Cameras to Event Cameras,” in *Proc. IEEE Winter Conf. on Applications and Computer Vision (WACV’19), 2019 (Oral)*.
- **Wang Qinyi**, Chen Shoushun, “A low power prediction SAR ADC integrated with DPCM data compression feature for WCE application,” in *Proceedings of IEEE Biomedical Circuits and Systems Conference (BioCAS), 2016*.

Bibliography

- [1] A. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Robotics: Science and Systems (RSS)*, 2018. [xii](#), [17](#), [18](#), [19](#)
- [2] Y. Wang, B. Du, Yiran Shen, K. Wu, G. Zhao, J. Sun, and H. Wen. Ev-gait: Event-based robust gait recognition using dynamic vision sensors. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019. [xii](#), [18](#), [19](#)
- [3] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. [xii](#), [18](#), [20](#)
- [4] L. Wang, S. M. Mostafavi I, Y. Ho, and K. Yoon. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019. [xii](#), [21](#), [22](#)
- [5] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu. Space-time event clouds for gesture recognition: from rgb cameras to event cameras. In *Proceedings of the IEEE Winter Conference on Applications and Computer Vision*, 2019. [xiv](#), [30](#), [65](#), [66](#), [67](#), [68](#), [69](#), [80](#), [82](#), [88](#), [96](#)
- [6] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos. Graph-based spatial-temporal feature learning for neuromorphic vision sensing. In *arXiv:1910.03579*, November 2019. [xvii](#), [27](#), [30](#)
- [7] Y. Sekikawa, K. Hara, and H. Saito. Eventnet: Asynchronous recursive event processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. [xvii](#), [10](#), [27](#), [31](#), [33](#), [36](#), [63](#), [67](#), [75](#), [81](#), [85](#)
- [8] A. F. Pelé. Neuromorphic promises better ai, 2019. URL <https://www.eetimes.com/neuromorphic-promises-better-ai/>. [1](#)
- [9] C. Mead. Neuromorphic electronic systems. In *Proceedings of the IEEE*, volume 78, page 1629–1636, 1990. doi: 10.1109/5.58356. [2](#)
- [10] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous mosaicing and tracking with an event camera. In *Proceedings of British Mach. Vis. Conf. (BMVC)*, 2014. [5](#)

- [11] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck. Asynchronous event-based binocular stereo matching. *IEEE Trans. Neural Netw. Learn. Syst.*, 23(2):347–353, 2012. 6
- [12] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza. “emvs:event-based multi-view stereo—3d reconstruction with an event camera in real-time. *IEEE Trans. Neural Netw. Learn. Syst.*, 126(12):1394–1414, 2018.
- [13] C. Reinbacher, G. Munda, and T. Pock. Real-time panoramic tracking for event cameras. In *Proceedings of IEEE Int. Conf. Comput. Photography*, 2017. 8
- [14] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Trans. Robot.*, 34(6):1425–1440, 2018. 6
- [15] J. Kogler, C. Sulzbachner, and W. Kubinger. Bio-inspired stereo vision system with silicon retina imagers. In *Proceedings of Int. Conf. Comput. Vis. Syst. (ICVS)*, 2009. 6
- [16] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.
- [17] M. Liu and T. Delbruck. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. In *Proceedings of British Mach. Vis. Conf. (BMVC)*, 2018.
- [18] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018.
- [19] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. Eklt: Asynchronous photometric feature tracking using events and frames. *Int. J. Comput. Vis.*, 2019. 6
- [20] E. Stamatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco. An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. In *Front. Neurosci.*, 2017. 8
- [21] S. B. Shrestha and G. Orchard. Slayer: Spike layer error reassignment in time. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1412–1421. Curran Associates, Inc., 2018. 8, 22, 66, 80, 92
- [22] APT Advanced Processor Technologies Research Group. Spinnaker project - the spinnaker chip. URL <http://apt.cs.manchester.ac.uk/projects/SpiNNaker/SpiNNchip/>. 11

- [23] Y. Yan, D. Kappel, F. Neumarker, J. Partzsch, B. Vogginger, S. Hoppner, S. Furber, W. Maass, R. Legenstein, and C. Mayr. Efficient reward-based structural plasticity on a spinnaker 2 prototype. *IEEE Transactions on Biomedical Circuits and Systems*, 13(3):579–591, 2019. [11](#)
- [24] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, and et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6179):668–673, 2014. [11](#), [25](#), [26](#)
- [25] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, and et al. S. Jain. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. [11](#)
- [26] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. [20](#), [21](#)
- [27] Z. Yi, H. R. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. [21](#)
- [28] S. Seifozakerini, B. Zhao W. Y. Yau, and K. Mao. Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. [21](#), [22](#)
- [29] H. Li, J. Pei, and G. Li. Real-time tracking based on neuromorphic vision. In *2015 15th Non-Volatile Memory Technology Symposium (NVMTS)*, pages 1–7, 2015. [22](#)
- [30] H. Wang, J. Xu, Z. Gao, C. Lu, S. Yao, and J. Ma. An event-based neurobiological recognition system with orientation detector for objects in multiple orientations. *Frontiers in Neuroscience*, 10:498, 2016. [22](#)
- [31] M. Osswald, S.-H. Ieng, R. Benosman, and G. Indiveri. A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems. *Sci Rep*, 7(1), 2017. [22](#)
- [32] J. Wu, Y. Chua, and H. Li. A biologically plausible speech recognition framework based on spiking neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, 2018. [22](#)
- [33] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. D. Nolfo, T. N., A. Andreopoulos, G. Garreau, M. Mendoza†, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, July 2017. doi: 10.1109/CVPR.2017.781. [22](#), [25](#), [26](#), [41](#), [44](#), [55](#), [56](#), [59](#), [62](#), [66](#), [76](#), [80](#), [84](#)

- [34] J. H. Lee, T. Delbruck, and M. Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 2016. [25](#)
- [35] G. K. Cohen, G. Orchard, S. Leng, J. Tapson, R. B. Benosman, and A. Schaik. Skimming digits: Neuromorphic classification of spike-encoded images. *Frontiers in Neuroscience*, 2016. [25](#)
- [36] B. Ramesh, H. Yang, G. Orchard, N. Thi, and C. Xiang. Dart: distribution aware retinal transform for event-based cameras. In *CoRR*, 2017. [25](#)
- [37] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, A. Amir P. Datta, B. Taba, M. D. Flickner, and D. S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016. [26](#)
- [38] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process*, 2015. [30](#)
- [39] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognit*, 2016. [30](#)
- [40] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognit*, 2017. [30](#)
- [41] C. R. Qi and H. Su, K. Mo, and L. J. Guibas. Pointnet: deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [32](#), [35](#), [50](#), [56](#), [60](#), [65](#), [68](#)
- [42] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017. [36](#)
- [43] A. Z. Zhu, D. Thakur, T. Zaslav, B. Pfrommer, V. Kumar, and K. Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2019. [36](#)
- [44] M. Van den Bergh and L. Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 66–72, Jan 2011. doi: 10.1109/WACV.2011.5711485. [38](#)

- [45] H. Liang, J. Yuan, D. Thalmann, and N. M. Thalmann. Ar in hand: Ego-centric palm pose tracking and gesture recognition for augmented reality applications. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 743–744, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2807972. URL <http://doi.acm.org.ezlibproxy1.ntu.edu.sg/10.1145/2733373.2807972>.
- [46] S. Mohatta, R. Perla, G. Gupta, E. Hassan, and R. Hebbalaguppe. Robust hand gestural interaction for smartphone based ar/vr applications. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 330–335, March 2017. doi: 10.1109/WACV.2017.43.
- [47] A. Sanin, C. Sanderson, M. T. Harandi, and B. C. Lovell. Spatio-temporal covariance descriptors for action and gesture recognition. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 103–110, Jan 2013. doi: 10.1109/WACV.2013.6475006.
- [48] H. Zhang, V. M. Patel, M. Fathy, and R. Chellappa. Touch gesture-based active user authentication using dictionaries. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 207–214, Jan 2015. doi: 10.1109/WACV.2015.35. 38
- [49] X. Zhu, W. Liu, X. Jia, and K. Y. K. Wong. A two-stage detector for hand detection in ego-centric videos. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, March 2016. doi: 10.1109/WACV.2016.7477665. 38
- [50] H. Cheng, L. Yang, and Z. Liu. Survey on 3d hand gesture recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(9): 1659–1673, Sept 2016. ISSN 1051-8215. doi: 10.1109/TCSVT.2015.2469551. 38
- [51] F. Destelle, A. Ahmadi, K. Moran, N. E. O'Connor, N. Zioulis, A. Chatzitofis, D. Zarpalas, P. Daras, L. Unzueta, J. Goenetxea, M. Rodriguez, M. T. Linaza, Y. Tisserand, and N. M. Thalmann. A multi-modal 3d capturing platform for learning and preservation of traditional sports and games. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 747–748, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2807975. URL <http://doi.acm.org.ezlibproxy1.ntu.edu.sg/10.1145/2733373.2807975>. 38
- [52] M. Yasen and S. Jusoh. A systematic review on hand gesture recognition techniques, challenges and applications. *PeerJ Computer Science*, 2019. 38
- [53] N. G. Wiseman. Gesture recognition technology and challenges to it, 2018. URL <https://www.techyv.com/article/gesture-recognition-technology-and-challenges-to-it/>. 38

- [54] X. Peng, B. Zhao, R. Yan, H. Tang, and Z. Yi. Bag of events: An efficient probability-based feature extraction method for aer image sensors. *IEEE Transactions on Neural Networks and Learning Systems*, 28(4):791–803, April 2017. ISSN 2162-237X. doi: 10.1109/TNNLS.2016.2536741. [8](#), [42](#), [43](#)
- [55] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate coding and coincidence processing—application to feedforward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2706–2719, Nov 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.71. [43](#)
- [56] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. *ACM KDD Conference*, 2005. [45](#), [46](#)
- [57] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Srpinger, 2009. [46](#)
- [58] C. C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001. [46](#)
- [59] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT’99)*, 1999. [46](#)
- [60] C. C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *ACM SIGMOD Record*, 30(1):13–18, March 2001. [46](#)
- [61] C. C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 2000.
- [62] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 1998. [46](#)
- [63] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30*, 2017. [52](#), [65](#), [69](#)
- [64] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 181–186. ACM, 1991. [53](#)

- [65] Q. Miao, Y. Li, W. Ouyang, Z. Ma, X. Xu, W. Shi, and X. Cao. Multi-modal gesture recognition based on the resc3d network. *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017. 63, 66, 85, 88
- [66] H. Wang, P. Wang, Z. Song, and W. Li. Large-scale multimodal gesture recognition using heterogeneous networks. *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017. 63, 66, 85, 88
- [67] P. Narayana, J. R. Beveridge, and B. A. Draper. Gesture recognition: focus on the hands. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 63, 66, 85, 88
- [68] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. C., K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. <http://arxiv.org/abs/1904.08405>, abs/1904.08405, 2019. URL <http://arxiv.org/abs/1904.08405>. 63, 85
- [69] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [70] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [71] Y. Wang, B. Du, Y. Shen, Kai Wu, G. Zhao, J. Sun, and H. Wen. Ev-gait: Event-based robust gait recognition using dynamic vision sensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [72] L. Pan, C. Scheerlinck, X. Yu, R. Hartley, M. Liu, and Y. Dai. Bringing a blurry frame alive at high frame-rate with an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 63, 85
- [73] Laptev and Lindeberg. Space-time interest points. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 432–439 vol.1, 2003. 64
- [74] Z. Ren, J. Meng, J. Yuan, and Z. Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 759–760, 2011. ISBN 978-1-4503-0616-4. 66
- [75] Z. Ren, J. Yuan, and Z. Zhang. Robust hand gesture recognition based on finger-earth mover’s distance with a commodity depth camera. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 1093–1096, 2011. ISBN 978-1-4503-0616-4. 66

- [76] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 416–421, 1998. 66
- [77] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. *IEEE conference on computer vision and pattern recognition workshops*, 2017. 66
- [78] P. Molchanov, S. Gupta, K. Kim, and K. Pulli. Multi-sensor system for driver’s hand-gesture recognition. In *Proceedings of the IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2015.
- [79] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. On-line detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 66
- [80] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 67
- [81] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. <http://arxiv.org/abs/1803.01271>, 2018. 67
- [82] Y. Rao, J. Lu, and J. Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 69
- [83] S. Lan, R. Yu, G. Yu, and L. S. Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 69
- [84] S. Miao, G. G., X. Ning, Y. Zi, K. Ren, Z. Bing, and A. Knoll. Neuromorphic benchmark datasets for pedestrian detection, action recognition, and fall detection. *Frontiers in neurorobotics*, 13:38, 2019. 76
- [85] A. S. Al-Shamayleh, R. Ahmad, M. A. M. Abushariah, K. A. Alam, and N. Jomhari. A systematic literature review on vision based gesture recognition techniques. In *Multimedia Tools and Applications volume*, volume 77, page 28121–28184, 2018. 84
- [86] S. S. Rautarayi and A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. In *Artificial Intelligence Review*, pages 1–54, 2015. 84

- [87] M. Ferron, N. Mana, and O. Mich. *Designing Mid-Air Gesture Interaction with Mobile Devices for Older Adults*, pages 81–100. Springer International Publishing, Cham, 2019. ISBN 978-3-030-06076-3. [85](#), [90](#)
- [88] K. G. Derpanis. A review of vision-based hand gestures. 2004. URL <https://pdfs.semanticscholar.org/bc80/871a9dfb0703ece28324a80bfc051243c947.pdf>. [86](#)
- [89] D. Saxe and R. Foulds. Toward robust skin identification in video images. In *IEEE international conference on automatic face and gesture recognition*, page 379–384, 1996. [87](#)
- [90] D. Chai and K. Ngan. Locating the facial region of a head and-shoulders color image. In *IEEE international conference on automatic face and gesture recognition*, page 124–129, 1998. [87](#)
- [91] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. In *IEEE asian conference on computer vision (ACCV)*, page 687–694, 1998. [87](#)
- [92] L. Sigal, S. Sclaroff, and Y. Athitsos. Skin color-based video segmentation under time-varying illumination. *IEEE Trans Pattern Anal Mach Intell*, page 862–877, 2004. [87](#)
- [93] J. M. Rehg and T. Kanade. Digiteyes: vision-based hand tracking for human-computer interaction. In *IEEE workshop on motion of non-rigid and articulated bodies*, page 16–24, 1994. [87](#)
- [94] S. Belongie, J. Malik, and J. Puzicha. Skin color-based video segmentation under time-varying illumination. *IEEE Trans Pattern Anal Mach Intell*, 24(4):509–522, 2004. [87](#)
- [95] A. A. Argyros and M. I. A. Lourakis. Binocular hand tracking and reconstruction based on 2d shape matching. In *IEEE international conference on pattern recognition (ICPR)*, 2006. [87](#)
- [96] J. Lee and T. L. Kunii. Model-based analysis of hand posture. *IEEE Comput Graphics Appl*, 15(5):77–86, 1995. [87](#)
- [97] J. Triesch and C. von der Malsburg. Robust classification of hand postures against complex backgrounds. In *Proceedings of IEEE International Conference on Computer Vision*, 1996. [87](#)
- [98] W. T. Freeman and C. D. Weissman. Television control by hand gestures. In *International Workshop on Automatic Face and Gesture Recognition*, 1995. [87](#)
- [99] Q. Yuan, S. Sclaroff, and V. Athitsos. Automatic 2d hand tracking in video sequences. In *IEEE Workshops on Applications of Computer Vision*, 2005. [87](#)

- [100] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004. [87](#)
- [101] H. Masnadi-Shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 779–786, 2010. [87](#)
- [102] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, 2008. [87](#)
- [103] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. Cheng, S. L. Hicks, and P. H. S. Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016. [87](#)
- [104] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. [87](#)
- [105] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, 2016. [87](#)
- [106] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen. Toward real-time particle tracking using an event-based dynamic vision sensor. *European Conference on Computer Vision*, 5:1465–1469, 2011. [87](#)
- [107] B. Ramesh, S. Zhang, Z. W. Lee, Z. Gao, G. Orchard, and X. Cheng. Long-term object tracking with a moving event camera. In *British Machine Vision Conference*, 2018. [88](#)
- [108] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018. [88](#)
- [109] W. Gerstner. Time structure of the activity in neural network models. *Phys. Rev. E*, 51:738–758, 1995. [92](#)
- [110] T. Delbruck and M. Lang. Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor. *Frontiers in neuroscience*, 7: 223, 2013. [93](#)
- [111] Celepixel: Bio-inspired computer vision. URL <https://www.celeapixel.com>. [109](#)
- [112] H. Duan, Q. Zhang, and W. Ma. An approach to dynamic hand gesture modeling and real-time extraction. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 139–142, 2011. [110](#)

-
- [113] Z. Lu, X. Chen, Q. Li, X. Zhang, and P. Zhou. A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices. *IEEE Transactions on Human-Machine Systems*, 44(2):293–299, 2014. [110](#)
 - [114] A. Aksaç, O. Öztürk, and T. Özyer. Real-time multi-objective hand posture/gesture recognition by using distance classifiers and finite state machine for virtual mouse operations. In *2011 7th International Conference on Electrical and Electronics Engineering (ELECO)*, pages II–457–II–461, 2011. [110](#)
 - [115] T. Chiang and C. Fan. 3d depth information based 2d low-complexity hand posture and gesture recognition design for human computer interactions. In *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, pages 233–238, 2018. [110](#)
 - [116] T. Tsai, C. Huang, and K. Zhang. Design of hand gesture recognition system for human-computer interaction. *Multimedia Tools and Applications*, 79: 5989–6007, 2020. [110](#)
 - [117] T. Zheng, C. Chen, J. Yuan, B. Li, and K. Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. [114](#)
 - [118] Stoffregen T, G. Gallego, T. Drummond, L. Kleeman1, and D. Scaramuzza. Event-based motion segmentation by motion compensation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. [115](#)