High Speed Event-based Visual Processing in the Presence of Noise

WESTERN SYDNEY
UNIVERSITY

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

SAEED AFSHAR

International Centre for Neuromorphic Systems
MARCS Institute for Brain, Behaviour and Development
Western Sydney University

25 May 2020

For Sami and Soren.

You are my light.

Statement of Authentication

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. I hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution.

Saeed Afshar

[redacted]

Abstract

Standard machine vision approaches are challenged in applications where large amounts of noisy temporal data must be processed in real-time. This work aims to develop neuromorphic event-based processing systems for such challenging, high-noise environments. The novel event-based application-focused algorithms developed are primarily designed for implementation in digital neuromorphic hardware with a focus on noise robustness, ease of implementation, operationally useful ancillary signals and processing speed in embedded systems.

One stream of research within the neuromorphic engineering space is modelling biological nervous systems in hardware. In this work however, instead of trying to model the complex phenomenological details of observed brain activity and circuitry, we focus on the constraints and requirements many high noise applications share with biological environments. By probing the large search space of potential hardware solutions to real-time temporal data processing in high-noise applications, this work seeks to develop functionally equivalent cognitive processes as occur in the brain in the same decentralized noise-robust manner and, importantly, through the use of time itself as the central processing variable.

Several new and challenging event-based datasets are designed and presented. The datasets all involve challenging tasks performed in high speed and in the presence of significant noise. These datasets are generated for investigation of different aspects of event-based processing and use a range of the event-based sensors from multiple providers as well as Single Photon Avalanche Diode (SPAD) imagers.

A wide range of event-based surface generation, surface processing and feature extraction methods are investigated, highlighting design trade-offs for the development of efficient hardware implementation. The performance details of a recognition system implemented in FPGA hardware is discussed. A novel event-based SPAD imager design, which was also implemented in custom hardware, is presented and discussed. In addition several other possible event-based SPAD imager designs are introduced and evaluated on a new SPAD dataset.

The first event-based space imaging dataset is presented. A number of detection and tracking algorithms are evaluated on this large challenging and highly noisy dataset. The dataset and the algorithm investigation results show the significant differences between controlled terrestrial event-based data streams and real-world uncontrolled environments, highlighting the different requirements and solutions involved in high-speed event-based processing in high-noise environments.

The datasets presented and the algorithms developed, demonstrate the utility of event-based processing in high speed noisy applications. The methods developed take advantage of the sparse temporal nature of event-based data streams to perform rapid processing during time intervals with high activity while limiting processing during periods low activity. Finally, redundancies in the types of features observed in event-based data enable optimizations that simplify the proposed algorithms and allow their implementation in neuromorphic hardware.

Acknowledgements

I wish to thank my supervisors, Professor André van Schaik, Associate Professor Tara Hamilton and Dr. Dennis Delic as well as former supervisor Professor Jonathon Tapson for their support, guidance and vision which were essential to the development of this work.

I especially wish to thank Associate Professor Gregory Cohen who has probably had the greatest influence on the direction of my research and without whom this work would not have been possible.

I would also like to express my gratitude to Langdon Davis, Ying Xu, Joyce Mau, Andrew Nicholson and Aaron Panella for their great help in the development of this work.

Finally I would like to thank my family for their love, support and extreme patience.

Contributions

The work presented in this thesis is entirely my own except for elements which are detailed below:

Professor André van Schaik, Associate Professor Tara Hamilton, Dr Dennis Delic, Associate Professor Gregory Cohen and Professor Jonathan Tapson contributed through their ideas, discussions and editing of the presented thesis.

Associate Professor Gregory Cohen also contributed in the collection of the ATIS Plane Dropping Dataset and also provided the event-based recordings of the Space Imaging Dataset that was used to design of the space object detection and tracking algorithms which this student developed for this thesis.

Dr. Andrew Nicholson contributed through his FPGA-based implementation of the event-based space object detection and tracking algorithms which this student developed for this thesis. While this hardware design is not discussed in this thesis, the implementation, and the hardware resource and timing constraints identified, did inform the design of the algorithms.

Joyce Mau and Aaron Panella from the Department of Defence Science and Technology (DST) contributed to this work through their implementation and testing on the NVIDIA TX2 platform of the real-time embedded feature extraction networks that this student developed for this thesis.

Dr Dennis Delic from the Department of Defence Science and Technology contributed to this work through his implementation of a Complementary Metal-Oxide-Semiconductor (CMOS) based Single Photon Avalanche Diode (SPAD) with supporting mixed signal and digital Integrated Circuit (IC) chip design based on the First-AND network design which this student developed for this thesis.

Langdon Davis from BAE Systems contributed to this work through his FPGA-based hardware implementation and testing of the event-based feature extraction network and classifier system which this student developed for this thesis.

List of Publications

1. Afshar, S., Hamilton, T. J., Davis, L., van Schaik, A. & Delic, D. Event-based Processing of Single Photon Avalanche Diode Sensors. *arXiv preprint arXiv:2001.02060* (2019).
2. Afshar, S., Nicholson, A. P., van Schaik, A. & Cohen, G. Event-based Object Detection and Tracking for Space Situational Awareness. *arXiv preprint arXiv:1911.08730* (2019).
3. Afshar, S. *et al.* Event-Based Feature Extraction Using Adaptive Selection Thresholds. *Sensors* **20.** ISSN: 1424-8220. https://www.mdpi.com/1424-8220/20/6/1600 (2020).
4. Cohen, G. *et al.* Event-based sensing for space situational awareness. *The Journal of the Astronautical Sciences* **66,** 125–141 (2019).
5. Xu, Y. *et al. A Binaural Sound Localization System using Deep Convolutional Neural Networks* in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)* (2019), 1–5.
6. Mau, J. *et al. Embedded implementation of a random feature detecting network for real-time classification of time-of-flight SPAD array recordings* in *Laser Radar Technology and Applications XXIV* **11005** (2019), 1100505.
7. Chakraborty, S. *et al. Neuromorphic object tracking architecture, based on compound eyes, and implementation on FPGA* in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)* (2018), 668–671.
8. Xu, Y. *et al. A Machine Hearing System for Binaural Sound Localization based on Instantaneous Correlation* in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)* (2018), 1–5.
9. Cohen, G. *et al.* Spatial and temporal downsampling in event-based visual classification. *IEEE transactions on neural networks and learning systems* **29,** 5030–5044 (2018).

10. Afshar, S., Hamilton, T. J., Tapson, J., van Schaik, A. & Cohen, G. Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition. *Frontiers in neuroscience* **12,** 1047 (2019).
11. Cohen, G., Afshar, S., Tapson, J. & Van Schaik, A. *EMNIST: Extending MNIST to handwritten letters* in *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), 2921–2926.
12. Cohen, G., Afshar, S. & van Schaik, A. *Approaches for astrometry using event-based sensors* in *Conf. Advanced Maui Optical and Space Surveillance Technologies* (2017).
13. Thakur, C. S. *et al.* Bayesian estimation and inference using stochastic electronics. *Frontiers in neuroscience* **10,** 104 (2016).
14. Thakur, C. S. *et al.* Sound stream segregation: a neuromorphic approach to solve the "cocktail party problem" in real-time. *Frontiers in neuroscience* **9,** 309 (2015).
15. Thakur, C. S. *et al.* An online learning algorithm for neuromorphic hardware implementation. *arXiv preprint arXiv:1505.02495* (2015).
16. Afshar, S. *et al.* Turn down that noise: synaptic encoding of afferent SNR in a single spiking neuron. *IEEE transactions on biomedical circuits and systems* **9,** 188–196 (2015).
17. Afshar, S., George, L., Tapson, J., van Schaik, A. & Hamilton, T. J. Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels. *Frontiers in neuroscience* **8,** 377 (2014).
18. Sofatzis, R. J., Afshar, S. & Hamilton, T. J. *The synaptic kernel adaptation network* in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (2014), 2077–2080.
19. Hamilton, T. J., Afshar, S., van Schaik, A. & Tapson, J. Stochastic electronics: A neuro-inspired design paradigm for integrated circuits. *Proceedings of the IEEE* **102,** 843–859 (2014).
20. Afshar, S. *et al.* The ripple pond: enabling spiking networks to see. *Frontiers in neuroscience* **7,** 212 (2013).
21. Tapson, J. C. *et al.* Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Frontiers in neuroscience* **7,** 153 (2013).

Publications

Chapters, 3 and 4 of this thesis have been published (see items 10 and 3 in the List of Publications) and chapters in 2 and 5 of this thesis are in the review process for publication.

For a full, up-to-date list of publications see Saeed Afshar - Google Scholar or visit:

`https://scholar.google.com.au/citations?user=a8FPrPwAAAAJ`

Contents

Statement of Authentication **iii**

Abstract **iv**

Acknowledgements **vi**

Contributions **vii**

Publications **x**

Contents **xi**

Table of Abbreviations **1**

Chapter 1 Introduction **3**

1.1 Neuromorphic Sensing 4

1.2 Brains vs Machines 5

1.2.1 AER Overcoming the Challenge of Connectivity 8

1.3 Neuromorphic Vision Sensors 9

1.4 Thesis Summary 13

Chapter 2 Event-based Object Detection and Tracking for Space Situational Awareness **15**

2.1 Introduction 16

2.1.1 Event-based Space Situational Awareness (EBSSA) 18

2.2 Methodology 18

2.2.1 Generation of the Space Imaging Dataset 19

2.2.2 Labelling the Dataset 22

2.2.3 Measuring Sensitivity, Specificity and Informedness 25

2.2.4 Artificial Space Imaging Dataset 27

2.2.5 Event Pre-processing . . . 29
2.2.6 Feature Detection . . . 31
2.2.7 Event-based Tracking . . . 40
2.2.8 Alternative Algorithms . . . 47
2.3 Results . . . 52
2.3.1 Algorithm and Expert Performance on Artificial Dataset . . . 52
2.3.2 Performance on Real World Space Imaging Dataset . . . 53
2.3.3 Processing Time Results . . . 58
2.3.4 Comparison with Alternative Algorithms . . . 60
2.4 Discussion and Future Work . . . 63
2.5 Conclusion . . . 65

Chapter 3 Investigation of Event-based surfaces 66
3.1 Introduction . . . 67
3.2 Methodology . . . 70
3.2.1 The ATIS Plane Dropping Dataset . . . 70
3.2.2 Time Surfaces vs. Index Surfaces . . . 74
3.2.3 Target Velocity vs. Surface Activation . . . 80
3.2.4 Event-Based Feature Extraction using SKAN . . . 82
3.2.5 Spatial Pooling of Feature Surfaces . . . 87
3.2.6 Classification . . . 89
3.2.7 Parameter Selection . . . 90
3.3 Results . . . 92
3.3.1 Results on the Full Dataset . . . 92
3.3.2 Frame Balanced Dataset . . . 93
3.3.3 Velocity Segregated Dataset . . . 96
3.3.4 The Decay Constants . . . 98
3.3.5 Feature Extractor Size and Number . . . 99
3.4 Discussion and Future Work . . . 102
3.5 Conclusion . . . 104

Chapter 4 Investigation of Feature Extraction using Adaptive Selection Thresholds 105

4.1 Introduction 106

4.1.1 Feature Extraction in Neuromorphic Systems 106

4.1.2 Feature Extraction via Adaptive Selection Thresholds 110

4.1.3 Noise Features and Network Size Selection 116

4.2 Methodology 118

4.2.1 Datasets 118

4.2.2 Classifiers 119

4.3 Results 121

4.3.1 N-MNIST Digit Classification Results 121

4.3.2 Plane Dropping dataset Results 122

4.3.3 Evaluating Feature Sets Via Feature Activation 126

4.4 Discussion and Future Work 129

4.4.1 Missed Events During Learning 129

4.4.2 Thresholds During Learning and Inference 129

4.5 Conclusion 130

Chapter 5 Event-based SPAD processing 132

5.1 Introduction 133

5.1.1 SPAD 133

5.1.2 Event-based Processing and SPAD 134

5.2 Methodology 135

5.2.1 The SPAD Dataset 135

5.2.2 First-AND Event Generation Method: Discarding Time and Transmitting Change 139

5.2.3 Noise in SPAD Imaging Motivates a Modified Noise Robust Design 144

5.2.4 Implementation of the First-AND Network in ASIC 152

5.2.5 Training Binary Feature Extractor Networks 154

5.2.6 Alternative Event-based Methods: On-Off Bi-polar and Uni-polar Events 160

5.2.7 Pooling, Surface Sampling and Classification 165

5.2.8 Event-based Processor Implementation on FPGA 169
5.2.9 Real-time Frame-based Feature Extraction Network Implemented on GPU 171
5.3 Results 173
5.3.1 Data Generation Rates 173
5.3.2 Classification 174
5.3.3 FPGA Implementation Results 181
5.4 Discussion and Future Work 184
5.5 Conclusion 185
Chapter 6 Conclusion 187
6.1 Future Work 189
Bibliography 190

Table of Abbreviations

AER	Address Event Representation
ASIC	Application-Specific Integrated Circuit
ATIS	Asynchronous Time-based Image Sensor
CMOS	Complementary Metal-Oxide-Semiconductor
CCD	Charge-Coupled Device
CPU	Central Processing Unit
DST	Defence Science Technology
DVS	Dynamic Vision Sensor
DTOF	Direct Time Of Flight
ELM	Extreme Learning Machine
EBSSA	Event-based Space Situational Awareness
EBSI	Event-based Space Imaging
FIFO	First In First Out
FEAST	Feature Extraction via Adaptive Selection Thresholds
FPGA	Field Programable Gate Array
GEO	Geosynchronous Equatorial Orbit
GPU	Graphical Processing Unit
HOTS	Hierarchy Of Time Surfaces
OOBU	On-OFF-Bipolar-Unipolar
ITOF	Indirect Time Of Flight
IC	Integrated Circuit
LEO	Low Earth Orbit
LUT	Look Up Table
ROI	Region Of Interest
SKAN	Synaptic Kernel Adaptation

SKIM	Synaptic Kernel Invese Method
SNR	Signal to Noise Ratio
SPAD	Single Photon Avalanche Diode
SSA	Space Situational Awareness
STDP	Synaptic Timing Dependent Plasticity

CHAPTER 1

Introduction

The field of Neuromorphic Engineering seeks to improve-state-of-the-art-sensing and processing technologies by borrowing solutions and principles from biology. Biological nervous systems, which Neuromorphic systems aim to model, are first and foremost weapons of war. From the beginning of their evolution five hundred and fifty million years ago, nervous systems have been under intense selective pressure for optimized speed, hardware and signal efficiency and survival in noisy low-information environments [1]. Biological visual systems, for instance, have no intrinsic use for high fidelity information capture or storage and count success exclusively as the ability to generate ecologically relevant output signals which prevent death before procreation under specific frequently encountered conditions. It so happens that the most ecologically critical visual environments, such as during predator-prey interactions, are noisy, dynamic and time-critical [2]. Furthermore, biological sensory systems such as the mammalian retina are themselves noisy event-based sensors which operate at close to their activation threshold in order to maximize information capture and minimize response time [3].

Given these factors, it would indeed be remarkable if biological vision systems were not optimized for operation in noisy environments. In contrast, conventional cameras and other imagers seek to encode the value of their observed inputs with perfect accuracy at the maximum possible resolution. Similarly, in the field of machine vision which typically operates on these high fidelity signals, algorithm accuracy is prized far above other performance measures with an outsized focus on small accuracy improvements on highly constrained carefully generated datasets. Thus, in contrast to most research in neuromorphic vision, in the wider machine vision field, hardware implementability and algorithm robustness in high noise environments are not critical design factors.

Given their contrasting origins and design requirements, it is unsurprising that conventional machine vision algorithms and biologically inspired vision systems operate on entirely different principles.

1.1 Neuromorphic Sensing

Neuromorphic sensors emulate the perceptual power of biological sensors via two critical principles. First, neuromorphic sensors efficiently encode sensory information into time itself. By converting a highly detailed sensory environment into spikes, bio-inspired sensors use the relative timing of spikes and/or their rate to encode a wide range of information with arbitrary precision, reliability, speed or bandwidth depending on the application requirements.

The second way neuromorphic sensors emulate the efficiency of biology is through focusing on encoding change. While conventional man-made sensors seek to continuously and reliably encode the absolute value of their inputs at the maximum possible resolution regardless of how much data is generated, biological sensors cannot afford to waste their sensory and processing hardware on useless, redundant information. Since living things live and die based on how quickly and energy efficiently they respond to their unpredictable noisy environment, their sensors are focused only on change because sensory change is the signal for initiation or modification of behavior. Similarly, by only encoding sensory change, bio-inspired sensors drastically reduce the sensory information reported to higher levels of processing and keep the processing system focused on high-speed reaction to behaviorally relevant stimuli at the expense of sensory fidelity and resolution. The two drastically different approaches taken in conventional sensing and neuromorphic event-based sensing are illustrated in Figure 1.1.

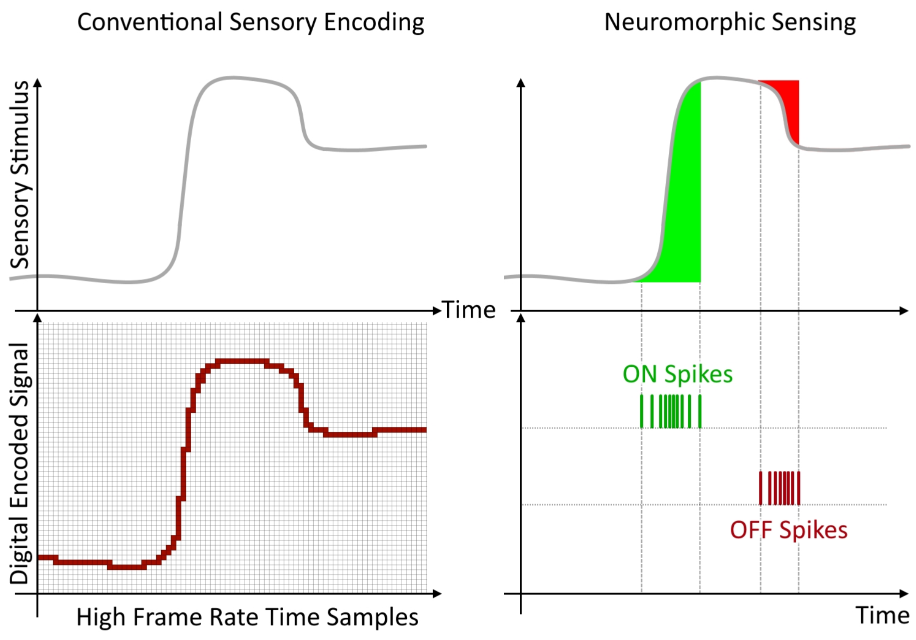


FIGURE 1.1: **Comparison of conventional sensing with Neuromorphic sensing.** Top left panel shows the true continuous signal. Bottom left shows the conventional approach to signal encoding via high resolution quantization. Right panels shows how neuromorphic sensing encodes sensory change into spikes or events.

1.2 Brains vs Machines

The field of machine vision is today the subject of intense research. This recent interest, enabled by the greater availability of ever more powerful computing resources and recent developments in machine learning, has resulted in an expansion of applications where computers directly interface with and process visual input. From smart phones that detect a smile to automated breast cancer screenings to driverless cars, the field of machine vision is increasingly entering daily life and is set to expand rapidly, enabling computers to interact with the external world without the need for constant human supervision, thus greatly expanding their utility.

Yet the visual processing performance delivered by these advances is dwarfed by the three-pound, twenty-watt "supercomputer" reading these words. The Google brain project is illustrative were the Google X labs using deep belief networks and sixteen thousand cores processing ten million YouTube videos over three days was able to achieve a 74.8 percent accuracy rate in identifying cats in YouTube videos [4]. The human brain is indeed the ultimate visual processor but even for us, vision is hard. Despite more than five hundred million years of evolution optimizing the visual processing system, nearly half the primate brain is devoted to vision [5]. The field of human and primate vision is one of the most intensely investigated areas in neuroscience. With the recent advent of new imaging technologies such as functional magnetic resonance imaging at the network scale and two-photon imaging at the micro-circuit scale, we have now begun to glimpse the inner workings of this complex system [6].

The way the brain processes visual information is radically different from the algorithms used in computer vision. The brain has no software, no CPU, no dedicated memory and signal speeds in the low meters per second. It processes all information in parallel through a distributed network of stochastic neurons utilizing the temporal information embedded in their spiking activity to produce a "neural code”. The simplest neural code which has traditionally been used in the field of Artificial Neural Networks is rate coding where a neuron encodes signal information in its output spike rate. This scheme, while mathematically amenable, incurs a significant energy cost by discarding the rich temporal information available in spike patterns and requiring many spikes per input channel to transmit a rate-based signal. In contrast, temporal coding encodes information in the timing of spike. Temporal coding plays a central part in the energy-efficient operation of the brain. This is because many more spikes per input channel are required in a rate coding scheme to transmit a real-valued rate compared to a single spike per channel which is needed in temporal coding for the transmission of inter-spike interval information[7][8]. In contrast to this efficient encoding scheme, Machine vision is based on the sequential execution of specialized software running on general-purpose deterministic processors with the canonical von Neumann architecture where instructions and data are fetched from memory and executed one at a time with the result stored back into memory. This so the called ‘von Neumann bottleneck’ in computing [9] which neuromorphic processors, as well as more conventional GPU systems, attempt to bypass. An area where engineered digital representation of information is superior to spike-based or any form of analog encoding is in the ability to encode arbitrary precision in the form of additional digital bits. Spike-based encoding schemes, as well as other analog representations of information, are always limited in precision by noise. However, in most real-world vision applications, the high precision provided by digital processors is wasted due to the inherent noise in the initial input signal which makes high precision computing irrelevant [10].

Vision, more than any other computing task, is inherently a parallel processing problem where many features in the entire visual scene can at any moment be potentially significant and require detailed processing [11]. Thus inherent incongruity between the problem of vision and our classical computing architecture is one of the reasons for the great gap in performance that still exists between biological and machine visual processing systems and serves as the motivation for investigation of alternate bio-inspired visual processing architectures. Over the last decade a range of such novel bio-inspired architectures have been presented for visual processing from the field of neuromorphic engineering.

Another central difference between brains and computers is the ways in which learning, memory and recognition are realized. In computers every piece of information or instruction exists as an exact digital value in a precise address in memory from which it can be accessed by the central processor in highly organized fashion during the execution cycle. While this precision model of memory is enormously powerful for solving deterministic problems like calculating the value of pi to the millionth decimal place, in the context of real-world vision it is entirely inappropriate where the incoming visual signal and even the concepts to be learnt and recognized (such as a tree) are inherently noisy and probabilistic. Here the extreme precision built into classical computing is wasted, witnessed by the fact that machine vision algorithms need to introduce significant noise into otherwise deterministic recognition systems in order to add generality and improve performance. This realization has led to the emergence of the field of stochastic computation and stochastic electronics where the probabilistic nature of electronic circuits is utilized to enhance system performance [10].

On the biological side, the mechanisms by which the brain performs learning, memory and recognition are still not well understood. There exist many proposed models with several overarching themes including synaptic adaptation, rate and temporal coding, localized and distributed information and statistical inference via system dynamics, but unfortunately these abstract models cannot capture the deep underlying complexities of neural systems and offer little predictive power. It is likely that the brain uses a wide range of highly complex mechanisms in different contexts to generate a very large dynamic space within which information is encoded.

Although lacking a complete model, there are important constraints on the brain which are useful for the purposes of engineering alternative learning and memory architectures. Among these is the lack of centralized control. From an engineer's perspective, this constraint can be seen as an extremely limiting handicap. Without a centralized controller such as the control unit in a CPU, the different elements of a dynamical system seemingly cannot be directed to work together in a coordinated fashion, the fundamental property by which systems are actually defined. Yet distributed, decentralized systems can indeed operate with remarkable effectiveness and can have the benefits of simplicity, scalability and speed [10][12].

Here again researchers in the field of neuromorphic engineering have proposed novel architectures which operate via distributed temporal architectures [13–15] and the Synaptic Kernel Adaptation Network proposed by this author [16]. In this context neuromorphic engineering takes a discovery-by-design approach to neuroscience, aiming to discover the underlying computational principles at work in the brain by designing efficient systems that perform functionally equivalent cognitive tasks as occurs in the brain in the same decentralized probabilistic manner and importantly by the use of time itself as the central processing variable.

1.2.1 AER Overcoming the Challenge of Connectivity

One of the most important bottlenecks in the design of neuromorphic vision systems is the requirement for high connectivity between a large number of neurons. One of the techniques used in neuromorphic engineering is the use of Address Event Representation. In this method, events generated by a sensor or a processor are time-stamped using a local encoder augmented with critical information such as their address which in vision is typically the x and y coordinates of the pixel generating the event. These events are then processed by arbiters and transmitted over digital buses as a stream of events. This is in contrast to the standard frame-based approach where the values of all sensor cells are communicated in serial or parallel at regular intervals regardless of their information content. AER has become the standard interfacing protocol in neuromorphic engineering, specifically for multi-chip systems [17] but also in terms of intra-chip communication in mixed-signal devices. AER is ideally suited to event-based neuromorphic applications which often process sparse events from a large number of sources through narrow bandwidth interfaces [18]. The AER protocol has been successfully used in neuromorphic processors such as SpiNNaker [19] and TrueNorth [20] and neuromorphic models of the auditory pathway [21]. But arguably the greatest utility of the AER protocol has been in event-based neuromorphic vision where the combination of high spatial resolution and sparse event generation rates gives the AER protocol the greatest advantage over conventional frame-based communication protocols [22, 23].

1.3 Neuromorphic Vision Sensors

The field of Neuromorphic vision sensors begun with the development of the first silicon retina by Mahowald and Mead [3]. This device incorporated many of the characteristics found in the current generation of silicon retina. These characteristics include adaptive photoreceptors and spatial smoothing networks. However, the device lacked the asynchronous communication paradigm which is central to the operation of current-generation silicon retinas and as was restricted to its output being scanned directly onto a multi-sync monitor.

Although that device was intended only for demonstrative purposes, it led to a succession of future sensors that refined and improved the underlying technologies. Zaghoul and Boahen implemented more detailed biologically-inspired models with both transient and sustained cell types [24, 25]. This circuit however suffered from large transistor mismatch, highly variable pixel firing rates and a relatively low dynamic range. The focus for this chip however was to model biology, rather than producing a device for real-world applications.

A more application-oriented device was the dual-output sensor from Rüedi et al [26]. This device was one of the first sensors to offer two outputs: a change measurement and an absolute illumination measurement. This device made use of a global integration time for all pixels, but presented the output in the order of the highest to lowest spatial contrast. Although not event-based, this sensor had the ability to stop the output from the absolute illumination circuit when enough spatial contrast data had been received. The temporal resolution of the sensor was still limited by the frame rate of the device and the camera itself did not perform any temporal redundancy suppression. The devices were later specialized for automotive applications [27].

Mallik et al. produced an imaging chip in which the individual pixels were capable of responding to quantized changes in absolute light intensity [28]. It offered both a traditional Active Pixel Sensor (APS) output and an output that responds to changes and motion in the scene. This sensor also included the change detection circuitry in each pixel, allowing for the technology to theoretically scale to larger arrays. The sensor only responded to changes in absolute illumination and not relative illumination, which limited the effective dynamic range of the device. The device was also frame-based, with an additional FIFO structure to handle the change events, which thereby limited the temporal resolution of the change detection to the frame rate.

It was the work by Kramer [29] which set the foundation for the current approach to event-based sensing. His paper introduced a 48×48 pixel imaging sensor that outputted ON/OFF events using an asynchronous binary address bus. The chip made use of a feedback loop, allowing it to respond to relative changes in illumination and effectively allowing the pixels to adapt to the background illumination. This circuit was then improved by Lichtsteiner, Delbruck and Kramer [30], increasing operating range and the symmetry of the ON and OFF channels. This sensor was followed by a 128 ×128 pixel event-based sensor from Lichtsteiner, Posch and Delbruck [31] which greatly improved the spatial contrast sensitivity. Elaborating on this work led to the first Dynamic Vision Sensor (DVS) [32]. As shown in Figure 1.2 this DVS sensor which has been the basis of most later silicon retinas uses an adaptive change detection circuit that generates events in response to the log change in illumination. In this design, every pixel operates independently and asynchronously producing ON and OFF events when the increase or decrease in illumination passes a preset upper or lower. Thus, the DVS camera does not operate using a global shutter but transmits information only in response to

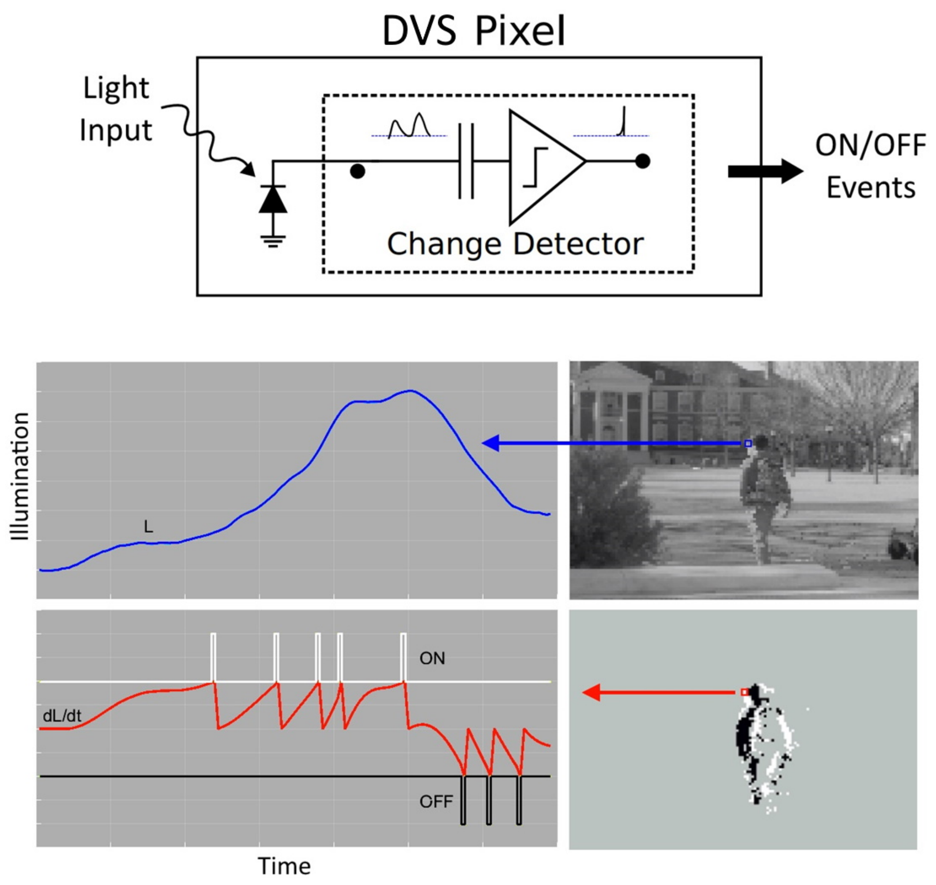


FIGURE 1.2: **The DVS pixel functional block diagram and description of the mode of operation of an event-based DVS pixel.** In the top panel, the block diagram of the DVS pixel is shown detailing how changes in illumination at the photodiode are passed to a change detection circuit which generates output events. The middle two panels show an example input stimulus where a person walks against a static background with the illumination level at the indicated pixel plotted in blue. The bottom right panel shows the event-based output generated by the scene demonstrating how increases in illumination cause ON events (white) and decreases in illumination cause OFF events (black). The red plot in the lower-left panel shows the change in illumination over time for the indicated pixel showing an initial rise and corresponding ON events followed by a fall in illumination which triggers OFF events.

new stimuli effectively performing data compression at the sensor. This event-based mode of operation where independent, asynchronous pixels generate events in response to stimulus change that crosses a threshold, is the basis of almost all current neuromorphic vision sensors.

Whereas the above technologies all directly influenced the development of event-based imagers currently being used, there were many other types of sensors developed. These sensors also embodied the neuromorphic approach and offered different scene representations, often providing additional capabilities over conventional imagers. Culurciello and Andreou produced an imager which reported the absolute light intensity at each pixel through an event-based mechanism using inter-event times [33]. However, the output bandwidth in these devices were inherently linked to the scene luminance, causing problems when imaging dark scenes with sparse but relatively bright features. This can be attributed to the lack of a reset feature for each pixel integration time, but did have the advantage of allowing for a small pixel size.

Another particularly relevant camera technology was the Time-To-First-Spike imager developed by Guo and Harris [34], in which the time between events for each pixel and a reference frame directly encodes the absolute light intensity. Chen and Bermak also produced a Time-To-First-Spike sensor which implements an enforced post-spiking delay to prevent saturating the output bus with the events generated by bright pixels [35]. This technology improved on many existing time-based imagers [36] and is similar to the methodology used in certain Single Photon Avalanche Diode (SPAD) devices.

There have also been devices that have used a "foveated" approach in which a high density of pixels are located in the center of the device, surrounded by less dense and more complex pixels [37]. Additionally, frame-based approaches to temporal change detection imagers have also been developed [38] in which the frame consists of the difference in the integration of the photocurrent between subsequent frames. These techniques were originally implemented in software [39] with Aizawa et al. implementing a prototype 32×32 pixel array with parallel frame-differencing pixels [40]. A 189×182 pixel sensor built by Gruev and Etienne-Cummings included both a conventional frame output and a frame-difference output [41]. Also of note are the spatial-contrast imagers [42], in which the pixels produce an output only when its illumination exceeds the weighted spatial average of its neighbors. A number of specialized neuromorphic sensors have also been developed including for motion detection [43] and multi-object center of mass detection [44].

Thus while the field of neuromorphic vision sensor has seen a wide range of approaches and methodologies one constant in the field has always been relatively low spatial resolution and low signal to noise ratios exhibited in the sensors. This will also be true of the sensors we investigate in this work, motivating the application-based processing approaches presented, which aim to provide robustness to these limitations while leveraging the high temporal resolution and speed of the sensors.

1.4 Thesis Summary

This thesis is organized as follows:

In Chapter 2, the field of Event-based Space Situational Awareness (EBSSA) is introduced. Following the introduction, a new large scale real-world event-based space imaging dataset is presented and discussed. A set of event-based detection and tracking algorithms are presented and evaluated on the presented dataset. The results and future work is then discussed.

In Chapter 3, the need for a rigorous investigation of event-based pre-processing techniques is introduced. Following this introduction, a range of event-based processing techniques are investigated. These techniques include a range of memory surface generation methods and kernels as well as a wide range of feature extraction architectures. These processing techniques are then evaluated in the context of a high-speed event-based airplane tracking and classification task.

In Chapter 4, a novel feature extraction algorithm using adaptive selection thresholds is introduced. The event-based algorithm is designed to simplify hardware implementation at the cost of some information loss while providing useful intermediary signals which are valuable in the context of neuromorphic hardware limitations. The introduced method is compared to several previous methods on the airplane dataset presented in the previous chapter as well as a neuromorphic handwritten digit classification task. Multiple configurations of the algorithm are tested with a range of back-end classifiers with the performances being analyzed at each processing stage.

In Chapter 5, Single Photon Avalanche Diode (SPAD) Sensor arrays and their use in flash LADAR are introduced. The problem of high data rates resulting from high-frame-rate

SPAD sensors is discussed and the use of on-chip event-based processing to overcome this problem is proposed. To evaluate this proposed approach, a large SPAD imaging dataset is presented involving a high-speed airplane tracking and classification task. Various sources of noise in the dataset are investigated and their effects are discussed. The frame-based SPAD imaging dataset is then converted via several alternative methods into event-based datasets and processed using a range of event-based feature extractor networks and pooling methods. The output of a proposed event generation method is then processed by a feature extraction and classification system implemented in FPGA hardware. The results and implications for future work are discussed.

CHAPTER 2

Event-based Object Detection and Tracking for Space Situational Awareness

Chapter Summary

Neuromorphic event-based sensors differ from conventional cameras by generating events in response to changes in illumination at each pixel, rather than through synchronous frames. This unique method of operation makes them well suited for tasks in terrestrial space situational awareness applications, primarily due to the ability of event-based cameras to exploit the sparseness of the data when performing sky imaging. These sensors also offer significantly lower bandwidth and power requirements, making them particularly well suited for use in remote locations and space-based platforms. Space imaging data differs significantly from that of conventional computer vision applications as it exhibits a sparser feature structure, a higher variance in target velocities and extremely low Signal to Noise Ratios (SNRs). The use of event-based cameras for space imaging therefore presents unique challenges and requires the development of specialized algorithms in order to take full advantage of the event-based paradigm offered by these sensors. Such algorithms are tailored to the resulting data, resulting in highly-efficient systems capable of being realized in hardware. This chapter also introduces the first event-based space imaging dataset, which includes recordings from multiple event-based sensors and from multiple providers, greatly lowering the barrier to entry for other researchers given the scarcity of such sensors and the complexity of operating telescope hardware. The dataset contains both day time and night time recordings, including simultaneous co-collections from different event-based sensors. Recorded at remote sites, and containing 572 labeled targets with a wide range of sizes, trajectories and signal-to-noise

ratios, this real-world event-based dataset represents a challenging detection and tracking task that is not readily solved using previously proposed methods. A highly optimized and robust feature-based detection and tracking method is proposed, designed specifically for space situational awareness applications, and implemented via a cascade of increasingly selective event filters. These filters rapidly isolate events associated with space objects, maintaining the high temporal resolution of the sensors. By measuring the unimodality of the angular activation of time surfaces around incoming events, objects are detected using a single method regardless of their size, direction of motion, speed, or SNR. To implement a multi-object tracking algorithm, an event-based line fitting method that operates over a rolling window of detection events is then used. The results from this simple yet highly optimized algorithm on the space imaging dataset demonstrate robust high-speed event-based detection and tracking which can readily be implemented on sensor platforms in space as well as terrestrial environments.

2.1 Introduction

Our increasing reliance on space-based technologies for communication, navigation and security tasks as well as the recent dramatic drop in the cost of space launches has created an immediate need for better methods for detecting and tracking objects in orbit around the earth [45]. The cost of collisions in space poses a significant risk to both our space infrastructure and future space missions.

Space Situational Awareness (SSA), and Space Traffic Management (STM) — its civilian counterpart — are therefore critical tasks for regulation and enforcement of the use of space, and to prevent a future catastrophic space event, such as described by the Kessler effect [46]. Space Situational Awareness is defined by the European Space Agency (ESA) as comprising three segments: Space Surveillance and Tracking (SST), Space Weather and Near Earth Objects (NEO) [47]. The work presented in this chapter contributes primarily to the task of space surveillance and tracking, specifically applied to satellites in orbit around the earth.

Currently, over 80 countries have a presence in space [48] and this is likely to increase, driven by both national space efforts and private industry [48].

Currently, the majority of SSA data originates from dedicated radar installations operated by the United States Air Force [49]. However, radars are an expensive technology to install and operate and there is an increased focus on looking toward optical telescopes to provide a more flexible, cost-effective and responsive means of obtaining accurate SSA data [48]. In previous work, we have demonstrated that event-based neuromorphic cameras offer a novel means of performing SSA tasks and provide capabilities that cannot be achieved using conventional astronomy cameras [50].

Event-based cameras operate in a different imaging paradigm, emitting data as a spatio-temporal pattern rather than using conventional frames [51]. The pixels report changes in log-illumination and are also independent and asynchronous, giving the device a high temporal resolution and a very high dynamic range [23]. The characteristics of these devices enable unique and novel approaches to satellite tracking [52], high-speed adaptive optics [53], satellite identification [54] and real-time in-frame astrometry [55].

This work builds upon those findings and presents two methods for tracking objects in the spatio-temporal output of an event-based camera. There exist many event-based trackers, such as those for long-term object tracking [56], real-time particle tracking [57], micro-particle tracking [58], corner detectors [59] and more complex kernel tracking algorithms [60]. These methods are all very specific to both their specific application and data, but do not generalize well and are not easily applicable to event-based space imaging (EBSI) data.

Event-based Space Situational Awareness (EBSSA) is a new and emerging field of study. The most relevant work to that presented here is the frame-based star tracking method proposed in [61]. In this work, an event-based camera captures simulated star data from a monitor and then uses the event-based camera to perform rotation averaging and bundle adjustment using frames made from the event stream. However, this method can only extract a single velocity from a star field not multiple independently moving objects. In addition, the algorithm was tested on simulated data which did not exhibit the noise and dynamics of real-world event-based space imaging environment.

2.1.1 Event-based Space Situational Awareness (EBSSA)

The application of event-based cameras to real-world space imaging leverages the unique nature of the hardware to perform tasks that cannot be undertaken with a conventional camera. It therefore allows for different and novel approaches to space imaging which can overcome many of the current limitations in space situational awareness systems. In previous work, we demonstrated the ability to detect a resident space object in orbits ranging from low-earth orbit (LEO) to geosynchronous orbits (GEO) [55]. We also demonstrated the ability to observe objects during the day with an event-based camera, and without any modifications to the optics.

Figure 2.1 provides a pictorial overview of the benefits of a neuromorphic approach to space imaging. The low-power and low-bandwidth operation of event-based sensors makes them highly suitable for use on orbital platforms, and the ability to synchronize cameras in a highly efficient manner also creates the potential for large distributed SSA observation networks.

The continuous nature of the imaging provided by event-based sensors allows for the camera to image whilst moving, and as a result, allows the device to operate in less stable environments than conventional astronomy cameras. This application requires robust real-time space object detectors and trackers that can operate reliably in the presence of unexpected and random jolts and in the presence of a wide range of noise conditions. This makes the task significantly different from conventional detection and tracking problems.

2.2 Methodology

This section describes the structure and nature of the events generated by the event-based cameras, the method used to generate the event-based space imaging dataset, the methodology used when labeling the dataset and the metrics used to report sensitivity, specificity and

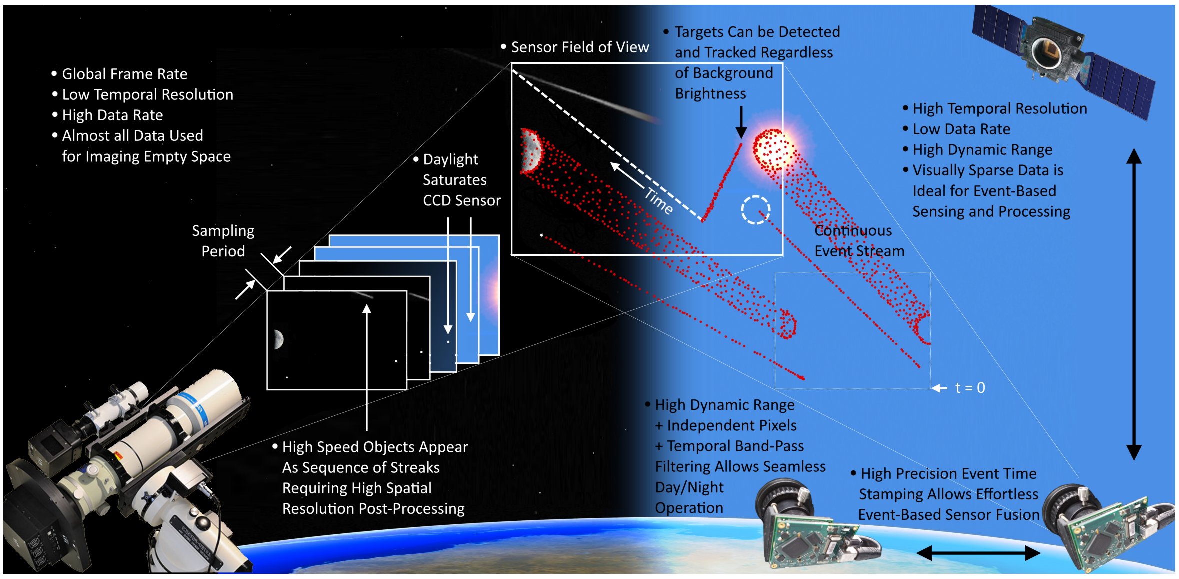

• Global Frame Rate
• Low Temporal Resolution
• High Data Rate
• Almost all Data Used for Imaging Empty Space
• Sensor Field of View
• Targets Can be Detected and Tracked Regardless of Background Brightness
• Daylight Saturates CCD Sensor
Sampling Period
Time
• High Temporal Resolution
• Low Data Rate
• High Dynamic Range
• Visually Sparse Data is Ideal for Event-Based Sensing and Processing
Continuous Event Stream
t = 0
• High Speed Objects Appear As Sequence of Streaks Requiring High Spatial Resolution Post-Processing
• High Dynamic Range + Independent Pixels + Temporal Band-Pass Filtering Allows Seamless Day/Night Operation
• High Precision Event Time Stamping Allows Effortless Event-Based Sensor Fusion

FIGURE 2.1: **Event-based Space Situational Awareness (EBSSA) compared to the standard CCD sensor approach.** Event-based sensors provide high temporal resolution imaging data of the sparse space environment allowing rapid sensor fusion, low bandwidth communication during continuous operation during day and night time.

informedness from the event streams. The section further details a complete event-based detection and tracking system, as well as a discussion of alternatives methods for benchmarking performance.

2.2.1 Generation of the Space Imaging Dataset

The space imaging dataset was captured using both ATIS sensors [23] and DAVIS sensors [32] and was undertaken at the DST Group's research facility in Edinburgh, South Australia, the experiments made use of their robotic electro-optic telescope facility, which was modified to support the event-based sensors and the existing astronomy equipment simultaneously.

The dataset setup and recording profiles are shown in Figure 2.2. Panels (a) and (b) show photographs of the equipment used in the recording of the space imaging data. Two identical telescopes were used for the ATIS and DAVIS event-based sensors alongside a conventional astronomy CCD camera (FLI Proline PL4710). (a) The ATIS camera is attached to the base of the lower telescope with the CCD camera shown at the top. (b) Shows the set up used in the simultaneous co-collects from both the ATIS and DAVIS cameras. Note that the optics for the telescopes for the event-based cameras were not altered between daytime and nighttime operations. Panels (a) and (b) adapted from [55]. Panels (c) and (d) show the distribution of the recordings in terms of duration and number of events respectively. (e) Plots the timestamp of all recordings in the dataset as a function of their index. (f) The Dimetric projection of the event stream from a two-minute recording of the rocket body SL-8 R/B [62] with time as the vertical axis. This projection of the data stream illustrates the high noise rate of a typical EBSI recording. Such non-ideal event timing behavior was observed with both the DAVIS and the ATIS sensors under different conditions.

The conventional telescope configuration comprised an Officina Stellare RH200 telescope and an FLI Proline PL47010 camera. This telescope and camera set-up was used to provide ground truth and to build an accurate mount and pointing model, allowing the event-based cameras to track and to be accurately pointed at objects as is the standard procedure for calibrating a mount and telescope in visual astronomy [63]. The telescopes were both mounted on a Software Bisque Paramount MEII robotic mount, as shown in Figure 2.2 (a). The system is housed in a 7ft Aphelion Dome which also contains a PC that controls the robotic telescope and controls the event-based cameras.

The event-based cameras were attached to an 8" Meade LX200 telescope, as shown in Figure 2.2. When performing co-collects with both event-based sensors, a second Meade LX200 was attached on the other side of the primary telescope as shown in (c). The DAVIS camera used to generate the dataset has a 240×180 pixel resolution at 18 μm with a 2000mm focal length = 7.44×5.58 arc-minutes = 0.124×0.093 degrees. The ATIS camera used has a 304×240 pixel resolution at 30 μm with a 2000mm focal length = 15.66×12.36 arc-minutes = 0.261×0.206 degrees.

With over 8 hours and 377 million events the presented dataset as detailed in Figure 2.2, is the first event-based space imaging dataset in the literature. The dataset consists of 84 separate labeled recordings, 45 using the DAVIS sensor and 39 using the ATIS. The full dataset, supporting material and all processing code proposed in this work can be accessed at [64]

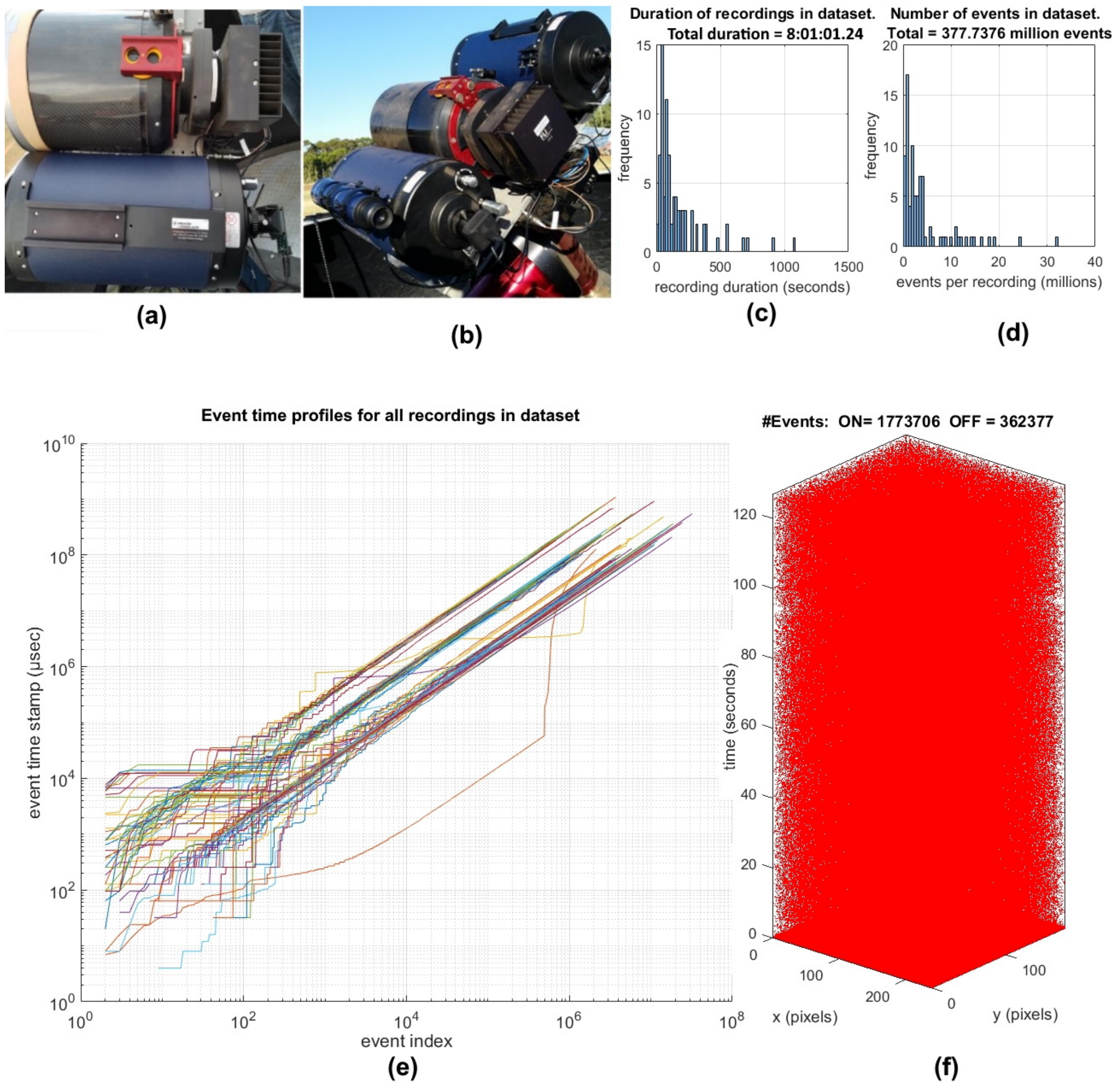


FIGURE 2.2: **Space imaging set up and resulting dataset.**

In addition, a further 152 unlabeled data streams containing 5 hours of recording and containing 2513 million events are provided. These include 15 recordings from the 180×240 DAVIS sensor, and 27, 100 and 7 using an original 304×240 pixel ATIS camera, a larger format 640×480 pixel ATIS prototype camera, and the BSI variant of the DAVIS sensor described in [65].

This larger unlabeled dataset enables further exploration of almost all currently available EBSI data by the research community. As shown in Figure 2.2(e), the time-stamp profiles of all recordings in the dataset show the heterogeneity and non-idealities in the dataset. The discontinuous staircase features in the time-stamp profiles represent event stream timing artifacts. These event stream ‘jumps’ and ‘dumps’ occur when multiple events are erroneously assigned simultaneous time-stamps often at periodic intervals. This effect is likely due to USB communication delays in the cameras.

Presented on a log-log scale, these discontinuities in time and event index can be observed more frequently at the lower scale at the lower-left corner but are present with decreasing frequency at the higher scales, as the plots move to the top-right corner where discontinuities represent more severe artifacts. The effects of these artifacts on the data stream are also illustrated in Figure 2.2(f) where at $t = 0$, a data dump can be observed in the form of a solid square. This particular recording of the rocket body SL-8 R/B is an especially instructive data stream in that it contains nearly all the sensor non-idealities, scene complexities and processing challenges that can be found in the dataset as a whole. It will therefore be used repeatedly in this work to illustrate many of the event-based processing problems and solutions presented in this work.

2.2.2 Labelling the Dataset

Generating ground truth labeling for real-world event-based space imaging data is a non-trivial task. Even when the true position, velocity, size and luminance of all targets in the field of view of the sensor are known, their detection by the event-based sensor is far from guaranteed. The clearest demonstration of this problem is in cases where within the same recording, the biases and circuitry of the camera are configured optimally for one event polarity such that space objects are clearly visible in one polarity but produce zero events in the other polarity. In these and analogous situations the use of any ground truth labels from external information sources such as the co-collects from the CCD camera (or a sky catalog or database as used in [61]), would likely result in an incorrect evaluation of any event-based algorithm operating on the actual observed real-world event stream.

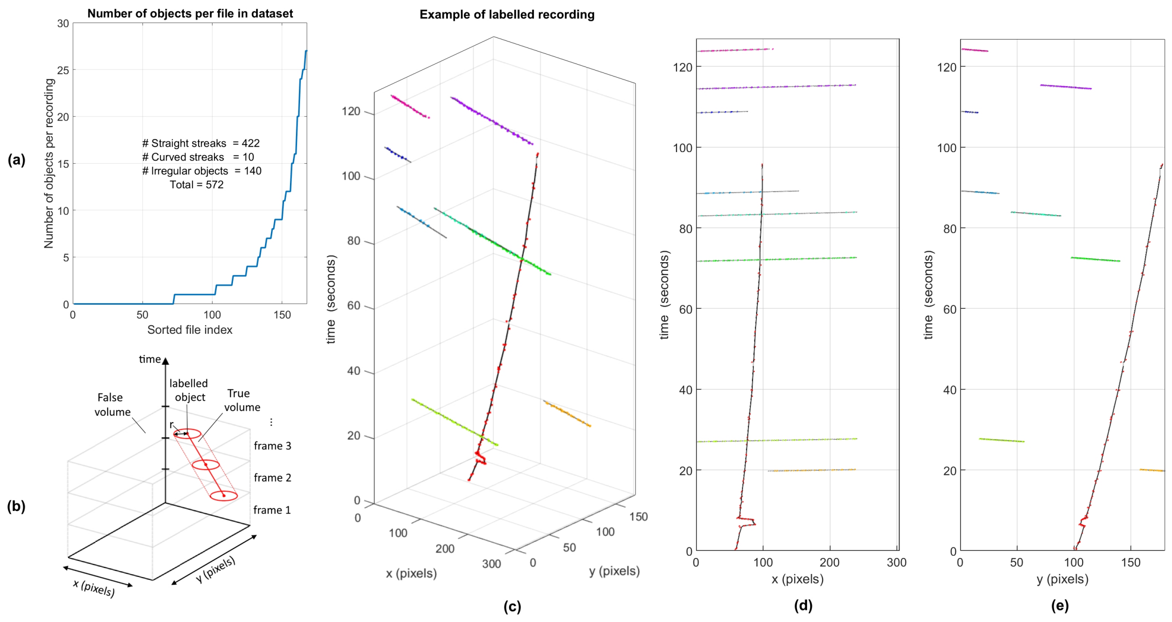


FIGURE 2.3: **Dataset labeling.** (a) Total number of sub-types and the number of objects per recording in the dataset shown as a sorted list. The 422 straight streaks represent objects that exhibit zero acceleration and move in a straight line in space-time. The 10 Curved streaks were object observed to exhibit uniform acceleration and 140 irregular objects exhibited non-uniform acceleration while in the field of view. (b) Illustrates the method used for calculating sensitivity and specificity of event volumes around labeled data points. A volume of radius r around a line connecting the labeled points marks the boundary between true and false volumes. The volumes are sliced at 10ms intervals. The event density of each sub-region designates its volume as a positive or negative volume depending on whether it is above or below the mean density of the recording as a whole. Panels (c), (d) and (e) show the expert labeled objects in the SL-8 R/B recording in a dimetric projection and across the x and y-axis respectively.

A single instance of such a comparison was recently performed in [66] where the DAVIS event-based sensor was estimated to have lower sensitivity relative to the CCD sensor. This difference in relative sensitivity was measured via the limiting magnitude, defined as the faintest magnitude of a celestial body that is detectable. The event-based sensor was estimated to provide sensitivity with magnitudes between 1.32 and 1.78 less than the CCD sensor. While these results do not necessarily generalize to other event-based sensor configurations and recording environments, they do highlight the general problem of using external labels to evaluate event-based data. This work evaluates the tracking algorithms and not the performance of the sensor. Hence, ground truth from a different sensor type (such as a conventional CCD) does not directly allow us to predict the accuracy of the tracker. Thus we require a ground truth related to the events generated from the camera, and not from external label sets.

For this reason, to generate a more appropriate label set for the observed event streams, hand-labeling of the data was performed, and a committee-of-experts approach was used to determine the ground truth labels. Thus expert human labeling of the highly noisy event stream is here set as a benchmark against which proposed event-based algorithms are tested.

In Section 2.3.1, the quality of expert human labeling performance is tested and quantified using an artificially generated space imaging dataset in which ground truth labels are analytically defined. The generated labeled dataset involves a multi-stage labeling and editing procedure where each of four experts sequentially view and label visible objects in each recording using a graphical user interface which allows the viewer to move forward or backward through 2D time surface frames of the event stream at arbitrary frame rates with a maximum sampling frequency of 1000Hz. The use of multiple experts and multiple stages of labeling and editing aimed to maximize the accuracy of the labeled dataset. Targets were tagged based on their motion profiles into straight streaks, curved streaks, or irregularly moving objects as detailed in Figure 2.3(a). Target entry and exit points, as well as segments of the trajectory exhibiting acceleration, were all marked manually.

These marked points were then linked programmatically via linear interpolation. After the first-round of labeling, the experts performed a second editing round with access to their first-round labeling information as well as those of the other experts. Before the commencement of the labeling procedure, a three-out-of-four voting protocol was devised for resolving any disagreement between the experts after the second round of labeling. Ultimately no such disagreements occurred, resulting in consensus for all labels without the need for the voting protocol.

After the expert labeling was finalized, the four interpolated label sets were averaged to generate a single, labeled dataset.

2.2.3 Measuring Sensitivity, Specificity and Informedness

The algorithms presented in this work are entirely event-based with all components from the sensors to the detectors and trackers operating entirely in the event-based domain. The microsecond time resolution of the sensor is therefore maintained throughout the processing chain. A brief explanation of event-based processing is provided below.

Following the notation in [60], events generated at the sensor, $\boldsymbol{e}_i$ can be described mathematically as:

$$\boldsymbol{e}_i = [\boldsymbol{x}_i, t_i, p_i]^T \tag{2.1}$$

where i is the index of the event, $\boldsymbol{x}_i = [x_i, y_i]^T$, is the spatial address of the source pixel corresponding to the physical location on the sensor, $p_i \in -1, 1$ is the polarity of each event indicating whether the log intensity has increased or decreased, and t_i is the absolute time at which the event occurred. The timestamp t_i has a temporal resolution of 1μs and is applied to the event in hardware within the event-based sensor.

The data from these sensors, therefore, have a high temporal resolution, with the event rate varying for each pixel and dependent on the activity in the scene. A robust method is required for measuring how well a given event stream sampled at 1 MHz matches the frame-based expert labeled dataset which is sampled at a much slower 1 kHz. This accuracy measure must also be invariant to the extreme differences in event rates produced by different recording conditions. The measure must also assess the highly noisy raw events of the sensor in the same manner as the extremely sparse detection and tracking output event streams. To achieve this, a metric based on relative event density in the event stream is proposed. This method assigns spatio-temporal volume slices to either a positive or a negative state. These states are then compared to the labeled dataset which indicates whether the corresponding volume contains target objects (True), or not (False).

As shown in Figure 2.3(b), for each frame, the spatio-temporal volume slice surrounding the trajectory of a labeled object by radius r is designated as True and the spatio-temporal volume outside this region and in frames with no labeled object is designated as False. If, for any spatio-temporal volume, the event density is above the global event density of the full recording, then the volume is activated as positive. Conversely, the volume is designated as negative if the event density in the volume falls below the global event density of the full recording (i.e. if there are relatively fewer events per pixel^2/second in the local volume slice than the total number of events divided by the total recording multiplied by the sensor area).

In this way, event streams with drastically different noise profiles and event densities can be directly compared and evaluated by calculating the mean True Positive ($\overline{TP}$), True Negative ($\overline{TN}$), False Positive ($\overline{FP}$) and False Negative ($\overline{FN}$) volumes of each recording. Using these volume-based measures, the event-based sensitivity and specificity of a particular event stream can be calculated using:

$$Sensitivity = \overline{TP}/(\overline{TP} + \overline{FN}) \tag{2.2}$$

$$Specificity = \overline{TN}/(\overline{TN} + \overline{FP}) \tag{2.3}$$

Using these measures, the informedness, or the Bookmaker Informedness of an event stream can be calculated using (2.4). Informedness, which is a generalization of the Youden's J statistic, provides a single statistic that captures the performance of a binary diagnostic test [67], and "quantifies how informed a predictor is for the specified condition, and specifies the probability that a prediction is informed in relation to the condition (versus chance)" [68].

$$Informedness = Sensitivity + Specificity - 1 \tag{2.4}$$

Informedness seeks to avoid biases of other common statistics, such as accuracy and precision, which are susceptible to population prevalence and label bias. This makes informedness an accuracy measure suitable for the highly imbalanced EBSI datasets in which the vast majority of the spatio-temporal volumes are labeled as False regions.

TABLE 2.1: **Event density activated volume statistics for measuring the performance of the event stream against labels.** Here the statistics are calculated from the raw events from the SL-8 R/B recording whose data stream is illustrated in Figure 2.2(f), and whose labels are shown in Figure 2.3(c). Due to the high disparity in data stream SNRs and event rates, the ON and OFF polarities are treated as independent data streams.

Polarity	**Sensitivity**	**Specificity**	**Informedness**	**# Events (ke)**
ON Events	0.69	0.68	0.37	1770
OFF Events	0.65	0.79	0.43	360

As an example, the event density activated volume statistics for the SL-8 R/B recording are detailed in Table 2.1 showing clear differences between the raw ON and OFF event streams.

2.2.4 Artificial Space Imaging Dataset

Given the difficulty of obtaining real-world space imaging data, the collected dataset was augmented and extended using a large analytically defined artificial dataset. The artificial dataset was designed to provide analytical ground truth and tested on both human experts via the same labeling protocol as used in the real space imaging dataset described in Section 2.2.2.

This additional artificial dataset serves to verify the quality of the expert labeling and enable a more extensive and detailed analysis of the proposed algorithms across analytically defined Signal-to-Noise Ratios (SNRs) and event rates.

Furthermore, the artificial dataset was designed to contain examples of the most important and challenging aspects of the real space imaging dataset, such as:

(1) **Multiple concurrent objects with independent trajectories and velocities**: In the SSA applications, where a target of interest is often being tracked, the target typically exhibits slow and often non-uniform relative motion whilst the background star field moves with a different velocity across the sensor field of view. Figure 2.3(c) is an example of such a tracking operation with SL-8 R/B as the target. To emulate this context, each recording in the artificial dataset contains a slow-moving target along with two other targets each moving with independent velocities.

(2) **Sharp discontinuities in object trajectories**: As shown in Figure 2.3 real-world space imaging data can contain high acceleration saccade-like shifts in the field of view due to mechanical vibrations or acceleration of the sensor field of view due to tracking. To replicate this effect, a discontinuity is introduced in the velocity of one of the objects.

(3) **Wide range of background noise event rates and target rates**: As real-world event-based space imaging data streams exhibit a wide range of event rates and SNRs, the artificial dataset must also test across a wide range of noise and target event rates. For each object, pixels within a three-pixel radius exhibit an event rate of λ_1 whereas the event rate of pixels outside this radius represents the background noise rate λ_0. In the artificial dataset experiments, the signal event rate is varied on a logarithmic scale from $\lambda_1 = 10^{-1}$ to 10^2 and the noise event rate from $\lambda_0 = 10^{-4}$ to 10^0 events per pixel per second. In comparison, the event rate of the real-world space imaging dataset is $\lambda_S = 0.240 \pm 0.197$ events per pixel per second.

The artificial dataset is described analytically as three objects whose trajectories are defined by (2.5), (2.6) and (2.7). The first, representing a slowly moving object being tracked, is defined by:

$$\boldsymbol{Q}_1 = [x_1, y_1]^T = [\beta_1^{(x)} + \alpha_1^{(x)} t/t_{max}, \beta_1^{(y)} + \alpha_1^{(y)} t/t_{max}]^T \tag{2.5}$$

where $\boldsymbol{Q}_1$ is the object location, $\alpha_1^{(x)}, \alpha_1^{(y)} \in \{-20, 20\}$ are the velocities of the object, $t_{max} = 10$ seconds is the duration of the data stream and $\beta_1^{(x)}, \beta_1^{(y)} \in [50, 150]$ is the random starting location of each of the object.

The second object, $\boldsymbol{Q}_2$ is defined as a circularly moving object, representing a more rapid and potentially non-linear motion of background targets:

$$\begin{aligned} \boldsymbol{Q}_2 = [x_2, y_2]^T = [&\beta_2^{(x)} + \alpha_2 cos(\omega_2 t/t_{max} + \phi_2), \\ &\beta_2^{(y)} + \alpha_2 sin(\omega_2 t/t_{max} + \phi_2)]^T \end{aligned} \tag{2.6}$$

where $\beta_2^{(x)}, \beta_2^{(y)} \in [50, 150]$ are the random starting locations, $\alpha_2 = 100$ is the diameter of the spiral, and $\omega_2 \in \{-3\pi, 3\pi\}$ and $\phi \in [0, 2\pi)$ are the angular velocity and phase respectively.

$$\begin{aligned} \boldsymbol{Q}_3 = [x_3, y_3]^T = [|\beta_3^{(x)} + \alpha_3 cos(\omega_3 t/t_{max} + \phi_3)|, |\beta_3^{(y)} + \\ \alpha_3 sin(\omega_3 t/t_{max} + \phi_3)|]^T \end{aligned} \tag{2.7}$$

Finally, the third object in the test introduces the sharp discontinuities in velocity which can result from sudden jerk-like motion of the sensor. This is visible in 2.3(c) from $t = 6$ and 8 seconds. This jerk-like motion is represented through the addition of a discontinuity in the form of the absolute value function operating on a circularly moving object with random initial position $\beta_3^{(x)}, \beta_3^{(y)} \in [50, 150]$, diameter $\alpha_3 = 100$, angular velocity $\omega_3 \in \{-2\pi, 2\pi\}$ and phase $\phi_3 \in [0, 2\pi)$ which together result in a zigzagging spiral pattern in space-time.

The randomized instantiations of these three objects together with the signal and noise event rates λ_1 and λ_0 define the artificial dataset. An example recording from the artificial dataset, as well as the associated ground truth labels and algorithm output, is shown in Figure 2.9 in the Results section.

2.2.5 Event Pre-processing

Event-based algorithms require as input, some form of memory of recent events. Such a memory can be generated via a range of methods that are investigated in Chapter 3. The method used in this section and one which typically outperforms other approaches is the exponentially decaying event time surface. This method weighs each pixel as an exponentially decaying function of the time since the most recent event as described by (2.8), (2.9) and (2.10).

$$\boldsymbol{T}_i = \mathbb{R}^2 \Rightarrow \mathbb{R} \tag{2.8}$$

$$\boldsymbol{x} : t \Rightarrow \boldsymbol{T}_i(\boldsymbol{x}) \tag{2.9}$$

$$\boldsymbol{S}_i(\boldsymbol{x}) = e^{(\boldsymbol{T}_i(x)-t_i)/\tau} \tag{2.10}$$

Here, $\boldsymbol{T}_i(\boldsymbol{x})$ is the matrix containing the time-stamp of the most recent event at each pixel $\boldsymbol{x}$ at the ith event and $\boldsymbol{S}_i(\boldsymbol{x})$ is the corresponding exponentially decaying time surface and $\tau = 0.4$ seconds is the decay constant. Note that in this work, as each event polarity is processed independently such that the time surface receives events of only one polarity. This approach is not typically used in event-based algorithms since it decouples ON and OFF event information at the lower processing stages and may potentially result in poorer performance. However, in the event-based space imaging context where the signal and noise event rates ON and OFF events can differ significantly depending on biases and the imaging environment, separating the polarities not only allows adaptive processing based on the event rate of each polarity, but also effectively doubles the dataset while better representing the difficulty of the real-world detection and tracking task. By splitting the event polarities and processing them independently, we can better replicate a wider range of observational environments where potentially the biases of both polarities are ill-suited to the recording environment. Given the sparseness of event-based space imaging data, this worst-case design approach aims to motivate the development of the more noise robust space object detection and tracking algorithms.

As shown in 2.4(a), after updating the time surface, a Region Of Interest (ROI) of size $D \times D$ around the event $\boldsymbol{e}_i = [x_i, y_i, t_i, p_i]^T$ is selected for processing. The $\boldsymbol{ROI}_i$ associated with event is defined as:

$$\boldsymbol{ROI_i} = \boldsymbol{S}_i(x_i + \boldsymbol{u}_x, y_i + \boldsymbol{u}_y) \tag{2.11}$$

where $\boldsymbol{u}_x = [-R : R]$ and $\boldsymbol{u}_y = [-R : R]$ subject to the constraint:

$$\sqrt{x^2 + y^2} \leq R, \forall x \in \boldsymbol{u}_x, \forall y \in \boldsymbol{u}_y \tag{2.12}$$

Thus the $\boldsymbol{ROI}_i$ generated by event $\boldsymbol{e}_i$ is defined as a disc containing the time surface values $\boldsymbol{S}_i$ at time t_i from all pixels within a distance R to the location of the current event $\boldsymbol{e}_i$. This $\boldsymbol{ROI}_i$ is then processed by a surface activation test which is defined as:

$$L < \sum_{x=x_i-R}^{x_i+R} \sum_{y=y_i-R}^{y_i+R} \big(\boldsymbol{T}_i(x,y) > \Phi\big) \tag{2.13}$$

where Φ is the event activation time interval, L is the number of activated pixels required and x and y are subject to the distance constraint given in (2.12). Thus, if the number of recently activated pixels on the time surface within a disc of radius R around the current event $\boldsymbol{e}_i$ is above L, then the ROI is accepted. Here recency is defined as a pixel that has received an event within Φ seconds. This surface activation test effectively acts as a noise filter and is a generalization of the nearest neighbor filter described in [69] where R was selected as 1. The expansion of the spatio-temporal activation test window is crucial here in the space imaging context due to the significantly lower SNR recording environments and the similarity of the most challenging targets (small dim geostationary satellites) to background noise.

2.2.6 Feature Detection

In the next stage of processing, a valid ROI is converted to an angular activation vector $\boldsymbol{\Lambda}$, generated by multiplying the ROI with each of N rotated half-bar templates shown in Figure 2.4.

The half-bar templates are designed to be triggered by events at the tip of a moving streak. The template consists of a bar of length $R + 1$ and three pixels wide with a magnitude of one. While setting the bar width parameter at three pixels appears an arbitrary choice, this pattern was found heuristically to produce the best performance across a wide range of object sizes and ROI sizes. This is likely due to the three pixels bar being close to the size of the smallest resolvable streak in the space imaging dataset.

Outside of the bar, the rest of the template has a negative magnitude of $s = -0.2$ to penalize activation from noise events. In practice the N, $D \times D$ templates are re-arranged into an

$N \times D^2$ Look Up Table (LUT) and the $D \times D$ ROI vector is rearranged to a $D^2 \times 1$ vector. This vectorization operation is here denoted as the vec() function. The multiplication of the ROI vector and the LUT results in an $N \times 1$ $\mathbf{\Lambda}$ vector as described by 2.14 and illustrated in Figure 2.4(f).

$$\mathbf{\Lambda}_i = \boldsymbol{LUT} \cdot \text{vec}(\boldsymbol{ROI}_i) \tag{2.14}$$

Note that since only the internal disc of radius $R = (D-1)/2$ is processed, the length of the LUT and the ROI vector can be reduced by a factor of $1 - \pi(1/2)^2 = 0.2146$ during hardware implementation. However when implemented in a software environment, the cost of retrieving the smaller circular ROI from the $D \times D$ time surface patch at each event outweighs the computational reduction provided by the smaller ROI, necessitating the use of the full D2 length template vector and LUT with zero padding for entries outside the disc. Thus by using the rearranged LUT, the calculation of angular activation vector $\mathbf{\Lambda}_i$ from $\boldsymbol{ROI}_i$ is converted to a single vector-matrix multiplication operation. Where libraries of optimized matrix routines are available, such LUT transformations can result in significantly faster processing.

Optimization of the subsystem that converts the ROI event timestamps to the angular activation vector $\mathbf{\Lambda}$ is critical in the performance of the proposed algorithm. The calculation of the angular activation vector is, regardless of the implementation environment, significantly more computationally expensive than that of the previous surface activation test, but unlike subsequent stages which are also computationally intensive, this operation is performed on the majority of the events from the camera. This makes the calculation of the angular activation vector the most computationally expensive step relative to the number of events processed, making it a computational bottleneck of the algorithm. This aspect of the algorithm is investigated in more detail in Results section 2.3.3.

The resulting angular activation vector $\mathbf{\Lambda}$ is compared to an angular activation threshold of Ψ and if no element of $\mathbf{\Lambda}$ exceeds Ψ the angular activation test fails, otherwise the variable m, which is defined as the index of the highest activated element of $\mathbf{\Lambda}$, is passed to the next stage of processing along with the vector $\mathbf{\Gamma}$ which contains the index of all elements in $\mathbf{\Lambda}$ above threshold Ψ.

The threshold used for calculating $\mathbf{\Gamma}$ can be chosen as a static parameter Ψ, or as a dynamic threshold Ψ_i which is defined as a scalar factor W of the difference between the minimum and maximum of elements of $\mathbf{\Lambda}_i$ as described in (2.15). Use of a static threshold Ψ simplifies the algorithm implementation whereas the use of a dynamic threshold can provide greater robustness to noise events. Except where stated, in this work, the dynamic method is used with $W = 0.5$.

$$\Psi_i = W(\max(\mathbf{\Lambda}_i) + \min(\mathbf{\Lambda}_i)) \tag{2.15}$$

The angular activation test serves as a filter that removes all ROIs with uniform activation in polar coordinates. This filter is useful in removing events not associated with a streak on the time surface. However, this filter does not distinguish between events which are on or near a streak and those at the streak's tip. To further extract these later events, a statistical unimodality test must be applied to the angular activation vector $\mathbf{\Lambda}$. Previously proposed unimodality tests include fitting of parametric mixture models [70], as well as non-parametric tests such as the commonly used Dip Test [71], use of kernel density estimates [72] and recursive methods based on unimodal template transformations [73]. While these methods can provide robust solutions to the unimodality test, they are too computationally expensive for the streak tip detection application where thousand of events must potentially be processed per second possibly by a low power processor on a space-based platform. Therefore a highly simplified hardware amenable circular unimodality test for our event-based application is proposed. This unimodality test simply measures the angular distance between the maximum value in $\mathbf{\Lambda}$ and the angular mean of all elements above a threshold Ψ_i.

As shown in Figure 2.4(a), the unimodality block takes as input m_i which is the index of the largest element of $\mathbf{\Lambda}_i$. It also takes as input a vector $\mathbf{\Gamma}_i$ containing the index of all elements in $\mathbf{\Lambda}_i$ higher than Ψ_i: $\mathbf{\Gamma}_i = \{n\}$ s.t. $\mathbf{\Lambda}_i(n) > \Psi_i$. The unimodality block then outputs a stream of filtered events f_j which have passed the unimodality test. As plotted in Figure 2.4(f), the unimodality block performs its test by calculating q_i which is the circular mean of $\mathbf{\Gamma}_i$ and testing whether the angular distance ζ_i between q_i and m_i is below a parameter δ. The distance ζ_i represents how far the peak angular activation is from the mean. This makes ζ_i a simplified yet robust measure of the unimodality of the angular activation vector $\mathbf{\Lambda}_i$.

Despite its simplicity, this unimodality test is highly selective and performs remarkably well at extracting space targets from the observed event-based space event streams while being robust to noise, object velocity, object size and orientation. If the event e_i passes this angular unimodality test, it is augmented with the mean orientation variable $\theta_i = q_i$ and results in an output detection event $\boldsymbol{f}_j$ as described by Algorithm 2.1, and illustrated in Figure 2.4(a). Note that in Algorithm 2.1, G denotes the number of above-threshold elements in Λ_i and is therefor the length of the vector Γ_i.

One important factor is the method used to calculate the circular mean value q. The most direct approach is via calculating the mean two-argument arctangent equation given in:

$$\overset{\circ}{\bar{x}} = \text{atan2}\left(\frac{1}{n}\sum_{j=1}^{n}\sin x_j, \frac{1}{n}\sum_{j=1}^{n}\cos x_j\right) \tag{2.16}$$

However, there are two drawbacks to this method for our event-based space imaging application. First, the method is computationally complex, making implementation in embedded hardware more difficult. Second, for some pathological input cases such as that shown in Figure 2.5, this direct method can result in an undesirably small circular distance $\zeta = |m - q|$ between the circular mean q and the maximum angular value m as shown in Figure 2.5. In the space imaging dataset, these pathological cases make up a small but consistent fraction of the observed ROIs occurring regularly whenever events are triggered late in the trail of a fast-moving target.

As shown in Figure 2.5, these trail events generate bimodal distributions of $\mathbf{\Lambda}$ which regularly have circularly symmetric elements that can cancel each other out. In such cases, the standard circular mean method results in the circular mean index q and circular max index m being close enough to generate false positive detections. To provide robustness to these streak trail events, a non-circular mean index q is calculated over the template indices vector $\mathbf{\Gamma}$ generated from the angular activation vector $\mathbf{\Lambda}$. A non-circular distance $\zeta = |m - q|$ between the

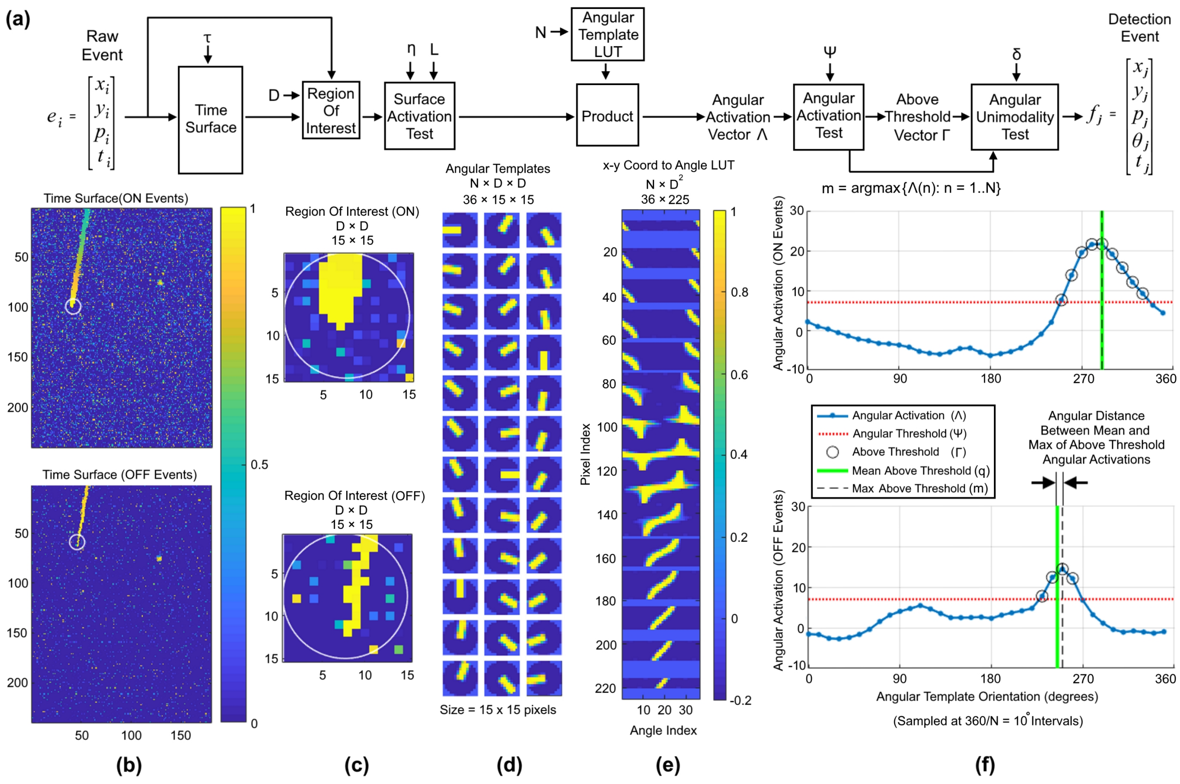

(a)
Raw Event
$e_i = [x_i, y_i, p_i, t_i]$
τ
Time Surface
D
Region Of Interest
η L
Surface Activation Test
N
Angular Template LUT
Product
Angular Activation Vector Λ
Ψ
Angular Activation Test
Above Threshold Vector Γ
δ
Angular Unimodality Test
Detection Event
$f_j = [x_j, y_j, p_j, \theta_j, t_j]$
m = argmax{Λ(n): n = 1..N}
Time Surface(ON Events)
Time Surface (OFF Events)
Region Of Interest (ON) D × D 15 × 15
Region Of Interest (OFF) D × D 15 × 15
Angular Templates N × D × D 36 × 15 × 15
Size = 15 x 15 pixels
x-y Coord to Angle LUT N × D² 36 × 225
Pixel Index
Angle Index
Angular Activation (ON Events)
Angular Activation (OFF Events)
Angular Activation (Λ)
Angular Threshold (Ψ)
Above Threshold (Γ)
Mean Above Threshold (q)
Max Above Threshold (m)
Angular Distance Between Mean and Max of Above Threshold Angular Activations
Angular Template Orientation (degrees)
(Sampled at 360/N = 10° Intervals)
(b)
(c)
(d)
(e)
(f)

FIGURE 2.4: **Orientation invariant space object detection algorithm and signals at each stage of processing.** Panel (a) shows the block diagram of the algorithm whereby a sequence of increasingly refined tests operate on an event e_i. If the event passes all test a detection event $\boldsymbol{f}_j$ is generated. (b) Shows an instance of the ON and OFF time surface for the SL-8 R/B recording. Note the different noise levels and target sensitivity of the two polarities. (b) Shows the local 15x15 Region Of Interest ($\boldsymbol{ROI}_i$) around the current event e_i for each polarity. (d) $N = 36$ Streak templates rotated at 10-degree intervals to calculate the angular activation of the ROI. (e) a stored Look Up Table (LUT) converts the ROI values to an angular activation vector $\boldsymbol{\Lambda}$ through a single vector-matrix multiplication operation. (f) the resultant angular activation is shown for each of the ON and OFF ROIs. If $\boldsymbol{\Lambda}$ exceeds the angular threshold Ψ, it passes the angular activation test after which the circular mean index q of all angles above the angular threshold $\boldsymbol{\Gamma}$, is calculated. If the distance ζ between q to the maximally activated angle m is below the threshold δ the event passes the angular unimodality test resulting in a detection event $\boldsymbol{f}_j$. Note that for visual simplicity, both the static and the dynamic angular activation thresholds are made static and equal with $\Psi = \Psi_i = 7$.

non-circular mean q and maximum angular index m is then calculated. The same operation is then performed on $\breve{\Lambda}$ which is $\boldsymbol{\Lambda}$ circularly shifted by $N/2$. These two operations result in two non-circular distances ζ and $\breve{\zeta}$ the smaller of which is compared to a threshold δ. This

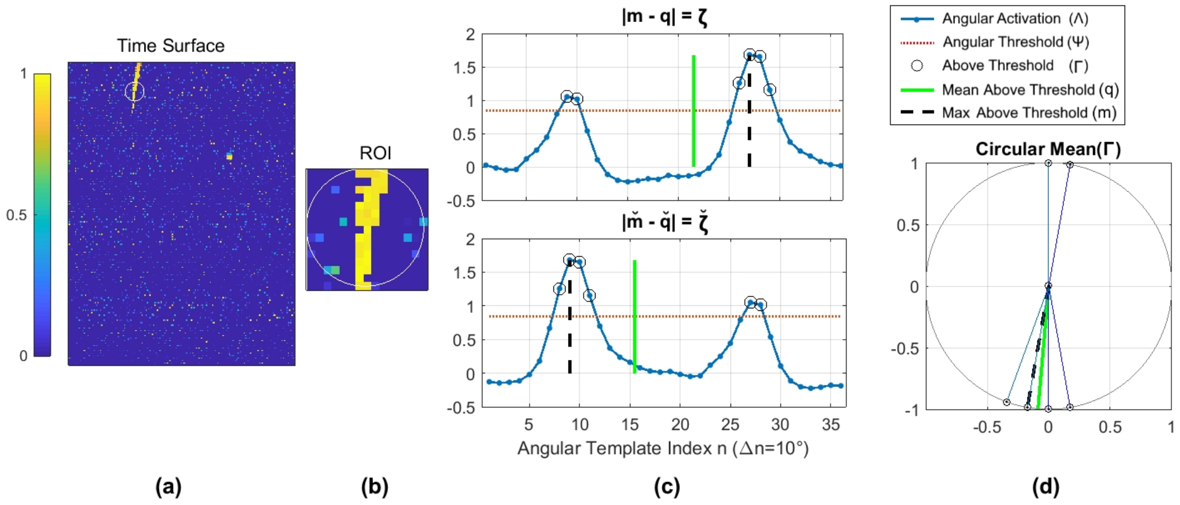


FIGURE 2.5: **Comparison of two methods of calculating the circular distance ζ to estimate unimodality of angular activation $\mathbf{\Lambda}$.**(a) Shows the OFF polarity time surface from the SL-8 R/B recording as a high-speed target enters the field of view. (b) A late-triggered event results in an ROI centered on a pixel on the trail of a streak. (c) Calculating the non-circular means of $\mathbf{\Gamma}$ and its circular shifted version $\breve{\mathbf{\Gamma}}$ results in q and $\breve{q}$ respectively and angular distances ζ and $\breve{\zeta}$ both of which are larger than δ, thus (correctly) failing the unimodality test. (d) When calculating the circular mean of $\mathbf{\Gamma}$ via (2.16) the symmetric entries of $\mathbf{\Gamma}$ cancel each other resulting in a circular distance ζ which is smaller than δ thus (incorrectly) passing the unimodality test and generating a false positive detection.

comparison between the minimum distance between the maximum element of $\mathbf{\Lambda}$ and the mean element of $\mathbf{\Gamma}$ represents the unimodality test as described in:

$$\min(\zeta, \breve{\zeta}) = \min(|m_i - q_i|, |\breve{m}_i - \breve{q}_i|) < \delta \tag{2.17}$$

To illustrate the response of this feature detection system to the most commonly observed ROI patterns in the space imaging environment, Figure 2.6 shows the response of the system to streaks of various sizes, lines and noise.

Note that due to the rotational invariance of the algorithm, the responses shown are nearly identical regardless of the orientation of the different features in the ROI. This feature-based detection method is detailed in Algorithm 2.1.

Algorithm 2.1
Feature Detection

```
Require: $\boldsymbol{e}_i = [x_i, y_i, p_i, t_i]^T, i \in \mathbb{N}$
Ensure: $\boldsymbol{f}_j = [x_j, y_j, p_j, \theta_j, t_j]^T, j \in \mathbb{N}$
  $j \Leftarrow 0$
  for every event $\mathbf{e_i}$ do
    $\boldsymbol{T}_i(x_i, y_i) \Leftarrow t_i$
    $\boldsymbol{ROI_i} \Leftarrow \boldsymbol{S}_i(x_i + \mathbf{u_x}, y_i + \mathbf{u_y})$ via (2.10)
    if $\boldsymbol{ROI}_i$ contains recent events verifying (2.13) then
      $\boldsymbol{\Lambda}_i \Leftarrow \boldsymbol{LUT} * vec(\boldsymbol{ROI}_i)$
      if $\max(\boldsymbol{\Lambda}_i) > \Psi$ then
        $\breve{\boldsymbol{\Lambda}}_i \Leftarrow \Big[\boldsymbol{\Lambda}_i\big((N/2+1):N\big), \boldsymbol{\Lambda}_i\big(1:N/2)\big)\Big]$
        $\Psi_i \Leftarrow W(\max(\boldsymbol{\Lambda}_i)+\min(\boldsymbol{\Lambda}_i))$
        $\boldsymbol{\Gamma}_i \Leftarrow \{\text{n}\}$ s.t. $\boldsymbol{\Lambda}_i(n) > \Psi_i$
        $\breve{\boldsymbol{\Gamma}}_i \Leftarrow \{\text{n}\}$ s.t. $\breve{\boldsymbol{\Lambda}}_i(n) > \Psi_i$
        $q_i \Leftarrow 1/G \sum_{h=1}^{G} \boldsymbol{\Gamma}_i(h)$
        $\breve{q}_i \Leftarrow 1/G \sum_{h=1}^{G} \breve{\boldsymbol{\Gamma}}_i(h)$
        $m_i \Leftarrow \text{argmax}_n \boldsymbol{\Lambda}_i(n)$
        $\breve{m}_i \Leftarrow \text{argmax}_n \breve{\boldsymbol{\Lambda}}_i(n)$
        if $\min(|m_i - q_i|, |\breve{m}_i - \breve{q}_i|) < \delta$ then
          $j \Leftarrow j + 1$
          $\theta_j \Leftarrow q_i$
          $\boldsymbol{f}_j \Leftarrow [x_i, y_i, p_i, \theta_j, t_i]^T$
        end if
      end if
    end if
  end for
```

TABLE 2.2: **Event density activated volume-based statistics for measuring the performance of the feature detection events $\boldsymbol{f}_j$.** The statistics calculated are from the detection events generated using the SL-8 R/B recording whose data stream is illustrated in Figure 2.7

Polarity	Sensitivity	Specificity	Informedness	# Events (ke)
ON Events	0.66	0.99	0.65	22.20
OFF Events	0.60	0.98	0.58	15.98

To illustrate in detail the behavior of the feature detector on a real space imaging data stream, the detection event stream generated from the SL-8 R/B recording is shown in Figure 2.7 with the associated statistics shown in Table 2.2.

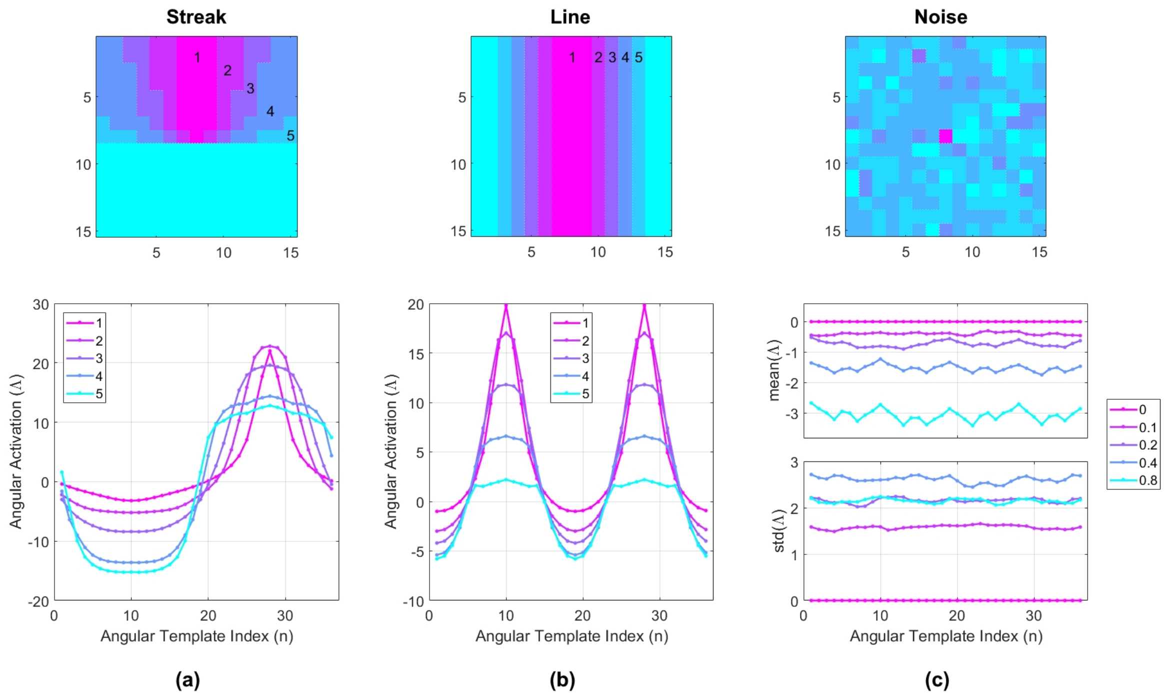


FIGURE 2.6: **Angular activation vectors $\mathbf{\Lambda}$ generated by different ROI content.**(a) Top panel: ROI containing streaks of increasing size from smallest (1) to biggest (5) which covers an entire half of the ROI. Bottom panel: The resulting $\mathbf{\Lambda}$ vectors demonstrate that irrespective of streak size, the unimodality test of (2.17) holds. (b) Top panel: ROI containing lines of increasing size from smallest (1) to biggest (5). Bottom panel shows the resultant bimodal distribution in $\mathbf{\Lambda}$ from the increasingly thicker lines. Note that as the line thickness increases the maximum magnitude angular activation vector falls such that the resultant $\mathbf{\Lambda}$ for lines 4 and 5 falls below a typically selected threshold Ψ=7. (c) Top panel: ROI with pure noise input. Bottom panel: Resultant $\mathbf{\Lambda}$ from noise distributions of different event densities with the probability of an event per pixel per τ seconds $\mathbf{P}[e]$ being varied from 0 to 0.8. Note that here, the mean $\mathbf{\Lambda}$ over 100 trials is always non-positive and the standard deviation begins at zero for $\mathbf{P}[e]$ = 0 and rises to just below 3 before falling again as the time surface becomes saturated with events and thus becomes uniform. Thus regardless of the noise level, the maximum activation of $\mathbf{\Lambda}$ remains significantly lower than a typically chosen static threshold $\Psi = 7$.

As shown in Figure 2.7, due to different noise characteristics and sensitivity of the ON and OFF polarities, significantly different detection event streams are generated from each of the polarities. Also, note that the high-velocity streaks exhibit discrete orientation distributions whereas the slow-moving object being tracked in the field of view generates a uniform

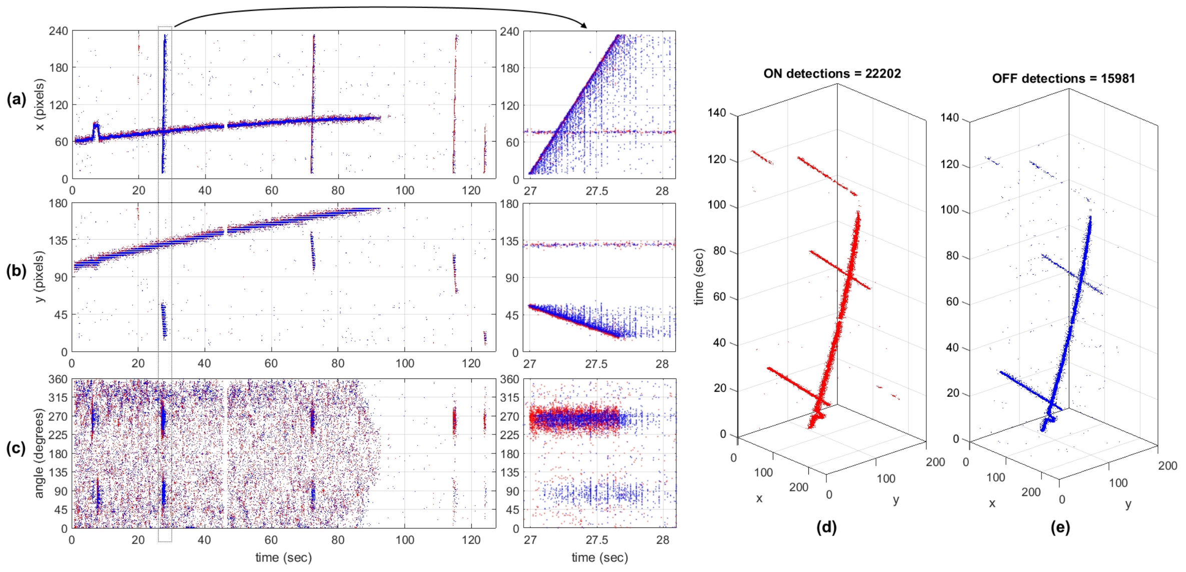


FIGURE 2.7: **Feature detection events from the SL-8 R/B recording.** The panels on the left in (a), (b) and (c) show the x and y location and orientation of the detection events respectively over time for the ON (red) and OFF (blue) detection events. The dashed rectangle marks the time interval around the detection of a high-speed object shown in the close-up right-hand side panels. The panels in (c) show the dominant orientation of the detection events, based on the mean index of above-threshold templates $\theta_i = q_i$. Note that the temporal event bands in the close-up panels are artifacts caused by the data interface. Panels (d) and (e) show the dimetric projections of the ON and OFF detection events respectively.

distribution $\theta \in [0, 2\pi)$ since the later generates a circular image on the time surface triggering detection events that are approximately equally in all directions.

As SL-8 R/B leaves the field of view, the uniform distribution also fades away, leaving only the orientation traces from the high-speed streaks. Also, note a 180 degree shifted angular ‘shadow’ generated by high-speed targets especially for the noisier OFF events. These false detections, which are pointed in the opposite direction to the true angle of the object’s trajectory, are due to late-triggered events along the tail of the streak. These detections have an equal likelihood of being oriented forward or backward. As shown in the close-up panels in Figure 2.7(a)(b) and (c), even whilst using high sensitivity parameter settings, these false detection events are significantly less frequent and more dispersed in space and time than the detections made at the tip of the streak generated by the fast-moving object and as such can be readily filtered by an event-based tracker.

Table 2.2 shows that whilst the sensitivity of the event stream is slightly lower than in Table 2.1, the much higher specificity results in significantly greater informedness than the raw events.

The proposed feature detector can be viewed as a highly refined filter designed to remove noise events passed to it from the upstream surface activation filter. The far sparser output event stream of the streak detection events can then be passed to a more computationally intensive event-based tracker. The event-based tracker, in turn, can be viewed as an even more restrictive filter capable of removing spurious detection events not associated with other nearby detection events of the same orientation and velocity. When viewed in this way, as a series of increasingly refined event-filters, the value of preserving true events generated by true targets outweighs the value of removing noise events at earlier filtering stages as long as the noise events will eventually be removed by a downstream filter. Thus, as long as the final filter can remove all remaining noise events, the only penalty to permissive parameter settings at the upstream stages is in the increased processing time of the latter filters. This motivates a conservative parameter selection regime which at the feature detection stage involves the selection of parameters that generate a significant level of false-positive detection events.

2.2.7 Event-based Tracking

The event-based tracking method used in this work continually generates, updates, and removes hypotheses in an event-based manner. The state of the hypotheses is modeled as a population of leaky integrate and fire neurons whilst the hypotheses trajectories are updated using a sequential least-squares fitting algorithm operating on incoming detection events.

Each active tracked object is modeled as a neuron containing a membrane potential which decays over time and is incremented via detection events $\boldsymbol{f}_j$ assigned to it as detailed later in this section. The membrane potential represents the level of recent observations of the object. If the membrane potential reaches the activation potential M_A, the object is activated.

Alternatively, if the membrane potential reaches zero the object is deleted.

$$\boldsymbol{M}_k^{(o)} = \begin{cases} \boldsymbol{M}_{k-1}^{(o)} - \gamma(t_j - t_{k-1}) + 1, & \text{if } \boldsymbol{f}_j \text{ is assigned to } \boldsymbol{H}_k^{(o)}. \\ \mathbf{0}, & \text{if } \boldsymbol{M}_{k-1}^{(o)} = 0. \\ \boldsymbol{M}_{k-1}^{(o)} - \gamma(t_j - t_{k-1}), & \text{otherwise.} \end{cases} \tag{2.18}$$

where $\boldsymbol{H}_k^{(o)}$ is the kth observation of the oth object, $\boldsymbol{M}_k^{(o)}$ is the membrane potential of $\boldsymbol{H}_k^{(o)}$ at t_k and γ is the decay factor for the membrane potential. If the object activation variable $\boldsymbol{M}_k^{(o)}$ reaches the activation threshold M_A, the object $\boldsymbol{H}_k^{(o)}$ is deemed a true tracked object. A variable $\boldsymbol{A}_k^{(o)}$ tracks the activation level of the object until it reaches M_A and $\boldsymbol{A}_k^{(o)}$ reaches 1. Thereafter $\boldsymbol{A}_k^{(o)}$ remains at 1, permanently indicating the activation of the object regardless of the value of the membrane potential $\boldsymbol{M}_k^{(o)}$.

This behavior is described by (2.19). In addition to indicating the activation of the object, $\boldsymbol{A}_k^{(o)}$ will be used weigh the angular distance relative to the spatial coordinates and as such plays an important role in reducing the weight of the angular distance in the earlier stages of tracking where the object's estimated angle tends to be unreliable.

$$\boldsymbol{A}_k^{(o)} = \begin{cases} \boldsymbol{M}_k^{(o)}/M_A, & \text{if } \boldsymbol{M}_k^{(o)} < M_A \text{ and } \boldsymbol{A}_k^{(o)} < 1. \\ 1, & \text{otherwise.} \end{cases} \tag{2.19}$$

where k denotes the number of previous observations assigned to the oth object and M_A is the object maturation threshold.

Given the jth detection event $\boldsymbol{f}_j = [x_j, y_j, p_j, \theta_j, t_j]^T$, $\boldsymbol{z}_j$ is defined as the vector containing the position and angular information excluding the polarity and timestamp entries:

$$\boldsymbol{z}_j = [x_j, y_j, \theta_j]^T \tag{2.20}$$

The position and velocity of each active object o in space and time, at the kth observation, is defined as:

$$\boldsymbol{H}_k^{(o)} = [\hat{z}_k, \boldsymbol{b}_k, p_k, t_k]^T, o \in \mathbb{N}, k \in \mathbb{N} \tag{2.21}$$

where $\hat{z}_k = [\hat{x}_k, \hat{y}_k, \hat{\theta}_k]^T$ and $\boldsymbol{b}_k = [d\hat{x}/dt_k, d\hat{y}/dt_k, d\hat{\theta}/dt_k]^T$ as estimated via Algorithm 2.3.

The predicted object position at time t_j is determined using:

$$[\hat{x}_k, \hat{y}_k, \hat{\theta}_k]^T = [\hat{x}_{k-1}, \hat{y}_{k-1}, \hat{\theta}_{k-1}]^T + \boldsymbol{b}_{k-1}(t_j - t_{k-1}) \tag{2.22}$$

where $\boldsymbol{b}_{k-1} = [dx/dt_{k-1}, dy/dt_{k-1}, d\theta/dt_{k-1}]^T$ as estimated via Algorithm 2.3.

When estimating the distance of a new detection event to each object $\boldsymbol{H}_k^{(o)}$, the weight of the angular distance in θ relative to the distance in x and y is proportional to each object's previous speed and the activation measure $\boldsymbol{A}_k^{(o)}$ as described in (2.19). Thus, the faster the velocity of an object, the higher the weight of the angular distance is with respect to the positional distance. Objects moving at close to zero velocity are assigned near-zero weight since the detection will be oriented at random, whereas objects moving at high speed have sharp clearly distinguishable angles.

$$\boldsymbol{w}_k^{(o)} = V\left(\sqrt{(dx/dt_k^{(o)})^2 + (dy/dt_k^{(o)})^2}\right)\boldsymbol{A}_k^{(o)} \tag{2.23}$$

where V is a scaling factor which in this work was selected as $V = 0.1$.

The distance between a new detected event $\boldsymbol{f}_j$ and the predicted position of each active object $\boldsymbol{H}_k^{(o)}$ at t_j is defined as:

$$\boldsymbol{d}_k^{(o)} = \sqrt{(x_j - \hat{x}_k^{(o)})^2 + (y_j - \hat{y}_k^{(o)})^2 + \boldsymbol{w}_k^{(o)}(\theta_j \ominus \hat{\theta}_k^{(o)})^2} < d_{max} \tag{2.24}$$

where d_{max} is the threshold acceptable distance to the detected event and the $\ominus$ symbol denotes circular subtraction.

In summary, at each detection event $\boldsymbol{f}_j$, the weighted Euclidean distance between the event and the projected x, y, θ position of every object $\boldsymbol{H}_k^{(o)}$ with an active membrane potential $\boldsymbol{M}_k^{(o)} > 0$ at time t_i, is measured. This distance is then compared to the threshold d_{max}. The detection event is assigned to the closest object with a distance below d_{max}. If no object is within d_{max} of the current detection event, a new object $\boldsymbol{H}_1^{(o+1)}$ is created. This algorithm is described by Algorithm 2.2.

Algorithm 2.2
Detection Assignment

```
Require: $\boldsymbol{f}_j = [x_j, y_j, p_j, \theta_j, t_j]^T, i \in \mathbb{N}$
Ensure: $\boldsymbol{H}_k^{(o)}$, $\boldsymbol{M}_k^{(o)}$ and $\boldsymbol{A}_k^{(o)}$ as defined by (2.21), (2.18) and (2.19)
  for every feature event $\boldsymbol{f}_j$ do
      for every object $\boldsymbol{H}_k^{(o)}$ where $\boldsymbol{M}_k^{(o)} > 0$ do
          compute current postition $[x_k^{(o)}, y_k^{(o)}, \theta_k^{(o)}]$ using (2.22)
      end for
      if $\exists$ any object satisfying (2.24) then
          $n \Leftarrow \text{argmin}_m \boldsymbol{d}_k^{(m)}$ s.t. $\boldsymbol{H}_k^{(m)}$ verifies (2.24)
          $\boldsymbol{M}_k^{(o)} \Leftarrow \boldsymbol{M}_{k-1}^{(o)} + 1$
          Update $\boldsymbol{H}_k^{(o)}$ with $[x_{k+1}, y_{k+1}, \theta_{k+1}]^T$ using Algorithm 2.3.
      end if
      for every object $\boldsymbol{H}_k^{(o)}$ with $\boldsymbol{M}_k^{(o)} > 0$ do
          Update $\boldsymbol{M}_k^{(o)}$ using (2.18)
      end for
  end for
```

In order to estimate the position of each hypothesis $\boldsymbol{H}_k^{(o)}$ at the time of each detection event $\boldsymbol{f}_j$, a sequential least-squares method is implemented involving the sequential calculation of the ratio of the covariance of the position and timing of the object over the variance of the timing. In this event-based approach, the covariance and variance measures are calculated online in a sequential manner. Each measure is calculated using a fixed rolling window of length K. This online approach allows the rapid calculation of the velocity of each object in x, y θ space without the need to perform least-squares on previous observations. The event-based tracker update method is described using Algorithm 2.3.

As shown in Figure 2.8, the event-based line fitting tracker algorithm removes virtually all false detection events remaining after the feature detection stage while correctly clustering events from each object. The output events $\boldsymbol{g}_l$ from the tracker can be represented in the form of an event stream defined as:

$$\boldsymbol{g}_l = [x_l, y_l, p_l, \theta_l, o_l, t_l]^T \tag{2.25}$$

Algorithm 2.3
Object Velocity Estimation Algorithm

```
Require: $\boldsymbol{z}_j$ from $\boldsymbol{f}_j$ , $j \in \mathbb{N}$ using (2.20) and $\boldsymbol{z}_k$ from $\boldsymbol{H}_k^{(o)}$ , $k \in \mathbb{N}, o \in \mathbb{N}$
  verifying (2.24)
Ensure: $\boldsymbol{H}_k^{(o)}$ as defined by (2.21)
  for every feature event $\boldsymbol{f}_j$ do
      $t_k \Leftarrow t_j$, $\boldsymbol{z}_k \Leftarrow \boldsymbol{z}_j$
      if $k = 1$ then
          $\bar{t}_k \Leftarrow t_k$, $\bar{\boldsymbol{z}}_k \Leftarrow \boldsymbol{z}_k$
          $\sigma_k^2 \Leftarrow 0$, $\Sigma_k^2 \Leftarrow 0$
          $\boldsymbol{c}_k^2 \Leftarrow 0$, $\boldsymbol{C}_k^2 \Leftarrow 0$
          $\boldsymbol{b}_k \Leftarrow 0$, $\hat{t}_k \Leftarrow t_k$, $\hat{\boldsymbol{z}}_k \Leftarrow \boldsymbol{z}_k$
      else if $k > K$ then
          $\bar{t}_k \Leftarrow \bar{t}_{k-1} + (t_k - t_{k-K})/K$
          $\bar{\boldsymbol{z}}_k \Leftarrow \bar{z}_{k-1} + (\boldsymbol{z}_k - \boldsymbol{z}_{k-K})/K$
          $\sigma_k^2 \Leftarrow \sigma_{k-1}^2 + (t_k - \bar{t}_{k-1})(t_k - \bar{t}_k) - (t_{k-K} - \bar{t}_{k-1})(t_{k-K} - \bar{t}_k)$
          $\boldsymbol{c}_k^2 \Leftarrow \boldsymbol{c}_{k-1}^2 + (\boldsymbol{z}_k - \bar{\boldsymbol{z}}_{k-1})(t_k - \bar{t}_k) - (\boldsymbol{z}_{k-K} - \bar{\boldsymbol{z}}_{k-1})(t_{k-K} - \bar{t}_k)$
          $\Sigma_k^2 \Leftarrow \sigma_k^2/(K-1)$
          $\boldsymbol{C}_k^2 \Leftarrow \boldsymbol{c}_k^2/K$
          $\boldsymbol{b}_k \Leftarrow \boldsymbol{C}_k^2/\Sigma_k^2$
          $\hat{t}_k \Leftarrow \bar{t}_k + (t_k - \bar{t}_{k+K})/2$
          $\hat{\boldsymbol{z}}_k \Leftarrow \bar{\boldsymbol{z}}_k + \boldsymbol{b}_k(t_k - \bar{t}_{k+K})/2$
      else
          $\bar{t}_k \Leftarrow \bar{t}_{k-1} + (t_k - \bar{t}_{k-1})/k$
          $\bar{\boldsymbol{z}}_k \Leftarrow \bar{\boldsymbol{z}}_{k-1} + (\boldsymbol{z}_k - \bar{\boldsymbol{z}}_{k-1})/k$
          $\sigma_k^2 \Leftarrow \sigma_{k-1}^2 + (t_k - \bar{t}_{k-1})(t_k - \bar{t}_k)$
          $\boldsymbol{c}_k^2 \Leftarrow \boldsymbol{c}_{k-1}^2 + (\boldsymbol{z}_k - \bar{\boldsymbol{z}}_k)(t_k - \bar{t}_k)$
          $\Sigma_k \Leftarrow \sigma_k^2/(k-1)$
          $\boldsymbol{C}_k^2 \Leftarrow \boldsymbol{c}_k^2/k$
          $\boldsymbol{b}_k \Leftarrow \boldsymbol{C}_k^2/\Sigma_k^2$
          $\hat{t}_k \Leftarrow \bar{t}_k + k(t_k - \bar{t}_1)/2(k-1)$
          $\hat{\boldsymbol{z}}_k \Leftarrow \bar{\boldsymbol{z}}_k + \boldsymbol{b}_k k(t_k - \bar{t}_1)/2(k-1)$
      end if
      Update $\boldsymbol{H}_k^{(o)}$ with $t_k$, $\boldsymbol{z}_k$, $\hat{t}_k$ and $\hat{\boldsymbol{z}}_k$
  end for
```

where o_l is the object index of the lth event generated by the tracker. Figure 2.8 compares the output event stream of the tracking algorithm to the labeled data. Figure 2.8(f) shows an example of a labeled object missed by the end-to-end system. In the example SL-8R/B

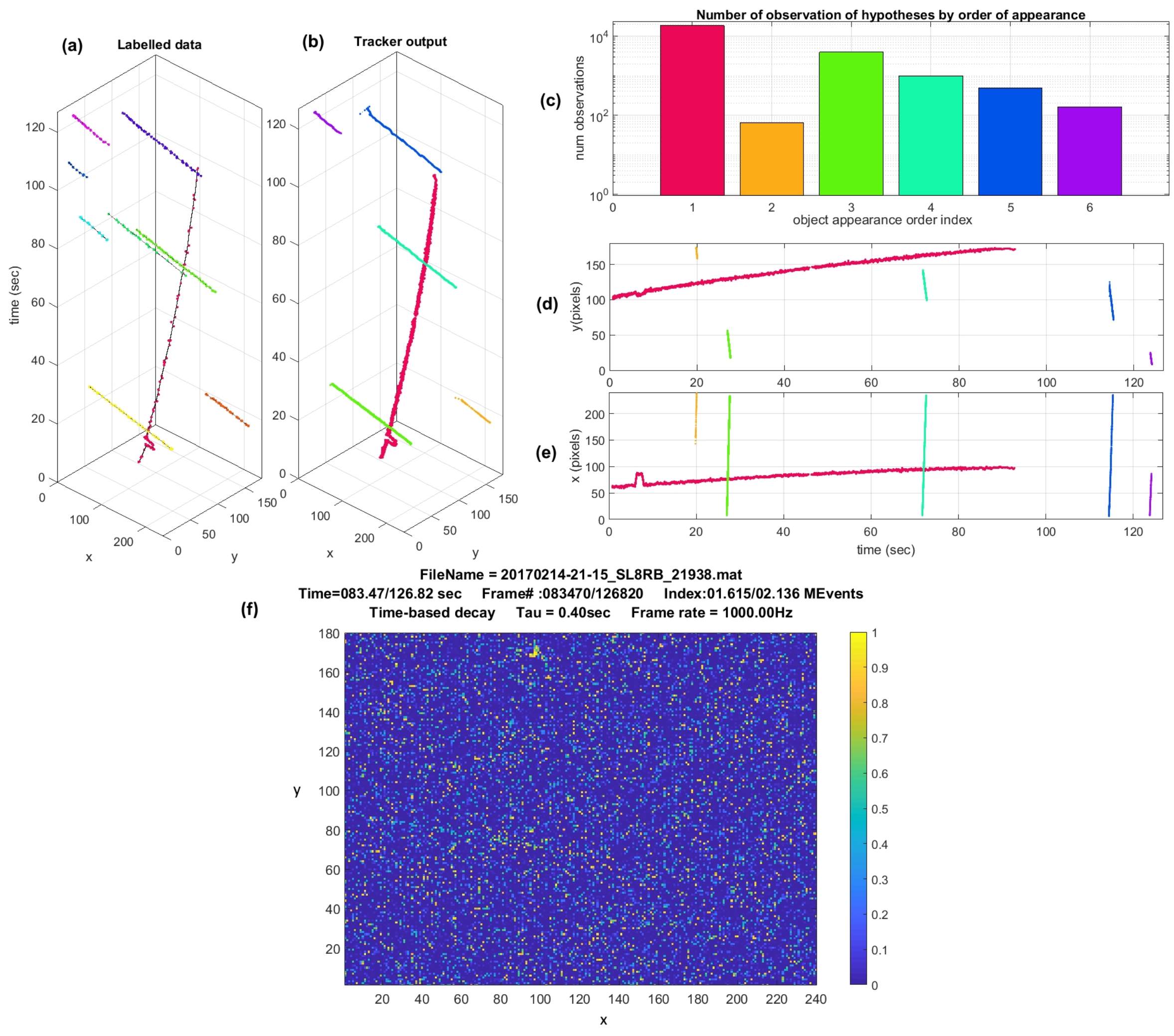


FIGURE 2.8: **Output of tracking algorithm.** Panels (a) and (b) show the dimetric projection of the labeled data and the output of the event-based tracker respectively for the SL-8 R/B recording. (c) Shows the number of tracking events per object. Panels (d) and (e) show the tracker event position in x and y respectively over time. (f) Example of an expert labeled object not detected by the algorithm showing the difficulty level at which the algorithm fails. Missed object at (158 , 56). SL-8 R/B is located at (97 , 170).

recording, three such faint high-speed objects are missed, demonstrating the superior performance of human experts over the proposed algorithm. Table 2.3 details the statistics generated for this particular recording demonstrating improved sensitivity, specificity and informedness with respect to the raw and detection event streams detailed in Tables 2.1 and 2.2 respectively.

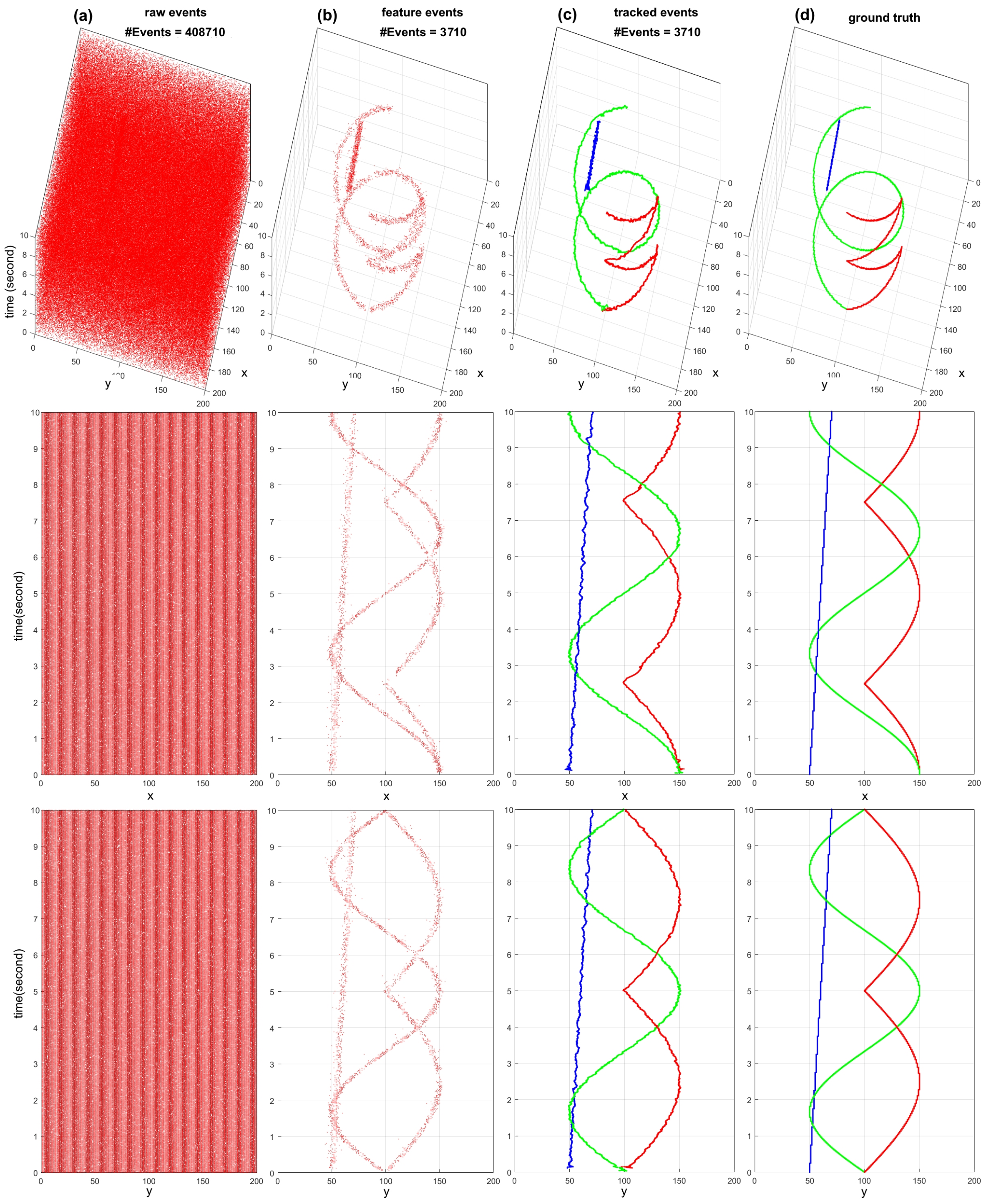


Figure 2.9: **An instance of the artificial dataset event stream and the output of the feature detection and tracking algorithms.**(a) The raw event stream in trimetric projection as well as along the x and y dimensions. (b) Shows the same projections of the detected feature events generated by Algorithm 2.1. (c) Tracked events. (d) Shows the analytically defined ground truth labels.

TABLE 2.3: **Event density activated volume statistics for measuring the performance of the tracking event stream g_l against labels.** The statistics calculated generated using the SL8R/B recording whose data stream is illustrated in Figure 2.8

.

Polarity	Sensitivity	Specificity	Informedness	# Events (ke)
ON Events	0.90	0.99	0.89	20.21
OFF Events	0.87	0.97	0.84	15.13

Figure 2.9 shows the performance of the detection and tracking algorithm on an artificial event stream with low SNR. For this recording, the three target objects are correctly detected. By gradually varying the SNR, the performance profile of the proposed algorithms can be tested against the analytical ground truth across a wide range of noise environments.

2.2.8 Alternative Algorithms

To further evaluate and benchmark the performance of the feature detection algorithm, three additional event-based methods were implemented and tested on the space imaging dataset. In all approaches described in this section, the events are first processed through the same time surface generation and surface activation filter described in section 2.2.6. This pre-processing and noise filtering removes slightly less than half the events for the entire dataset. Following each of the alternative feature detection algorithms, the same tracking algorithm described in 2.2.7 was used on the detection event stream providing an unbiased comparison between the methods.

2.2.8.1 Global Maximum Detector

The first, alternative method examined is a simple event-based Global Maximum Detector (GMD). Given the significant sparseness of space imaging data, the narrow field of view and the stereotypical shapes of space objects, simply looking for the most activated region of the time surface is an ideal baseline method for investigation. To perform the global maximum detection in an event-based manner, at every event that passes the surface activation test, the sum of the ROI activation is compared to a previous global maximum of G_{max}.

Algorithm 2.4
Global Maximum Detection

```
Require: $\boldsymbol{e}_i = [x_i, y_i, p_i, t_i]^T, i \in \mathbb{N}$
Ensure: $\boldsymbol{f}_j = [x_j, y_j, p_j, t_j]^T, j \in \mathbb{N}$
  $G_{max} \Leftarrow -\infty$
  for every event $\mathbf{e_i}$ do
      $\boldsymbol{T}_i(x_i, y_i) \Leftarrow t_i$
      if $\boldsymbol{e}_i$ passes the surface activation test in (2.13) then
          $\boldsymbol{\Sigma}_{ROI}(x_i, y_i) \Leftarrow \sum_{x=x_i-R}^{x_i+R} \sum_{y=y_i-R}^{y_i+R} \boldsymbol{S}_i(x, y)$ with $\boldsymbol{S}_i(x, y)$ calculated via (2.10)
          if $\boldsymbol{\Sigma}_{ROI}(x_i, y_i) > G_{max}\boldsymbol{S}_i(x_{max}, y_{max})$ then
              $G_{max} \Leftarrow \boldsymbol{\Sigma}_{ROI}(x_i, y_i)$
              $x_{max} \Leftarrow x_i$
              $y_{max} \Leftarrow y_i$
              $\boldsymbol{f}_j \Leftarrow [x_i, y_i, p_i, t_i]^T$
          end if
      end if
  end for
```

With each new event e_i, the total ROI activation is measured. This measure is then compared to the current value of G_{max} decayed exponentially with a time constant of τ. Here τ is the same decay factor used to generate the time surface $\boldsymbol{S}_i(x, y)$ in 2.10. If the activation sum of the current ROI exceeds the decayed value of the previous G_{max}, then it replaces G_{max}. This continued exponential decay ensures the global maximum is continually refreshed without searching the entire time surface. The GMD algorithm is described in Algorithm 2.4.

In the space imaging dataset, where many recordings are taken during satellite tracking or sidereal tracking, significant portions of each recording contain a single object (the one being tracked) moving very slowly across the field of view. This tracking event stream is often punctuated by high-velocity objects, passing rapidly through the field of view[1].

While the GMD is far from robust, under this narrow set of conditions which makes up a significant minority of the real-world space imaging data, this simple method performs quite well. This is illustrated in Figure 2.10 showing the different tracking methods on the SL-8 R/B

[1]When tracking satellites, these high-speed objects are often background stars and during sidereal tracking, the high-speed objects are typically Low Earth Orbit (LEO) objects.

recording where the extremely simple and fast GMD method performs very well under the narrow but common conditions when only the SL-8 R/B is in the field of view. However, in the presence of multiple targets, the detector focuses only on the brightest, typically fast-moving, object. Furthermore, as discussed in Section 2.2.6, due to mismatch in the pixel circuitry, the time response of nearby pixels to near-identical changes in illumination can vary. In the space imaging context, this can result in high-speed objects generating late-triggered events on the object trail slightly behind the tip of the streak since some pixels respond later than others. These late events resulting from variance in pixel response times often causes surface activation patterns that are stronger along the trail of the streak than its tip. This causes the GMD to detect objects with a delay, on the trail of the streak and often in a disorganized non-sequential manner instead of sequentially at its tip. This effect is shown in Figure 2.10(a) and (c).

Finally, when no object is in the field of view, the GMD simply detects random local clusters of noise events. While this can be avoided by setting higher surface activation threshold parameters Φ and L, this, in turn, results in reduced sensitivity in the context of faint or slow-moving space objects.

2.2.8.2 Hough Transform Detector

The second most common class of objects observed in space imaging event streams, after single slow-moving targets, are the high-velocity streaks. Since these streaks generate relatively straight lines across the time surface, a high-speed line detecting algorithm such as a Hough transform serves as an ideal candidate for comparison to the proposed feature-based detection method. Previous event-based implementations of event-based Hough transform for the detection of straight lines include [74], where a Hough transform was used to detect and control a balanced pencil. In [75] a spiking neural network was used to generate local inhibition in a neural implementation of the Hough space and in [76], the event-based Hough transform was combined with an efficient end-point generation algorithm to detection line segments. Here, by projecting the event activation of the detected line onto the x or y edge of the sensor (depending on the orientation of the line), two endpoints can be found on

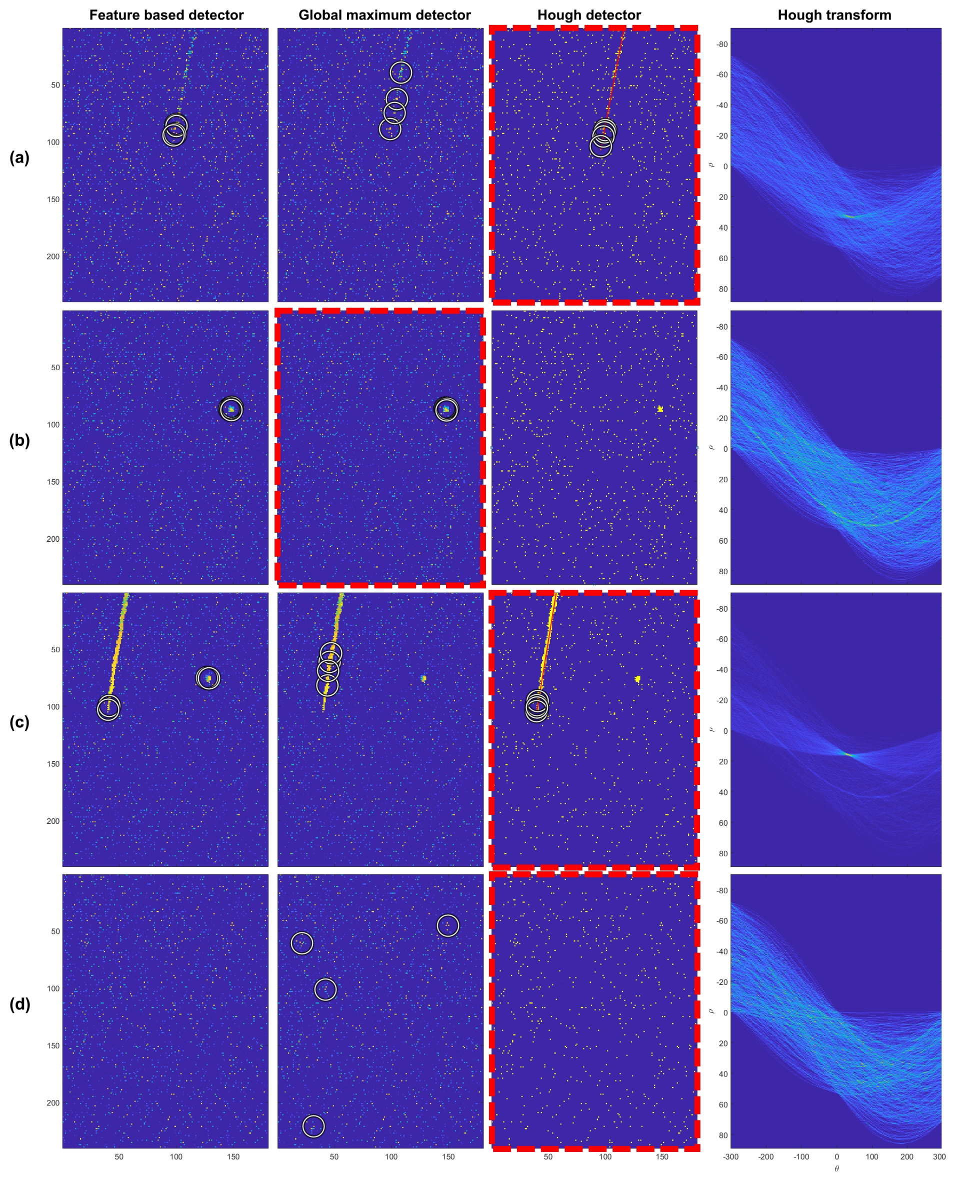


FIGURE 2.10: **Behavior of tested algorithms in common space imaging conditions.**(a) When a high-speed object moves across the field of view, the Hough and feature-based detectors correctly detect the tip of the streak while the GMD incorrectly detects events along the trail. The Hough transform of the image is shown on the right-hand side with a clearly visible peak. (b) With a single slow-moving object, the GMD and the feature-based detector operate correctly and the Hough fails to detect any object. (c) In the presence of both slow and fast-moving objects, the Hough detector only detects the tip of bright fast-moving streak which generates a large peak in the Hough space. The GMD again detects late events on the trail and the feature-based detector correctly detects both objects regardless of velocity. (d) With no objects in the field of view, the GMD incorrectly generates detections around clusters of noise events while the Hough and feature-based detectors correctly generate no detections. The red dashed box indicates which of the two alternative algorithms was automatically selected in the post hoc combined detector.

the projection, based on the location at which the line activation drops below a pre-defined threshold. For the implementation of the event-based Hough transform for space imaging data, the method as proposed in [76] is used with the minor modification that after the detection of a line segment, the endpoint with the lower number of recent events[2] is considered to be the trail of the streak and discarded. This is because, in the space imaging context, we are only interested in the tip of the streak which will have more recent events than the tail.

2.2.8.3 Combining the GMD and Hough Detectors

The event-based Hough detection algorithm is clearly capable of rapidly detecting streaks on an event-based time surface and as shown in Figure 2.10 the event-based GMD method provides a complementary capability for finding slow-moving objects. Given their extreme simplicity, efficiency and suitability for sparse space imaging event streams, the combination of these two complementary algorithms would provide robust performance benchmarks in terms of processing speed and accuracy against which the proposed algorithm can be compared. However, since the stimulus type that will be observed for any segment of a recording is unknown, it is not possible to determine a priori which algorithm must be used. Even after the data is observed and processing by the algorithms is complete, there is still no simple way of determining which algorithm performed better without access to the ground truth labeling. For the purposes of testing the feature-based detection algorithm, these discrepancies are overlooked and with the aim of providing the best alternative algorithm, a combined metric is generated where for each 1ms of the dataset, the alternative algorithm with the highest detection event informedness measure was selected in a post hoc manner. In this way, the output of the feature detector algorithm can be compared with the best-consolidated results from the two alternative algorithms. After this post hoc combination of the best detection event streams from the two algorithms, the same tracker as that used in Section 2.2.7 was run on the output of the post hoc combined GMD-Hough detector.

[2]To determine the line endpoint with the lower number of recent events, the value of the 75th percentile of the ROI at each line endpoint is compared and the line end with the lower value is discarded.

2.3 Results

In this section, the expert labeling procedure and the proposed algorithm are first evaluated on the artificial dataset described in Section 2.2.4. The performance of the algorithm on the real-world event-based dataset is then investigated in detail. Finally, the performance of the proposed algorithm is compared with the alternative algorithms described in Section 2.2.8.

2.3.1 Algorithm and Expert Performance on Artificial Dataset

Figure 2.11 details the performance of our expert labeling procedure on the artificial dataset across a range of SNR configurations. Each data point represents mean and standard deviations over 20 trials with each trial being a random instantiation of the event stream defined in Section 2.2.4.

In (a), the top panel shows mean informedness as a function of the signal per-pixel event rate λ_1 for three, per-pixel background noise event rates λ_0 on the raw event stream. The bottom panel in (a) shows the standard deviation. Panel (b) shows the same mean and standard deviation results on the feature detection stream and panel (c) shows these for the tracking event stream. In panel (c), the results from the algorithm are augmented with performance measures of expert labelers with $\lambda_0 = 10^{-2}$ events/second. Note the logarithmic scale on the bottom standard deviation panel where results with zero variance are not shown. Panels (d), (e) and (f) show the mean and standard deviation specificity for the raw, detection and tracking event streams respectively with (g), (h) and (i) showing the same for sensitivity.

As panels (a), (b) and (c) show, the informedness improves in all cases with increased signal event rate λ_1. The effect of the noise event rate λ_0 on informedness is somewhat mixed in raw and detection event streams due to a stochastic resonance effect [10] where random noise events assist in activating a proportion of true volume slices above the recording mean density increasing sensitivity while decreasing specificity to a smaller extent resulting in higher informedness. In all cases however, the informedness results in (a) and (b) are low when compared to the tracking event stream of (c). Here the behavior of the full system becomes clear erasing any stochastic resonance effects. The informedness of the tracked events generated by the algorithm and shown in (c) increases monotonically with increased per-pixel signal event rate λ_1 and is invariant to the per-pixel noise event rate up to approximately $\lambda_0 = 1$ event per second where it begins to fall. For comparison, the mean event rate of the real space imaging dataset is approximately 0.24 events per second. This value can also be assumed to be the noise event rate given the sparseness of signal events in space imaging data. The expert results also show the performance of experts on the artificial dataset against the analytically defined ground truth labels. The expert results demonstrate accuracy that is approximately three times higher (in terms of signal strength λ_1) than the proposed algorithm with perfect specificity at all levels and high sensitivity even at very weak signal strengths. Altogether these results validate the labeling procedure used for the real space imaging dataset.

2.3.2 Performance on Real World Space Imaging Dataset

The detailed results for all recordings in the dataset are summarized in Figure 2.12. The first three rows of results (a), (b) and (c) plot informedness, specificity and sensitivity respectively. The results demonstrate how each stage of processing shifts the distribution toward 1 resulting in a more informative event stream. The bottom row (d) shows how, at each stage of processing, the event density of the recordings is reduced into an ever more efficient representation of the data. Together these results demonstrate that over the wide range of heterogeneous input event streams, the proposed algorithm generates a sparse yet informative output event stream. (b) Shows the per recording specificity distribution is shifted from a mean of 0.63 for the raw events to 0.98 and 0.99 for the detection and tracking events with most results at 1. Similarly (c) shows how the per recording sensitivity distributions for the raw, detection and tracking event streams. Here the sensitivity distribution is actually reduced in the detection stream in comparison to the raw events. This is primarily due to the relative sparseness of the detection stream. When the sparser detection event stream is interpolated via the tracker, the sensitivity rises above the raw events. Together the higher sensitivity and specificity result in a significantly higher informedness distribution as shown in (a). These results demonstrate the effectiveness of the end-to-end system in transforming noisy raw input events from space

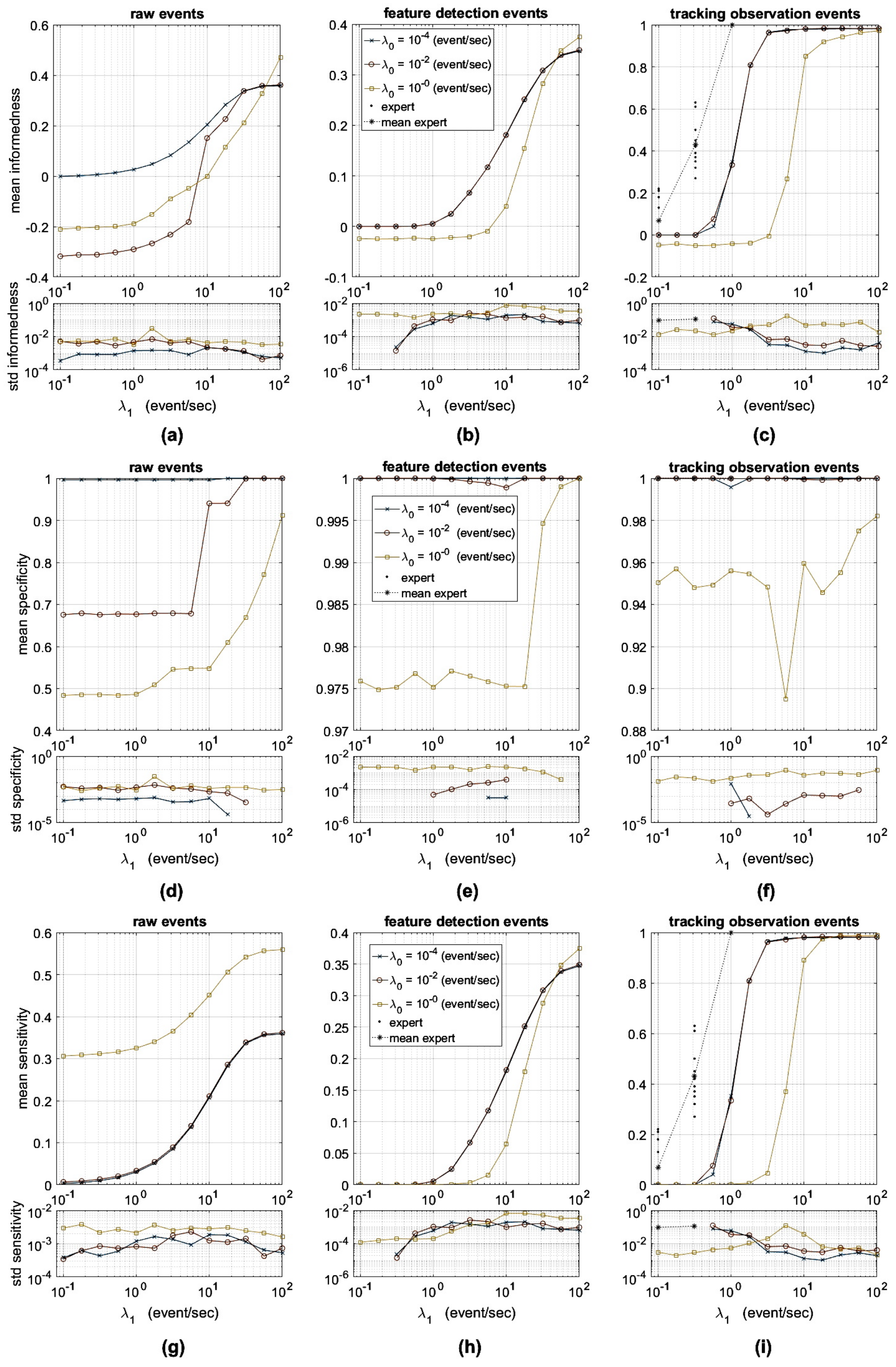


FIGURE 2.11: **Detailed results of experts and proposed algorithm on the artificial dataset.** See text for details.

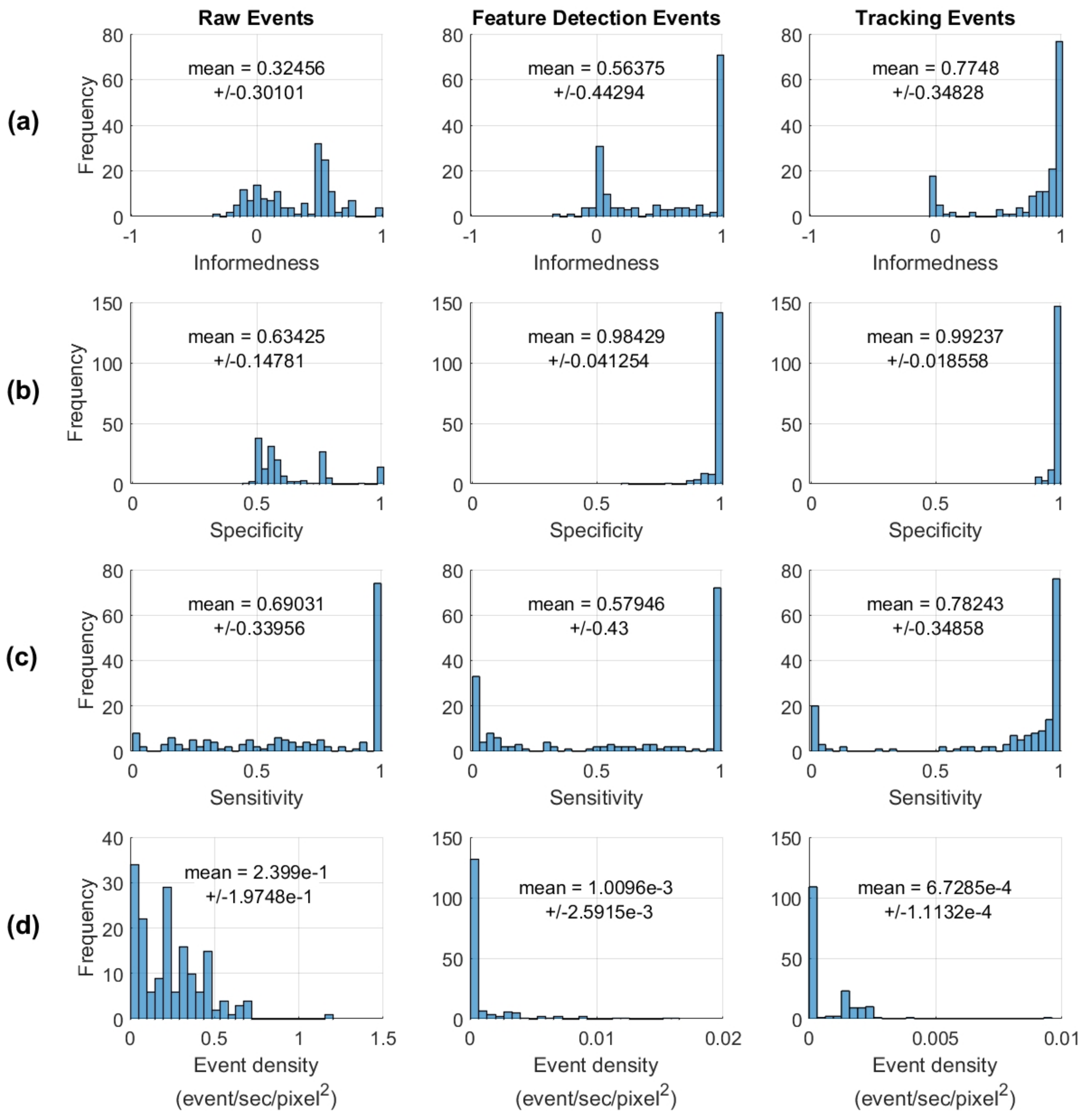


FIGURE 2.12: **Per Recording Histogram Results on the Space Imaging Dataset.** From left to right, the panels show results from the raw events, the detection events and the tracking events. From top to bottom the panels show (a) informedness, (b) specificity, (c) sensitivity and (d) the event density of each of the event streams.

imaging data, into sparse highly informative noise-free event streams using a series of simple hardware implementable filters.

An important parameter in evaluating the space imaging dataset is the selection of the acceptance distance from an object r marking the boundary of the True and False volumes.

The radius is dependant on the size of the objects in the dataset and based on inspection of the data, a value of $r = 10$ pixels was selected for this parameter. To validate the robustness of the results and to investigate the effect of r selection, all tests were repeated across all possible values of r with the results shown in Figure 2.13. These results not only validate the parameter selection but also provide insights about the spatial structure of the dataset. Figure 2.13(a) shows an expected rise and fall of the informedness statistic as a function of r in the raw event stream. At the extreme low radius range, the likelihood of events from a labeled object falling exactly at the labeled point is low, especially given that the objects themselves often cover an area which is many pixels across. As the acceptance range is increased, the majority of events from the objects fall within the acceptance threshold activating the space-time volume as per Section 2.2.3. As the acceptance radius is further widened into regions around the object where no object exists, the event density of the region falls reducing the probability that events from the object activate the volume slice. As a result, informedness falls back toward zero. Note that due to the greater sparseness, the informedness in the detection and tracking event streams do not show the same drop after r=10 seen in the raw events since there are almost no noise events in these event streams. As shown in Figure 2.13(d), due to the dominance and uniform presence of noise events in the dataset, the specificity of raw event stream changes very little[3] as a function of acceptance distance. In contrast to the raw event stream, the detection and tracking results show a clear increase in specificity between r=[1:10] with little increase thereafter. The bottom row panels detail sensitivity results showing a rise and fall in sensitivity for the raw events with a peak around r=10. Similarly, the sensitivity results in panels (h) and (i) for the feature detection and tracking events respectively, show rise at r=[1:10] with little change thereafter. The results from all panels demonstrate the validity of the selection by inspection of r=10. This selection produces near-optimal results on the raw events stream and further increases of r providing little change on the more selective detection event streams and even less change on the tracking event streams. Together, the

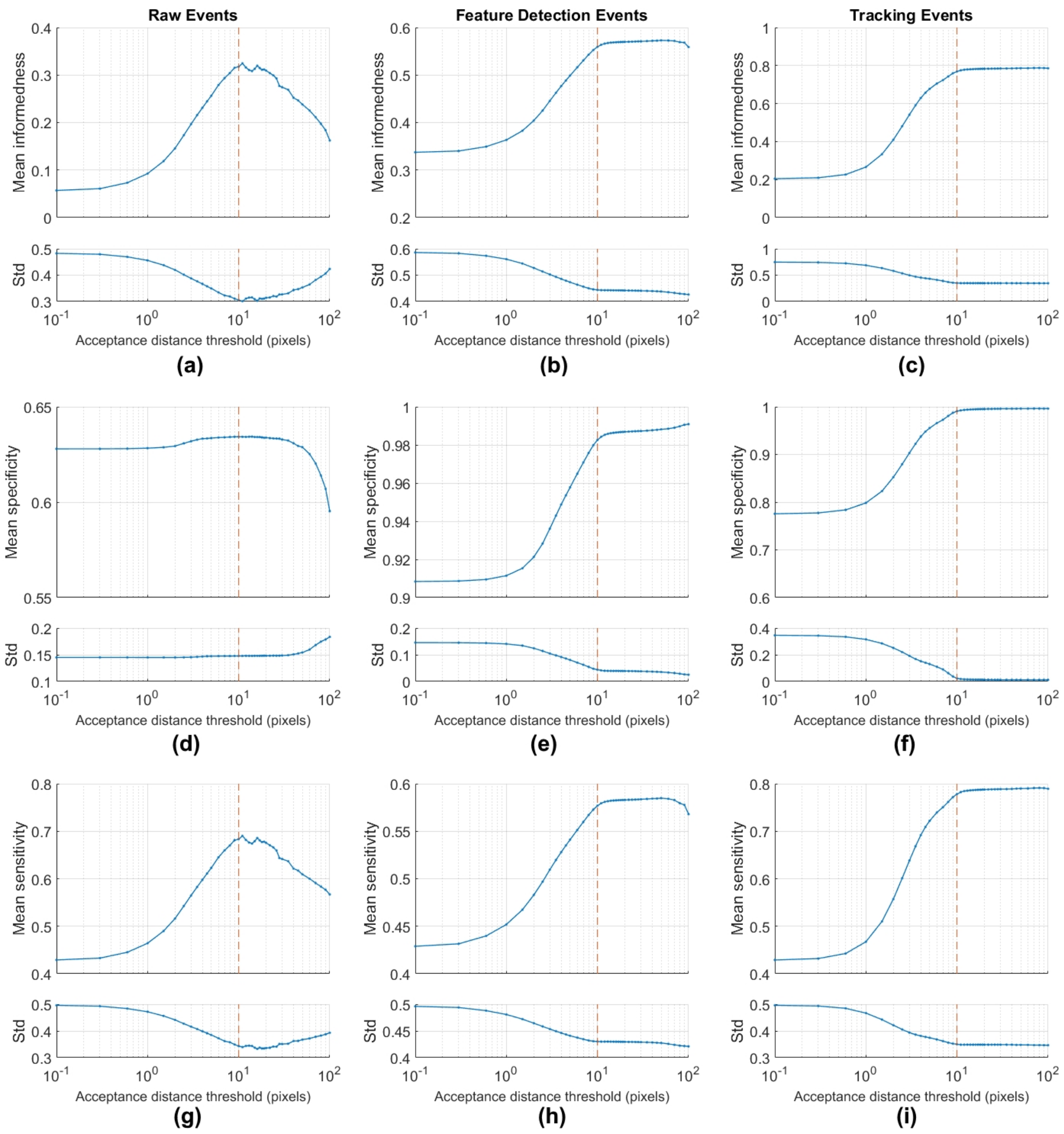


FIGURE 2.13: **Results on space imaging dataset as a function of acceptance distance** r**.** (a) The per recording mean and standard deviation in informedness on the raw event stream as defined in Section 2.2.3 is found to be maximal at acceptance radius r=10 pixels. The red dashed line marks the acceptance radius r=10 pixels, chosen from inspection of the data. (b) Shows informedness of the detection event streams, (c) informedness of the tracking event streams (d), (e) and (f) show the per recording mean and standard deviation specificity of the raw detection and tracking event streams respectively as a function of acceptance radius. (g), (h) and (f) show the sensitivity statistic on the raw events, detection events and tracking events respectively all as a function of acceptance radius r.

precise pattern of results shown in Figure 2.13 serves to validate the volume-based statistical measures used to evaluate event streams in this work.

Given the wide range of velocities and event rates observed in the space imaging dataset, the effect of the value of the surface decay parameter τ on the performance of the algorithm requires investigation. During the labeling procedure described in Section 2.2.2, a value of $\tau = 0.5$ seconds was chosen for viewing the dataset. This value was chosen by inspection of the data. In Figure 2.14 the algorithm results on the space imaging dataset are shown across a range of τ values. At shorter time constants, the memory of recent events fades so quickly on the time surface that faint objects, which generate fewer events over time, generate too short a trace to be distinguishable from noise clusters and thus are rejected, resulting in lower sensitivity. This faster decay also rejects true noise events which also results in slightly higher specificity, but this is outweighed by the fall in sensitivity and thus results in lower informedness overall. At the other extreme, with very large time constants, the memory from the background noise events remain on the surface for so long that random clusters of noise events begin to dominate the signal from the true objects. This significantly reduces specificity but also sensitivity since the adaptive angular activation threshold Ψ_i described in (2.15) adapts the sensitivity of the system depending on the amount of activation. The results show that the informedness metric changes by only 8% across the wide range of time constant values tested demonstrating the desired robustness of the overall algorithm to poor parameter selection. Finally, note that the peak informedness of the system occurs at $\tau = 0.4$ seconds which is very close to the value of $\tau = 0.5$ seconds chosen by inspection during the labeling process.

2.3.3 Processing Time Results

In this section the processing time and filtering operation of the algorithm is detailed. The processing times were tested in the MATLAB 2017a environment on a laptop with a 64bit

[3]The slight drop in specificity at the extreme acceptance radii results from the increasing probability of extremely active 'hot pixels' falling onto the acceptance region and activating the positive volume. Performing the same test on the raw events processed by a hot pixel removing algorithm reduces this drop to varying degrees depending on the permissiveness or severity of the hot pixel removing algorithm used.

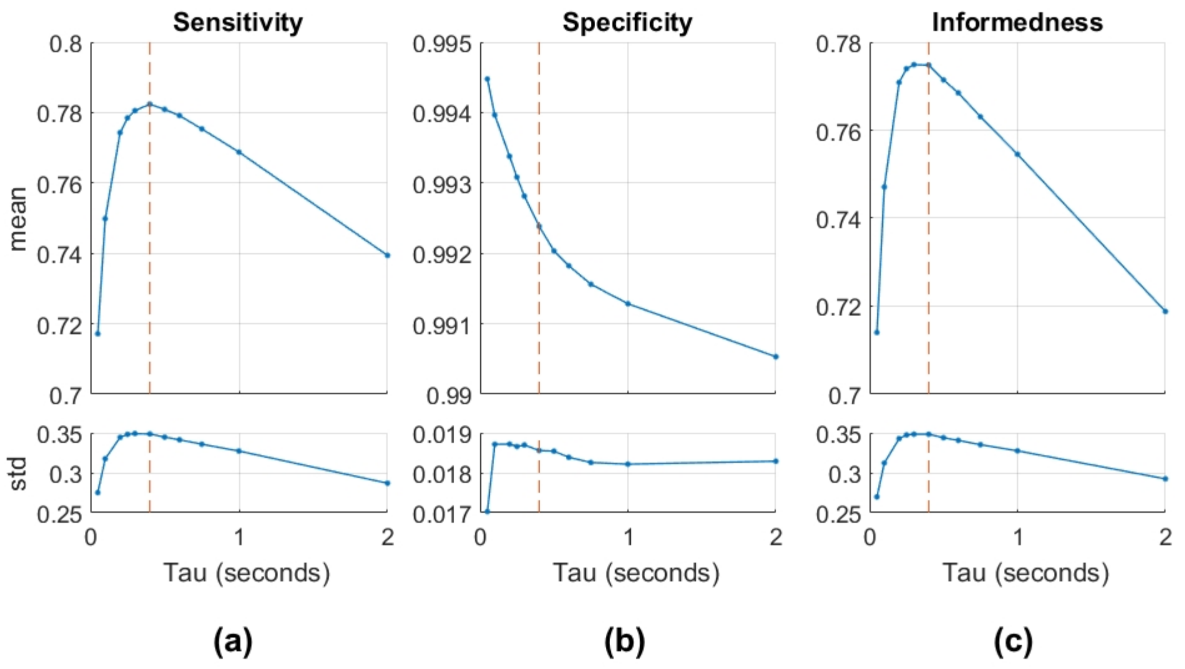


FIGURE 2.14: **Final results on the space imaging datgaset as a function of exponential decay factor τ.** The dashed red line indicates the $\tau = 0.5$ value chosen during labeling.

4.00 GHz i7-6700 CPU processor and 64GB of RAM. Figure 2.15 shows the cascaded event filter design of the proposed system, where at each of the increasingly refined processing stages, the increased computation time is accompanied by a corresponding reduction in events.

As the distributions shown in panels (a) to (d) of Figure 2.15 demonstrate, the event rates at each stage of processing of the space imaging dataset becomes reduced requiring an ever-smaller number of events to be processed by the subsequent stage. Furthermore, as panel (e) shows for the example SL-8 R/B recording, due to the sparseness of activation in space imaging event streams, the processing speed of the algorithms is remarkably stable over time within a recording. In other words, given the small size of the area occupied by space objects relative to the entire field of view, the presence or lack of even bright target objects in the field of view makes little difference in the global event rate of the raw events. This is in contrast to terrestrial applications where, due to the complexity and the relative size of the objects in the visual environment, the event rate can vary by many orders of magnitude depending on the relative velocity of the visual scene. The relative stability of event rates within EBSI recordings can be exploited at every stage of processing. This property of the data provides yet another important distinction between EBSI processing algorithms and more general event-based systems. Panel (f) shows the timing response of the entire system for each processing stage. Here we can observe that as envisioned, at each stage of processing, the increase in complexity of the following stage is accompanied by an approximately commensurate reduction in input event rate such that the entire end-to-end system can process all events at slightly faster than an eighth of the speed of the first simple surface activation test. This is despite the fact that the last processing stage, the tracker, processes events at a rate that is more than 230 times slower than the first stage. Finally, note the position of the angular activation test above and to the right of the diagonal formed by the other tests. This position identifies this stage as the bottleneck in the system as discussed in Section 2.2.6.

2.3.4 Comparison with Alternative Algorithms

To provide a benchmark for comparison Table 2.4 details the results of the feature-based detection and tracking algorithm against alternative event-based high-speed algorithms that could be used on the space imaging dataset against expert labeling. All algorithms operated on the same event stream which was pre-processed with the same initial local surface activation filter and were paired with an identical event-based tracker for the tracking results. The first row in the table sets the raw events as a baseline showing low informedness primarily due to the low mean specificity of the event streams. Given the high noise rate, the Hough line detection algorithm is the worst performing algorithm in this context with informedness lower than the raw events. This, however, is primarily due to the poor sensitivity of the Hough detector to a great number of the observed objects in the dataset that are extremely slow-moving. These slow objects generate faint point source-like activation patterns which the Hough detector can not detect. When augmented with the event-based tracker, the sensitivity of the system is slightly reduced but specificity rises to close to 1 resulting in a near doubling of the informedness. In contrast to the Hough detector, the GMD detector performs best on the more common slower-moving targets thus resulting in significantly higher sensitivity and thus informedness. The GMD detector, however, performs poorly in noise filtering. This

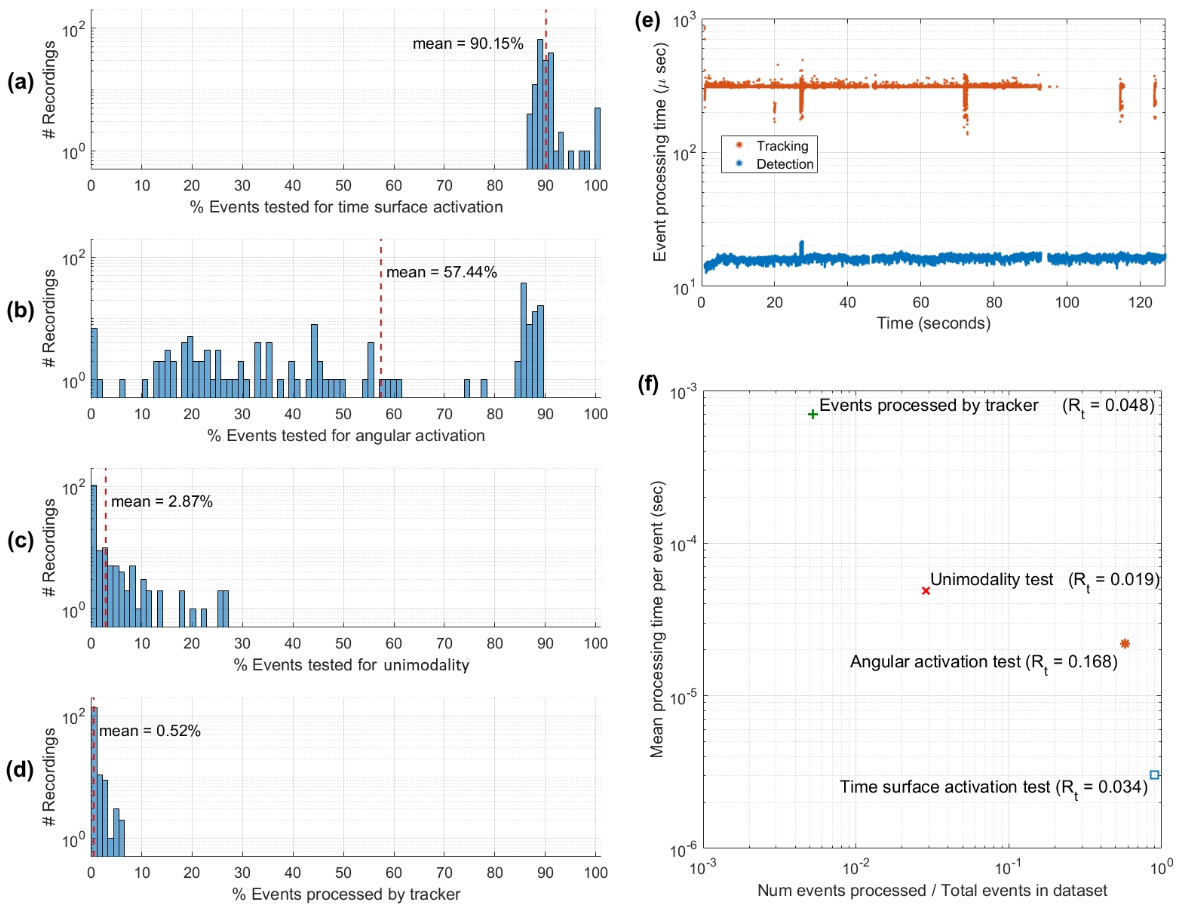


FIGURE 2.15: **Reduction in event numbers and associated processing time at each stage of the algorithm.** Panel (a) shows the distribution of the number of events processed in each recording by the initial time surface activation test. (b) Shows the angular activation test, (c) the angular unimodality test and (d) the tracker. (e) Shows processing time per event of the detection and tracking algorithm for the SL-8 R/B recording. (f) The number of events at each stage of processing against the mean processing time per event at that stage. The processing time ratio R_t is the total processing time of each stage divided by the duration of the recording being processed.

is especially the case for neighboring clusters of noise events from overactive ‘hot pixels’ on the sensor which are a challenging feature of the dataset and which the GMD fails to remove. Furthermore, these localized stationary clusters of noise activation are also difficult for the tracker to remove. For this reason, the specificity of the GMD system is about the same with or without the tracker. However, the tracker does slightly improve sensitivity mainly through interpolating between periods of higher activity of slow-moving objects. Next, when the performance of the GMD and the Hough detector are combined in a post hoc

TABLE 2.4: **Summary of results of tested algorithms on the space imaging dataset.**

Algorithm	Informedness		Sensitivity		Specificity		Processing
	mean	std	mean	std	mean	std	time ratio
Raw Events	**0.324**	0.301	**0.690**	0.340	**0.634**	0.148	1
Hough Detector	**0.244**	0.343	**0.552**	0.408	**0.692**	0.224	**0.632**
Hough Detector + Tracker	**0.417**	0.478	**0.442**	0.488	**0.975**	0.073	**0.781**
GMD Detector	**0.609**	0.323	**0.756**	0.284	**0.853**	0.117	**0.113**
GMD Detector + Tracker	**0.664**	0.374	**0.813**	0.314	**0.851**	0.223	**0.671**
max(GMD,Hough) Detector	**0.617**	0.309	**0.754**	0.286	**0.863**	0.103	**0.755**
max(GMD,Hough)Detector+Tracker	**0.753**	0.344	**0.804**	0.331	**0.950**	0.096	**1.174**
Feature Detector	**0.564**	0.443	**0.580**	0.430	**0.984**	0.041	**0.222**
Feature Detector + Tracker	**0.775**	0.348	**0.782**	0.349	**0.992**	0.019	**0.270**

manner the highest informedness is achieved. When the output of this detection system is processed by the tracker, a result of 0.804 sensitivity, 0.95 specificity, and 0.753 informedness is achieved. The performance of this artificially created system serves as a benchmark for comparison to the feature-based detection algorithm. When the feature-based detection event stream is evaluated alone we observe a low sensitivity value of 0.58 but the highest specificity so far at 0.984. However, when combined with the tracker the sensitivity jumps to 0.782, the specificity to 0.992 and the informedness to 0.775 with the latter two being the highest achieved measures on the dataset, exceeding even the combined GMD-Hough system. Together these results show that after the tracking stage is completed, the proposed feature-based detection approach outperforms all other methods tested including the post hoc combined Hough-GMD detector with unrealistic access to ground truth demonstrating the performance of the proposed approach on this challenging space imaging dataset. Finally, as detailed in the last column of Table 2.4, the processing time of the feature-based detector at 0.222 real-time duration, is approximately double the much simpler and lower performing GMD detector. When augmented with the tracker the feature-based detection and tracking system process events faster than all other approaches at only 0.27 times real-time duration. This is less than half the processing time of the GMD detection and tracking system which passes through a significant number of noise detection events to the more computationally expensive tracker as is evidenced by lower specificity of the GMD detector relative to the feature-based detector. This best of both world's performance, of high processing speed and high algorithm complexity resulting in high accuracy, is only possible due to the highly optimized cascaded event-based filtering design described in Section 2.3.3.

2.4 Discussion and Future Work

In terrestrial event-based recording conditions a typically complex, feature-rich scenery is observed at a relatively high SNR, generating event streams with high variance in event rates. In contrast, event-based space imaging typically contains sparse simple featured scenes with low SNR and very stable event rates. In this context, the primary challenge is not the processing of a complex environment, but the extraction of simple faint detections from a highly noisy random event stream. In this context, even the most simple event-based algorithms such as hot pixel filters can become problematic given the similarity of noisy pixels to the stationary point sources targeted in EBSSA. Thus EBSSA is to a significant degree an exercise in SNR enhancement. Two entirely independent solutions to this problem of low SNR are of course the design of specialized event-based space imaging sensors and more immediately online automated optimization of current event-based sensor biases to recording conditions. Among the recordings in the dataset are instances where due to the incidental alignment of sensor biases to the recording conditions extremely faint low earth orbits objects exhibiting random trajectories are observed. In theory, such LEO observations should populate all recordings in the dataset, yet they are present in only a few. On the other hand, regardless of future improvements in sensor technologies, improved observing conditions and future implementations of online sensor bias optimization systems, there will always remain fainter space objects to be observed and extracted from the event stream. This perpetual requirement for higher sensitivity will continue to motivate the configuration of sensor biases for higher sensitivity (and higher noise) in the space imaging context. This ensures that such event-based datasets will continue to be noisy and in need of robust detection and tracking algorithms like those described in this work.

One important hyperparameter in the algorithms presented, and in all low SNR event-based applications, is the size of the ROI patch used. While small ROIs with faster decaying memory suffice in high SNR contexts, in low SNR applications such as EBSI, larger-sized ROIs with slower decaying memory collect more information from a larger spatio-temporal volume which typically results in better performance. On the other hand, increasing a system's ROI size reduces its speed. Through heuristic testing of the space imaging dataset and the algorithms presented in this work, an ROI of fifteen pixels was found to provide a reasonable trade-off between performance and speed. In future work, we aim to investigate the use of non-binary ROI collection windows which weigh events continuously with spatio-temporal distance to the current event.

Another important hyperparameter that was investigated in detail was the shape and weights of the LUT templates used to generate the angular activation vector $\mathbf{\Lambda}$. Initially, it was assumed the precise image used for the template and its fidelity to observed space object shapes would significantly impact the accuracy of the overall system and be highly specific for each particular class and size of the objects observed. In practice, it was found through experimentation with a range of different bar shapes, lengths, widths and template values, that as long as the template was strongly uni-directional, the precise shape of the template did not significantly impact performance.

In this work the proxy signal ζ estimating unimodality of the angular activation $\mathbf{\Lambda}$ was used for scale, speed and rotation invariant detection of point sources. In typical terrestrial event-based contexts with their higher SNRs and more complex features a more local plane fitting optical flow algorithm is used as the first step in detecting events on moving edges [77]. These events are then augmented with orientation information that is analogous to θ_i in this work. In future work, we apply the optimized hardware implemented feature-based detection algorithm presented here to extremely low SNR terrestrial contexts where the larger ROIs are likely to provide improved performance over more localized optical flow detection algorithms.

The number of streak templates which in this work was chosen as $N = 36$ was determined heuristically. In general, with other factors kept constant, smaller values of N resulted in faster template matching, but reduced feature detection accuracy especially when detection streaks. Above $N = 36$, little improvement in performance was observed and for this reason, $N = 36$ was selected to generate ten-degree offsets for neighboring templates.

In this work, the angular activation of the detection stream θ_i was utilized by the tracker in a straight forward manner as just another spatial dimension albeit a circular one thus helping to remove spurious delayed trail events. This orientation information, however, can potentially be utilized further to update the tracker estimate potentially providing better performance by incorporating the orientation of motion θ_i especially in informationally sparse conditions such as where the velocity of a newly detected faint object has not yet been ascertained. In such initial conditions where gaps in the trajectory of a faint object are common, further incorporation of orientation information into the tracker could provide improvements in performance. Investigation of this approach is the subject of future work.

2.5 Conclusion

In this work, the first event-based space imaging dataset was presented. The labeled dataset, augmented with a larger unlabeled dataset, provides a test bench for investigation of event-based algorithms for the unique and challenging space imaging environment. Statistical measures were introduced where event density activated spatio-temporal volume slices can be used to compare the sensitivity, specificity and informedness of extremely heterogeneous event streams. In this way, the output of the proposed detection and tracking systems can be directly compared to the raw input events quantifying improvements at each stage and providing insights into properties of the dataset as well as the operation of the algorithm. The expert labeling procedure used was validated using an artificial dataset with analytically defined ground truth. The expert labeling procedure was shown to provide a highly accurate label set across a wide range of SNR environments. Several high-speed event-based algorithms with different levels of complexity were tested on the dataset with the feature-based detection and tracking method outperforming the other methods combined, both in terms of accuracy as well as in speed of operation. By measuring an optimized proxy measure for the unimodality of angular activation over a fairly large, slow decaying local time surface region, the feature-based method was shown to provide a scale, rotation and speed invariant target detection capability that is ideal for the event-based space imaging context. In terms of speed of operation, the cascaded event-filter design of the detection and tracking system provides a high-speed event processor.

CHAPTER 3

Investigation of Event-based surfaces

Chapter Summary

In this chapter, we investigate event-based feature extraction through a rigorous framework of testing. We test a hardware efficient variant of Spike Timing Dependent Plasticity (STDP) on a range of spatio-temporal kernels with different surface decaying methods, decay functions, receptive field sizes, feature numbers and back end classifiers. This detailed investigation can provide helpful insights and rules of thumb for performance versus complexity trade-offs in more generalized networks, especially in the context of hardware implementation, where design choices can incur significant resource costs. The investigation is performed using a new dataset consisting of model airplanes being dropped free-hand close to the sensor. The target objects exhibit a wide range of relative orientations and velocities. This range of target velocities, analyzed in multiple configurations, allows a rigorous comparison of time-based decaying surfaces (time surfaces) versus event index-based decaying surface (index surfaces), which are used to perform unsupervised feature extraction, followed by target detection and recognition. We examine each processing stage by comparison to the use of raw events, as well as a range of alternative layer structures, and the use of random features. By comparing results from a linear classifier and an Extreme Learning Machine (ELM) classifier, we evaluate how each element of the system affects accuracy. To generate time and index surfaces, the most commonly used kernels, namely, event binning kernels, linearly and exponentially decaying kernels, are investigated. Index surfaces were found to outperform time surfaces in recognition when invariance to target velocity was made a requirement. In the investigation of network structure, larger networks of neurons with large receptive field sizes were found to perform best. We find that a small number of event-based

feature extractors can project the complex spatio-temporal event patterns of the dataset to an almost linearly separable representation in feature space, with best performing linear classifier achieving 98.75% recognition accuracy, using only twenty-five feature extracting neurons.

3.1 Introduction

The last decade of development in the field of event-based cameras has motivated the development of a range of event-based or spiking feature detection and recognition algorithms [78][79–82]. These feature-based algorithms and indeed all event-based algorithms, require as their initial step, a method for storing memory of recent events. In [83], Tapson et.al proposed the potential use of a wide range of such memory preserving kernels in event-based systems. These kernels included exponential kernels, Gaussian functions and other more biological plausible kernels. The first of these kernels, which decay exponentially with respect to time was also used in [81], where they were labeled as 'time surfaces' and combined with unsupervised feature extraction and classification to form an event-based convolutional network which was named Hierarchy of Time Surfaces (HOTS).

These time surfaces which are a particularly effective method of implementing event-based memory systems are the subject of the investigation in this work. Here, we set out to rigorously quantify in detail the share in performance improvement attributable to each element of an event-based surface generation, convolution, feature extraction and classification system. More precisely we investigate the memory generation and decay methods, commonly used memory kernels, use of raw events relative to the use of feature events, the event-based convolutional structure of the feature extractor networks and the additional classification performances provided by the back-end classifiers.

An important question arising at every stage of any event-based algorithm is whether the event rate should inform the progression of the algorithm through time. In this work, we investigate this question through comparisons of time surfaces and index surfaces where the memory of events decay either as a function of time or event index, respectively.

Processing event memory as a function of time is straight-forward and intuitive. By decaying event memory as a function of time, all elements of an event-based system operate in a uniform time-based manner regardless of the informational content in any part of the sensor's field of view. The behavior of time-based decaying memory does not vary as a function of sensor size or any aspect of the visual scene that alters the event generation rate, such as scene contrast or texture. However, once the sensor event rate is incorporated into the operation of the system, these invariances may no longer hold, since a change in event rate may alter the decay rate of the memory of the event stream, potentially resulting in information loss. Therefore, algorithms using event rate information in memory decay require more careful testing, parameter selection and potentially secondary solutions such as localized memory decay mechanisms to mitigate information loss. On the other hand, processing event memory as a function event count or index does have one crucial advantage over a purely time-based processing system. In general, event-based vision sensors generate more events in response to faster-moving objects when holding other variables constant. This approximately proportional relationship between local event rate and local velocity allows an algorithm operating as a function of event index to effectively make computational decisions at approximately the same speed as the object being observed. Previous works have suggested that the use of event index to decay memory provides greater robustness in the presence of such variance in target velocity [84, 85]. In [85] an event-based Hough transform was used for tracking and in [86] this was augmented with an event-based particle filter to improve tracking performance. The Hough transform in these works was implemented using a window of fixed event size, thus incorporating the event-rate information into the algorithm. The results showed that higher target velocities increased the update rate of the algorithm, allowing better tracking performance at high velocity. In [84], windows of fixed event number and fixed time windows were compared in their performance in simultaneous tracking and recognition, and a slightly higher recognition accuracy was achieved when the algorithm was tested for velocity invariance. Such robustness to observed velocities in the data can be critical in a range of real-world applications. These results, and the potential utility of velocity robust algorithms in real-world applications of event-based sensors, motivate a central element of the investigation presented in this work. One such example is one of the few current applications of event-based sensors: the field of event-based Space Situational Awareness (SSA) introduced in [87] and described in Chapter 2 where the number of event-based observations is extremely limited. A major aspect of this limitation is that particular targets may only have been observed at a single velocity relative to the sensor. Yet such objects must be detected, tracked and identified robustly regardless of their relative velocity. This requirement of robustness to target velocity variations motivates the detailed rigorous examination of time and index surfaces in combination with a range of commonly used decay kernels.

Another important element in a wide range of event-based algorithms is the use of feature extractors. The contribution of the feature extraction layer as a whole is the simplest to determine and yet can often be missing in the literature as a baseline performance measure. This involves directly feeding sensor events into the final stage classifiers in the same manner as the output feature layer, skipping the intervening feature extraction layer(s). A more subtle question is how effective the learnt features are. In other words, how well does the learning algorithm orient the feature set with respect to the data so as to cover the underlying non-linearities in the dataset? This can be ascertained by comparing the mean recognition performance of multiple independently learnt features against random instantiations of features with the same network structure and feature weight distribution. The power of random features to cover non-linear feature spaces has been demonstrated by the Extreme Learning Machine ELM [88] literature. By comparing feature extraction algorithms to a baseline of random features a better understanding of the relative improvement can be ascertained. An alternate baseline for learnt features is hand-crafted features which are based on standard basis functions such as 2D Gabor filters. However, unlike random features, the performance of hand-crafted features is highly dependent on the precise design of the feature. This approach would thus require the testing of a very large class of features in order to provide a meaningful comparison and as such is not used in this work.

Finally, the most complex measure that is investigated is the role of the classifier on the performance. While there are a wide range of potential back-end classifiers that may be used, we propose that the combined use of linear classifiers and large hidden layer ELMs have particular utility in providing a rigorous measure of residual non-linearity following each stage of processing. This is because, unlike other classifiers, which through learning orient their non-linear features toward the training data, the random non-linear projections of the ELM's hidden layer create projections that are approximately uniform with regard to the structure of the data. As such the size of the hidden layer provides a reasonably "unbiased" measure of the residual non-linearities present after each a processing layer.

3.2 Methodology

3.2.1 The ATIS Plane Dropping Dataset

The system presented in this work constitutes an event-based high-speed classification system and makes use of a noisy object recognition dataset to demonstrate and characterize its performance.

A variety of event-based datasets now exist, such as the N-MNIST and N-Caltech101 [89], MNIST_DVS [90] and the event-based UCF-50 datasets [91]. One common facet of these datasets is that they have been generated under highly constrained conditions, especially with respect to the range of target object velocities. For a static image, event-based cameras only produce data in response to motion and therefore require either the static image or the camera itself to be moving. Therefore, the velocities involved in many of the event-based datasets are strictly controlled. This is often a desirable trait to ensure consistency across all samples, but this constraint is a strongly artificial one. Other event-based datasets, such as the visual navigation dataset found in [92], do not control velocity in the same manner, but represent a fundamentally different task and are therefore not well-suited to exploring detection and feature extraction mechanisms.

The need to explore the effect of variances in velocity is important as these tend to produce significant variance in the spatio-temporal event patterns generated by event-based cameras. This can have a significant impact on the performance of a classifier or detection algorithm. A primary focus of this work is on the comparison of different event-based processing approaches in the presence of such variance. This required the creation of a new dataset designed to test event-based classification algorithms under conditions that are less constrained and closer to those found in real-world tasks. However, as well as being reasonably difficult, the dataset was also designed to be constrained enough to allow a rigorous comparison of the various parameters and architectures of interest. As such the dataset was specifically designed to act as a proxy for a noisy local region in a larger real-world dataset.

The task is to identify model airplanes as they rapidly pass through the field of view of an ATIS camera. The airplanes were dropped free-hand, and from varying heights and distances from the camera, as shown in Figure 3.1(a). Four model airplanes were used, each made from steel and all painted uniform gray, as shown in Figure 3.1(b). This served to remove any distinctive textures or marking from the airplanes, thereby increasing the difficulty of the task. The airplanes are models of a Mig-31, an F-117, a Su-24 and a Su-35, with wingspans of 9.1, 7.5, 10.3 and 9.0 cm, respectively.

The recordings were captured using the same model of ATIS camera and the same acquisition software used in capturing the N-MNIST dataset in [89], and the recordings were stored in the same file formats, thereby maximizing compatibility with other neuromorphic algorithms and systems. The models were dropped 100 times each from a distance ranging from 120 to 160 cm above the ground and at a horizontal distance of 40 to 80 cm from the camera. This ensured that the airplanes passed rapidly through the field of view of the camera, with the planes crossing the field of view in an average of 242 ± 21 ms. No mechanisms were used to enforce consistency of the airplane drops, resulting in a wide range of observed speeds from 0 to >1500 pixels per second. Additionally, there were variable delays before and after each drop, resulting in recordings of varying lengths. The dataset was augmented with left-right flipped versions of the recordings, resulting in 200 drops for each airplane type. An example of the variability in the airplane drops is demonstrated in Figure 3.1(c), which shows binned events in the same 3 ms slice of data from 20 randomly selected recordings from the dataset. The samples demonstrate significant variations in the positions of the airplanes, their orientations and their sizes. No attempt was made to fine-tune the sensor's biases for the particular light condition or target velocities. This lack of tuning is likely in real-world environments where the recording conditions may not be known a priori. An example of this

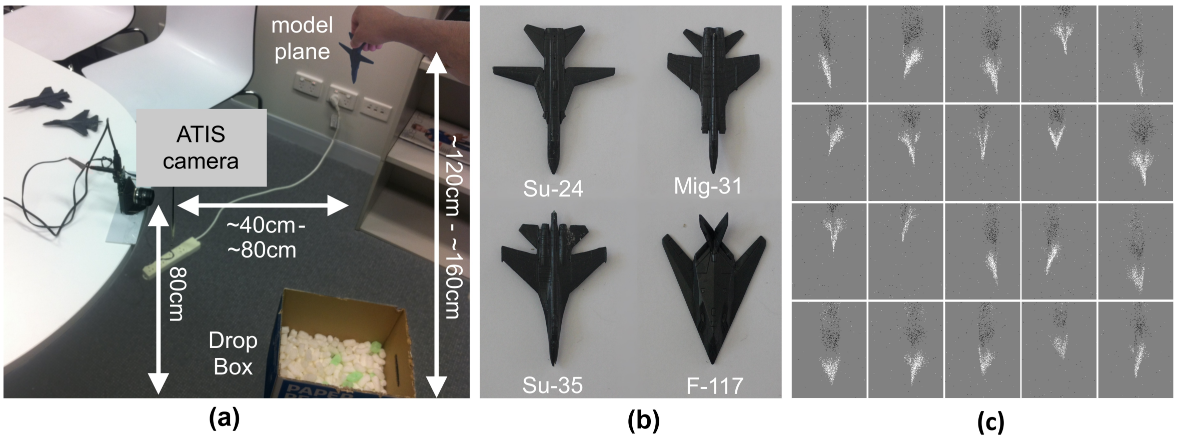


FIGURE 3.1: **Data collection setup and samples of the airplane dropping dataset.** (a) The physical setup used for recording dataset in which an ATIS camera is attached to a table and the airplanes dropped freehand in front of the camera. (b) A top-down and labeled view of the four model airplanes used to generate the dataset. (c) Examples of the variation in the dataset in terms of position, scale, orientation and speed. Each image represents a frame rendered from the same 3 ms of events extracted from each recording with ON events represented with white pixels and OFF events represented with black pixels. The twenty random samples clearly demonstrate the difficulty of the recognition task. Unlike most event-based datasets, the camera was not tuned or biased for the application, simulating real-world noisy dynamic environments where such fine-tuning would be difficult or impossible. As a result of this arbitrary untuned camera configuration the OFF events (black) in the entire dataset produced essentially noise clouds and as such were discarded. Airplane class key ordered from top left to bottom right, Mig-31:$\{2, 3, 7, 11, 12\}$, F-117:$\{9, 15, 16, 18, 19\}$, Su-24:$\{1, 5, 8, 14, 20\}$, and Su-35: $\{4, 6, 10, 13, 17\}$.

is the previously mentioned SSA application, where acquired data is inevitably noisy, often with one of the sensors polarities entirely unable to capture useful events from the target due to the sensor biases not being matched to the lighting or velocity profile of the target. Even when the sensor biases are ideal for the lighting and temperature conditions of the recording, there are always fainter targets of interest in the field of view which can only be viewed by lowering sensor biases and "delving deeper into the noise" to accumulate events from these fainter objects. Thus, allowing noise and un-tuned biases into datasets, additional real-world challenges, such as structured noise and unevenly performing polarities, become apparent, motivating the implementation of robust solutions and new network behaviors that would otherwise be missed.

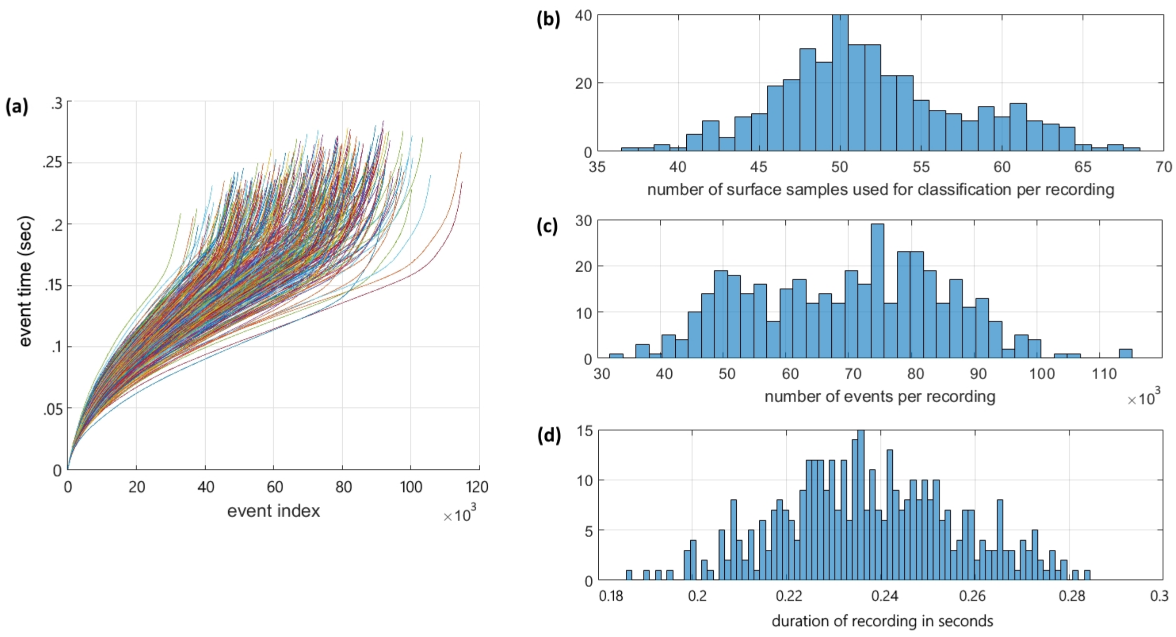


FIGURE 3.2: **The Dataset Summary.** (a) Event timestamp profiles of all airplane drops in the dataset showing the event timestamps of each recording as a function of event index. The timestamp profiles demonstrate the variable rates of event generation within and across the recordings. These differences are a function of the speed, size and shape of the airplanes and the distance from the camera. Note the color assigned to each recording profile is arbitrary. (b) Distribution of the number of frames per recording for each recording in the dataset. (c) Distribution of the number of events per recording for each recording in the dataset. (d) Distribution of the duration of each recording in the dataset.

Figure 3.2(a) shows the event time vs. event index profiles of all recordings in the dataset showing the significant inter and intra-recording variance in data-rate present in the dataset. While the number of recordings in the augmented dataset is 800, the number of frames making up the data sample points presented to the detection and recognition algorithm is >20,000. The free-hand drop methodology resulted in significant variance in velocity and orientation of the model airplane within each recording. As a result, the spatio-temporal output patterns varied significantly through each recording, as shown in Figure 3.2(a) and discussed in later sections. The distribution of the number of frames per recording is shown in Figure 3.2(b). These frames represent event-based sampling operations presented to the classifier. Figure 3.2(c) and (d) show the distribution in the number of events per recording and recording duration for the dataset. The full dataset is freely available at [93].

3.2.2 Time Surfaces vs. Index Surfaces

Following notation in [60], an event e_i from the ATIS camera can be described mathematically by:

$$e_i = [\boldsymbol{x}_i, t_i, p_i]^T \tag{3.1}$$

where i is the index of the event, $\boldsymbol{x}_i = [x_i, y_i]$ is the spatial address of the source pixel corresponding to the physical location on the sensor, $p_i \in \{-1, 1\}$ is the polarity of the event indicating whether the log intensity decreased or increased, and t_i is the absolute time at which the event occurred.

Event-based algorithms require iterative processing of each event and therefore require that each new observation be combined with previously observed local events, both in space and in time. This is accomplished using a variation of the time surfaces extended to cover surfaces decaying based on time and also based on event index (index surfaces). The timing and polarity information contained in each event, as shown in (3.1), allows the generation of two useful surfaces, based on time and polarity, from which more complex surfaces can be constructed. The first surface, referred to as $\boldsymbol{T}_i$, maps the time of the most recent event to spatial pixel location and is described in (3.2), with the corresponding surface $\boldsymbol{P}_i$ for event polarity given by (3.3). Note that as discussed above, due to the excessive noisiness of the OFF events due to untuned biases, only ON events with p_i = 1 were used.

$$\begin{aligned} T_i &: \mathbb{R}^2 \rightarrow \mathbb{R}, \\ x &: t \rightarrow T_i(x) \end{aligned} \tag{3.2}$$

$$\begin{aligned} P_i &: \mathbb{R}^2 \rightarrow \{-1, 1\}, \\ x &: p \rightarrow P_i(x) \end{aligned} \tag{3.3}$$

Here, we compare the time surfaces which decay as a function of time, with index surfaces, where the surface values for all pixels decay not as a function of time, but in response to new incoming events. We then define the analogous function to 3.2 for index surfaces. This surface, $\boldsymbol{I}_i$, is defined in 3.4 and stores the indices of the incoming event for each pixel.

$$\begin{aligned} I_i &: \mathbb{R}^2 \to \mathbb{R}, \\ x &: i \to I_i(x) \end{aligned} \tag{3.4}$$

In addition to exploring time-based decay and index-based decay, three different transfer functions or temporal kernels are investigated. These kernels are event binning ($\boldsymbol{BTS}/\boldsymbol{BIS}$), linear decay ($\boldsymbol{LTS}/\boldsymbol{LIS}$) and exponential decay ($\boldsymbol{ETS}/\boldsymbol{EIS}$). As a point of reference, the HOTS algorithm makes use of exponential decaying time kernels $\boldsymbol{ETS}$.

In all surface generation methods, when a new event arrives, the surface at $\boldsymbol{x}_i$ is set to $\boldsymbol{P}_i$. When using the event binning technique, the value on the surface maintains its value over a temporal window τ_e or index window N_e, after which it is reset to zero. The event binning method for surface generation is described by equations (3.5) for the time-based binning surfaces ($\boldsymbol{BTS}$) and (3.6) for the index-based binning ($\boldsymbol{BIS}$).

$$\boldsymbol{BTS}_i(\boldsymbol{x}, t) = \begin{cases} \boldsymbol{P}_i(\boldsymbol{x}), \text{if} & t - \boldsymbol{T}_i(\boldsymbol{x}) \leq \tau_e \\ 0, \text{if} & t - \boldsymbol{T}_i(\boldsymbol{x}) > \tau_e \end{cases} \tag{3.5}$$

$$\boldsymbol{BIS}_i(\boldsymbol{x}) = \begin{cases} \boldsymbol{P}_i(\boldsymbol{x}), \text{ if} & i - \boldsymbol{I}_i(\boldsymbol{x}) \leq N_e \\ 0, \text{ if} & i - \boldsymbol{I}_i(\boldsymbol{x}) > N_e \end{cases} \tag{3.6}$$

For the linearly decaying time surface ($\boldsymbol{LTS}$) and linearly decaying index surface ($\boldsymbol{LIS}$), the initial value set on the surface in response to a new event instead decays toward zero linearly as a function of time. These surfaces are described by (3.7) for time-based linear decay or in response to incoming events as described by (3.8) for index-based linear decay.

$$\boldsymbol{LTS}_i(\boldsymbol{x}, t) = \begin{cases} \boldsymbol{P}_i(\boldsymbol{x}) \cdot \left(1 - \frac{t - \boldsymbol{T}_i(\boldsymbol{x})}{2\tau_e}\right), \text{ if} & t - \boldsymbol{T}_i(\boldsymbol{x}) \leq 2\tau_e \\ 0, \text{if} & t - \boldsymbol{T}_i(\boldsymbol{x}) > 2\tau_e \end{cases} \tag{3.7}$$

$$\boldsymbol{LIS}_i(\boldsymbol{x}) = \begin{cases} \boldsymbol{P}_i(\boldsymbol{x}) \cdot \left(1 - \frac{t-\boldsymbol{I}_i(\boldsymbol{x})}{2N_e}\right), \text{ if} & i - \boldsymbol{I}_i(\boldsymbol{x}) \leq 2N_e \\ 0, \text{ if} & i - \boldsymbol{I}_i(\boldsymbol{x}) > 2N_e \end{cases} \tag{3.8}$$

The exponential decay method works in a similar manner to the linear decay, with the value placed on the surface decaying exponentially instead of linearly with respect to either time or event. This results in the equations for the exponentially decaying time surface ($\boldsymbol{ETS}$) shown in (3.9, and the exponentially decaying index surface ($\boldsymbol{EIS}$) shown in (3.10).

$$\boldsymbol{ETS}_i(\boldsymbol{x}, t) = \boldsymbol{P}_i(\boldsymbol{x}) \cdot e^{\frac{\boldsymbol{T}_i(\boldsymbol{x})-t}{\tau_e}} \tag{3.9}$$

$$\boldsymbol{EIS}_i(\boldsymbol{x}) = \boldsymbol{P}_i(\boldsymbol{x}) \cdot e^{\frac{\boldsymbol{I}_i(\boldsymbol{x})-i}{N_e}} \tag{3.10}$$

The equations for these surfaces make use of a constant parameter, time constant τ_e for time-based methods and index constant N_e for the index-based methods and the chosen values for these parameters are shown in Figure 3.2(a) and (b). The plots show the time surface and index surface generation kernels which have an area under the curve of 3 ms in (a), and 554 events in (b), respectively. These values were chosen based on the mean data rate over all recordings.

Given the 184.5 k event/s event rate for the entire dataset the area under the curves in Figure 3.3, τ_e = 3 ms and N_e = 554, respectively were chosen to be approximately equal, thus resulting in approximately equal total surface activation for the time and index-based decay methods over the entire dataset, but not for any individual recording or section thereof.

To illustrate the difference in the two decay methods, Figure 3.4 shows the index surface subtracted from the time surface for a single recording from the dataset. The figure shows that the binning time surface has a lower activation than the binning index surface when the speed of the airplane is low (at the start of the recording). As the airplane speeds up through its fall, the total time surface activation continues to increase whilst the index surface remains

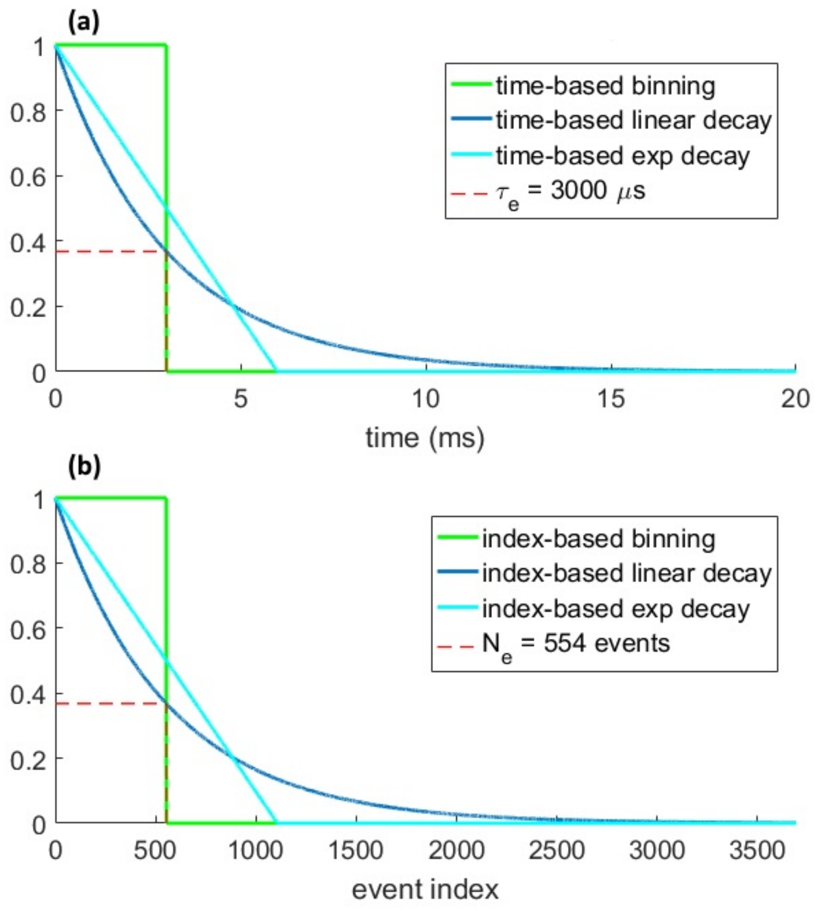


FIGURE 3.3: **Graphs of the six methods for generating time and index surfaces.** (a) Shows the three time-based kernels over time. Note that the area under all kernels is the time constant τ_e = 3 ms. (b) Shows the value of the index-based kernel as a function of event index. Here the mean dataset event rate over all recordings (184.5 k events/s) was used to obtain equivalent sized kernels with index constant N_e = 554 events.

approximately constant. In fact, at t = 157 ms, the total activation on the time surface is approximately twice that of the index surface which remains relatively stable throughout the recording. This stability of index surface activation is the direct result of the decay process. Since both the increase and decrease in surface activation are a function of event index, all decay kernels with a finite impulse response will inevitably generate stable surface activations. This is in contrast to the time decay method where no coupling exists between the activation and decay of the surface. Panels (d), (e) and (f) in Figure 3.4, show that the differences between the two decay methods are greatest for the binning method, followed by linear decay and finally exponential decay, which is the result of a slight reduction in surface activation from binning to linear to exponential decay for the time surfaces. This reduction is due to the kernel width such that the arrival of a new event overwrites the entry for a pixel that has recently been activated. The motivation for overwriting the previous surface value, as opposed to the accumulation of event kernels over the surface, is that the former approach allows event-based surface generation at arbitrary points in time using only the timestamp of the last event at each pixel. The latter approach, however, requires either continuous calculation of the surface activation or the storage of all previous events. The overwrite effect is more pronounced for kernels with a longer time window as the surface maintains the value for longer. This same effect is also present in the index surfaces but is less prominent due to the lower variance of the index-based activation plots. Figure 3.4 also highlights the event-overwrite effect for different decay methods and kernels, as well as the significantly lower variance of index surface activation in the presence of change in velocity (due to gravity) relative to time surfaces. Such lower variance potentially allows downstream processing stages to be optimized for the stable operating point of the index surface.

Panels (a),(b) and (c) of Figure 3.4, show the surface differences ($\boldsymbol{BTS}_i$ - $\boldsymbol{BIS}_i$), ($\boldsymbol{LTS}_i$ - $\boldsymbol{LIS}_i$) and ($\boldsymbol{LTS}_i$ - $\boldsymbol{LTS}_i$), respectively at the beginning of the recording (t = 36 ms). This moment in the recording is marked (1) on panel (d) which displays total surface activation for the binning method $\Sigma_{x,y}\boldsymbol{BTS}_i$ and $\Sigma_{x,y}\boldsymbol{BIS}_i$. The two traces in (d) show that at the beginning of the recording when the target airplane's speed is low the binning time surface has a lower activation than the binning index surface. However, as the target speeds up, the total time surface activation also increases, while the index surface remains approximately stable, such that by t = 157 ms the time surface activation $\Sigma_{x,y}\boldsymbol{BTS}_i$ is approximately twice that of $\Sigma_{x,y}\boldsymbol{BIS}_i$. Panels (e) and (f) show a similar but slightly less pronounced relative increase for the linear and exponential decay surfaces. Panels (g), (h) and (i) show this relative increase for the binning, linear and exponential decay surfaces by plotting the differences of (a), (b) and (c) at t = 157 ms.

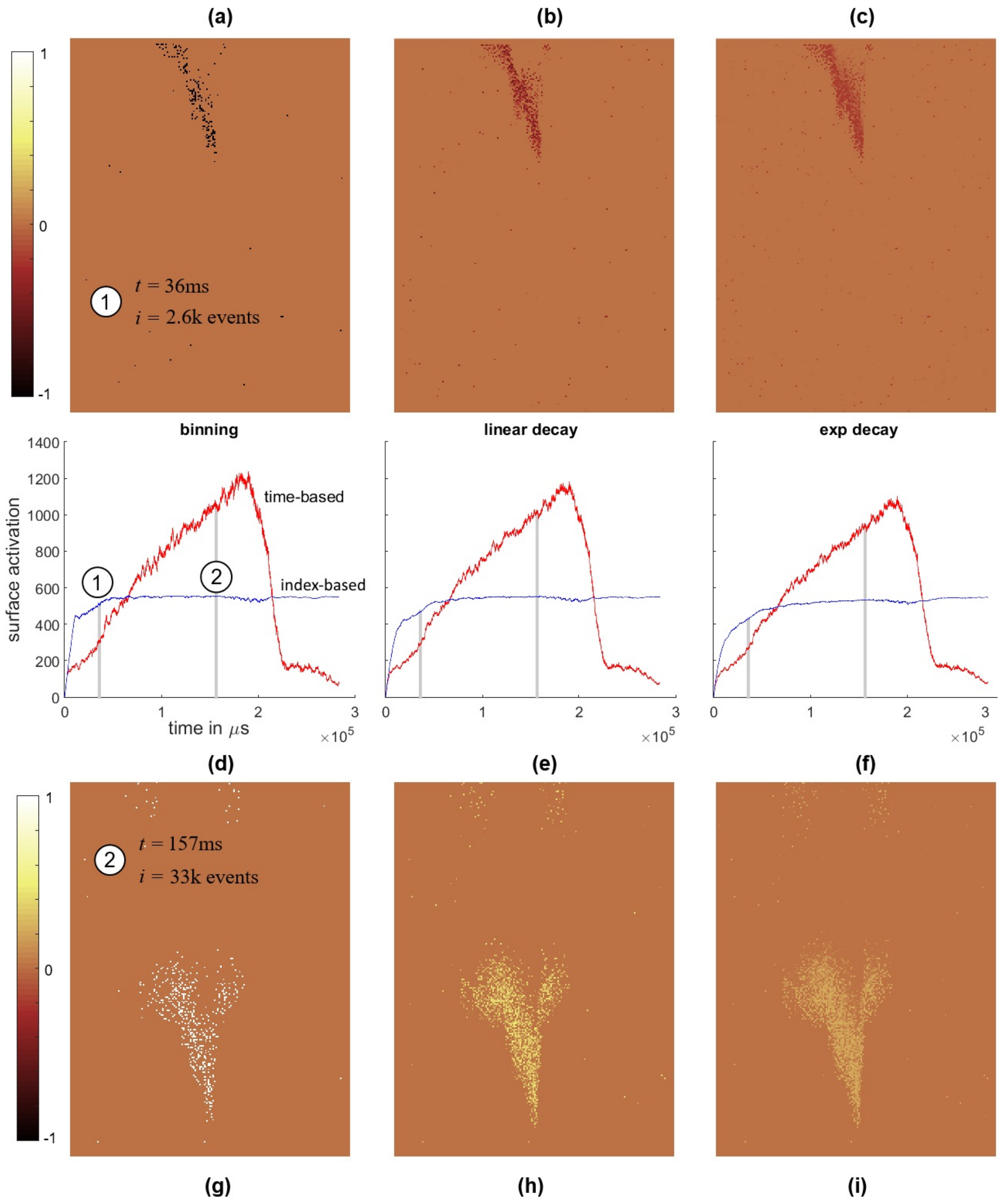


FIGURE 3.4: **Examples of the six methods for generating time and index surfaces.**

3.2.3 Target Velocity vs. Surface Activation

Prior to the feature extraction and recognition, the airplane is detected and the location within the field of view is determined. The speed of the airplanes is much faster than any other stimulus expected within the field of view of the camera, such as the body of the author accidentally entering the frame, as can be seen in the lower right pane of Figure 3.5(c). Therefore, the summation of events across the rows and columns of the camera's field of view (after normalization and thresholding as shown in Figure 3.5(a) and (b) provides a simple method to detect the boundary of the airplane in the limited context of this investigation. While the presence of slow-moving objects in the background can be rejected as shown in Figure 3.5(c), complex background objects with similar velocities to the target would impair this simple object detector.

In terms of limitations, the presented dataset is constrained in the sense of having only a single high-speed object in the field of view against an effectively blank background. This restriction allows a more focused investigation of different methodologies as well as of the sources of variance in the data such as target orientation and velocity. While the restriction may appear to limit the generalization of the results to more complex scenes, the dataset and the resulting network solutions should be viewed as investigating a local region within a more complex visual scene and the processing required for it which would represent a small section in a larger system. Alternatively, a local index-based decaying method can be used that only decays the surface within a local region of an incoming event thus avoiding the limitations outlined. Such an approach is likely to provide advantages but also have limitations. However, these are beyond the scope of this work.

By using the detection method described we can plot the estimated vertical position of each target airplane as shown in Figure 3.6, both in terms of time in Figure 3.6(a) and event index Figure 3.6(b). These vertical position profiles serve to further highlight the difference between the index-based and time-based approaches in the context of local velocity. Whereas the estimated position plots take on their expected parabolic shape when plotted against time, when plotted against event index, the trajectories are linear to a first approximation. The

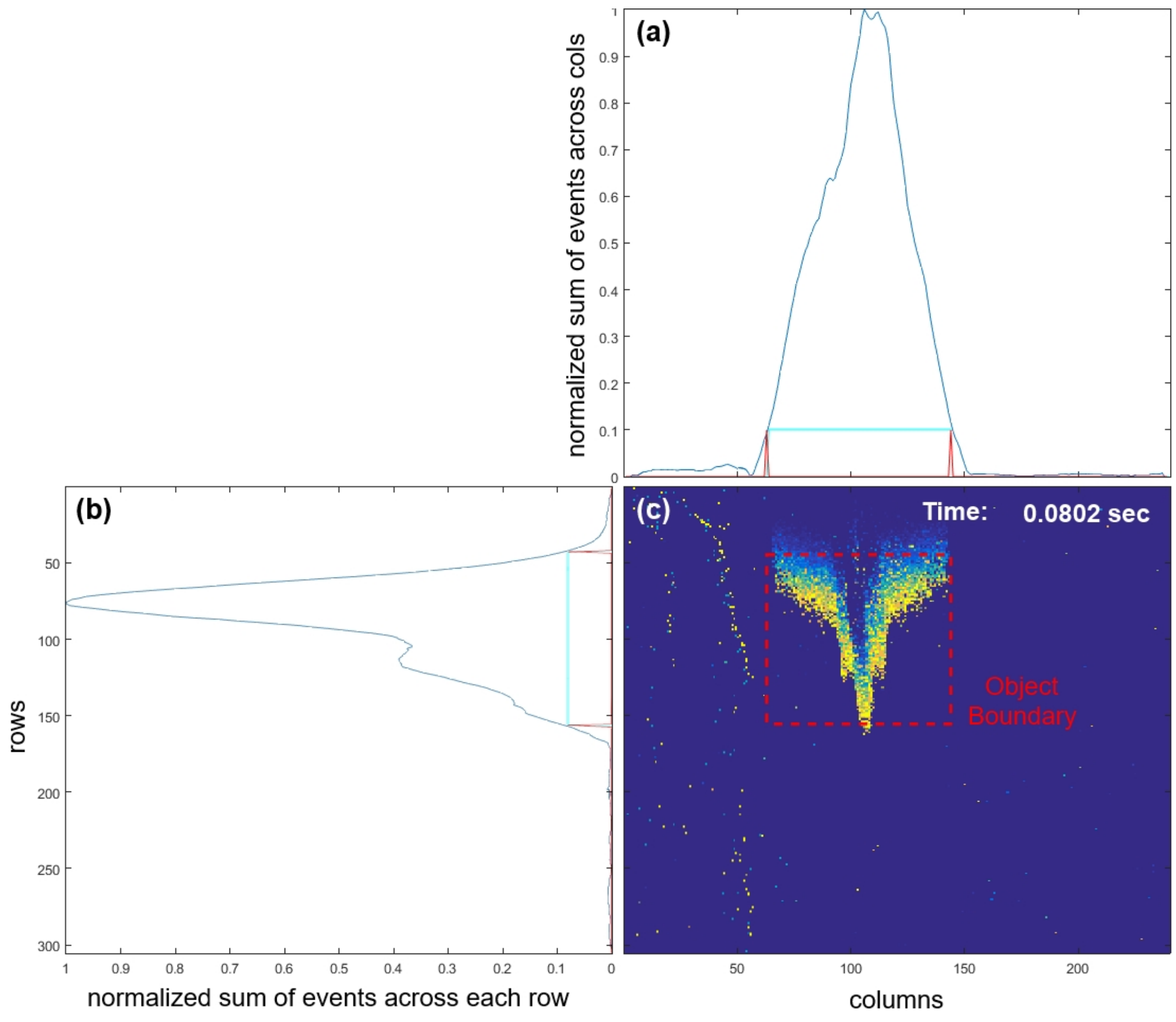


FIGURE 3.5: **Example of an airplane drop test after 0.08 s.** Panels (a) and (b) show a smoothed summation of recent events across columns and rows, respectively. The smoothing was performed by using an 8-pixel wide rectangular moving average window. Due to the relatively high speed of the airplane these summations, when normalized and thresholded at 0.1, could reliably be used to extract the fast-moving airplane from the static background or slower moving objects. The generated target object's boundary is shown in (c). Note that movement of the body of the author (light vertical trace on the left) as he drops the airplane is slow relative to the airplane and generates relatively few events and does not reach detection threshold.

linearity of target position with respect to event index provides an interesting insight into the potential use of index surfaces for tracking, however, this is beyond the scope of the work presented here, which focuses on detection and recognition.

Figure 3.7 illustrates the wide range of velocities in the dataset and the associated mean rate of change in surface activation for time surfaces, index surfaces. The exponentially decay kernel was used for this test. The line of best fit through the data demonstrates different relationships between velocity and change in surface activation which arise from the different geometries of the airplanes. In all cases, however, surface activation is significantly more sensitive to velocity when using time surfaces than index surfaces. This invariance hints at the potential utility of index surfaces for velocity invariant feature generation, where features learnt from a dataset with a particular velocity distribution operate equally well on a dataset with an entirely different velocity distribution, which is not the case for time surfaces. We explore the ramifications of this invariance further in section Velocity Segregated Dataset.

3.2.4 Event-Based Feature Extraction using SKAN

In this work an unsupervised event-based feature extractor was used to learn the most common spatio-temporal features generated by the recordings. Unsupervised feature extraction or feature learning refers to methods where no supervisory signal is used to adapt the neurons or features of a neural network. The unsupervised spike-based feature extraction algorithm was developed for hardware implementation, as previously described in [16]. In the Synapto-dendritic Kernel Adaptation Network (SKAN) algorithm, the adaptive synaptic kernels and adaptive thresholds allow the neurons to compete in the temporal domain to learn commonly observed spatio-temporal spike patterns. These adaptive synapto-dendritic kernels provide an abstracted representation of the coupling of pre- and post-synaptic neurons via multiple synaptic and dendritic pathways allowing unsupervised learning and inference of precise spike timings. By conceptually combining multiple synapses, the most numerous elements of any neuromorphic system, into a single adaptive kernel, the SKAN algorithm allows an efficient yet reasonably complex model of STDP to be realized in hardware. In [94], the algorithm was extended using a simplified model of Spike Timing Dependent Plasticity (STDP) [95] to

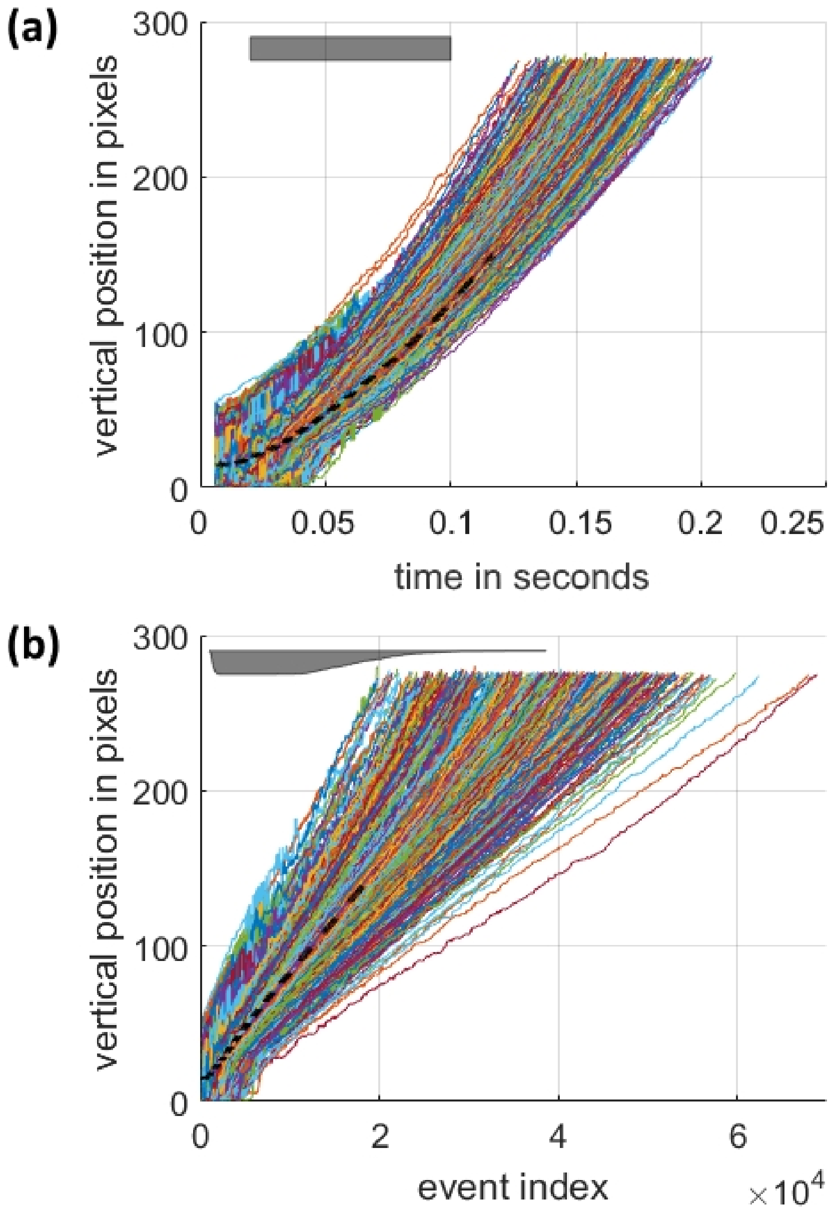


FIGURE 3.6: **Vertical position of targets in the dataset.** Estimated vertical position of the target as a function of time (a) and as a function of event index (b). The dashed black line marks the mean position over all recordings. For the entire dataset, the mean time interval from the first valid object boundary detection event to the last was 156.2 ms with a standard deviation of 17.8 ms. The target's position was defined as the midpoint between the object boundaries as shown in Figure 3.5(c). The gray bar at the top left in (a) indicates the time window used for investigating the effect of target velocities on surface activation in Figure 3.7. The same gray time window bar is shown in the lower (b) panel as a function of event index. The relative thickness of the bar is proportional to the number of recordings in the time window of (a) at each event index. Each colored line indicates a single recording.

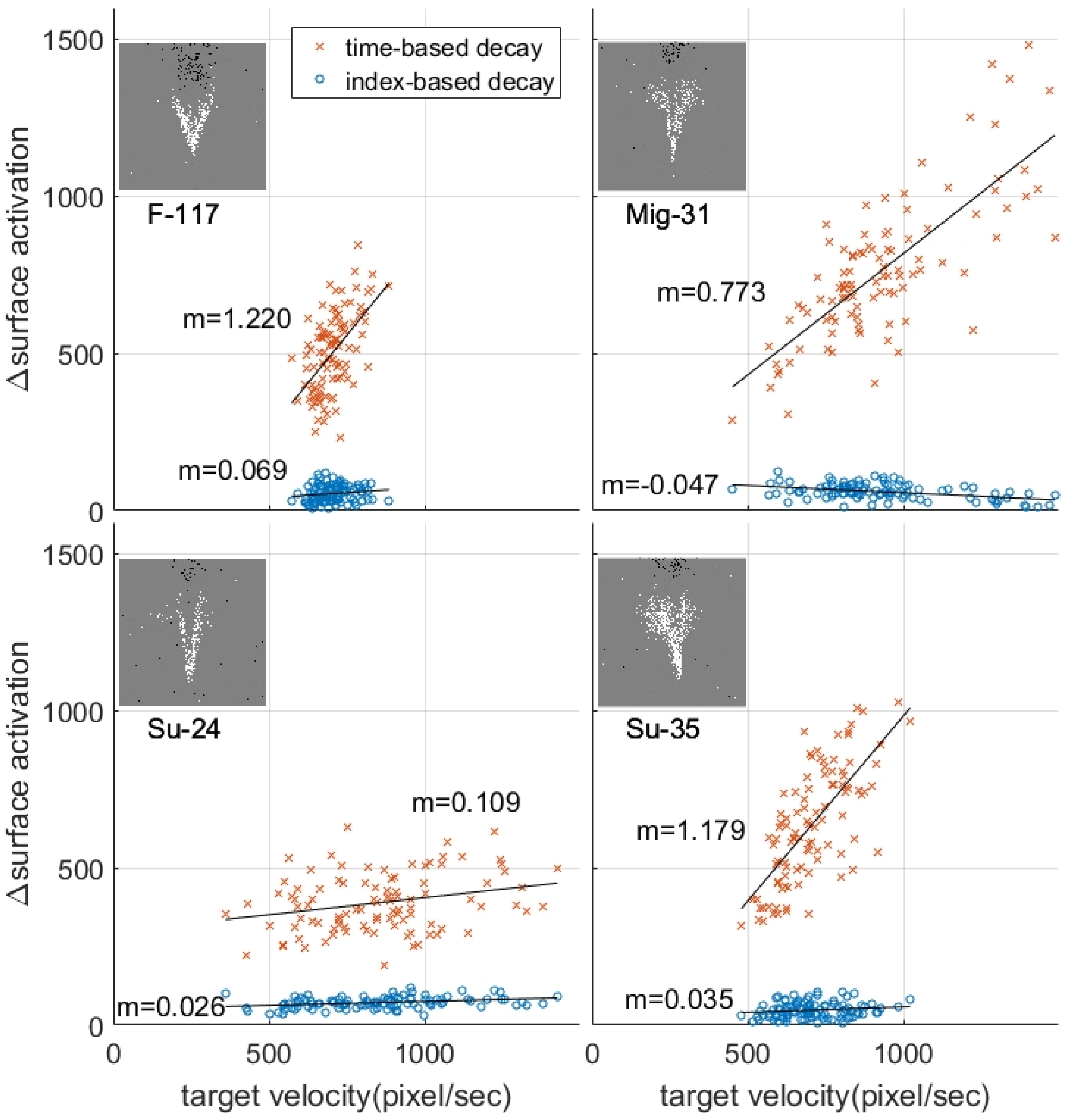


FIGURE 3.7: **Relationship between change in surface activation and target velocity and the resultant mean rate of change in surface activation.** Each point represents a single recording in the dataset. The mean value of target velocity and change in surface activation was calculated over the time window indicated in Figure 3.6(a). For each panel m indicates the slope of the line of best fit.

provide synaptic encoding of afferent Signal to Noise Ratio. In [96], the algorithm was used to perform real-time unsupervised hand gesture recognition using an FPGA. In this work, the event-based approach is continued at the feature extraction layer with the output spike of the winning neuron representing a feature event.

The SKAN layer operates via two simple feedback loops: a synaptic kernel adaptation loop and a threshold adaptation loop. Each input event $u_i(t)$ in a spatio-temporal pattern activates a triangular post synaptic kernel $r_i(t)$ as described by (3.11) and (3.12) where the kernel value $r_i(t)$ rises up to the upper-bound value w_i and then decreases back to zero. Note that in this work, the value of w_i does not adapt and the network operates purely through spike timing adaptation. The kernels generated from this process are then summed at the soma to generate a membrane potential. While this membrane potential is above the neuron's adaptive threshold $\Theta(t)$, the neuron output $s(t)$ goes high, which is analogous to a series of action potentials or a neuronal burst, as described in (3.13). While the neuron output $s(t)$ is high, the kernels perform their temporal adaptation operation as described by (3.12). According to this rule every time step where the neuron output is high and the kernel is rising (q_i = 1), the synaptic kernel's slope Δr_i is reduced by a small amount ddr, thus moving the kernel peak later in time to better match the observed pattern. Conversely, if the event arrives before the peak of the membrane potential, the kernel's slope Δr_i is raised contracting the kernel and moving its peak earlier in time.

$$q_i(\boldsymbol{t}) = \begin{cases} 1, \text{ if} & \Big(u_i(t) = 1 \wedge q_i(t-1) = 0\Big) \vee \Big(q_i(t-1) = 1 \wedge r_i(t-1) < w_i\Big) \\ -1, \text{ if} & \Big(q_i(t-1) = 1 \wedge r_i(t-1) \geq w_i\Big) \vee \Big(q_i(t-1) = -1 \wedge r_i(t-1) > 0\Big) \\ 0, \text{ otherwise} & \end{cases} \tag{3.11}$$

$$\begin{bmatrix} r_i(t) \\ \Delta r_i(t) \end{bmatrix} = \begin{bmatrix} r_i(t-1) \\ \Delta r_i(t-1) \end{bmatrix} + q_i(t-1) \cdot \begin{bmatrix} \Delta r_i(t-1) \\ ddr \cdot s(t-1) \end{bmatrix} \tag{3.12}$$

$$s(t) = \begin{cases} 1, \text{ if} & \Sigma_i r_i(t) > \Theta(t-1) \\ 0, \text{ otherwise} & \end{cases} \tag{3.13}$$

The neuron's thresholds adapt via a similar mechanism to the kernels. At each time step where the neuron output is high the neuron's threshold also rises. In addition at the falling edge of the neuron's output pulse, the threshold falls by a small value. A single inhibitory neuron prevents multiple neurons from spiking at the same time thus preventing duplicate learning of the same pattern by multiple neurons.

$$\Theta(t) = \begin{cases} \Theta(t-1) + \Theta_{rise}, \text{ if } & \Sigma_i r_i(t) > \Theta(t-1) \\ \Theta(t-1) - \Theta_{fall}, \text{ if } & \Sigma_i r_i(t) = 0 \wedge \Sigma_i r_i(t) > 0 \\ \Theta(t-1), \text{ otherwise} & \end{cases} \tag{3.14}$$

This simple hardware implementable rule-set allows the neurons to orient their spatio-temporal receptive fields from a random starting point toward the most commonly observed patterns, thus attempting to optimally represent the observed data given a limited number of features. It is in the class of unsupervised training algorithms used in wide range of neuromorphic algorithms such as STDP. For a detailed description of the hardware implementation of the algorithm and resultant behaviors see [97].

When the camera detects a new event, a 13 $\times$ 13-pixel region of the surface around it is converted to a temporally coded spatio-temporal spike pattern. This value to time encoding method was originally used in [98]. The normalized real-valued intensity of the surface is first rescaled from 0–1 to 0–255 and then mapped to an 8-bit unsigned integer. This 8-bit encoding of the surface allows for potential hardware implementation of the SKAN kernels, without needing floating-point operations. This integer representation of the local surface region is then encoded into spike delays forming a spatio-temporal spike pattern. The resultant pattern is then used as the input to a 25-neuron network. The neurons were trained 10 times independently using half the dataset consisting of 50 recordings from each plane type augmented by the left-right flipped version of these recordings. After training, the weight and threshold adaptation mechanisms in the neurons were disabled. Independent training of SKAN on randomly selected sections of the dataset consistently resulted in similar spatio-temporal features being learnt. Given the shallow network structure, feature sizes significantly affect how much of an object is seen by any of the feature detecting neurons. For this reason, a range of feature sizes were investigated. The panels in Figure 3.8 show the resulting feature set from two independent trials at different network sizes to demonstrate this. As the comparison of the trained feature sets shows the same consistent features were learnt at each network size, with the features coding for the leading edge of the airplane nose cones and wings dominating the feature sets. In addition, variants of a solitary noise spike produced often by the ATIS camera are represented as noise features appearing in the top left of Figure 3.8(b),(c) and (d). This consistency was also observed over training epochs of the individual trials. As the number of neurons is increased some of the neurons no longer code for the same features, as can be best seen in the bottom right neurons of Figure 3.8(d). Note also the increasing number of variants of the “noise feature” as the network size is increased. These variants of the “noise feature” encode weak traces of features which are too weak to show in the full-color scale.

Of the many network sizes shown in Figure 3.8 the $N = 25$ neuron network was chosen for the investigation of the other parameters in the system. In section Feature Extractor Size and Number, we return to investigate the effect of network and feature sizes in greater detail. Following feature extraction, and with learning disabled, the neurons compete to recognize incoming spatio-temporal event patterns generated from the same 13×13-pixel region of the surface following each new event with the spike output of the winning neuron representing a feature event. These feature events were then stored onto 25 separate feature time surface or feature index surfaces, which were generated identically to the event surfaces described in section Time-Surface vs. Index Surfaces using the same decay method and decay factor.

3.2.5 Spatial Pooling of Feature Surfaces

In order to reduce the required processing and speed up simulation, the subsystems following the feature surfaces were operated in a frame-based manner such that at periodic intervals the estimated target region from each feature surface was sampled to generate feature frames. The interval used for sampling was the same as the time surface decay constant $\tau_e = 3$ ms. The frame generation was time-based for both the time and index surfaces so as not to bias the

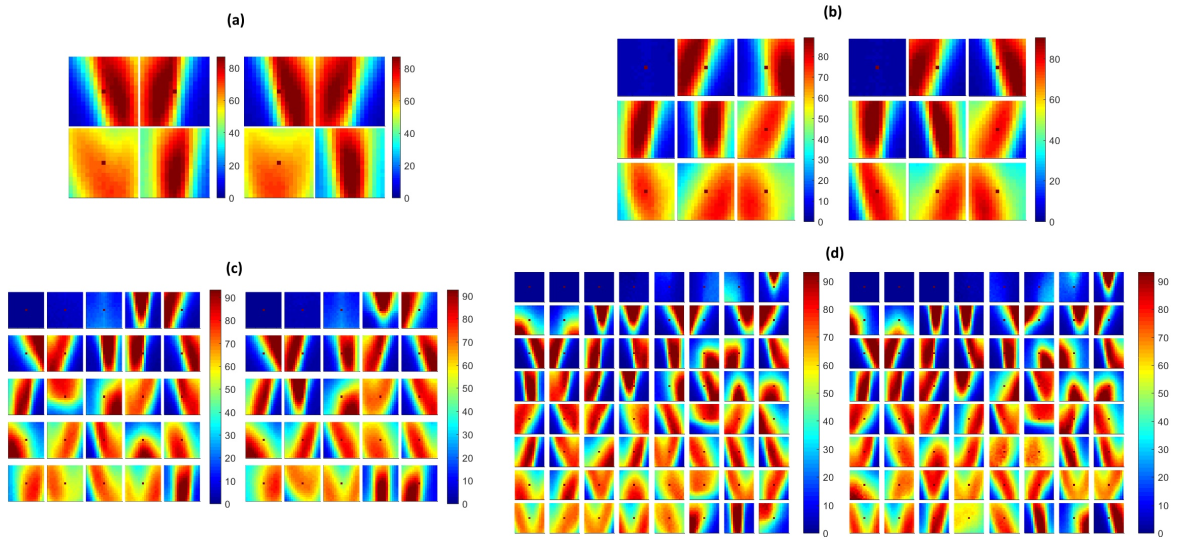


FIGURE 3.8: **Consistency of feature generation at multiple network scales.** Panels (a) to (d) show 4, 9, 25 and 64 spatio-temporal features, respectively, extracted from the ATIS airplane drop dataset. Each panel show results from two independent trials. To allow for a visual comparison of the two feature sets, the features from the first trial have been ordered based on the sum of the squares of the weight of each pixel in each feature. The features of the second trial were then sorted based on cosine distance to the first feature set. Only the feature-set obtained from two instances of the time-based, exponentially decaying surface is shown above for brevity. The features resulting from the other kernels resulted in qualitatively similar features dominated by wing edge, nose cone tail features as well as features coding for noise.

comparison. To reduce the input size to the classifier, spatial pooling of the feature surfaces was performed. To perform this spatial pooling, the estimated object boundary region was summed along the rows and columns, generating two one dimensional feature vectors, one for the rows and one for the columns. The length of these vectors would vary at each feature frame depending on the size of the estimated target region. Thus, in a network with N neurons for each feature a target region of size R rows and C columns would generate two one-dimensional vectors (of length R and C, respectively) resulting from the summation of the image region across rows and columns for each of N surfaces. In order to provide the classifier with a uniform input layer size, the varied length feature vectors R and C need to be resampled to a uniform length. This was done using linear interpolation and the uniform vector length chosen was 72 for each of the row and column vectors. This vector size (72) were multiplied by the number of pooling dimensions (2), and the number of features (25), produced a 3,600-input layer for the classifier.

3.2.6 Classification

The choice of a back-end classifier used to map feature outputs to classes can play a critical role in the performance of a convolutional feature extraction layer or network. Well-regularized high-capacity classifiers with internal non-linearities can provide significant improvement in performance over and above the underlying feature extractors used. In many proposed event-based recognition systems, only a single type of classifier is tested and often only a single instance of such a classifier (the best performing configuration) is reported. While this approach encourages greater attention to the presented work, it can also overstate the performance of the overall system, due to fine-tuning. What's more, the use of well-optimized powerful classifiers without concurrently testing simple linear classifiers obscures the role of the event-based feature extractors in the system performance. Here, we propose a dual classifier testing protocol, which ideally should be applied before and after each stage of processing, to provide insights into the effectiveness of the elements under test. For the baseline test, a simple linear classifier is used to measure how linearly separable the underlying data is before and after processing. In addition to this baseline classifier, we utilize a large capacity ELM, which, by virtue of the large number of random hidden layer neurons, is likely to project the non-linearities of the dataset into a linearly separable higher dimensional feature space. In addition, the lack of feature learning in the ELM allows a reasonable unbiased estimate of the residual non-linearity in data. This framework of testing provides significant insights, as detailed in the results section, which would not be revealed if only the results from the best performing classifier were reported.

To evaluate the performance of the system, two measures of recognition accuracy were considered: per-frame accuracy and per-drop accuracy. For the per-frame measure, the feature vectors described in Section 3.2.5 were presented to the classifier at periodic time intervals τ_e. At each frame, the class with the largest output was selected as the winner for that frame.

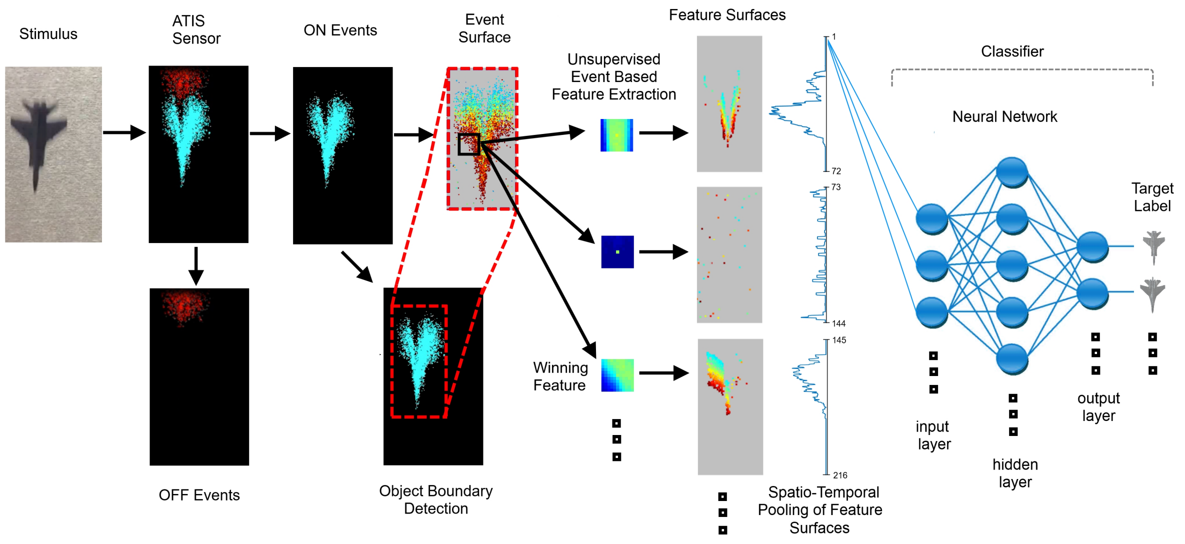


FIGURE 3.9: **Block diagram of the full event-based detection, feature extraction and recognition system.** The target is sensed by the sensor and the generated ON events are processed using a time or index surface. Each event triggers a comparison of a local patch around the event with a set of features or neurons. The winning neuron outputs an event which in turn is placed on a feature surface. The feature surfaces are summed across the rows and columns and presented to the back end classifier. The classifier is here depicted as a network with a hidden layer but we also use a linear classifier. Note that in the feature surface pooling stage only the vector summing the feature surfaces across columns is shown, with the second vector showing the summation across rows omitted for clarity.

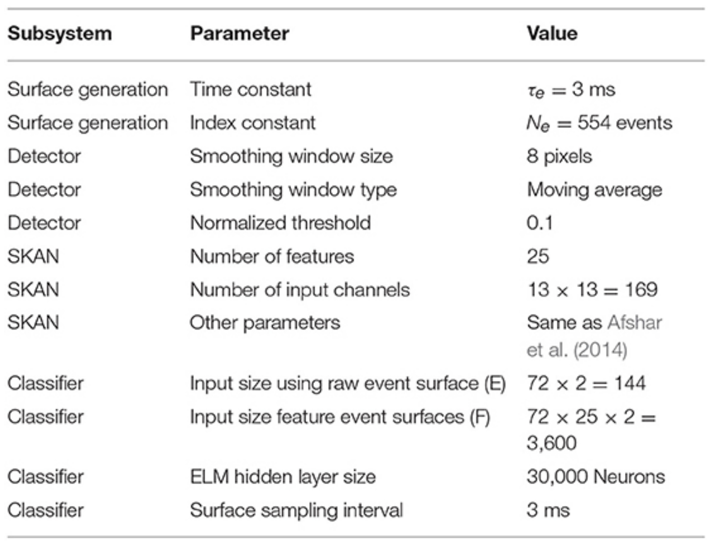
For the per-drop accuracy measure, the class with the highest number of per-frame during the entire recording was selected.

A linear classifier and an ELM classifier with a hidden layer size of 30,000 neurons were trained using the time-based binning method to achieve the highest per-frame recognition accuracy.The resultant end-to-end system is shown in Figure 3.9.

3.2.7 Parameter Selection

In order to fairly evaluate the relative performance in terms of recognition accuracy resulting from different decay kernels, surface decay methods, feature extractor numbers and their receptive field sizes, a large number of free system parameters must first be selected. These parameters, listed in Table 3.1, are used to implement event and feature surface generation,

TABLE 3.1: **Free parameters used in the system.**



Subsystem	Parameter	Value
Surface generation	Time constant	$\tau_e = 3$ ms
Surface generation	Index constant	$N_e = 554$ events
Detector	Smoothing window size	8 pixels
Detector	Smoothing window type	Moving average
Detector	Normalized threshold	0.1
SKAN	Number of features	25
SKAN	Number of input channels	$13 \times 13 = 169$
SKAN	Other parameters	Same as Afshar et al. (2014)
Classifier	Input size using raw event surface (E)	$72 \times 2 = 144$
Classifier	Input size feature event surfaces (F)	$72 \times 25 \times 2 = 3{,}600$
Classifier	ELM hidden layer size	30,000 Neurons
Classifier	Surface sampling interval	3 ms

frame generation, object detection, feature extraction, spatial pooling, regularization and classification. In order to ensure that the selected parameters do not advantage the index-surfaces or the feature extraction methods that are the focus of this work, all subsystem parameters would need to be evaluated in terms of their combined effect on the performance of each method under testing. However, this represents a prohibitively large search space to explore in a brute force fashion. Instead, to remove possible parameter selection bias in favor of the proposed methods, the approach taken in this work involved optimization of all parameters to achieve the highest recognition accuracy on what may be considered the null hypothesis: that simple time-based binning kernels used on raw input events outperform other kernels, decay methods and feature extractors. To this end, the parameters in Table 3.1 and all algorithm design choices were selected via a manual heuristic search for optimal recognition performance using the time-based binning surface $\boldsymbol{BTS}_i$ whose spatially pooled output was fed directly to the classifier without the use of feature extractors. The classifiers were then selected for optimal performance on the output data generated by the selected parameters. Once optimized in this way for the "null" hypothesis, these same parameters and network structures were used for all other tests, ensuring that recognition results were biased in favor of the simple time-based binning approach and not those proposed in this work.

3.3 Results

3.3.1 Results on the Full Dataset

The per-frame recognition results on the full dataset are shown in Figure 3.10. For each of the panels, the same performance pattern is observed: when operating on raw event surfaces as inputs, the large capacity ELM (ELM-E) significantly outperforms the linear classifier (L-E). This demonstrates the non-linearity of the classification boundaries in this case. In comparison, when feature surfaces are used as inputs, the improvement margin gained by the ELM (ELM-F) is small relative to the linear classifier (L-F) suggesting that the output of the 25 feature extractors is significantly more linearly separable, with less room for improvement through further non-linear expansion. Also noteworthy is that the linear classifier operating on feature surfaces (L-F) outperforms the ELM operating on the raw event surfaces (ELM-E) for all surface generation methods. This shows that the application of a small number of trained local feature extractors is more effective than using a much larger globally connected network of neurons with random input weights. The ratio of errors between the ELM and the linear classifier indicated at the bottom of each panel quantifies this reduction in error for each case.

Comparing the results across the panels for the linear classifier operating on events (L-E), the exponentially decaying surfaces outperform linearly decaying surfaces by a margin of 1.75% for the index surfaces and 0.24% for the time-surfaces. In turn, the linearly decaying surfaces outperform the binning method by 3.06% and 1.36% for the index surfaces and time surfaces, respectively. For the case of the linear classifier operating on feature surfaces (L-F), the exponentially decaying surfaces outperform linearly decaying surfaces by a margin of 0.57% for the index surfaces and 0.22% for the time-surfaces, and in turn, the linearly decaying surfaces outperform the binning method by 3.07% and 1.91% for the index surfaces and time surfaces, respectively. Also, consistently, the improvement of exponential kernels over linear kernels is not as significant as their margin with the binning method.

It is worth noting that, when the ELM is chosen as the back-end classifier, the margin in performance improvement obtained from feature extraction is reduced. This result is expected since the randomly situated hidden layer neurons of the ELM have a greater chance of improving the linear separability of segments of the dataset if such segments are not already linearly separable due to processing in the preceding layer. This effect of obscuring the performance of other subsystems is not limited to the ELM. A similar effect would be expected with any other classifier performing non-linear expansion. This underlines the need to include results from a simple linear classifier when comparing alternative systems. Also worth noting is that for the preceding results (features outperforming raw events, and exponential and linear kernels outperforming binning) all system parameters were optimized for the time-based binning method. These results, therefore, confirm the suitability of exponential kernels for time and index-surface generation. This conclusion is also supported by results in Akolkar et al. (2015), where the information from the visual scene is found to rapidly rise within a small initial temporal window, but thereafter fall gradually with increasing window size, as is best described by an exponentially decaying kernel. By weighing events in an approximately compensatory manner to their information content as described in [99], the exponentially decaying kernel results in the highest information content for the classifier. Another observation from Figure 3.10 is that all time-based decay methods outperform the index-based decay methods by approximately 1% on the full dataset with the largest performance disparity observed between the index-based binning method $\boldsymbol{BIS}_i$ and the time-based binning method $\boldsymbol{BTS}_i$. This would be expected since the latter method was used during all parameter optimizations and would be most advantaged by the selected parameters. Based on the results shown in Figure 3.10 we narrow further investigations by selecting linear classifiers L-E and L-F and focus on exponentially decaying surfaces $\boldsymbol{LTS}_i$ and $\boldsymbol{LTS}_i$.

3.3.2 Frame Balanced Dataset

In order to generate a balanced dataset, an equal number of frames from each recording was selected. In this way, the total number of presentations to the classifier for each class was equalized. Figure 3.11 shows that 1, 2, 4, 8, 16 and 32 frames were sampled from each of the airplane recordings and presented to the linear classifier operating on events surfaces L-E and feature surfaces L-F for each of the $\boldsymbol{LTS}_i$ and $\boldsymbol{LTS}_i$ surfaces.

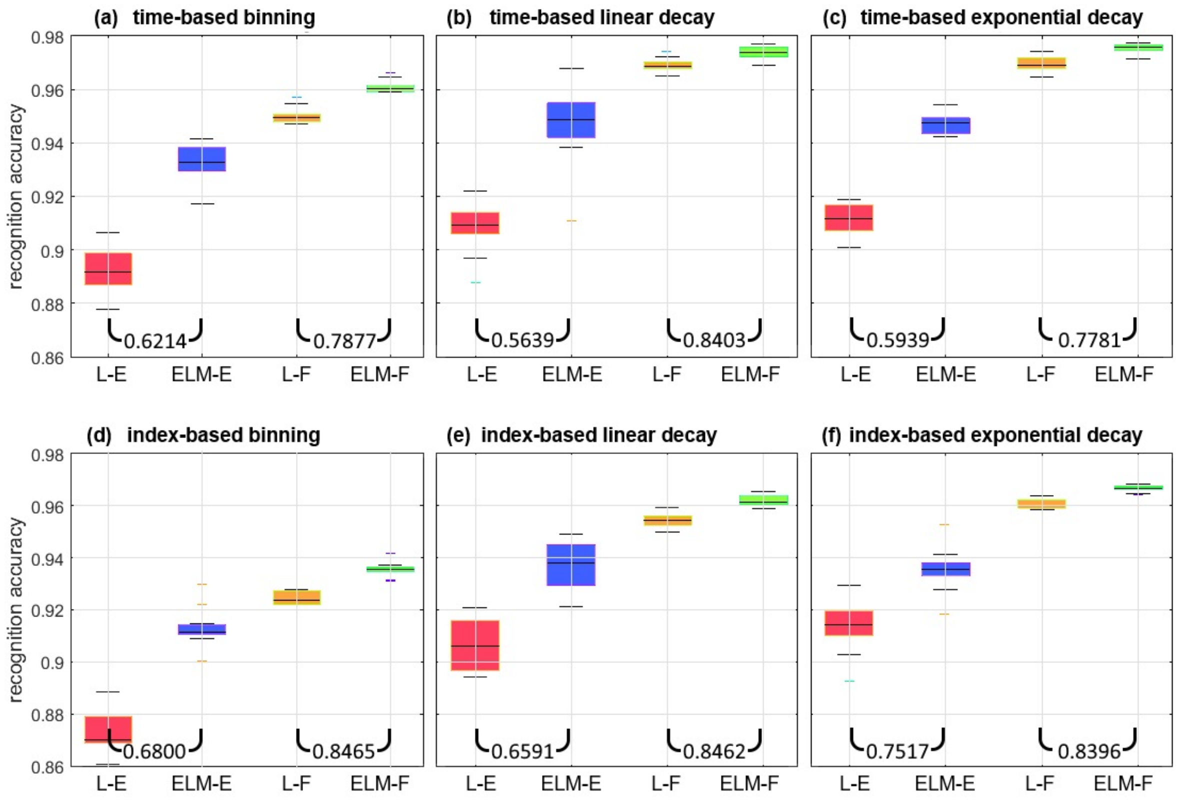


FIGURE 3.10: **Per-frame recognition accuracy on the full dataset over n = 20 independent trials.** Each panel shows results from four network arrangements. The box plots show the median (center-line) maximum and minimum values at the top and bottom respectively. The height of the box captures the 25th to 75th percentile. In (L-E), and (ELM-E) the linear classifier and the 30,000 hidden layer ELM operate on inputs from raw event surfaces. In (L-F), and (ELM-F) the same classifiers use 25 feature surfaces as inputs. Each panel shows results for a different surface generation method: The top three panels show time-based methods using (a) binning, (b) linearly decaying and (c) exponentially decaying surfaces. The bottom three panels show corresponding index-based binning (d), linearly decaying (e) and exponentially decaying surfaces (f). The two ratios at the bottom of each panel indicate the median error ratio of the ELM over the linear classifier. The black horizontal lines above and below the box plots indicate the minimum and maximum accuracy values. The shorter lines indicate outliers.

As Figure 3.11 shows, both the per-frame and per-drop accuracy increase as a function of the number of frames used during training. Additionally, a sharper increase and a higher final accuracy is observed for the per-drop accuracy measure as would be expected since the per-drop measure is analogous to a max-pooling operation which benefits from increased pool size. The relative performance margin of the network using feature surfaces over raw

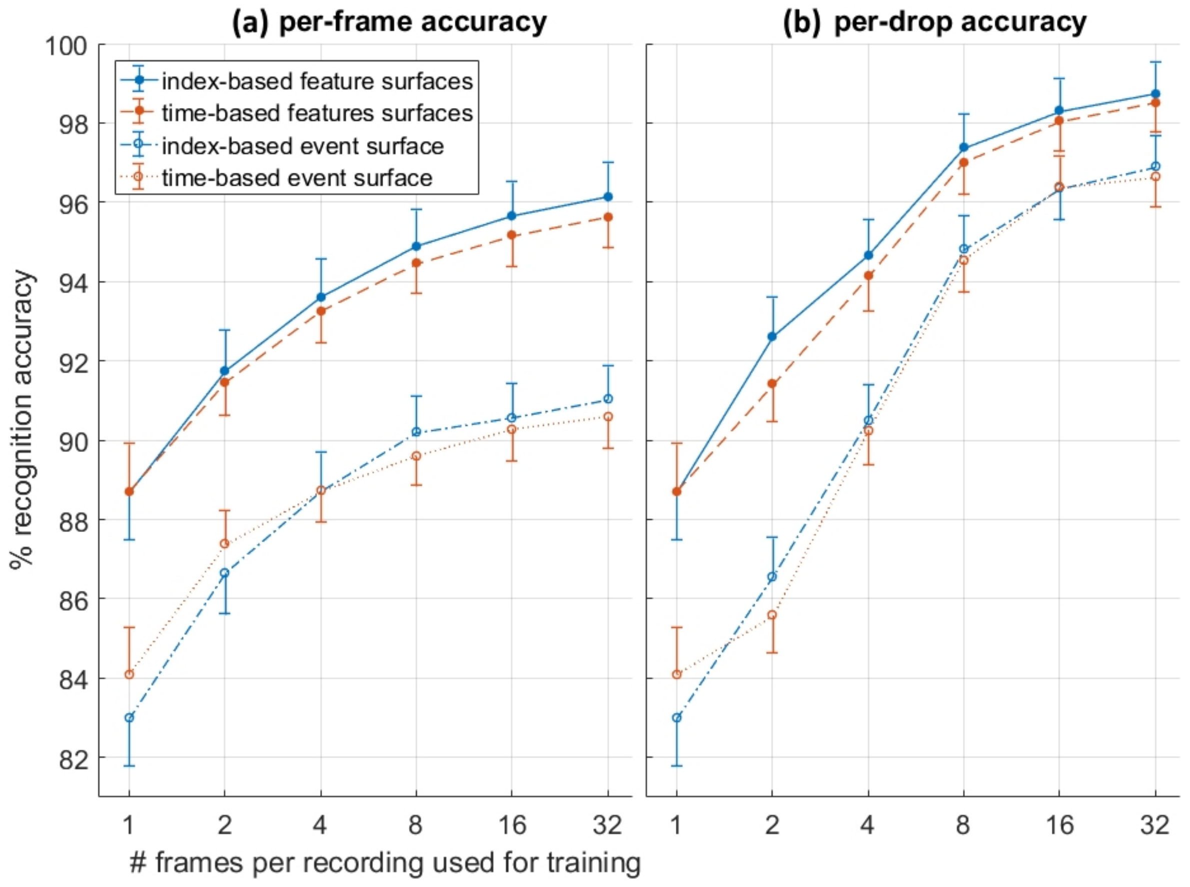


FIGURE 3.11: **Comparison of per-frame and per-drop and recognition accuracy as a function of the number of randomly selected frames used during training from each recording.** The index-based $\boldsymbol{LTS}_i$ surface and time-based $\boldsymbol{LTS}_i$ surfaces are compared. Results shown are over $n = 20$ trials. A linear classifier was used in all test.

event surfaces is reduced in the per-drop measure, as more information is accumulated over a recording, reducing error. The highest number of random frames used per recording was 32, as this was approximately equal to the total number of frames in the shortest recording as shown in Figure 3.2(b). Table 3.2 details the accuracy results for this balanced dataset while Figure 3.12 shows misclassified recordings for one instance of the highest performing network using index-based decaying feature surfaces and a linear classifier, illustrating that some drops are almost impossible to classify correctly.

Interestingly, in contrast to the full unbalanced dataset results detailed in section Results on the Full Dataset, the per-frame balanced results in Figure 3.11 and Table 3.2 show little significant difference in accuracy between the index-based and time-based surfaces for either

TABLE 3.2: **Selected Linear Classifier Results.** Per-frame and Per-drop accuracy results on the frame balanced dataset for four selected systems: Linear classifier operating on events surfaces (L-E) and feature surfaces (L-F) for each of the $\boldsymbol{LTS}_i$ and $\boldsymbol{LTS}_i$ surfaces.

	Per-frame (%)	**Per-drop (%)**
Time-based Event surface	90.60 +/−1.02	96.64 +/−1.47
Index-based Event surface	91.03 +/−0.89	96.90 +/−1.34
Time-based Feature surfaces	95.64 +/−0.79	98.52 +/−0.75
Index-based feature surfaces	96.15 +/−0.84	98.75 +/−0.78

Number of trials used is 20.

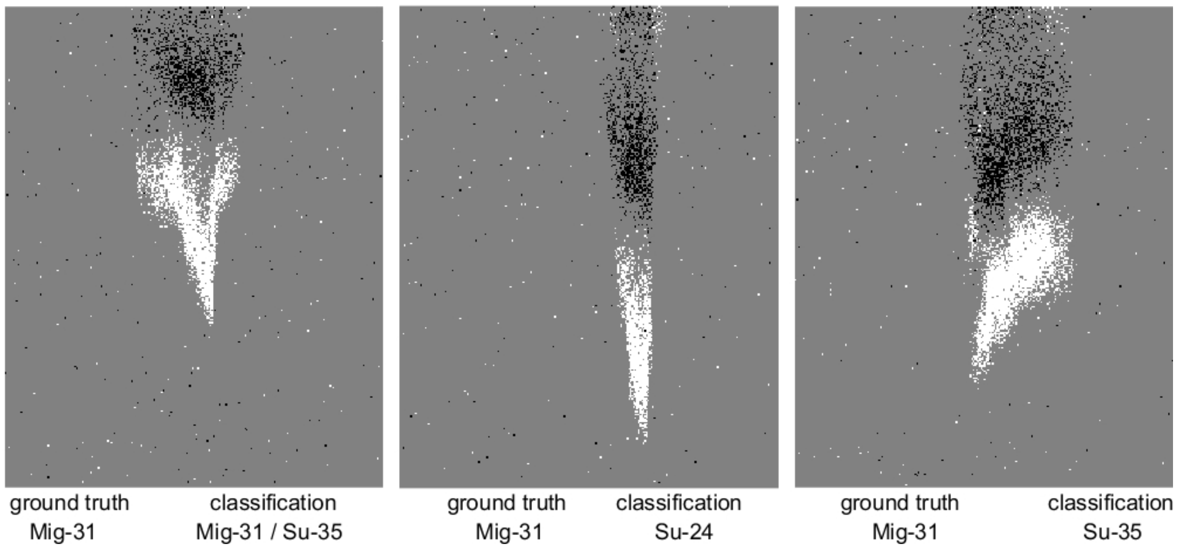


FIGURE 3.12: **The three drops misclassified by an instance of a linear classifier using 25 exponentially decaying index-based feature surfaces.** Captured frames show airplanes at mid-point (in time) of recording.

the per-frame or per-drop measures, suggesting that the observed slight advantages in accuracy on the full dataset may be due to the use of time-based surfaces during parameter selection.

3.3.3 Velocity Segregated Dataset

As outlined in section 3.2.3, the apparent velocity invariance property of index surfaces motivates a test using a modified dataset which is split in terms of target velocity. Thus, in order to compare index-based and time-based surfaces in terms of target velocity invariance, the recordings were divided into 200 “slow” and 200 “fast” recordings based on the estimated

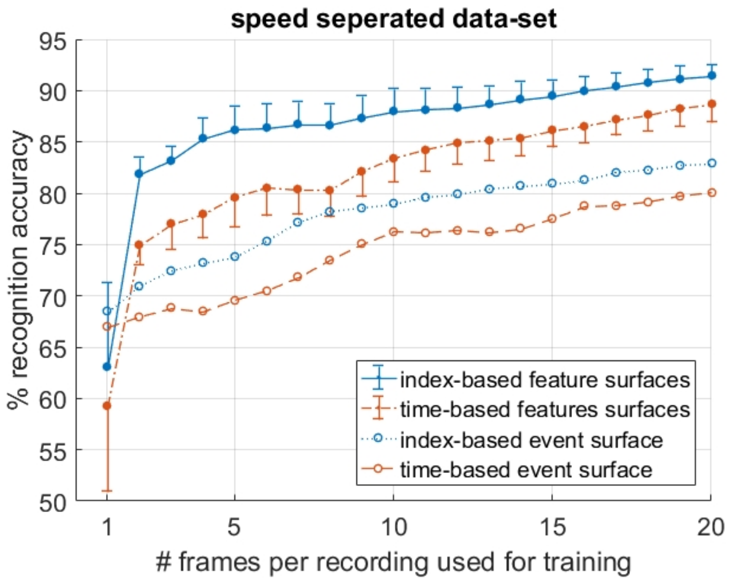


FIGURE 3.13: **Mean and standard deviation per-frame accuracy on a speed segregated dataset.** The results demonstrate superior performance of index-based surfaces in the presence of variance in target velocity.

vertical airplane velocity at the midpoint (in time) of each recording. Since the airplanes speed up during the fall, the system was trained on the n-first (slowest) frames of the slow recordings and tested on the n-last (fastest) frames of the fast recordings. In this way, by varying the number of n frames, datasets with different degrees of velocity segregation could be tested. The resulting recognition accuracies in Figure 3.13 demonstrate that with increasing number of frames, and thus decreasing velocity segregation in the data, the recognition accuracy of all systems rises. Figure 3.13 further shows that although training on a speed segregated dataset significantly reduces accuracies for all systems in comparison to training using a randomly sampled dataset (such as shown in Figure 3.11), the decline is significantly larger for time-based decaying surfaces. This difference demonstrates the relative robustness of index-based decay surfaces to variance in velocity and their utility in applications where the full range of potential target velocities to be encountered during testing is not available in the training data.

Therefore, given the results in the previous section, it can be concluded that, at the local scale, with a single target in the field of view, systems using index-based decay surfaces tend to match equivalent systems using time-based decay surfaces, when presented with an adequately wide range of velocities in the training data, since their advantage of velocity invariance is effectively neutralized. But when the available range of velocity distributions for training is incomplete, index-based decay surfaces tend to produce more robust performance. Given this finding, and in order to limit the scope in the next section, we narrow our focus exclusively on index-based surfaces and investigate the effect of different feature extraction networks and their effect on recognition accuracy. This is also supported by findings in [84], where a small superiority was found when using fixed event windows over time windows. However, those tests were performed using a randomly sampled training set, likely containing data with velocity distributions that were similar to the test set. As such their results are similar to the full dataset results examined in section Frame Balanced Dataset of this work, which only showed a slight improvement due to the velocity variance available in the training dataset. In this work, by additionally testing the algorithms using a range of velocity segregated datasets, the robustness of the index surface method is more completely investigated.

3.3.4 The Decay Constants

An important element of any event-based surface is the value of its decay constant. In this work the value of decay constants, τ_e = 3 ms and N_e = 554 events were effectively chosen arbitrarily. This raises an important question about the optimality of the chosen decay constants and the robustness of the generated features and recognition accuracy to different values of these decay constants. A closely related question, which applies only to index surfaces, is whether targets which generate more or fewer events, e.g., due to different object size or contrast, could still be learnt and recognized with the decay constants chosen. To investigate these questions a wide range of decay constants across six orders of magnitudes were tested on a frame balanced randomized training and testing dataset. The resulting recognition accuracies and selected feature sets are shown in Figure 3.14. The results show a similar pattern for time and index surfaces with little significant difference in classification accuracy. At the extreme decay rate of 10 events and 54 μs the systems perform little better than chance since virtually all event information is decayed away before it can be extracted. This leaves all the features coding for variants of the noise feature. As the decay constant increases by two orders of magnitude, coherent features begin to emerge coinciding with a rapid increase in recognition accuracy. At this event rate, there are still multiple features coding for a single noise spike. Index decay constants of between three and four orders of magnitude of events correspond with a plateau in recognition performance. This region coincides with the range where the noise feature is only represented by one or two neurons with all remaining neurons coding for complex features. After four orders of magnitude increase in the decay constant, the accuracy begins to decline slightly. In this region, the noise features begin to be represented once more but this time with a highly activated background which is a direct result of the much slower decay rate.

As Figure 3.14 illustrates, when sweeping the decay constant, the number of variants of the noise feature in the network roughly correlates to the feature extraction performance of the network. The feature set with the fewest representations of the noise feature (ideally only one) performs the best. This is expected since the noise feature is unlikely to be correlated to any particular class of object and the frequency of its representation in a feature-set reduces the efficiency of that feature-set, leaving fewer neurons to represent classification relevant feature information. Figure 14 also shows a wide central region of stable performance that is robust to the choice of τ_e and N_e. The results also show that overestimating the optimal value of the decay constant is less detrimental than underestimating with significantly less reduction in classification accuracy.

3.3.5 Feature Extractor Size and Number

In order to characterize the effectiveness of the feature extraction subsystem in an unbiased manner, a range of feature sizes and a number of feature extractors were investigated and assessed in terms of the resultant recognition accuracy. In addition, for each point on the feature size-feature number space, the results of the learning algorithm described in section Event-Based Feature Extraction was compared to those of equivalent sized networks using random feature sets. The mean accuracy results in Figure 3.15 (top panels) demonstrate that learnt features outperform random features at every scale while exhibiting slightly lower variance in accuracy (bottom panels).

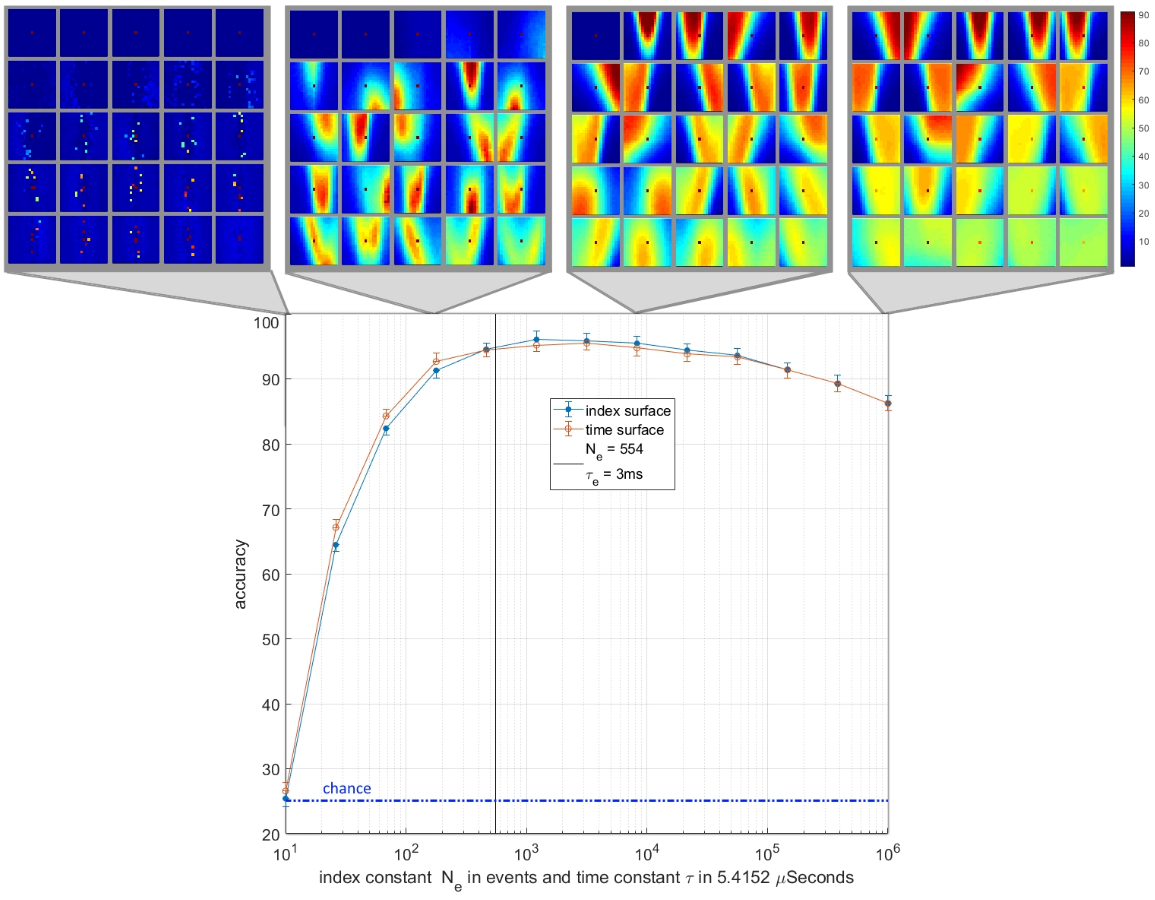


FIGURE 3.14: **Classification accuracy and typical feature sets as a function of the decay constants for time and index surfaces.** The lower panel shows accuracy plotted as a function of the index decay constant N_e on a logarithmic scale. The time surface results are plotted on the same logarithmic scale where a 5.4152 μs/event conversion rate is used to align the results. This conversion rate is based on the average event rate over the entire dataset. The vertical solid line at N_e = 554 and τ_e = 3 ms (τ_e = 554 $\times$ 5.4152 μs) indicates the value of the index and time decay constants used in rest of the work. The horizontal dotted line indicates chance accuracy. All tests were performed over n = 20 independent feature extraction trials. The feature sets above the panel show instances of the feature sets for four points on the decay constant axis.

In addition, while the results from the random features suggest a slight trend toward increased accuracy as a function of both feature numbers and feature size, the learnt feature results clearly show that the larger feature sizes (17 $\times$ 17 and 13 $\times$ 13) generate higher accuracy with increasing number of features, while the smallest feature sizes (3 $\times$ 3 and 5 $\times$ 5) exhibits a weak downward trend with the number of features. When the feature size is small, only a few

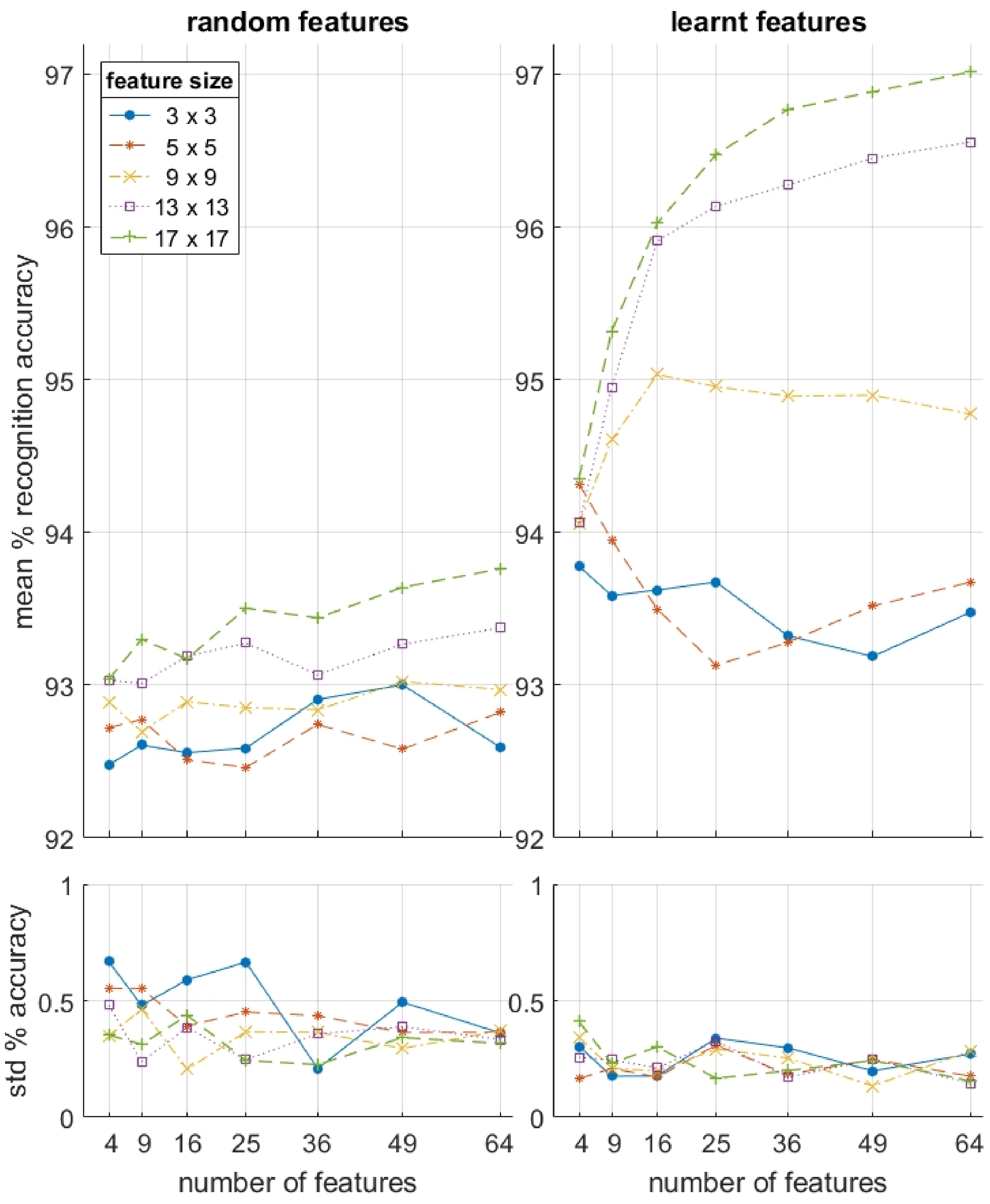


FIGURE 3.15: **Per-frame accuracy on the full dataset as a function of feature size and number of features used in the feature extraction layer for both learnt and random features.** n = 10 independent feature sets with 10 cross validating classifications per feature set. Note that the baseline linear accuracy using the raw event surface with no feature extraction layer was 91.38 +/-0.81% as shown in Table 3.2.

distinct combinations exist. Therefore, when and a large number of them are trained, several features will be very similar, resulting in near-identical input generating very different input to the classifier. This reduction in accuracy resulting from the addition of more redundant features is due to the OR operation which must be performed by the back-end classifier. This insight demonstrates that convolutional features layers can, if poorly configured, “over-fit” the data by representing overly specific variants of the same pattern. This effect only becomes apparent with the combined use of a large number of features, small feature sizes and relatively small datasets. But this might be an issue in future applications of event-based convolutional networks, where resource efficiency of a hardware implementation may allow a very large number of features in a layer to be trained (especially in the first layer) while the level of independent features in the recorded data may be limited.

We can also note that for both the random and learnt feature sets, the feature size has little effect on accuracy when the number of features becomes very small. This is because there is very little additional discriminatory information that can be captured by the larger sized features when a wide range of unrelated, heterogeneous spatio-temporal patterns become effectively averaged together to generate the (too) few features used in the network. Thus, local spatial complexity of observed data determines optimal feature size and feature number relationships, which, if not considered during hardware implementation, can result in inappropriately scaled network architectures and effectively wasting hardware resources.

3.4 Discussion and Future Work

As detailed in this work, the selection and optimization of system parameters were performed on a "null" hypothesis which involved time-based binning surfaces acting on raw event streams without features. The results have shown that the improvements in accuracy resulting from exponential surfaces, index-based decay and feature extraction more than compensated for this biased parameter selection procedure. It is, however, worth noting that if the system parameters were optimized instead on the elements proposed in the work i.e. features operating on index-based exponentially decaying surfaces, then the improvements over the "null" hypothesis would almost certainly be even larger than the ones presented in this work.

While binning methods examined in this work were shown to perform less well than linearly decaying surfaces and exponentially decaying surfaces, the significantly simpler implementation of the binning method allows for much more efficient implementations of event surfaces in neuromorphic hardware. In a similar fashion, the selection of feature sizes and the number of features implemented at any layer of a multi-layer event-based network generates trade-offs between hardware resources and performance. In this context, the network and feature size investigations presented here provide guidelines for such network designs.

The four-class dataset presented allows reasonably accurate classification using a single layer of feature extraction in combination with a linear classifier; the task can be made increasingly difficult by increasing the number of classes in the dataset. In such a case, the output of the feature extraction layer would retain significantly greater residual non-linearity. This would increase the performance gap between the linear classifier and the large ELM. Conversely, adding additional feature extraction layers will work in the opposite direction, producing output that is more and more linearly separable and thus reducing the performance gap between the linear classifier and ELM.

The presented recordings in the dataset were varied to cover a wide range of target speeds. As a result, any random splitting of training and testing data provided an overlapping range of target speeds in both sets. This overlap removed any advantage of index-based decaying surfaces that provide robustness to target velocity. However, in many applications, such as the SSA applications discussed in Chapter 2, the range of velocities in the training set is limited so that features trained on this limited set of target velocities must generalized to a wide range of as yet unobserved velocity profiles. In this work, such a condition was simulated by iteratively segregating the data based on speed to highlight the utility of the index-based decay method.

3.5 Conclusion

In this work, we investigated in detail an event-based feature extraction layer. In order to rigorously investigate the effects of different kernels, decaying methods, classifiers and feature sizes and numbers, we limited the exploration to a single layer network. Yet the design of deeper networks can be informed by these single layer results. Using a dataset featuring a range of target shapes, scales, orientations and velocities, it was observed that exponentially decaying kernels outperform other kernels, and that index-based decaying surfaces perform equally as well as time-based decaying surfaces, when robustness to target speed is not required, and outperform them when it is required. We also showed the clear superiority of learnt features over random features and showed that the largest networks of neurons with the largest receptive fields using the most complex kernels outperform all other configurations.

CHAPTER 4

Investigation of Feature Extraction using Adaptive Selection Thresholds

Chapter Summary

Unsupervised feature extraction algorithms form one of the most important building blocks in machine learning systems. These algorithms are often adapted to the event-based domain to perform online learning in neuromorphic hardware. However, not designed for the purpose, such algorithms typically require significant simplification during implementation to meet hardware constraints, creating trade-offs with performance. Furthermore, conventional feature extraction algorithms are not designed to generate useful intermediary signals which are valuable only in the context of neuromorphic hardware limitations. Here we investigate an event-based feature extraction algorithm with a focus on these issues. The algorithm operates via simple adaptive selection thresholds which allow a simpler implementation of network homeostasis than previous works by trading off a small amount of information loss in the form of missed events that fall outside the selection thresholds. The behavior of the selection thresholds and the output of the network as a whole are shown to provide uniquely useful signals indicating network weight convergence without the need to access network weights. A novel heuristic method for network size selection is proposed which makes use of noise events and their feature representations. The use of selection thresholds is shown to produce network activation patterns that predict classification accuracy allowing rapid evaluation and optimization of system parameters without the need to run back-end classifiers. The feature extraction method is tested on both the N-MNIST benchmarking dataset and a less controlled dataset of airplanes passing through the field of view. Multiple configurations with different classifiers are tested with the results quantifying the performance gains at each processing stage.

4.1 Introduction

Feature detection is a fundamental building block required for a wide range of computer vision tasks. These tasks rely on computationally efficient algorithms capable of detecting features in a reliable, repeatable, and robust manner.

Conventional feature extraction is an active and well-researched field of study and has produced sophisticated and robust algorithms. Event-based feature extraction poses a different and perhaps more challenging task. The output of an event-based camera constitutes neither a frame, as in traditional computer vision, nor a stream of frames, as in conventional video. As a result, the majority of existing feature detectors are a poor fit for event-based vision data and require the events to be converted into standard image frames before processing. In contrast, this work explores the use of event-based features using an unsupervised and data-driven approach. Originating from the concepts underpinning the Synaptic Kernel Adaptation Network (SKAN) [97][94], this method, called the Feature Extraction with Adaptive Selection Thresholds (FEAST) algorithm, makes use of neurons or features with individually adaptive selection thresholds that are iteratively updated using a competitive control strategy. These adaptive neurons act as feature extractors that learn data-specific features in an online, event-based, and unsupervised manner. Adaptive selection thresholds provide a simple way of maintaining homeostasis between the activation patterns of a large number of neurons without the need to store or share information about previous neuronal activity or the internal parameters of the individual neurons. This simplicity also enables efficient implementation of the algorithm in neuromorphic hardware.

4.1.1 Feature Extraction in Neuromorphic Systems

Feature detection is commonly defined as the process of identifying and describing sections of an image representation for the purposes of identification, tracking, or classification. When dealing with conventional cameras, these image representations are often frames of illumination intensity, containing either monochrome or color information for each pixel.

Feature detection in the context of event-based cameras operates on a fundamentally different representation of the visual scene in which the encoding of information includes a pixel-level temporal component not present in conventional frame-based data.

Event-based vision systems, therefore, need a class of feature detectors that exploit the event-based and activity-driven nature of neuromorphic vision systems. Tasks such as tracking and object recognition still require the identification and matching of local visual features, but these features must represent commonly observed spatio-temporal patterns of events instead of static images. As with conventional feature detection, the most desirable property of a feature is still its ability to be uniquely distinguished in feature space [100].

The field of neuromorphic vision has seen a significant increase in interest over the past few years, resulting in a number of innovative approaches to the task of feature detection and extraction. Features based on corner detection, such as the Harris corner detector [101] and cortex-like Gabor filters [102], are examples of an explicit feature detection method commonly used in conventional computer vision. This approach has been adapted for event-based sensors, with notable examples including an event-based implementation of the Harris Corner detector [103], and a novel corner detection method based on finding the intersections of planes fitted to the event stream from the cameras [104]. Related to corner detection is the process of edge detection, and this class of algorithms have also been implemented in an event-based manner. Examples include the Canny edge detector presented in [105] and the event-based line segment detector presented in [106].

Whereas some feature detection methods have sought to make use of a combination of event-based and frame-based approaches [59], this work restricts itself to operating only on the output of the change detection circuity from the event-based camera. The mechanism for measuring absolute illumination in an event-based sensor varies from device to device, whereas the change detection produces compatible output across all current event-based vision sensors. By restricting the algorithms to only the change detection events, this maximizes the versatility and applicability of the algorithms by allowing them to be compatible with most existing event-based vision devices, such as the DAVIS event-based cameras [107].

The HFIRST algorithm [78] is an example of a multi-layer network in which appropriate features are learned directly from the event-based data. The algorithm is based on the HMAX algorithm [108] and implements an analogous first-to-spike operation in place of the maximum pooling operation from which the algorithm derives its name. Another example of an unsupervised learning method capable of learning spatio-temporal features makes use of recurrent reservoir networks and a winner-take-all approach [80] allowing the network to maintain and preserve the high temporal nature of the events throughout the feature detection system. The Event-Based GASSOM [109] algorithm extends the ASSOM [110] algorithm to the output of an event-based sensor and successfully demonstrated the ability to learn features invariant to fast changes in the input signal. Both methods were tested on event-based datasets similar to, or superseded by, the datasets used to verify the feature extraction method presented in this work.

The Hierarchy of Event-based Time Surface (HOTS) algorithm [81] represents the closest work to the event-based feature detection method described in this work. The HOTS algorithm uses the neuron update learning rule introduced in [111] where neurons are updated in proportion to the cosine distance of their weights to the input, and where the learning rate gradually decays as a function of time. The HOTS algorithm makes use of multiple layers of feature extractors based on unsupervised feature clustering with the output of each layer fanning out and feeding into deeper feature detectors with longer exponential time constants. The algorithm successfully demonstrates the ability to use these feature layers to perform accurate feature classification in an entirely event-based manner.

Here we highlight the significance and benefits of using an adaptive thresholding approach to feature extraction as well as highlighting the novel use of readily accessible network signals for estimating weight convergence and predicting network classification performance. The adaptive thresholding technique presented allows a simple yet robust implementation of network homeostasis. Unlike in the learning method proposed in [111] and used in HOTS, the neurons or features in this work do not need to continuously keep count of their updates. Instead the FEAST algorithm adapts its selection thresholds for incoming events such that the features are constantly being contracted by accepted events and expanded by rejected or missed events. This event-based competition means that the neuron thresholds do not decay exponentially as a function of time but only in response to missed input data which represents information not yet well incorporated into the network.

Furthermore whereas the HOTS learning algorithm updates every neuron in proportion to the cosine distance of the input to the neuron, in FEAST only the winning neuron is updated. Designating a single neuron as the winner of an input event, and only updating this winner, not only performs a max-pooling operation, but also significantly reduces computation and potential hardware costs of such an online learning system.

The major difference between the proposed FEAST algorithm and the SKAN algorithm on which it is based, is the latter operates directly on spatio-temporal spike patterns whereas the former operates on continuous values extracted from time surfaces.

The FEAST algorithm was first used in [112] and validated through classification tasks using the controlled benchmarking datasets. Here, the operation of the algorithm is investigated in more detail and also tested on the noisy Plane Dropping dataset. In addition a subset of the tests performed in [112] are repeated. Here we find slightly better performance relative to random features on the same tests.

By using adaptive selection thresholds, the proposed network provides reliable signals for the detection of network convergence without requiring access to the network weights. The most direct indication of network convergence is a stable steady state in the values of the network weights. In general the change in the network weights does not reach zero but a steady-state around a small value depending on the magnitude of learning rate. In this work three additional signals, the selection thresholds, the missed spike rate, and the variance in the output spike rates, are shown to provide alternative signals for convergence detection during online learning. This useful property is the direct consequence of the dynamics of the adaptive selection thresholds. Such proxy signals for weight convergence are unnecessary when a network is trained offline or if network weights are readily available for inspection and convergence analysis. However, in neuromorphic hardware applications, continuous access to network weights may be limited and costly, due to the large number of weights and limited number of output channels. The proxy signals examined in this work are more accessible and easier to calculate than the change in network weights, enabling more efficient implementations of neuromorphic on-chip learning hardware.

Thus the FEAST algorithm trades a small amount of information loss for a simpler implementation of network homeostasis and robust measures of fitness to data and early proxies for classification accuracy which are shown to predict classification accuracy directly. These novel uses of intermediary signals are particularly important in the context of on-chip event-based neuromorphic systems in real-world online learning applications where hardware resources and opportunities for detailed network investigation are limited.

4.1.2 Feature Extraction via Adaptive Selection Thresholds

As described previously, the output of an event-based camera can be viewed as a continuous stream of events $\boldsymbol{e}_i$, each of which have the form $\boldsymbol{e}_i = [\boldsymbol{x}_i, t_i, p_i]^T \quad i \in \mathbb{N}^+$ where $\boldsymbol{x}_i = [x_i, y_i]$ denotes the location of the pixel generating the event, $p \in [-1, +1]$ indicates the polarity of the change in illumination at the pixel causing the event, and t represents the time at which the event occurred. In a hardware event-based system, the time-stamp would not need to be explicitly stored for each event, as the time would be implicit as the arrival time of the event during processing.

The stream of events e_i can be used to generate a range of time or index surfaces as described in Chapter 3. Here, we use the exponentially decaying time surface with a τ of 10 ms and τ of 3 ms were used for the N-MNIST dataset and Plane dropping dataset respectively.

After generating an event-based time surface, a local Region of Interest (ROI) patch can be extracted for each event e_i such that only a small region of the image requires processing.

Extracting an event ROI from the time surface for an incoming event produces an $R \times R$ ROI I_e containing spatio-temporal information from the neighborhood surrounding the pixel generating the event. In this work the ROI patch size was selected as the neighboring 11×11 pixels. In order to perform further processing on this event, the ROI region is converted into a descriptor in the form of a one dimensional, $1 \times R^2$ vector as follows:

$$d = vec(I) \equiv [I_{1,1}...I_{R,1}, I_{1,2}...I_{R,2}, I_{1,R}...I_{R,R}]^T \tag{4.1}$$

In the next step, this descriptor is normalized through a division by its norm to achieve invariance to temporal scaling.

$$d = \frac{vec(I)}{||vec(I)||} \tag{4.2}$$

The time scale invariance resulting from normalizing the descriptor is an important operation in the weight update step of the FEAST algorithm. By effectively normalizing the descriptor I with respect to time, velocity information is discarded in favor of feature robustness.

With the normalized descriptor d as input, online unsupervised feature extraction can be performed. The FEAST algorithm uses two mechanism for capturing the N most dominant spatio-temporal patterns observed in an event stream. The first mechanism involves adaptation of the feature neuron weights w_n by way of re-orientation toward observed patterns. Here each of the N neuron has exactly $R \times R$ feature weights corresponding to the ROI input size. As with the vectorized descriptor, the neuron weights can be stored and processed as $1 \times R^2$ vectors. The second mechanism involves balancing the rate of learning across the neurons through the use of the neurons' adaptive selection thresholds θ_n which contract and expand to make each neuron more or less selective around their orientation in $\mathbb{R}^{R^2}$ space.

At each event e_i, the normalized descriptor d is compared to each feature w_n via a dot product calculation $\delta_n = d \cdot w_n$. For each neuron δ_n is then compared to its selection threshold θ_n. If for any neuron, the dot product δ_n is larger than the selection threshold θ_n, the neuron with the largest dot product is selected as the winner. In this way, the neuron with the smallest cosine distance to the observed input is the selected as the winner. If on the other hand, for all neurons, the dot product δ_n is larger than the selection threshold θ_n, then no neuron wins and the event is missed by the network since the input descriptor is not within the selection threshold of any neuron.

In order for the network to learn to orient its features toward the observed data, at each event e_i, if a neuron is selected as the winner, a small mixing rate η is used to slightly move the winning neuron toward the descriptor:

$$w_n = (1 - \eta)w_n + \eta d \tag{4.3}$$

where w_n denotes the weights of the winning neuron to the current input ROI descriptor d. As a point of reference, a mixing rate of $\eta = 0.001$ is used in this work. Following this operation, the feature weights are normalized to prevent the biasing of subsequent dot product measurements. In this way, all neurons have an equal magnitude of 1. The normalization of the descriptor serves a similar function across events such that the input descriptor d generated from each event injects an equal amount of information into the network by 4.3. Normalizing each input descriptor prior to mixing prevents larger change due to events generated in regions of higher velocity where the ROIs exhibit larger magnitudes. Without the normalization step, faster-moving segments of the scene, which have higher magnitude, would have a larger effect on the learned feature weights in comparison to slow-moving features. This in turn would result in feature sets that were biased toward faster moving objects at the expense of slower ones. The full algorithm is described by Algorithm 4.1. Note that for the purposes of brevity, the event polarities are not included here.

For $R \times R$ size ROIs the feature weights and descriptors are located on the unit $R \times R$ hypersphere. In this work the On and Off event-based time surfaces are processed separately such that descriptors are extracted from positive valued time surfaces. For this reason the feature weights and descriptors are all placed on the positive quadrant of the unit hypersphere.

In addition to updating the feature weights, the threshold for the winning feature is also increased by $\Delta\theta^+$. This contraction of the selective threshold of the winning feature, slightly reduces the receptivity of the features to new inputs forcing it to become more selective with each win.

If the input does not match any existing feature, the input is discarded and the thresholds for all features is decreased by $\Delta\theta^-$. This has the effect of increasing the receptivity of all

ALGORITHM 4.1
Event-based Feature Extraction via Adaptive Selection Thresholds

```
Require: $\boldsymbol{e}_i = [x_i, y_i, t_i]^T, i \in \mathbb{N}$
Ensure: $\boldsymbol{w}_n(\boldsymbol{x}, \boldsymbol{y}),\ n \in 1..N$
  where $N$ is the number of neurons in the layer.
  and $\boldsymbol{x} = [-R..R]$ and $\boldsymbol{y} = [-R..R]$
  and $R$ is the radius of the $\boldsymbol{ROI}$
Initialize:
      $\boldsymbol{T}_i \Leftarrow -\infty, \boldsymbol{w} \Leftarrow \boldsymbol{w}^0, \boldsymbol{\theta} \Leftarrow \boldsymbol{\theta}^0$
  where $\boldsymbol{w}^0$ and $\theta^0$ are random arrays with values between 0 and 1.
  for each event $\boldsymbol{e}_i$ do
    $\boldsymbol{T}_i(x_i, y_i) \Leftarrow t_i$
    $\boldsymbol{ROI}(\boldsymbol{x}, \boldsymbol{y}) \Leftarrow e^{(t - \boldsymbol{T}_i(\boldsymbol{x}+x_i, \boldsymbol{y}+y_i)/\tau_0}$,
    $\boldsymbol{d} \Leftarrow \boldsymbol{ROI}(\boldsymbol{x}, \boldsymbol{y}) / ||\boldsymbol{ROI}(\boldsymbol{x}, \boldsymbol{y})||$
    for each neuron $n \in 1..N$ do
      $\delta_n \Leftarrow \left\langle \boldsymbol{d}, vec(\boldsymbol{w}_n(\boldsymbol{x}, \boldsymbol{y})) \right\rangle$
    end for
    $m \Leftarrow \mathrm{argmax}_n(\delta_n)$
    if $\delta_n < \theta_n$ for any $n \in 1..N$ then
      $\boldsymbol{w}_m(\boldsymbol{x}, \boldsymbol{y}) \Leftarrow (1-\eta)\boldsymbol{w}_m(\boldsymbol{x}, \boldsymbol{y}) + \eta \boldsymbol{ROI}(\boldsymbol{x}, \boldsymbol{y})$
      $\theta_m \Leftarrow \theta_m + \Delta\theta^+$
    else $\theta_n \Leftarrow \theta_n - \Delta\theta^-$ for all $n \in 1..N$
    end if
  end for
```

features making them more receptive to input ROIs with greater distance to their current location in the feature space. Thus, with each 'missed' input event, the network as a whole becomes less selective and more receptive to change.

The effect of this dynamic thresholding serves to ensure that after convergence the rate of firing of for all features is approximately equal, as decreasing the threshold on matching serves to reduce the receptiveness of each feature to new data. This balance between expansion and contraction of the thresholds results in features with weights placed at the center of the most commonly observed regions of the input space, with selective thresholds that match the dispersion of observed inputs around those points. This adaptive method takes advantage of the abundance and informational redundancy in event-based data. By trading off a small fraction of events which are missed by all neurons, the algorithm finds the appropriate set of selective thresholds which balances the feature activation over the dataset.

The top panel in Figure 4.1 shows the evolution of useful system parameters during network training where after every 100 events, four signals were sampled and the magnitude of their inter-sample change is plotted over input event index. The signals plotted include the variance (across neurons) of the output spike rate (divided by ten thousand), the magnitude of the change in the synaptic weights of all neurons, the magnitude of the change of the selection thresholds and the missed spike rate (divided by ten thousand).

The bottom panel shows the evolution of the learnt features during training. Note the presence of three variants of the "noise feature" in the feature sets. These empty features with a single high value at the central triggering pixel are learnt from events which are not correlated with any recent adjacent events. While the features appear empty and flat except for the central pixel, all but one feature per network typically exhibit very weak structure in their empty regions distinguishing them from each other and the dominant truly flat noise event

As shown in Figure 4.1, at the beginning of training the features are initialized to random points on the unit hypersphere. Since the selection thresholds are initialized at random, a minority of the neurons will have thresholds so wide that every input event causes a neuron to fire, preventing other neurons from spiking and thus learning. Because of their greater receptivity, these neurons capture all input events such that there are no missed spikes and no change in the selection thresholds of the more selectively initialized neurons. This is evidenced by the top panel of Figure 4.1 where during the initial stage the relative magnitude of change in the thresholds is low since only the thresholds of a few neurons are adapting (becoming ever more selective).

The magnitude of the change in the feature weights is similarly low due to the early unbalanced activity of the network, allowing only a few neurons to learn. An additional signal shown in the panel is the standard deviation of spike rates across neurons. At the early stage of learning, with only a few highly receptive neurons firing, the variance in the firing rates across neurons the neurons is low.

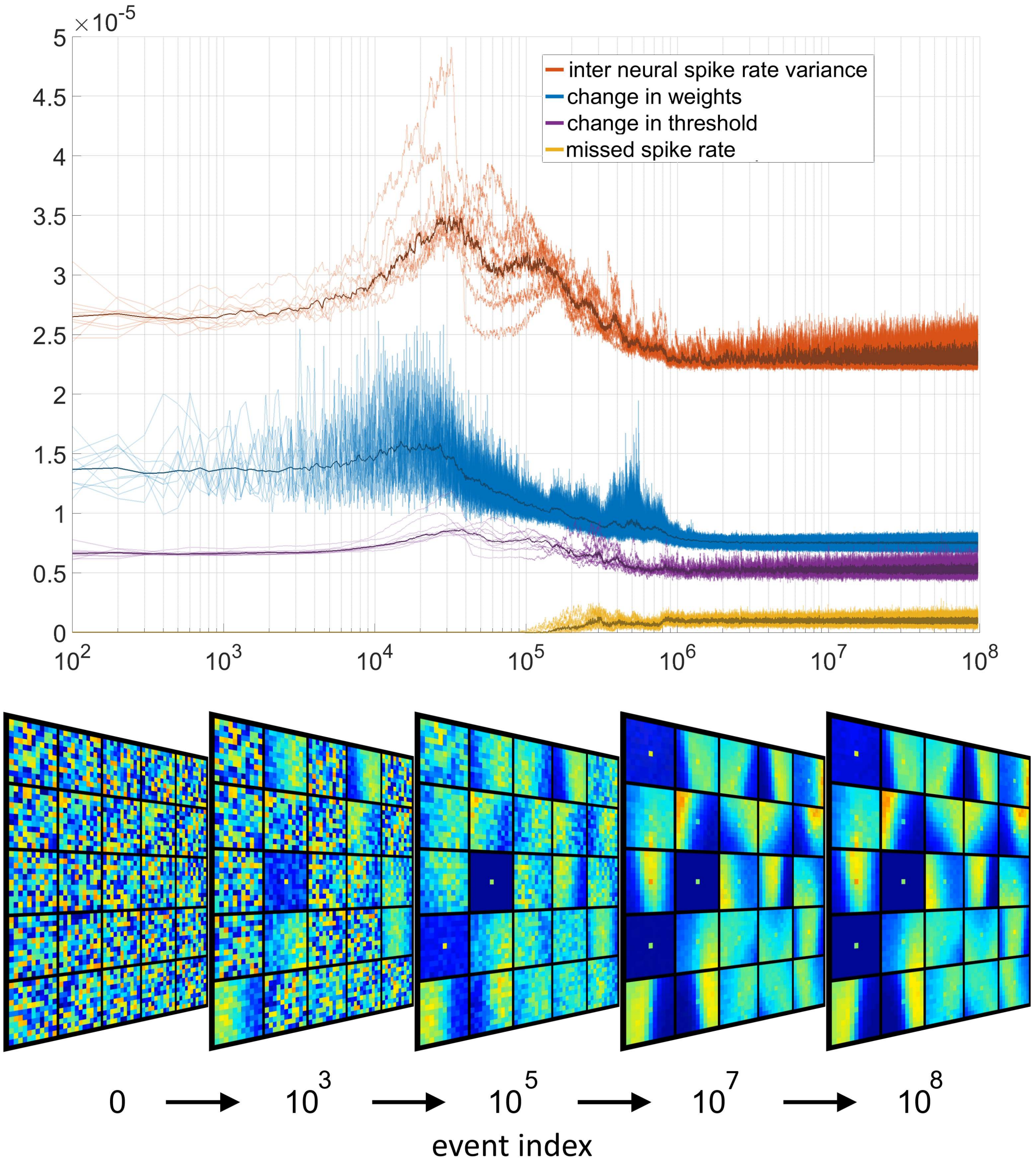


FIGURE 4.1: **Evolution of the network variables.** Top panel shows the adaptation of various neural signals in the network over 10 independent trials as well as the mean over the trials. The bottom panel shows the evolution of the feature weights for one of the trials.

As the early highly receptive neurons contract their thresholds and become more selective, more neurons with more selectively initialized thresholds become activated and begin learning. This learning results in an increasing rate of change in the feature weights and the selection thresholds. Even greater change is observed in the variance of the output spike rate across neurons, as more and more neurons become activated while the most selective neurons have still not fired a single output. Eventually, as the number of activated neurons with decreased thresholds increases to a tipping point, the magnitude of the change in thresholds begins to decline as fewer and fewer highly receptive neurons are left for adaptation. Simultaneously, the change in weights and the variance in the spike rate of the neurons also falls, as the neuron weights and thresholds begin to take on the statistics of the input dataset such that the neurons orient toward the centroids of the most common spatio-temporal pattern clusters while the thresholds take on values in proportion to the spread of the patterns around these centroids. Eventually all neurons become so selective that some input events start to fall outside the selection threshold, causing the first missed spikes. With these missed spikes the thresholds of all neurons increases causing the final most selectively initialized neurons to respond to input and begin adapting their weights. After this point all signals move toward their final steady-state values demonstrating the convergence of the network. In this state the change in weights and thresholds and inter-neuron spike variance reach their lowest value, while the missed spike rate reaches a steady state of approximately 2 percent.

Once training is completed, the selection threshold can be discarded such that during inference, the feature with the largest dot product (smallest cosine distance) to the input is assigned to the incoming event, regardless of the absolute value of the adapted selection threshold.

4.1.3 Noise Features and Network Size Selection

A heuristic developed during the testing of the FEAST algorithm was to use the number of noise features to select the appropriate network size. Optimal feature receptive field size and the corresponding layer size are among the most difficult event-based network meta parameters to optimize. This is due to the large potential parameter search space, the strong interdependence of the parameters, and the long feedback loop guiding the parameter selection. For multi-layer networks the search space increases in a combinatorial manner. Furthermore, each data point in the network structure search space requires development and convergence of multiple independent feature extractor networks and subsequent multiple classification operations.

To bypass this search, we use a heuristic method of observing the noise features shown in Figure 4.1 for selecting network size. These noise features are often one of the dominant features in networks trained on noisy real-world event-based datasets. By representing noise events that are triggered by the noise in the sensor and not by changes in illumination, such noise features effectively perform unsupervised noise filtering. In addition, in this work these noise features are used for the novel purpose of selecting the number of neurons in each layer for a given dataset. This method is based on the observation that irrespective of the complexity and structure of the dataset the input descriptors d generated by noise events are highly correlated with each other and contain mostly redundant information. This would ideally be described by a single noise feature neuron with all other neurons coding for the complex structured features present in the dataset. Thus in an extreme case, a large network whose trained features are all variants of the noise feature is evidence of a training dataset that contains no structured information beyond noise events. At the other extreme, again assuming a non-ideal sensor which generates some noise, a network containing only complex features with no noise features is evidence of a network containing too few neurons since it has yet to incorporate the information from the noise feature (and presumably, other more significant non-noise features also). In this heuristic approach, if an event-based sensor is assumed to generate noise events, the target number of noise features learnt from any dataset should be at least one and possibly slightly more (to ensure that non-noise features less common than the noise feature are also incorporated).

In practice, as the number of neurons in the feature set is increased and representation of unique complex structured features in a dataset is exhausted, the number of additional noise-like features tends to increase. After this point, given that noise features by definition do not correlate to any target class, further increase in neuron numbers is likely to produce diminishing returns. In this work a target of 2-4 noise features was selected for the Plane Dropping dataset. This target resulted in respective layer sizes of 100 and 25 neurons per polarity for the N-MNIST and Plane Dropping datasets respectively. This heuristic method of selecting layer sizes by observing the number of features becomes particularly important for feature extraction networks developed for noisy real-world event-based data and in applications where exhaustive search of network structures is unavailable.

4.2 Methodology

This section presents a brief introduction to the datasets and classifiers used to evaluate the FEAST algorithm.

4.2.1 Datasets

Two different datasets were used to explore and a validate the FEAST algorithm. There are a growing number of standardized event-based datasets captured with event-based cameras. These include datasets for a wide range of vision-related tasks, such as action recognition [91], optical flow [113], face recognition [81], and visual navigation [92]. The FEAST algorithm was tested on the N-MNIST event-based dataset [89] to present and characterize the feature extraction process. The N-MNIST dataset is a conversion of the original and widely disseminated MNIST dataset [114] to an event-based format. Whereas the letter and digit dataset presented alongside the HFIRST algorithm contained only a small subset of digits, the N-MNIST dataset contains the full 70,000 training and testing samples. The dataset is converted to an event-based representation by projecting the digits onto an LCD screen and then recording them with an ATIS camera as it proceeded through three defined saccade-like movements forming the triangle.

While the N-MNIST dataset represents a good benchmarking task for event-based classification systems, with results reported in multiple papers [109, 115–117], it represents a heavily controlled classification task. The use of the fixed and predictable saccade motion creates an unrealistic assumption on which to test feature detection algorithms destined for less controlled real-world tasks. In addition, the N-MNIST dataset is an extremely clean dataset containing virtually no noise events. While this noise-free property of the dataset allows a greater focus on event-based classification, it limits the dataset's application to more noisy environments. In order to better evaluate the performance of the feature detectors, the less controlled Plane Dropping classification task was added which introduces more noise and greater natural variation into the task. In the plane dropping experiment, the random noise events, which are inherently generated due to the non-ideal response of the camera, are not cleaned. Furthermore, the biases of the sensor which control the pixel event generation threshold at each pixel are not optimized for the lighting condition of the recorded scene. These non-optimized settings cause additional noise events which make the dataset more challenging to process and more similar to real-world recording conditions.

4.2.2 Classifiers

Three different classifiers were used to perform the learning and classification tasks on the feature events generated from the FEAST algorithm.

The first classifier is an iterative implementation of the Extreme Learning Machine (ELM) [88]. ELM networks consist of a standard three-layer configuration and use random weights to project from the input layer to a hidden layer. This hidden layer input is passed through a nonlinear activation function, typically a sigmoid function. A set of linear output weights are then learned to map the hidden layer output to the output classes, thus performing classification.

The classifier uses the Online Pseudo-Inverse Update Method (OPIUM) [118] to iteratively update the linear output weights which project from the hidden layer neurons to the output neurons. The use of an iterative method of solving the pseudo-inverse for the ELM allows the classification network to be updated in response to each individual event, however, the scale of the number of input channels and the size of the event-based dataset make direct application of the ELM network to the event-based data prohibitively difficult, motivating the dimensionality reduction provided by the feature extractor network.

The second classifier used in this work is the Synaptic Kernel Inverse Method (SKIM) [13], which is a neural synthesis technique designed to operate directly on spike-based inputs and is therefore directly compatible with the event-based output of these event-based sensors. The SKIM method is inspired by the process of dendritic computation and has a similar three-layer structure to the ELM network with fixed and random connections from the input layer to a much larger hidden layer. A set of learned linear weights connect the hidden and the output layer neurons. The hidden layer neurons in SKIM represent dendritic synapses and implement a nonlinear activation function, whereas the random input weights model the axonal connections to other neurons. The linear output weights represent the dendritic connections to each output neuron.

The SKIM network also makes use of OPIUM to learn these weights, although any gradient descent method would be suitable. The original SKIM network as proposed by Tapson et al. provided a means of synthesizing networks capable of producing specific spatio-temporal patterns in response to specific input patterns. The implementation of the network requires a number of modifications in order to utilize the algorithm for a classification task. A full discussion of these alterations is provided in [115]. In addition to these two classifiers a linear classifier was also used as a baseline for the performance of the other two classifier.

The two major alterations involve the nature of the training signal and the means by which the classification output is determined in a multi-class classification task. The original SKIM network made use of a single spike as the training target, which performed well on simple spike pattern recognition tasks but did not extend well to larger and more complex datasets, suffering from a high degree of sensitivity to noise.

These single output spikes were replaced with a multi-step training signal which imparts more energy into the system and greatly increases the robustness of the learning mechanism. A range of possible transfer functions can be used for spreading the energy of the events to later training time steps. These include linearly decaying, exponentially decaying or Gaussian transfer functions. For this purpose the SKIM classifiers presented in this use Gaussian transfer functions were used as training signal which was amended to the end of each recording. A second adaptation required to handle multi-class classification tasks involves the means by which the winning class is determined. Whereas in ELM networks, the class with the maximum activation at each time step is the winning output class, this assumption does not readily translate to the event-based paradigm used in the SKIM network. In this work, additional time-steps are appended to the end of the training and testing spike trains. During this augmented section at the end of the spatio-temporal pattern, the supervisory signal indicating the winning class is activated. For the purposes of this work, the winning class is determined by the output neuron with the highest cumulative activation during the augmented period. This is referred to as the Area determination method, as described in [115].

4.3 Results

4.3.1 N-MNIST Digit Classification Results

For the purposes of the classification experiments, the FEAST method was used to generate features on the N-MNIST dataset. Through the heuristic examination described in Section 4.1.3, 100 features were selected as the feature layer size for each polarity for the N-MNIST dataset. Only the training samples were used to generate the features, and made use of the feature extraction parameters configured as $\Delta\theta^{+} = 0.001$ and a $\Delta\theta^{-} = 0.003$ with 200 features (100 for ON events, 100 for OFF events) of size 11×11 pixels.

An ELM network was used as one of the back-end classifiers. ELM networks do not intrinsically operate on event-based data, and therefore the input sequences from the N-MNIST dataset cannot serve directly as input for an ELM network. The most direct approach requires the use of a separate input channel for each pattern at each time-step, and when dealing with the original N-MNIST sequences, the image size of 34×34 pixels and the 316 time-steps (the maximum number of millisecond time-steps in the N-MNIST dataset) results in a required input size of 365,296 per digit which is prohibitively large. However, by mapping the data into the feature domain, the size of the input layer is reduced to a single feature per time-step. For a network containing 100 features, this results in an input pattern size of 31,600, reducing the input layer by more than an order of magnitude.

Testing with the ELM classifier involved the two different sets of features for each polarity, with the same event to feature mapping method used for each polarity.

Table 4.1 presents the results of the linear classifier, the ELM classifier and the SKIM network. The results show that the learnt FEAST features outperform the random features. The ELM is shown to outperform a SKIM network of the same hidden layer size. As expected, both

TABLE 4.1: **Summary of classification accuracies on the N-MNIST dataset.** Three classifiers, a linear classifier, an 8000 hidden neuron SKIM network and an 8000 hidden neuron ELM classifier were used on the output of 200 11×11 pixel features.

Classifier	Random	FEAST
Linear	**63.49** +/-0.33 %	**71.42** +/-0.28 %
SKIM 8K	**90.64** +/-0.18 %	**93.89** +/-0.17 %
ELM 8K	**90.34** +/-0.13 %	**95.11** +/-0.11 %

networks outperform a simple linear classifier. The results in Table 4.1 generally confirm those performed in [112] with a minor difference that for the 8000 hidden layer ELM, the FEAST features provide slightly higher performance relative to random neurons. These results exceed those achieved using the same number of hidden layer neurons with the SKIM algorithm alone as previously reported in [115]. Combining the learnt features with the SKIM network creates a fully event-based network from end to end. The network operates on each spike, updating the features, and learning in a feed-forward manner. The SKIM network is also particularly well suited to the nature of the events produced by the adaptive threshold clustering, as they are inherently sparse spatio-temporal patterns. Where the ELM required the vectorization of the resulting spatio-temporal pattern in feature space, the SKIM network can operate on the feature events directly, and therefore has only a single input channel for each feature.

4.3.2 Plane Dropping dataset Results

Algorithms tested and carefully tuned for ideal datasets can produce unrealistic performance expectations, and fail when tested in such challenging real-world applications. For this reason we augment our testing with the Plane Dropping dataset which provides a less controlled, more noisy dataset for classification than the N-MNIST digits dataset. It is intended to showcase the ability of the FEAST algorithm to generalize to more real-world conditions with fast, unregulated motion and unpredictable recording environments that do not match the tuned biases and controlled environments used in the generation of most event-based datasets such as N-MNIST. Additionally, although the N-MNIST dataset includes motion through saccade-like movements used to collect the dataset, this repeated tightly controlled motion profile generates repeating predictable patterns for the classifier. In the plane dropping task, the lower SNR (Signal to Noise Ratio), the varying relative orientations of the similar-looking targets, and the varying velocity profiles increase the difficulty of the classification task in ways that are more similar to real-world conditions.

The algorithm used for processing the plane drop dataset was the same as that used for the N-MNIST dataset, with only three parameter modifications.

Firstly, the higher target velocities and noise levels in the Plane Dropping dataset required a shorter time constant than the N-MNIST dataset, specifically 3ms in place of 316 ms.

Secondly, whilst the same 11×11 feature size as for N-MNIST was used, the number of features selected was 25 per polarity, as networks with a higher number of features generated a large number of representations for the "noise feature" shown in Figure 4.1. For the Plane Dropping dataset, using 11×11 pixel features, 25 neurons consistently resulted in 2-4 variants of the noise feature which is the target range set out in the heuristic described in Section 4.1.2.

Finally, due to the non-optimized tuning of the biases of the sensor, the OFF events exhibited very low SNR and carried little information. As a result, only the ON events were used for this dataset, thereby resulting in only 25 features used in total as opposed to 200 for the N-MNIST dataset.

As with the N-MNIST dataset, the system was trained on a subset of the airplane dataset. The training set consisted of random sets of 400 recordings, with the remaining 400 making up the test set. There is significant variance in the spatio-temporal patterns generated by the airplanes within each recording of the airplane dataset, due to significant change in velocity, pose, and the periods of partial occlusion as the planes enter and exit the field of view. This intra-recording variance significantly adds to the complexity of the dataset. To capture this variance, the feature surfaces were sampled at 3ms time intervals during each recording, resulting in approximately 50 classification operations for each recording, such that approximately 20000 unique training and 20000 unique testing samples were presented to the classifier.

An example of the features generated for the plane dataset is presented in Figure 4.1 where the network produced three variants of the noise features. Since the output of these features do not correlate with any particular class, they effectively act as naturally evolved noise detectors, leaving only clean data for the rest of the network and being essentially ignored by the classifiers. While a similar functionality can be hardcoded using noise filters, the FEAST algorithm extracts multiple variants of the noise feature from event stream. These features point to subtle statistical structure in the noise which likely depends on the dynamic recording environment and the sensor. Such structured data would not be amenable to hard coding and could only be learnt and detected in an online manner. After the convergence of the feature detector, the training data was converted to feature space through the FEAST algorithm and presented to the classifier through a supervised training regime. The training order was randomly selected. Once the training step was complete, the unseen test set was passed through the same FEAST layer and the classifier output determined. Two measures of accuracy were used. First a per frame measure calculating accuracy of each classified frame and a second per recording accuracy measure which performs a majority voting on the frames of each recording assigning the recording to the class with the highest number of winning frames.

In addition to the linear classifier and the 8000 neuron hidden layer SKIM and ELM networks, a large 30000 hidden neuron ELM network was also tested on the same feature output data in order to quantify the level of residual nonlinearity after the feature extraction operation. Finally, to separate the efficacy of the FEAST algorithm from the improvement gained via the event-based convolution operation, 25 random weighted features with identical weight distributions were also tested against FEAST while keeping all other aspects of the system unchanged. These results are shown in Table 4.2

As the results in Table 4.2 show, the highest per-frame classification accuracy is achieved using the large ELM operating on the FEAST output, resulting in 92.81% accuracy. More remarkable than the absolute value of the highest accuracy is the relative improvements each element of the system delivers. When random features are used as feature extractors 64.2% of the samples become linearly separable. The use of the large ELM on the random features

TABLE 4.2: **Summary of classification accuracies on the Plane Dropping dataset.** Four classifiers, a linear classifier, an 8000 hidden neuron SKIM network, one 8000 and one 30000 hidden neuron ELM classifier were used on the raw events, on the output of 25 random features, and on the 25 FEAST features.

	Per Frame		Per Drop	
Classifier	Random	FEAST	Random	FEAST
Linear	**64.2** +/-4.9 %	**83.8** +/-2.5 %	**69.6** +/-5.8 %	**87.9** +/-2.7 %
SKIM 8K	N/A	N/A	**74.4** +/-5.0 %	**77.0** +/-3.8 %
ELM 8K	**67.9** +/-5.0 %	**87.2** +/-1.9 %	**75.9** +/-5.3 %	**90.1** +/-2.2 %
ELM 30K	**69.2** +/-4.7 %	**92.8** +/-1.8 %	**77.8** +/-5.5 %	**96.2** +/-2.0 %

only provides an additional 5% improvement. This result provides an insight into the utility and also into the limitation of the event-based convolution operation. Despite not being effective representations of the data, the random neurons still significantly improve accuracy by aggregating local information around incoming events. Yet this aggregation is highly inefficient, with a significant amount of information lost due to the lack of specificity of the neurons to the dataset structure. This is shown by the fact that the 30000 hidden layer neuron ELM can only extract a slight improvement on the available data despite its large hidden layer. In contrast, by orienting the features toward the data, the FEAST neurons alone manage to linearly separate 83.8% of the frames and provide enough information to the ELM for it to linearize a further 9% of the data. In this configuration of the system, when all frames of the recording are combined in a majority voting operation, a per drop accuracy of 96.2% is achieved on the 30000 hidden layer neuron ELM. Table 4.3 details the confusion matrix for this configuration of the system for the 4-way Plane Dropping dataset.

While the SKIM classification algorithm operates on time steps that are analogous to the frames used for the linear and ELM classifier, no meaningful per frame (or per time-step) accuracy measure can be deduced from the SKIM algorithm. This is due to the nature of the algorithm which is trained to output a classification signal at the end of each recording only. On the random features, the SKIM network's accuracy of 74.4% was between the linear classifier and the tested ELMs with the equivalently sized 8000 hidden layer ELM performing slightly better than the SKIM classifier. This result is in contrast to those from the N-MNIST dataset. It is likely the result of the great variance in target velocity and the noise present in

TABLE 4.3: Confusion matrix for mean performance of the per-frame 30000 hidden layer ELM classifier on the 4-class plane dropping results.

Results averaged over 20 trials.

		Predicted				
		F117	Mig-31	Su-24	Su-35	Accuracy
Actual	F117	**24.5**%	0.1%	0%	0.5%	97.7%
	Mig-31	0%	**20.8**%	0.5%	3.1%	85.3%
	Su-24	0%	0%	**24.79**%	0.07%	99.1%
	Su-35	0.5%	2.0%	0.3%	**21.7**%	88.7%
	Precision	97.9%	90.8%	97.0%	85.3%	

the Plane Dropping dataset compared to the N-MNIST dataset. Here, the random kernels of the SKIM may work against the classifier by increasing the already high variance in the time scales of the observed spatio-temporal patterns caused by the varying target velocity as well potentially extending the effect of noise events via slow decaying kernels. These differing relative performances between the classifiers highlights the utility of testing algorithms on datasets of dissimilar design.

Finally, when tested on an equal number of FEAST features that are well oriented towards the data, SKIM performs worse than a all classifiers tested. This is possibly due to the fact that the output activation of the FEAST features already provides a linearly separable mapping to the output classes but the high variance of velocity in the dataset together with the late supervisory signal in SKIM, which, on this dataset, arrives as the airplane is leaving the field of view likely impacts the algorithm's accuracy below the other per frame-based methods which perform their learning at all stages of each recording providing greater invariance to target velocity.

4.3.3 Evaluating Feature Sets Via Feature Activation

The most direct measure of the utility of a feature set for any classification dataset is the recognition accuracy achieved by the classifiers. However, in many circumstances, this measure can be significantly more computationally expensive than the development of the feature set itself. Acquiring a rigorous figure of merit for any feature set can require repeated training of back-end classifiers. This long feedback loop in the evaluation of feature sets can be time-consuming and can limit the range of feature extraction parameters that can be investigated. This same issue was encountered in this work, where the rigorous evaluation and comparison of feature sets through the calculation of recognition accuracy consumed significantly more time and computational resource than the development of the features themselves. However, it was found that the output of the FEAST neurons provided an easily accessed alternative signal that correlated strongly with final recognition accuracy measure.

The adaptive selection thresholds of the FEAST neurons force the network features to capture the most commonly observed patterns, while also compensating for the frequency of the observed patterns. This means that during the learning phase the neurons are constantly being pushed toward equal activation. During inference, however, without the adaptive thresholds enforcing equal activation, the network spike rate can vary significantly across neurons with some neurons spiking more than others. This spike inequality was found to correlate strongly to the classification accuracy over the dataset, allowing rapid coarse evaluation of feature-sets and network meta parameters. The measure used for quantifying inequality in spike output was the Gini coefficient [119]. This measure, commonly used to quantify wealth and income inequality [120] is defined as the mean absolute difference of all pairs of items in a population divided by the mean of the population to normalize the scale. The Gini coefficient G is defined by 4.4 where N is the number of neurons and x_i and x_j are the output spike counts of neurons i and j. This measure can easily be calculated for the FEAST neurons at any point during inference.

$$G = \frac{\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}|x_i - x_j|}{2\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}x_j} \tag{4.4}$$

Figure 4.2 shows the strong, relationship between the Gini coefficient and classification accuracy for two thousand random, independently parameterized feature sets. Each data point represents results from a feature set instantiated with random threshold parameters, feature sizes, learning rates, training dataset size. The feature sets were randomly selected to have

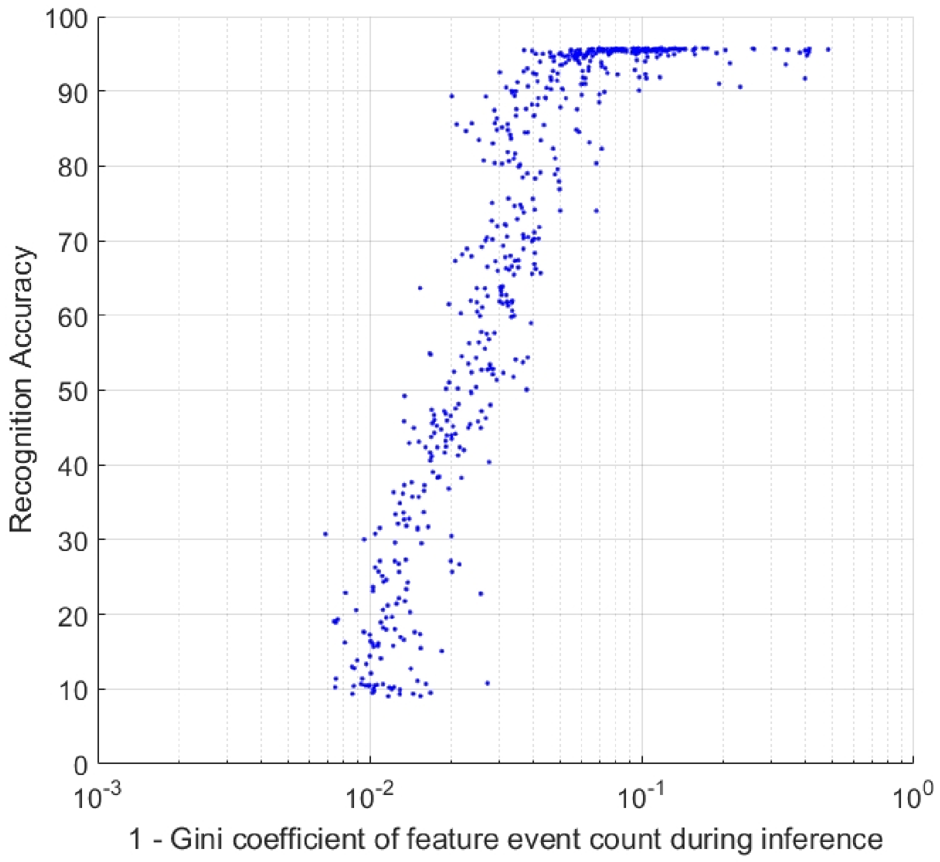


FIGURE 4.2: **Final classification accuracy and the Gini coefficient of feature event counts.** Each point on the plot represents results from an independent instantiation of a FEAST network with randomized parameters on random subsections of the N-MNIST dataset.

sizes between 1 and 100 neurons. with feature sizes randomly ranging from 3 to 19 pixels across. The training and testing dataset were split randomly with splits ranging from 0.1 and 0.9 of the full dataset. The threshold rise and fall parameters as well as the learning rate were selected randomly from 0.0001 to 0.01. The results cover almost the entire accuracy range from chance accuracy to the highest optimized accuracies achieved in this work on the N-MNIST and the Plane Dropping datasets. Yet across the entire accuracy range, the relationship between accuracy and the Gini coefficient is remarkably robust, suggesting that the Gini measure can provide a reliable rapid evaluation and comparison of feature layers, without the need for further processing and computation. Not only is the Gini coefficient useful during the algorithm design stage, where many of the interdependent parameters of the larger system need to be instantiated, but can also serve as a reinforcement signal in online learning applications, by quantifying in real-time the relevance of a feature set to any batch of observed data.

4.4 Discussion and Future Work

4.4.1 Missed Events During Learning

As detailed in Section 4.1.2, events that fall outside the threshold of all features reduce all thresholds but do not result in an adaptation of the weights. These ’missed’ events can be treated in different ways. They may be viewed as outliers with respect to the features learned by the network. This is the simplest approach in the context of hardware implementation and the one taken in this work. Another approach is to assume the missed events hold important residual information useful for classification. Being an unsupervised algorithm, the learned weights of FEAST and the output classes have no direct relationship. As such, the relative importance of unincorporated outlier events can only be determined empirically through their effect on resultant feature sets and classification performance. In all our tests, the number of missed events constituted less than 5 percent of events. Experiments with a larger number of training epochs or in which missed events were re-included a second or third time into the dataset produced no observable change in the feature set or recognition accuracy. Such a result would be expected for the tests performed due to the large number of events and the significant informational redundancy in data generated by event-based sensors. Thus, while it is possible that with an extremely small and informationally sparse dataset the FEAST algorithm may not exhibit the same robustness due to missed events, this was never observed in our testing.

4.4.2 Thresholds During Learning and Inference

In our tests the initial values for the threshold were randomized. Other tests of threshold initialization included initializing the threshold at equal values at very high, or low, or zero, using uniform or Gaussian distributions. In all such tests, no two neurons were ever detected to be in identical states, due to the random initialization of the large number of weights. Because of the adaptive nature of the thresholds and the large size of the training data used, no significant difference was observed in the behavior of the signals tested across the wide range of initialization procedures and threshold adaptation parameters. In general, the threshold adaptation mechanism was found to be robust to parameter selection choices, such that after a rapid initial adaptation period the thresholds of different features reached a final steady-state without exception.

After training, a choice arises as to whether the selectivity information contained in the thresholds should be used during inference or simply discarded and replaced with the simple cosine distance matching rule. In this work the thresholds were disabled. Methods to incorporate the information encoded in the selection thresholds is the subject of future work.

4.5 Conclusion

The results presented in this work demonstrate the applicability and capabilities of the FEAST algorithm for extracting useful features in an unsupervised manner. The algorithm converts the event stream into efficient feature representations that outperform random features with the same architecture. The different datasets tested are shown to have significantly different feature information and noise properties. These aspects of the dataset were demonstrated in the resultant trained feature sets and used to select network size. On the N-MNIST dataset, the SKIM classifier operating on FEAST features was shown to outperform all other configurations, including the ELM classifier, while on the Plane Dropping dataset the ELM on FEAST outperformed other configurations. Yet on both datasets and in all cases tested, the FEAST features outperformed raw events and random features. The adaptive selection threshold approach used in FEAST also illustrated a number of interesting properties of event-based visual classification and demonstrated the ability to perform integrated noise filtering, the generation of proxy signals for weight convergence, and ready measures for the prediction of classification performance via the Gini coefficient of the FEAST output event count.

CHAPTER 5

Event-based SPAD processing

Chapter Summary

Single Photon Avalanche Diode sensor arrays operating in direct time of flight mode can perform 3D imaging using pulsed lasers. Operating at high frame rates, SPAD imagers typically generate large volumes of noisy and largely redundant spatio-temporal data. This results in communication bottlenecks and unnecessary data processing. In this work, we propose a set of neuromorphic processing solutions to this problem. By processing the SPAD generated spatio-temporal patterns locally and in an event-based manner, the proposed methods reduce the size of output data transmitted from the sensor by orders of magnitude while increasing the utility of the output data in the context of challenging recognition tasks. To demonstrate these results, the first large scale complex SPAD imaging dataset, to the author's knowledge, is presented involving high-speed view-invariant recognition of airplanes with background clutter. Various sources of noise in the data are investigated and their effects on the proposed event-based generation algorithm are discussed. The frame-based SPAD imaging dataset is converted via several alternative methods into event-based data streams and processed using a range of feature extractor networks and pooling methods. The results of the event-based processing methods are compared to processing the original frame-based dataset via frame-based but otherwise identical architectures. The results show the event-based methods are superior to the frame-based approach both in terms of classification accuracy and output data-rate. Among the several event-based methods examined, systems with higher output event rates, more resource-intensive pooling and larger, more complex network structures produce higher classification accuracy. The detailed investigation of

the event-based processing methods informs the implementation of high-speed event-based architectures in hardware for high noise applications.

5.1 Introduction

5.1.1 SPAD

A Single Photon Avalance Diode (SPAD) is a type of photo-detector that comprises of a reversed biased photo-diode operated just above the breakdown voltage and as such is able to detect individual incoming photons from the environment [121]. This ability to detect single photons enables SPAD cells to calculate precise photon timing information. Integrating an array of SPAD detectors onto a single CMOS chip and using high precision laser illuminators allows the development of SPAD cameras, which can capture high-speed 3D images under extremely low-light conditions. SPAD array cameras have a broad range of applications from military, meteorology, space, augmented reality, remote sensing, autonomous robotics [122][123].

The SPAD camera can operate in two modes: Indirect Time of Flight (ITOF) or photon counting mode and Direct Time Of Flight (DTOF) or photon timing mode. In photon counting mode, counters integrated to each SPAD cell keep a count of the number of arriving photons at each cell and effectively provide a measure of illumination from the probing laser pulse. In timing, or Direct Time of Flight mode, the integrated counters are triggered to start counting at the clock speed by the laser pulse and stop counting at the detection of the first photon as shown in Figure 5.1(a).

The data generated by a SPAD array in DTOF mode consists of a three-dimensional time surface corresponding to the relative distance of the visual scene to the camera and illuminator. Traveling at the speed of light, these time surfaces activate every SPAD cell within the space of a few hundred nanoseconds after each laser pulse such that the relative timing of the activations (or inter-spike time intervals) holds all the encoded spatial information. This means the data is inherently non-sparse, temporal and requires high-speed parallel processing.

Additionally, due to device mismatch and stray photons from the environment, the signal has a high noise floor. In typical applications of SPAD DTOF imaging, due to the multiple noise sources, high frame rates are required to gather enough data so that a useful signal can be attained through averaging over a large number of frames. The high amount of data generated through high frame rate imaging causes significant bandwidth problems and limits the scalability of the sensor.

5.1.2 Event-based Processing and SPAD

The conventional approach to date has been to encode the time of flight of the arriving photons using high precision counters for each SPAD cell and to transfer this timing data off-chip for processing. This approach typically involves as a first step, some form of averaging over a large number of frames which would significantly increase the cost of on-chip processing. This transfer process also creates an information bottleneck which is currently one of the major limiting factors in the speed of operation of high frame rate SPAD cameras. In addition, the use of conventional CPUs or GPUs for processing this temporal data makes processing SPAD data computationally intensive using conventional signal processing techniques and results in significant power and hardware requirements.

Yet the attributes that make SPAD data challenging for conventional processors, when combined with the significant level of temporal redundancy present in real-world visual data, makes the SPAD cell activation patterns ideal for event-based and spiking neuromorphic processors that are designed to operate directly on noisy temporal data in a parallel fashion.

While the conversion of high data-rate DTOF SPAD data into local event-based features is entirely novel, previous works have demonstrated the utility of taking a bio-inspired approach to SPAD processing. In [124], Berkovich et al. present a scalable 20×20 SPAD imaging array using asynchronous Address Event Representation (AER) readout. In [125] the same approach is proposed for use in Positron Emission Tomography applications. The AER protocol is an efficient communication protocol for sparse event-based data that reports events as they occur removing the need for global frames [17]. In these works, the SPAD cells operate in an ITOF photon counting mode where an analog photon-counting circuit counts incoming photon until the counter reaches a preset threshold causing the pixel to generate an event indicating a preset level of illumination. This mode of operation is similar to previously proposed non-SPAD event-based sensors [32] albeit with the advantage of the SPAD's high quantum efficiency. In contrast, the design proposed in this work seeks to combine the inherently temporal nature of DTOF SPAD spatio-temporal data with neuromorphic event-based feature extraction and processing.

In the proposed approach instead of encoding, storing and transferring the high-resolution, (typically 16 bit) photon time of flight data off-chip for processing, the measurement of the time of flight of the laser pulse is abandoned entirely in favor of a neuromorphic processor that operates directly in the time domain and on the inter-spike intervals within local regions of the SPAD array. The proposed approach illustrated in Figure 5.1 motivates the development and hardware implementation of event-based feature extraction algorithms and circuits that generate sparse event-based local representations from the non-sparse event-based SPAD activation data and in this way drastically reduce the I/O requirements of the overall system.

5.2 Methodology

5.2.1 The SPAD Dataset

In this work, we tackle the challenge of performing classification of a large complex SPAD imaging dataset generated using frame-based and event-based approaches. The task involves the recognition of fast-moving model airplanes. The view-invariant classification of the fifteen classes of target airplanes and one distractor represents a challenging problem given the similarity of the classes, the low spatial resolution, the presence of partial occlusions and the high noise level in the dataset.

The 32$\times$32 pixel SPAD camera used in this work was fabricated on a standard CMOS chip with each pixel integrating one SPAD and one time-to-digital converter as illustrated in Figure 5.1(a) and described in [126]. The SPAD camera was fitted with a Navitar NMW-12WA lens

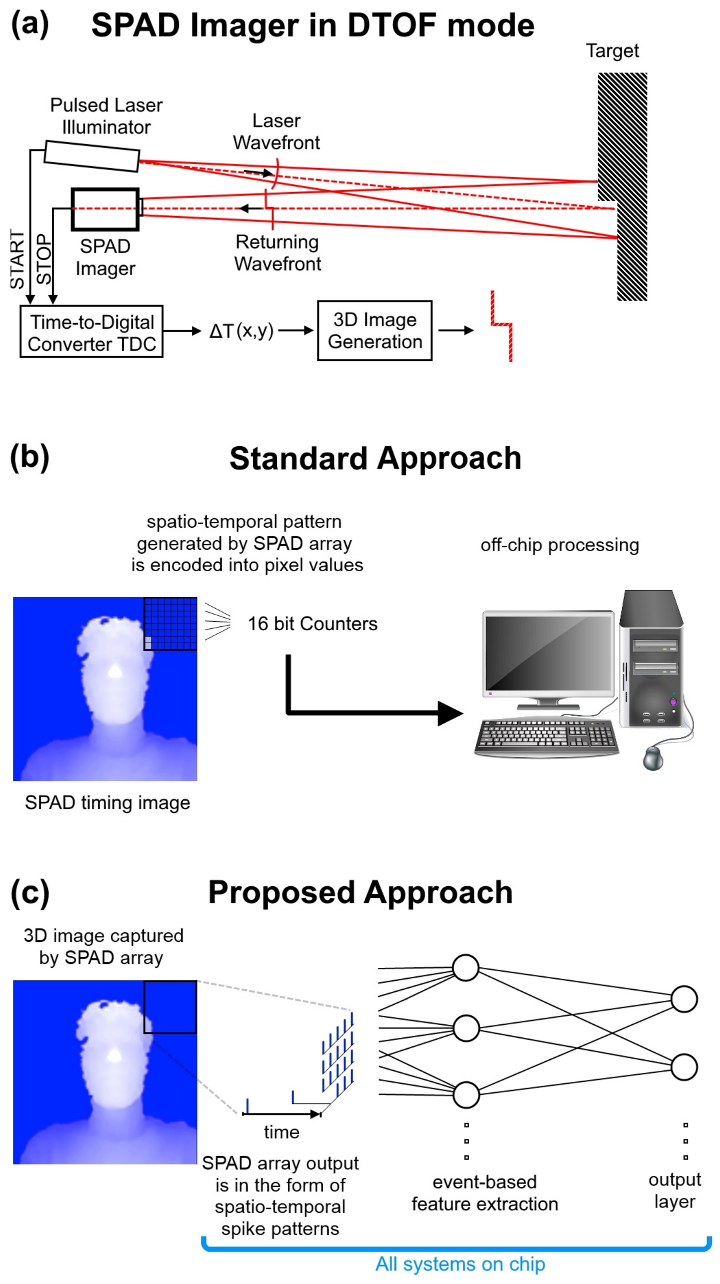


FIGURE 5.1: **Conventional and Neuromorphic SPAD DTOF data processing.** (a) SPAD imager in DTOF mode using a pulsed laser illuminator. Using SPAD sensors in Direct-Time of Flight (DTOF) mode enables the capture of three-dimensional images with a single camera. (b) Standard approach to processing SPAD imaging data using on-chip counters and off-chip processing. (c) Proposed event-based approach to SPAD data processing.

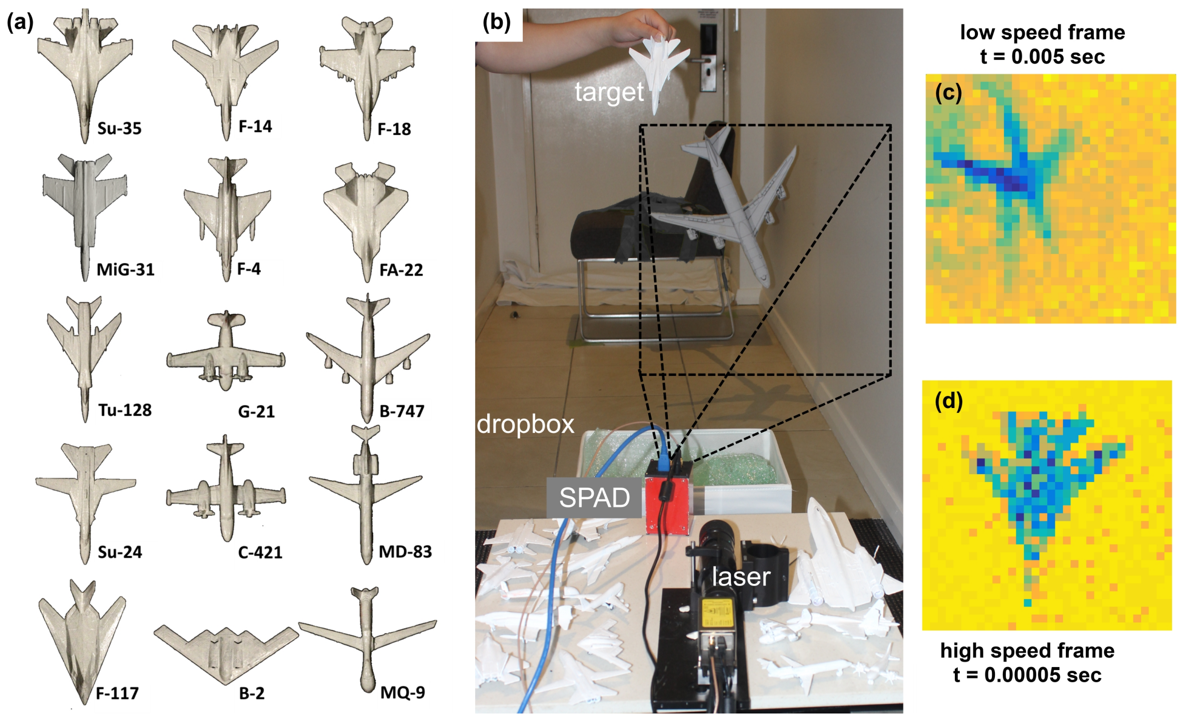


FIGURE 5.2: **SPAD sensor airplane drop classification experiment.** (a) Fifteen model airplane types make up the 15 classes in the detection and classification task. (b) Experiment set-up. Metallic model airplanes painted a uniform white were dropped in front of the SPAD sensor at close range (approximately 30-40cm) resulting in high relative velocity. SPAD field of view is marked by the black dotted line. In the background (approximately 3 meters from the camera) a large model B-747 airplane serves as a distractor. (c) SPAD image generated from averaging 500 raw frames representing 5ms of recording time. The background B-747 model is clearly visible. (d) SPAD image showing of the rapidly moving F-14 model generated through averaging 5 raw frames representing 50μs of recording time.

and a Thorlabs 660 nm filter. The SPAD camera's field of view was set to 26.22 degrees. The laser used to obtain DTOF data was a 100 mW 660 nm Coherent CUBE diode laser using a 12$\times$ zoom lens such that the region of laser illumination and the SPAD camera field of view were overlayed as shown in Figure 5.2(b).

The targets in the dataset are imaged using the SPAD sensor in a photon timing mode where each SPAD pixel operates as a LIDAR sensor. The illuminating laser is pulsed at 100 kHz providing photon time of flight information at an extremely high frame rate. By dropping the model airplane at high speed close to the sensor the high temporal resolution of the sensor can be leveraged and investigated. As shown in Figure 5.2(b), the experiment involves the use of a larger more distant background stationary B-747 model as a distractor. This distractor becomes increasingly more prominent as the number of frames collected for an image is increased. The inclusion of the larger distractor with the free moving high-speed target classes ensures that the dataset can only be processed at extremely high frame rates (for example 10 kHz) which results in a high noise floor that better represents real-world imaging environments. Unlike controlled image collection environments typically used in machine vision research, real-world imaging environments are unpredictable, dynamic and noisy precluding many commonly used image enhancement methods such as arbitrarily long frame averaging. This experiment design aims to encourage the development of algorithms that are more robust to noise and can more readily be applied to challenging real-world imaging environments.

The originally captured dataset involved 3000 individual uncontrolled free hand drops of the 15 airplane classes with 200 drops per class. In the recorded dataset, the targets airplanes passed rapidly through the field of view with a mean duration of only 40.5 milliseconds and a standard deviation of 6.4 milliseconds. The dataset and associated supporting files are available for download at [127]. This 3000 recording dataset was augmented via mirror reflection as well as 90, 180 and 270 degree rotation resulting in an augmented dataset of 24000 recordings. Here, dataset augmentation refers to the common procedure of appending an original dataset with a transformed version the same to generate a larger dataset to capture a greater amount of variance than was present in the original dataset. The transformations used in this experiment involved vertical reflection in combination with three rotation operations which in total results in a dataset that is $2 \times (1 + 3) = 8$ times larger than the original. Sample recordings from the dataset are shown in 5.3 illustrating the significant visual complexity due to variance in target orientation, occlusions and the similarity of the tested classes. This complexity is even greater when the entire video of each recording is considered due to the change in the relative orientation of each target during each recording and due to the occlusions present at the beginning and end of the recordings as the targets enter and exit the field view.

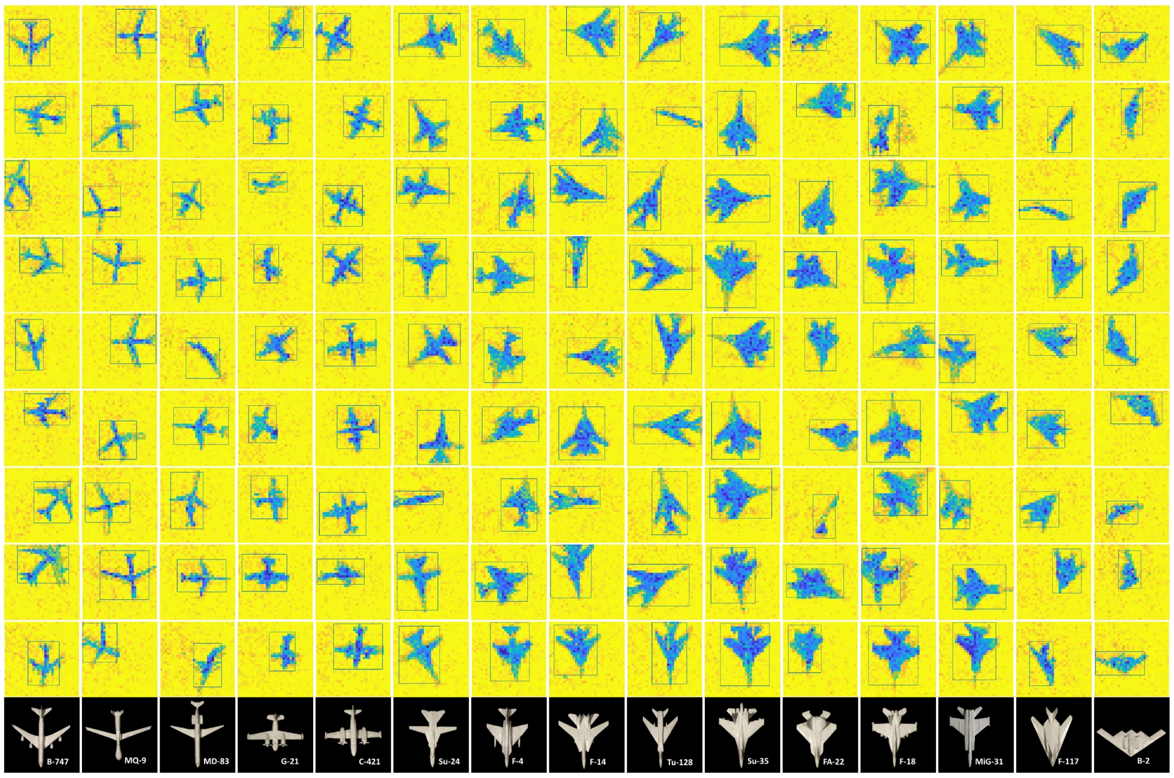


FIGURE 5.3: **Sample recordings from the SPAD dataset.** Each of the fifteen columns shows random samples of each airplane class in the dataset. The images show the wide range of observed orientations, sizes and partial occlusions in present in the dataset as the model airplanes pass through the field of view. The bottom row shows a photo and label of each model. Note that the images show only the midpoint of the sample recordings.

5.2.2 First-AND Event Generation Method: Discarding Time and Transmitting Change

In previously implemented SPAD DTOF systems and in our proposed system, when a SPAD pixel is activated, it enables a latch which stays high until it is reset. The reset is typically performed after all data from the current laser pulse has been transmitted off the sensor. This data and the associated delay can be significant especially as the number of pixels on the imager becomes very large. In the first proposed system, which we call First-AND, instead of recording and transmitting the time interval from the initial laser pulse, only the inter-pixel photon arrival order (not the time) is detected. In this way, the requirement for precise measurement and transmission of the photon time of flight is removed along with the resource-consuming high-precision, on-chip, per-pixel counters and memory circuits.

This simplification is achieved through the use of multiple AND gates which take as input a local group of pixels. The number of input pixels per AND gate must be equal so as to provide an equal probability of gate activation. The pattern of connectivity and its correlation to the observed spatio-temporal order of SPAD activation also determines AND gate activation. For example, an edge bar is more likely to be activated in a natural environment than a checkerboard pattern since the latter is not typically observed in the visual environment. In this way, each AND gate encodes a local feature and its activation indicates that all its input signals have been activated. The choice to use digital AND gates, as opposed to an analog summing and comparator circuits was made to simplify the Integrated Circuit design of the system and to ensure a deterministic output for each gate. This choice has significant implications on the robustness of the system which are discussed in the relevant sections.

The AND gate pattern used in this work have overlapping receptive fields and are tiled across the visual-spatial field to form a convolutional layer. Thus, the same pattern of AND gate connectivity is repeated across the visual field. Each AND gate can be interpreted as a neuron in a local single-layer network of N_0 (in this case N_0=4) neurons connected to a local $r \times r$ (in this case $r = 4$) receptive field. For each receptive field as soon as all the input pixels of a single AND gate latch high, i.e. as soon as the all SPADs feeding an AND gate detect a photon, the AND gate goes high.

For illustrative purposes Figure 5.4(a) shows a small imager with $5 \times 5 = 25$ pixels. This imager uses four 4×4 overlapping receptive fields. Each receptive field has four AND gates. One of the four AND gates takes as input the left/west 8 pixels of the 16 pixels of the 4×4 patch. Another AND gate on the receptive field takes as input the right/east 8 pixels, yet another the lower/south and another the top ones/north. If we consider the imaging environment illustrated in (b), with the 5$\times$5 imager viewing a scene with a back wall and a box in the foreground that is seen by pixels $row = [3 : 5]$ and $col = [2 : 5]$, then the lower/south AND of rf(1,2) (green) and rf(2,2) (blue) will latch which can be expressed as AND(1,2,3) and AND(2,2,3) latching.

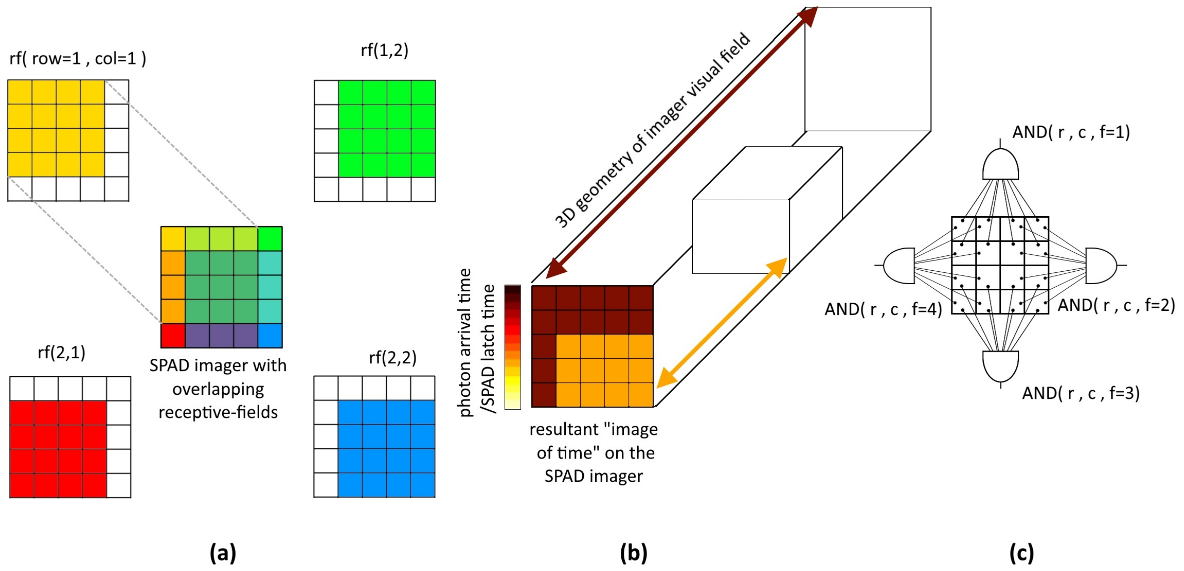

rf(row=1 , col=1)
rf(1,2)
rf(2,1)
rf(2,2)
SPAD imager with overlapping receptive-fields
(a)
3D geometry of imager visual field
photon arrival time /SPAD latch time
resultant "image of time" on the SPAD imager
(b)
AND(r , c , f=1)
AND(r , c , f=2)
AND(r , c , f=3)
AND(r , c , f=4)
(c)

FIGURE 5.4: **Illustrative example of the First-AND event-based imager.** (a) A hypothetical 5x5 imager with four 4x4 overlapping receptive fields. (b) An example 3D visual scene and the resultant SPAD timing pattern. (c) A single First-AND receptive field showing the connectivity of the four AND gate feature detectors.

In the proposed design, the latching of the first AND gate at each receptive field at each laser pulse, prevents subsequent latching of any later gate via a recurrent enable connection that gates all AND gates. This temporal inhibitory feedback structure was introduced in the SKAN network [97] and demonstrated in FPGA hardware. At the beginning of the pulse cycle when the laser pulse is sent, the enable signal to all AND gates for all the receptive fields is high, allowing any AND gate to latch. Following the laser pulse, at each receptive field, as soon as the first AND gate latches, the enable to the other AND gates of this receptive field is set low. Note that the enable flag going low does not affect the first AND gate that caused the lowering in the first place. This is achieved through the use of a positive feedback loop that latches the first, triggering gate to high as shown in Figure 5.5. This is realized through the following logic: An AND gate can only be high if all its input pixels are high and the local receptive fields enable flag is high or the AND was already high in the previous clock cycle in a synchronous system or via an asynchronous mechanism in an asynchronous implementation. Note that the unit delay elements in the feedback path provide time for the feedback circuit to stabilize.

An important edge case is where two or more AND gates latch at exactly the same time. For the synchronous case, this means that the AND gates latch on the same clock cycle and in the asynchronous case this means one or more AND gates latch during the time it takes for the disable signal to travel back to the inputs of the AND gates.

For this edge case, an arbitration logic must be used to either randomly select a winning AND gate or to discount the result of the offending receptive field for this laser pulse. The latter choice is preferable to prevent biasing AND gate activation patterns. This can be achieved through the use of a one-hot gate at the AND gate output as shown in Figure 5.6. This rule-set creates three possible states for each receptive field in the system:

- When, after a laser pulse, no AND gate has yet latched (i.e. no photons has been detected) then the one-hot gate is off and the NOR of the AND gates is high which means this receptive field is still awaiting a winning AND gate event.
- If only one AND gate latches over a clock cycle, then the NOR of the AND gates will be set high and the one-hot gate will also be also high indicating a successful feature detection.
- If two or more AND gates have latched at the same clock cycle in the synchronous case or very close in time in the asynchronous case, then there are multiple AND gates latched. In this case, the NOR of the AND gates is low but the one-hot gate is also low indicating a detection failure. When this multi-latch fail state occurs after a laser pulse, the pulse is effectively ignored.

The detail logical diagram of this competitive AND gate network is shown in Figure 5.5 and 5.6 for one receptive field.

In this approach, each receptive field only requires memory storage for a two-bit address of the feature which was detected most recently. In theory the feature memory would hold an accurate representation of the 3D geometry in front of the camera at the most recent pulse. If this ideal case held in general, and if we wished to minimize the output data-rate, after each successful laser pulse where only a single feature was detected, the logic would check to see if this most recently detected feature is the same as the one in memory. If this is so, there is

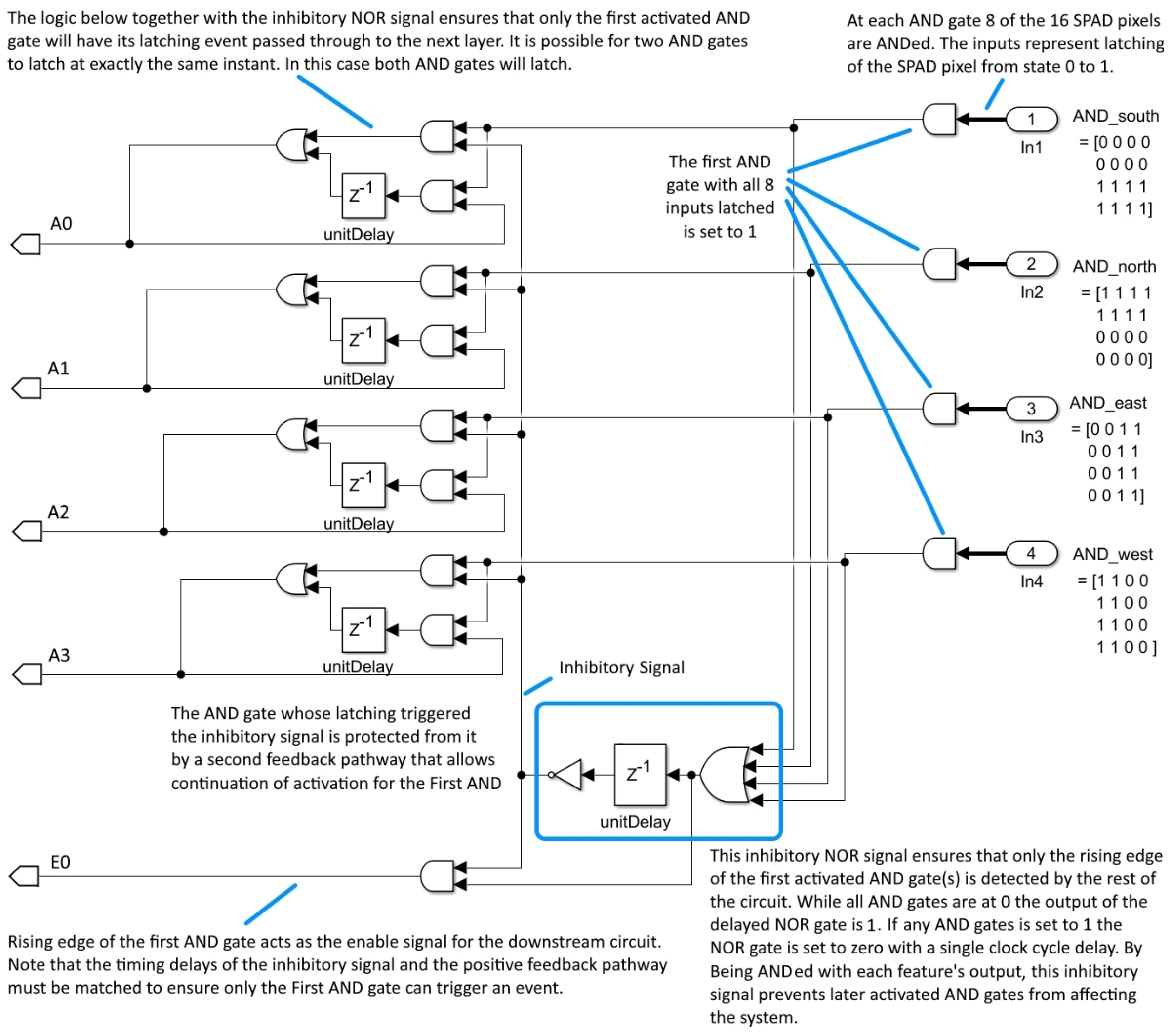

The logic below together with the inhibitory NOR signal ensures that only the first activated AND gate will have its latching event passed through to the next layer. It is possible for two AND gates to latch at exactly the same instant. In this case both AND gates will latch.
At each AND gate 8 of the 16 SPAD pixels are ANDed. The inputs represent latching of the SPAD pixel from state 0 to 1.
The first AND gate with all 8 inputs latched is set to 1
AND_south = [0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1]
AND_north = [1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]
AND_east = [0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1]
AND_west = [1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0]
1 In1
2 In2
3 In3
4 In4
A0
A1
A2
A3
E0
z-1
unitDelay
Inhibitory Signal
The AND gate whose latching triggered the inhibitory signal is protected from it by a second feedback pathway that allows continuation of activation for the First AND
This inhibitory NOR signal ensures that only the rising edge of the first activated AND gate(s) is detected by the rest of the circuit. While all AND gates are at 0 the output of the delayed NOR gate is 1. If any AND gates is set to 1 the NOR gate is set to zero with a single clock cycle delay. By Being ANDed with each feature's output, this inhibitory signal prevents later activated AND gates from affecting the system.
Rising edge of the first AND gate acts as the enable signal for the downstream circuit. Note that the timing delays of the inhibitory signal and the positive feedback pathway must be matched to ensure only the First AND gate can trigger an event.

FIGURE 5.5: **Logical diagram of stage one of the First-AND receptive field block.** The subsystem shown ensures that only the first AND gate(s) can be triggered and that only a one clock cycle output pulse can be generated. Outputs A[0:3] signal the winning AND gate has been detected while the E0 output acts as an enable signal for the generation of a feature event in subsequent the feature event generation subsystem.

no need to transmit it out since nothing has changed at this receptive field. If on the other hand, the feature detected at this receptive field at this most recent pulse is different from the one in memory, then we generate a feature event by storing the new feature in memory, setting the event flag high and sending the 2-bit feature address out on the AER bus. By only sending out events when a new feature is detected, a significant amount of redundant data is no longer transmitted from the sensor.

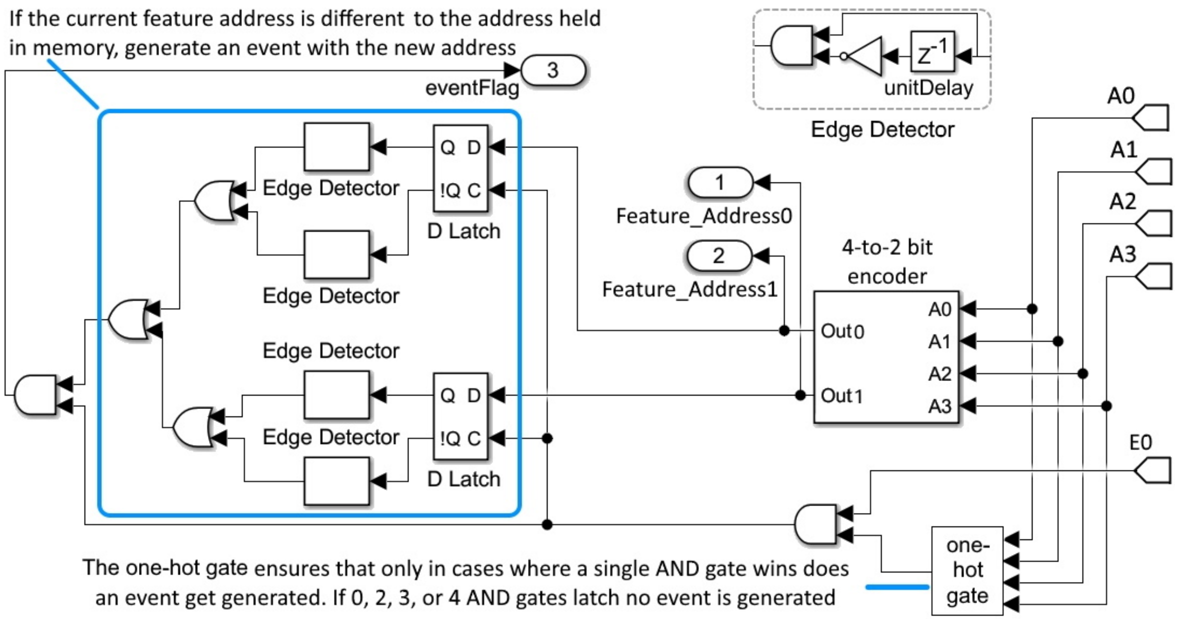

If the current feature address is different to the address held in memory, generate an event with the new address
3
eventFlag
Edge Detector
Q D
!Q C
D Latch
Edge Detector
Edge Detector
Edge Detector
Q D
!Q C
D Latch
z-1
unitDelay
Edge Detector
1
Feature_Address0
2
Feature_Address1
4-to-2 bit encoder
Out0
Out1
A0
A1
A2
A3
A0
A1
A2
A3
E0
one-hot gate
The one-hot gate ensures that only in cases where a single AND gate wins does an event get generated. If 0, 2, 3, or 4 AND gates latch no event is generated

FIGURE 5.6: **Logical diagram of stage two of the First-AND receptive field block** The subsystem shown acts to detect a change in the feature encoded at the receptive field. If a change relative to the previous laser pulse is detected an event is output. This change detection is performed via the Edge Detector circuit shown in the inset (top) which only goes high if the previous input was low and the current input is high.

An important element of the First-AND system is its hybrid mode of operation. The feature detection and event generation circuits described in this section operate asynchronously i.e. without clock synchronization. Here, feature events are generated and placed on the bus as they occur. In contrast, the input to the First-AND system, i.e. the latching operation of the SPAD cells is synchronous and occurs at the rising edge of high-speed (600 MHz) synchronizing clock. Finally, the output of the First-AND system is processed by a synchronous event-based FPGA processor which uses a 100 MHz clock. This hybrid mode of operation results from the integration of distinct systems.

5.2.3 Noise in SPAD Imaging Motivates a Modified Noise Robust Design

Based on the data recorded during our experiment, a range of sensor non-idealities and noise sources were found. These non-idealities could be broadly divided into false-positive and

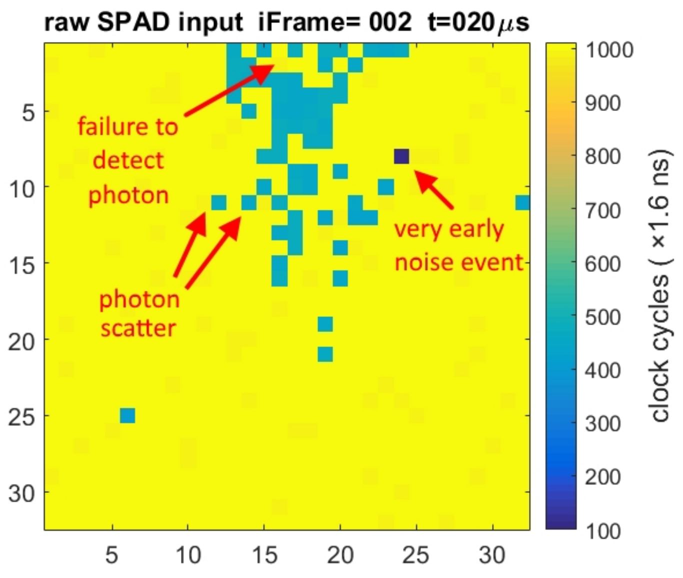


FIGURE 5.7: **Three common type of noise in DTOF SPAD data.**

false-negative latching events, imprecise timing in the latching of the SPAD pixels (jitter) and persistent non-ideal timing patterns across the array pixels.

One source of noise identified is that of stray photons emanating not from the laser pulse but from the environment. These photons detection events are not correlated with the illuminating laser and can trigger the SPAD cells at any moment. These latching events appear to occur in a random fashion. This effect is shown in Figure 5.7 where a SPAD cell is activated within 100 clock cycles due to a stray photon from a light source that is not the laser. Another source of noise is photon scatter whereby a photon from the laser is reflected via an indirect light path and lands on an incorrect pixel. An example of this is marked in Figure 5.7. The second source of noise in the SPAD imaging data is the failure of a SPAD cell to detect a photon where a reflective object should cause a reflected photon to trigger a SPAD pixel. An example of this false-negative case is shown in Figure 5.7 where the internal regions of the airplane, which should have been detected, are not. A simple solution to this issue is to increase the illuminator power to ensure the activation of pixels. This, however, can have drawbacks in terms of power and the safety requirements of using a high power laser illuminator.

Imprecise measurement of the time of photon detecion is another source of noise. This temporal jitter could be due to the noise in the time measurement circuitry, or possibly due to mismatch in the SPAD latching circuitry, or both. This effect can be seen for the case of the detected nearby target (the airplane) whose timing distribution is shown in the measurements marked as 'airplane detections' in Figure 5.8(c). This variance in timing measurement can be seen in Figure 5.8(d) and spans 40 clock cycles whereas the timing should at most be distributed over two clock cycles since each clock cycle represents a distance of 0.48 meter. The imprecise measurement of photons in time is also seen in the background where an apparent sawtooth wave is seen in Figure 5.8(c) and in Figure 5.8(d) where it spans approximately 28 clock cycles. Figure 5.9 shows two different sections of the recording, from pixel (16, 32) shown in Figure 5.8(a). For the First-AND system, small amounts of jitter, noted in the SPAD latching times, does not significantly affect the performance of the system as long the effect is smaller than the relative photon flight delays due to the geometry in the imaging scene.

The latching times plotted in Figure 5.8(c) show that in this experiment, the probability of an error due to missed detection is significantly higher than the error due to stray photons. Figure 5.9 illustrates this more clearly by splitting the recording based on the presence of the target at the indicated pixel. This particular example was chosen as it exhibited an extremely high false-negative rate of 71.56%. While this particular instance was an outlier in the observed dataset, such false-negative errors are consistently observed in the dataset and require consideration during system design.

Another potential source of noise, and one which can potentially have the most impact on the proposed design is persistent non-ideal timing patterns in SPAD activation timings. These are the non-zero additive spatio-temporal patterns which can be observed at almost every laser pulse. An example of this issue, potentially resulting from systematic offsets in the timing counter or due to mismatched delay paths, can be seen in Figure 5.10. If the observed persistent non-ideal timing patterns are due to offsets in the timing counters, the removal of the counters in the proposed design would also remove the effect. If however, they result from unequal delays in the circuitry, this effect could systematically delay some pixels relative to

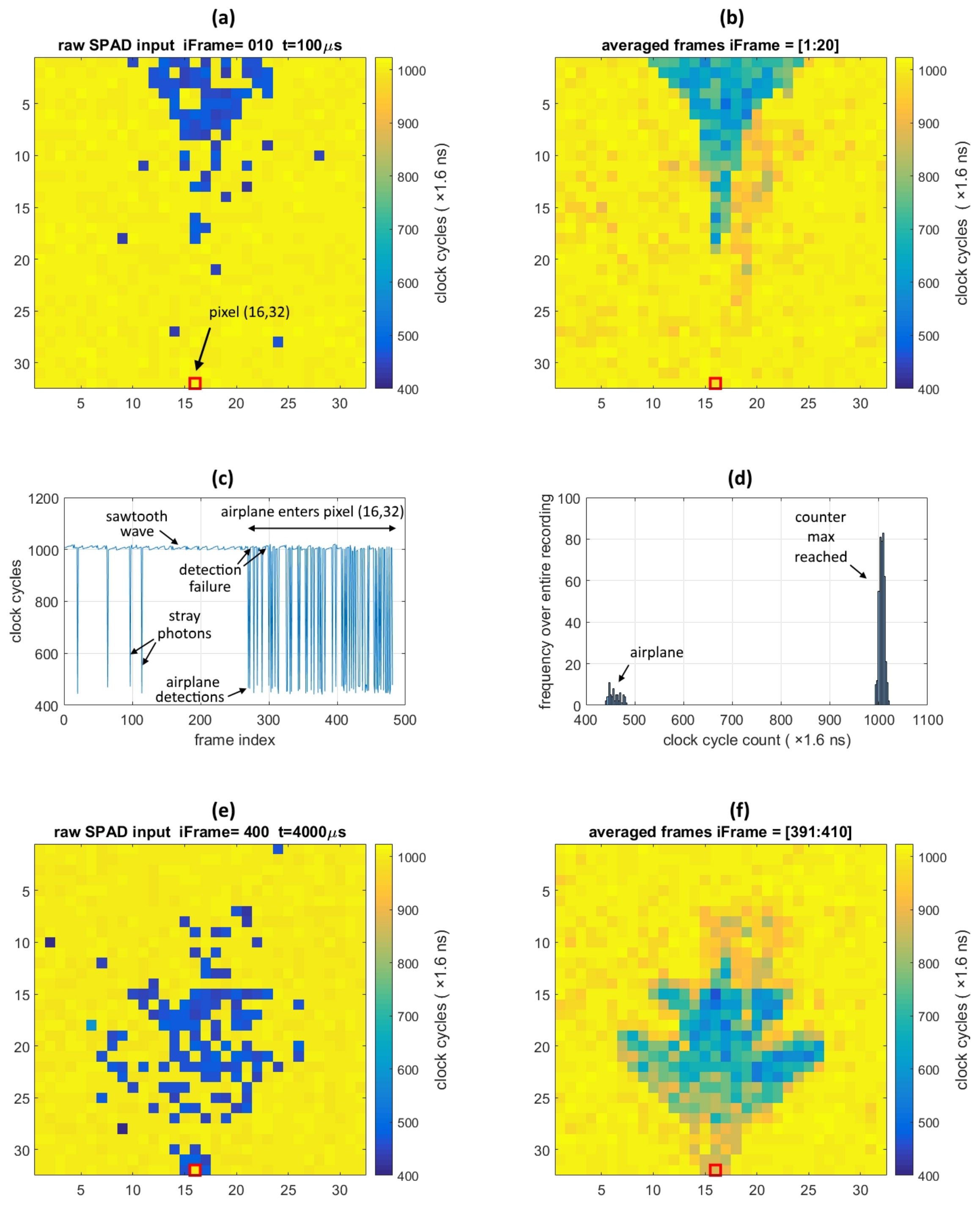


FIGURE 5.8: **SPAD cell activation across pixels and over time for a single pixels.** (a) Shows the raw SPAD input frame as the target enters the field of view. The indicated pixel at (16, 32) should, but for stray photons, consistently time out at 1023 clock cycles during this section of the recording. (b) Shows an image generated via averaging the first 20 frames of the recording. (c) The recorded latch time at the pixel at (16, 32) during the entire 481 frame recording. The plane passes over the pixel at frame 270. (d) Histogram of latch times at pixel at (16, 32) during the entire 481 frame recording.

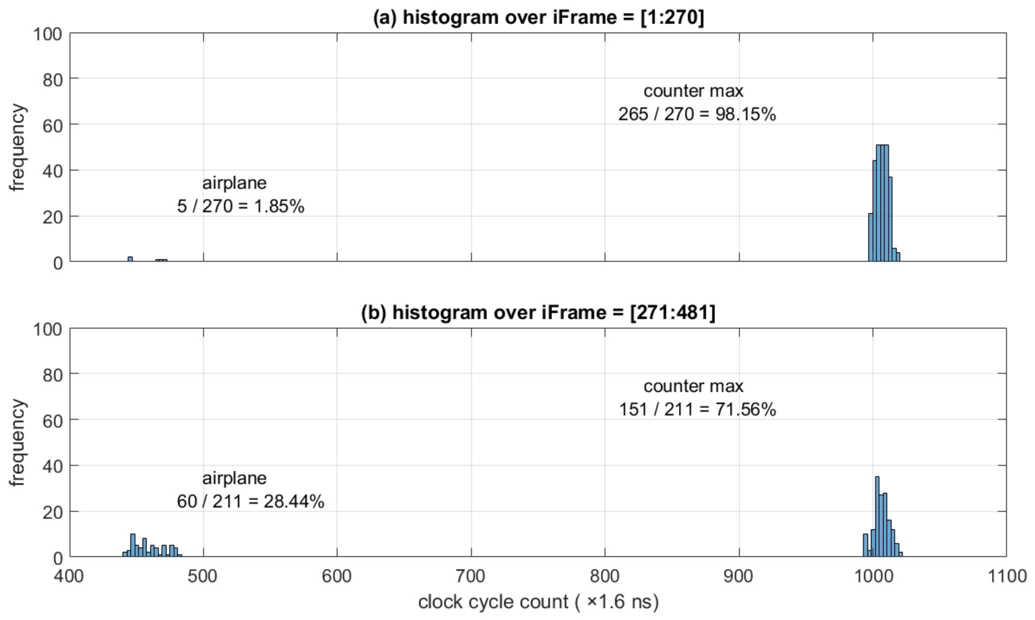


FIGURE 5.9: **Prevalence of stray photons versus missed detections for the recording shown in Figure5.8.** (a) Latching times for the section of the recording with the airplane not in front of pixel (16, 32) showing a false positive rate of 1.85%. (b) Latching times with the airplane in front of pixel (16, 32) showing an extremely high false negative rate of 71.56%.

others. The problem with such systemic delays can best be illustrated when combined with an equidistant object in the field of view i.e. an imager looking directly at a flat wall. Such an environment would (in the absence of all other noise sources) cause spurious spatio-temporal patterns to be detected by the proposed feature detecting AND gates across the entire wall due to the imperfect delays within the chip. This would result in many of the receptive fields generating feature events when they should in fact not generate any events. In the flat wall example, all the AND gates should latch, in theory, at precisely the same moment thus disabling the one-hot gate shown in Figure 5.6. This significance of this source of noise greatly depends on the details of the hardware implementation and the success of design solutions for mitigating this source of noise. However, the details of these circuit design choices are beyond the scope of this work.

Another source of noise in SPAD imaging data is highly noisy ’hot pixels’. In addition to being triggered by incoming photons from the laser and the environment, these pixels,

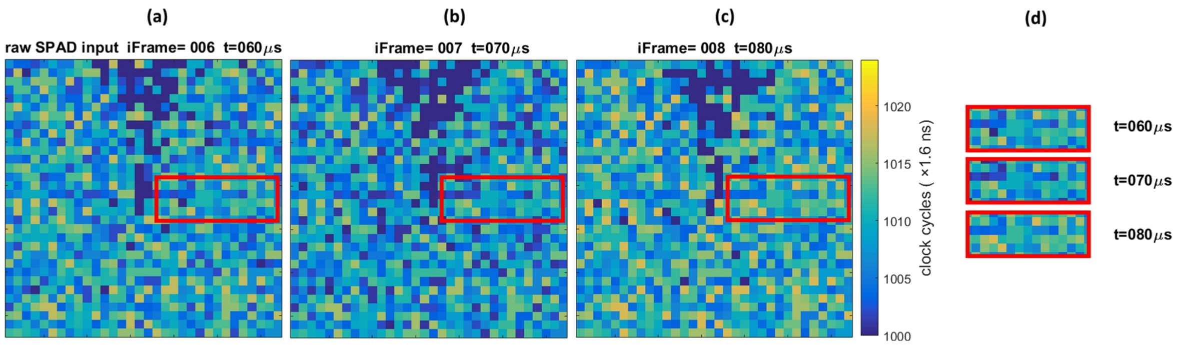


FIGURE 5.10: **Persistent background pattern SPAD timings likely due to hardwired delays in the imager** Panels (a),(b) and (c) show SPAD latch timings generated at sequential laser pulses. (d) The timings shown in the red boxes are from the sequence of frames in (a), (b) and (c). The consistent timing patterns generated be the imager can potentially generate spurious events in a local feature detection network such the First-AND system.

frequently latch randomly at a high rate regardless of the imaging environment. These non-ideal latching events are different to the case of stray photons originating from the laser pulse or photons from other light sources. The activation of these noisy pixels is not correlated with activation of their normally acting neighbors or the presence of any light, as would be expected if a common external noise source was triggering the random latching events. Figure 5.11 illustrates the behavior of these noisy pixels as captured in an image of 3600 averaged frames. Here the five marked noisy pixels on Figure 5.11(a) show an on average earlier arrival time/reduced depth compared to their immediate normal neighbors. The histogram of two of the noisy pixels, shown in (b) and (d), the noisy latch times which are conspicuous when compared to the normal functioning neighboring pixels, shown in (c) and (e). Note that the target high-speed airplane only activated the four examined pixels from clock cycle 370 to 485. The noisy pixels show higher activation at this distance relative to other distances and to the nearby non-noisy pixels suggesting an additive noise at work in the noisy pixels where in this case about a third of the correct on-target activations are in fact due to noise.

The effect of randomly latching noisy pixels is yet another non-ideality that must be handled by the First-AND feature detector. Fortunately the relative rarity of these random latching events, in comparison to other sources of noise such as detection failures, makes this source less of a concern. In addition, the additive and uncorrelated occurrence (across neighboring

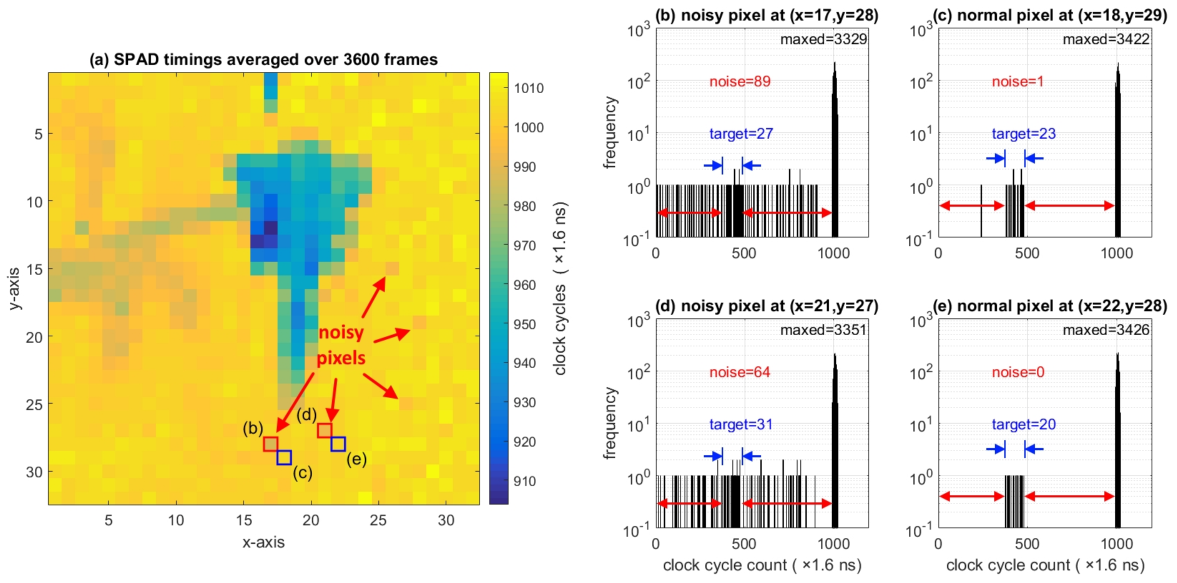


FIGURE 5.11: **Comparison between noisy pixels and normal pixels over all frames in a recording.** (a) Long exposure averaged frame highlighting five hot pixels and two nearby normal pixels. Panels (b) and (d) show SPAD latch times for two noisy pixel indicated in (a). Panels (c) and (e) show SPAD latch times for two nearby normal pixels indicated in (a).

pixels) of this noise source means that a competitive AND gate network can readily handle this form of noise.

The all or nothing behavior of the AND gate design is an advantage in this context. In the proposed four 8-input AND gate design, 8 noisy pixels of a non-winning AND gate must erroneously latch together and earlier than the true photon flight time from laser to sensor in order to set their AND gate before a ’correct’ competing AND gate which is viewing a truly closer section of the scene. This makes it highly unlikely for this noise source to generate false-positive First-AND events in the proposed design.

In contrast, to noisy pixels, ’dead pixels’ which rarely or never latch, can significantly harm the proposed design. This is because, unlike in a simple pixel-based imager, in the proposed receptive field-based convolutional design of the First-AND system, the detrimental effect of a single dead pixel expands with the receptive field size as shown in Figure 5.12. Here, an analog summing and comparator circuit which could activate on say, 7 out of 8 activated pixels, would provide the robustness to dead pixels but at the cost of a more complex design.

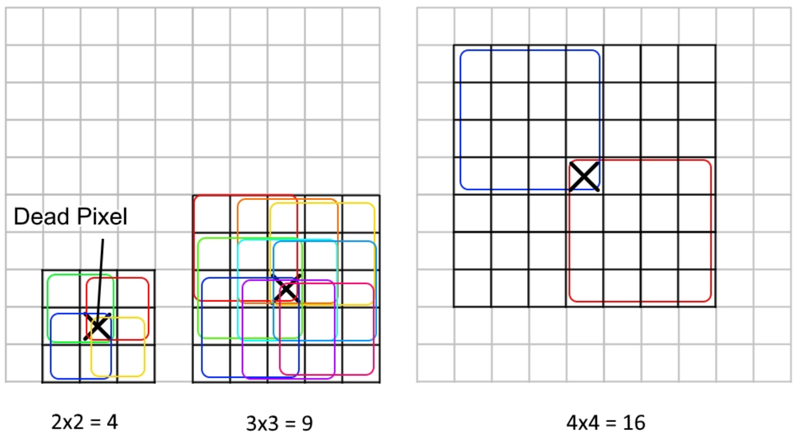


FIGURE 5.12: **Number of receptive fields impaired by a single dead SPAD at different receptive field sizes**X marks the dead pixel the black grid indicated the pixel region affected by the dead pixel and the colored squares indicate receptive fields affected by the dead pixel.

The selection of the simpler AND gates as feature detectors means that for the proposed 4×4 pixel receptive fields, each dead pixel will disable a maximum of 7×7 pixel region containing 16 receptive fields. Fortunately, no such low activation pixels were observed in SPAD imaging data.

With the information about the likely sources of noise, we now revisit the event generation method of the First-AND system in order to introduce robustness to these noise sources. In the ideal noise-free model of the SPAD sensor array, every time a newly detected feature is found to be different from the one already stored in memory, that feature (or more precisely its 2-bit address) should be transmitted via AER since in the absence of noise. This change must indicate an informative change in the field of view.

However as highlighted, the many noise sources observed in the dataset can act to erroneously result in a change in the detected feature. In order to make the proposed First-AND system more robust to these anticipated noise sources, a feature detection success counter is added such that every time a detected feature in a receptive field is the same as the one already in memory for that receptive field, the detection counter increments by one. Every time the newly detected feature is different from the one already in memory, the counter is decremented by one. If the feature detection counter of a receptive field reaches a pre-set threshold value, the receptive field creates a feature event and the counter is set to zero. Conversely, if the counter reaches zero after a decrement, the newly detected feature which caused the decrement replaces the old feature that was in memory. In this way, a constantly noiselessly detected feature will periodically send out a confirmatory feature event, whereas receptive fields where no feature consistently wins will not output any features. By decreasing the global threshold, we can decrease the number of times a feature must be detected before it triggers a feature event. This reduction in threshold increases the data-rate and allows features whose verity is less certain to be transmitted. Conversely, a higher threshold increases the certainty about the transmitted features and reduces the data-rate. In this way a global feedback control to the system can be implemented. While this aspect of the design was not explored it forms an avenue of investigation in future work.

5.2.4 Implementation of the First-AND Network in ASIC

The First-AND system described in Section 5.2.2 was implemented as part of a Complementary Metal-Oxide-Semiconductor (CMOS) based SPAD cells with supporting mixed-signal and digital Integrated Circuit (IC) chip design. The First-AND system was implemented as an Application-Specific Integrated Circuit (ASIC) using a Silterra High Voltage CL130H32 Process Technology by Dennis Delic based on the design developed by this student. The chip operates in time of flight mode, in this configuration the start of a new acquisition cycle is synchronized to a laser pulse being fired at the target (i.e. Flash LIDAR). The implemented 128×128 SPAD array contained 125×125 4×4 receptive fields each with 4 silicon digital AND based feature detectors consisting of North, South, East and West. The system features priority encoding, a 3-bit feature counter and an adjustable threshold for the detection of winning features. The network readout is implemented via the AER protocol allowing an asynchronous DTOF SPAD sensor array with real-time event-based feature extraction for 3D imaging applications. The inclusion of the 3-bit feature success counter and a globally adjustable feature count threshold provides robustness to noise while minimizing the readout of noise from the sensor. The adjustability of the threshold aim provides control over the output data-rate of the sensor with higher threshold values reducing the number of output events by

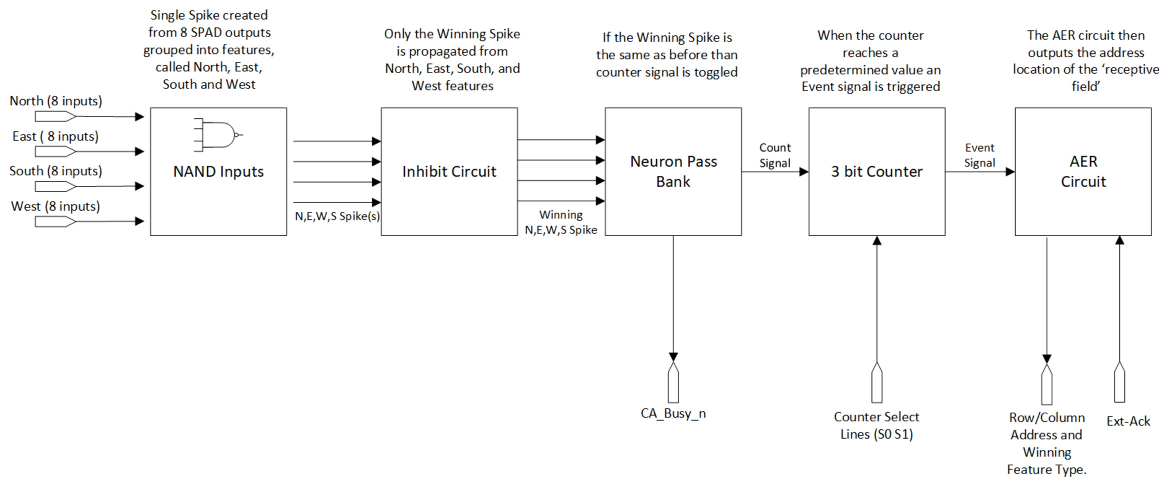

Single Spike created from 8 SPAD outputs grouped into features, called North, East, South and West
Only the Winning Spike is propagated from North, East, South, and West features
If the Winning Spike is the same as before than counter signal is toggled
When the counter reaches a predetermined value an Event signal is triggered
The AER circuit then outputs the address location of the 'receptive field'
North (8 inputs)
East (8 inputs)
South (8 inputs)
West (8 inputs)
NAND Inputs
N,E,W,S Spike(s)
Inhibit Circuit
Winning N,E,W,S Spike
Neuron Pass Bank
Count Signal
3 bit Counter
Event Signal
AER Circuit
CA_Busy_n
Counter Select Lines (S0 S1)
Row/Column Address and Winning Feature Type.
Ext-Ack

FIGURE 5.13: **Functional block diagram of a single receptive field of the ASIC implemented First-AND Network.**

requiring a higher number of successive activations by the same feature in the same receptive field. The implementation also removes the need for a high bit counter while providing SPAD noise immunity through receptive field inter-connectivity. The novel design seeks to circumvent pitch and array size limitations and problems associated with the miniaturization of conventional SPAD sensor arrays. These benefits come at the cost of added connectivity complexity between the overlapping receptive fields and the interconnected SPADs. However, these challenges were addressed and resolved in the integrated circuit design which is beyond the scope of this work.

Figure 5.13 shows a functional representation of each receptive field circuit or cell. The 16 SPADs connections are not shown. When the SPADs avalanche or fire, they are synchronously latched to an on-chip CLOCK (configured via PLL or fed via external CLOCK signal). Within each receptive field whichever feature (North, South, East, West) detects its particular shape first and blocks other features from activating.

In this implementation, if the winning feature is the same as the previous one (i.e. from the previous laser pulse or acquisition cycle) the 3-bit counter is then incremented. Once the counter reaches a predetermined threshold it triggers and event. The value of the threshold is set by the input S0/S1 lines shown in Figure 5.13 with the default value is set to 6. The event is indicated by the CA_Busy_n output line on the chip. When an event is generated, the encoded row and column address of the location of this receptive field is output and read off-chip as well as the class of winning feature North, South, East or West. When the user has read the address/data information, and acknowledgment is sent to the ext-ack input line which resets the counter and releases the CA_Busy_n line. The event generator in a particular receptive field is reset once it receives row and column acknowledge signals as well as a global acknowledge signal which is generated off-chip. Although the acquisition cycle is synchronized to CLOCK, events are asynchronously generated off the chip. Thus, while the SPAD latching events are synchronized to a high-speed (600 MHz) clock, the event generation circuit operates asynchronously and the on-chip arbiter processes the order of events as they occur, and events can be generated asynchronously and as such, it is possible that not all events will be captured and read by the monitoring FPGA between laser pulses.

Figures 5.14 and 5.14 show the layout of a single SPAD pixel and a $4\times$ 4 pixel receptive field respectively while Figures 5.16 shows the fabricated First-AND chip.

5.2.5 Training Binary Feature Extractor Networks

In order to extract higher-level spatio-temporal patterns generated by the SPAD imager, an event-based feature extraction network was trained on the event-based dataset. The Feature Extraction using Adaptive Selection Thresholds (FEAST) method used was detailed in Chapter 4. This simple event-based unsupervised learning algorithm uses competition between adaptive neurons to generate balanced network activation in response to incoming data.

As detailed in Chapter 4, the algorithm operates via of the network's selection thresholds and the gradual adaptation of the weights of the network weights to the observed local spatio-temporal patterns results in a trained network where all neurons fire at approximately equal rates given a spiking training dataset. By balancing network activation, the FEAST algorithm ensures that the neurons in the feature extractor network represent the most commonly observed spatio-temporal patterns resulting in a feature set that best represents the underlying

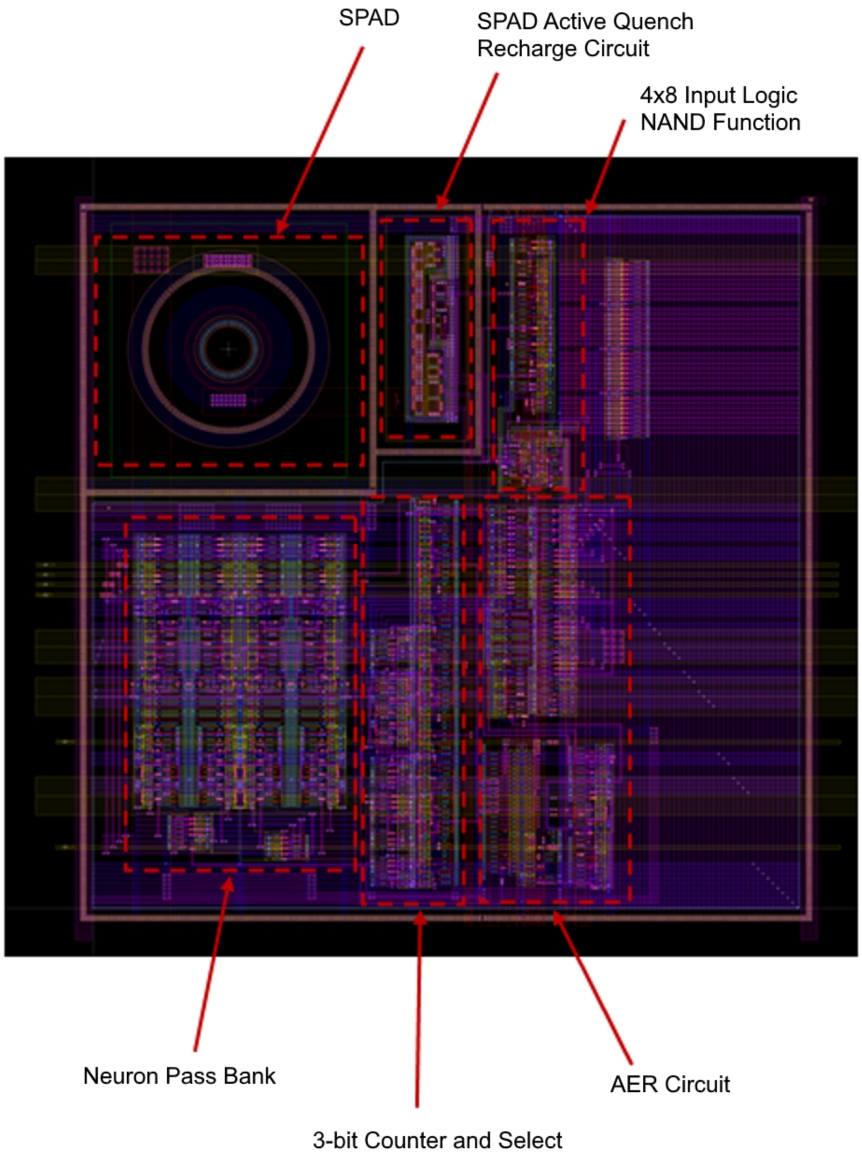

SPAD
SPAD Active Quench Recharge Circuit
4x8 Input Logic NAND Function
Neuron Pass Bank
AER Circuit
3-bit Counter and Select

FIGURE 5.14: **Single Receptive Field Cell showing a SPAD sensor and Circuitry Blocks, size 75 μm $\times$ 75 μm.** Here, in the ASIC implementation, a 4×8 input logic NAND function is used to implement the first half of the First-AND design where four AND gates compete in time to be the first to detect one of four features. This functional behavior of this subsystem is detailed in Section 5.2.2 and illustrated in Figure 5.5. The neuron pass bank subsystem shown in the bottom left of the panel implements the second half of the First-AND design which is functionally described in Figure 5.6. The neuron pass bank determines whether the current detected feature is the same as the previous detection. Courtesy of the Department of Defence Science and Technology. Implemented by Dennis Delic based on designs by this student.

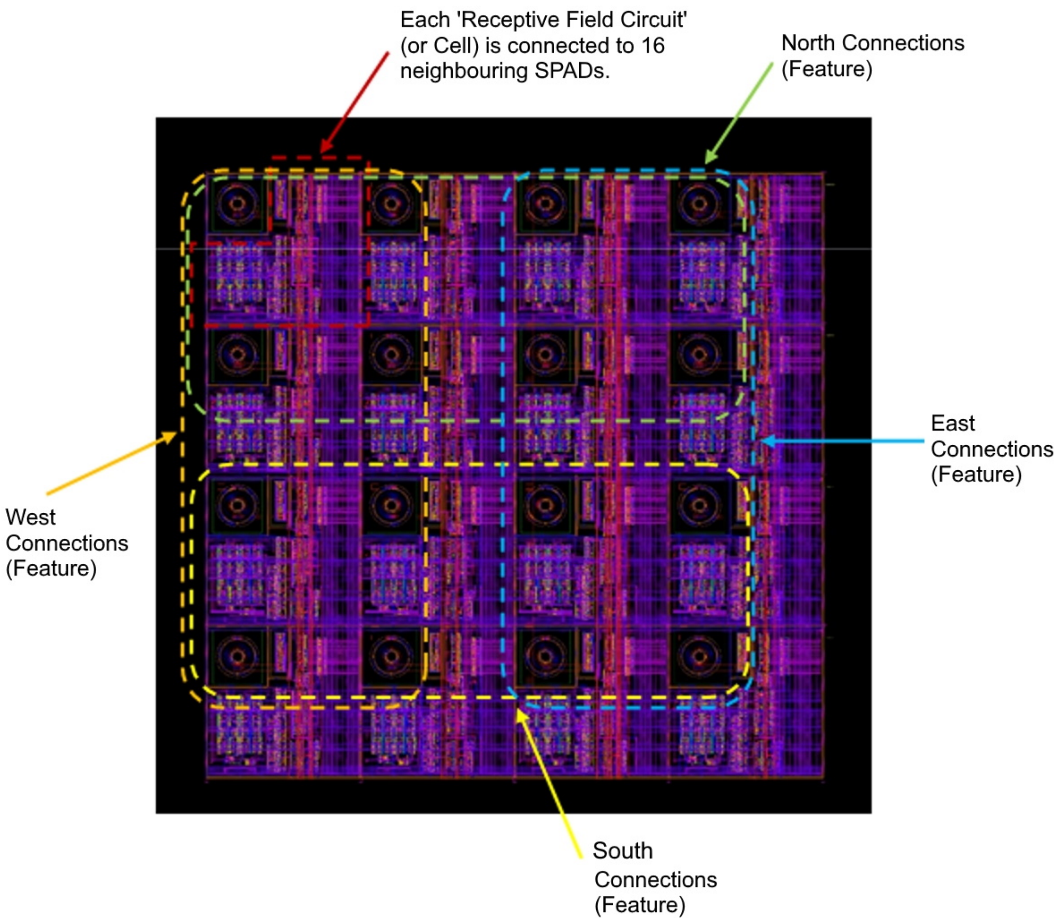

Each 'Receptive Field Circuit' (or Cell) is connected to 16 neighbouring SPADs.
North Connections (Feature)
East Connections (Feature)
West Connections (Feature)
South Connections (Feature)

FIGURE 5.15: **Each receptive field is comprised of (connected to) 16 (4$\times$4) neighboring SPAD detectors 30μm in diameter (5μm active area).** Each of the north, south, east and west connected blocks route to one of the four AND gates of the First-AND system as described in Section 5.2.2 and illustrated in Figure 5.4. Courtesy of the Department of Defence Science and Technology. Implemented by Dennis Delic based on designs by this student.

training data. Figure 5.18 shows an example of the FEAST algorithm training a 16 neuron network on the four-polarity First-AND event-based airplane dataset.

When implementing the FEAST algorithm, the best fitting neuron to an incoming ROI pattern must be determined. This can be achieved most directly via dot product operation which requires $D_1 \times D_1 \times N_1$ multiplication operations followed by N_1 summation operations. However to reduce the hardware resource requirements for this neuron matching operation and to remove the need for hardware-implemented multipliers, the continuous-valued feature weights $\boldsymbol{w}_1$ shown in 5.18 are converted to binary-valued features. Methods for binarizing

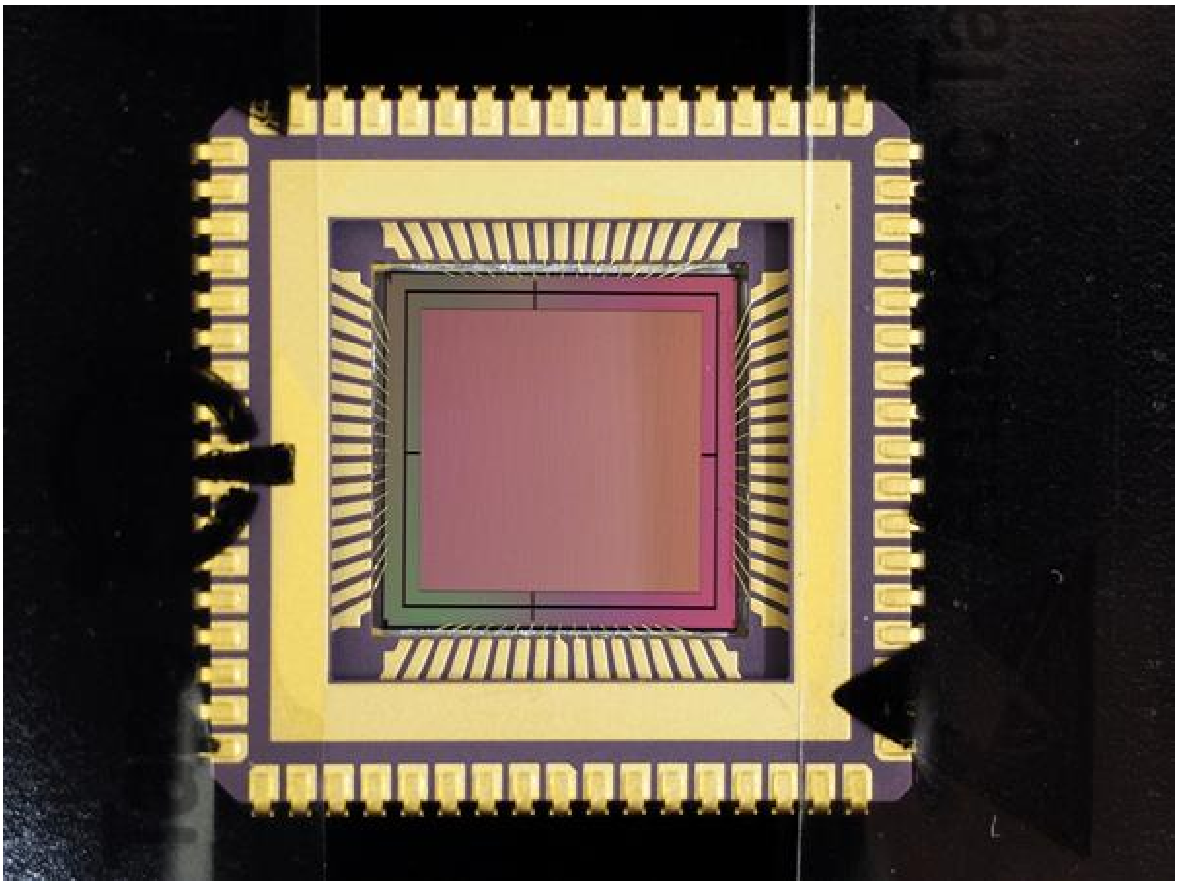


FIGURE 5.16: **Manufactured 128×128 First-AND Event Based CMOS ASIC.** Courtesy of the Department of Defence Science and Technology. Implemented by Dennis Delic based on designs by this student.

images include the Otsus method [128], Kittler and Illingworth's minimum error thresholding method [129] and the Adaptive Binarization method [130]. Here we use a much simpler equal activation method where for each neuron, the number of 1 valued pixels m is equal. During training, at each presentation, the largest m weights on each neuron are set to one. This method of equal neuron activation allows the unbiased use of AND gates instead of multipliers. When using AND gates as multipliers, if the number of active pixels per feature is not equal, neurons with a lower number of on pixels would activate on more patterns than those with a larger number of on pixels regardless of fitness to the input pattern.

An alternative approach to using AND gates is the use of XNOR gates together with binary $\{1, -1\}$ weighted neurons to replace multipliers as used in [131]. This allows other non-equal

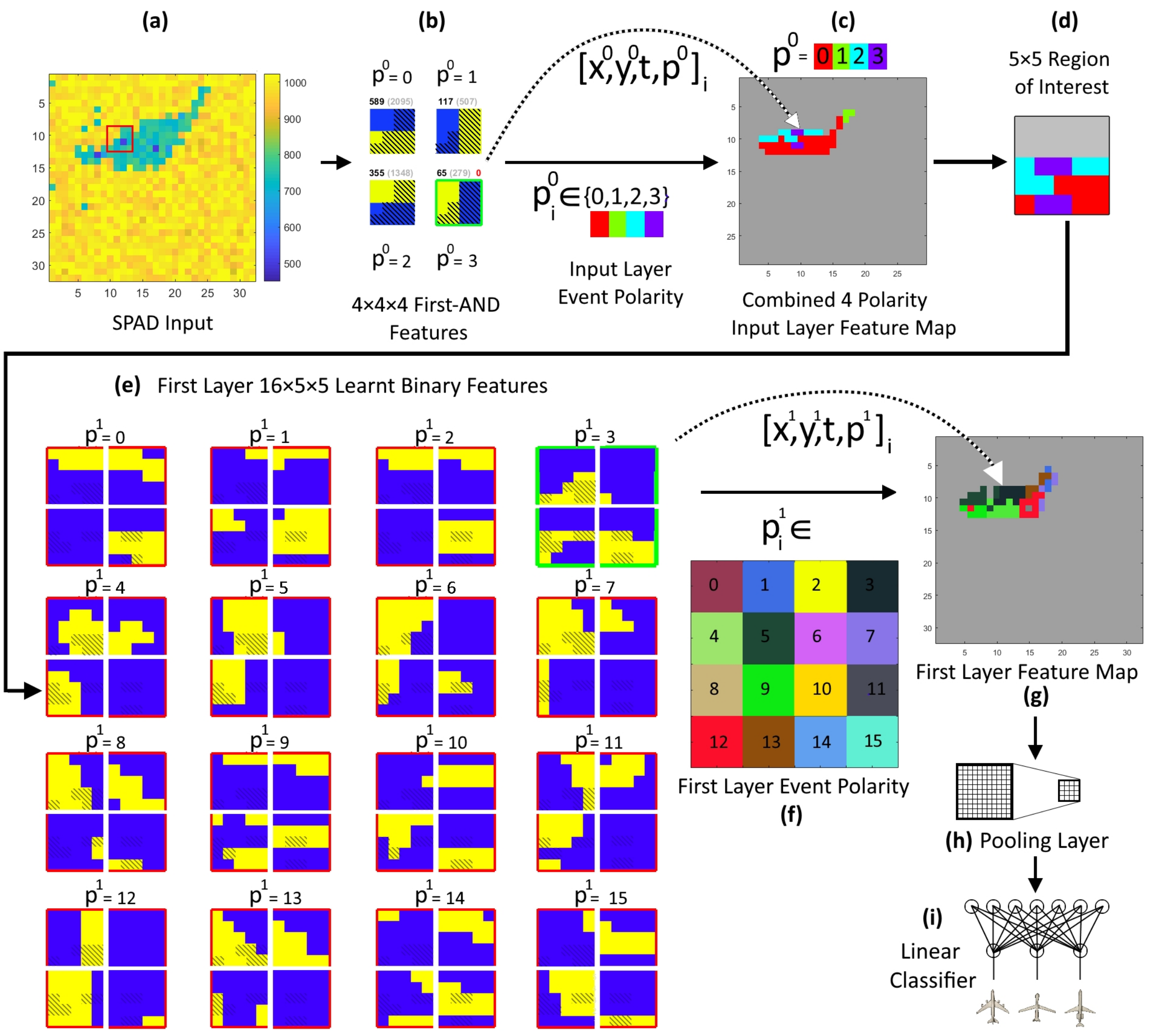


FIGURE 5.17: **Block diagram of the end-to-end First-AND event-based processing system.** (a) Shows the raw image generated by the SPAD sensor in time of flight mode as a B-2 model enters the field of view. The red box indicates the receptive field of the current generated event. (b) Shows the four First-AND features and their binary bar-shaped weights. Superimposed are the state of the latched SPAD pixels at the moment the First-AND feature generates an event (diagonal black lines). The third feature is the first AND gate to latch disabling the others and passing its event to the next layer. (c) Shows $\boldsymbol{S}_i^0$, the binary-valued four-polarity time surface with activation over τ_0 seconds. This surface serves as a feature map for the next layer of processing. (d) Shows the 5×5 Region of Interest ROI extracted from $\boldsymbol{S}_i^0$. (e) Shows the 16 four-polarity binary event-based features which operates on $\boldsymbol{S}_i^0$. (f) Shows the encoding of the 16 features. (g) Shows the 16 polarity binary-valued time surface $\boldsymbol{S}_i^1$. Panels (h) and (i) show the pooling and classification layers respectively.

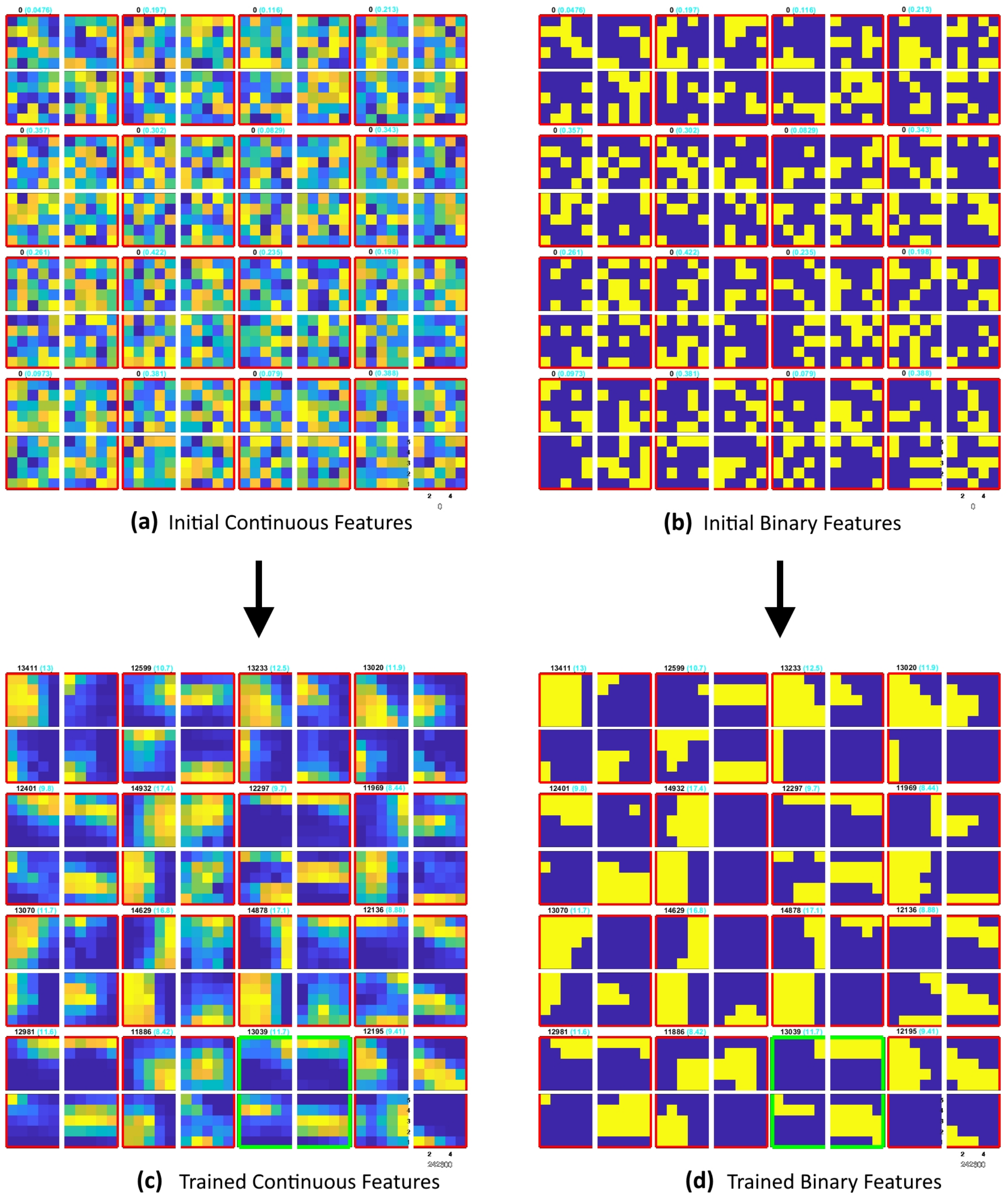


FIGURE 5.18: **Generating 16 binary-valued features on a four-polarity event-based dataset.** (a) Shows the initial random weights of the sixteen four-polarity 5×5 features $\boldsymbol{w}_1$. (b) The binary weighted feature set $\ddot{\boldsymbol{w}}_1$ with the number of on pixels per feature $m = 32$. (c) Final state of the continuous features $\boldsymbol{w}_1$ after training. (d) Final binarized feature set $\ddot{\boldsymbol{w}}_1$.

ALGORITHM 5.1
Event-based Surface Generation

```
Require: $e_i = [x_i, y_i, p_i, t_i]^T, i \in \mathbb{N}$
Ensure: $S_i^0, i \in \mathbb{N}$
  $S_i^0 \Leftarrow \mathbf{0}$, $T_i^0 \Leftarrow \mathbf{0}$
  for each event $e_i$ do
      $T_i^0(x_i, y_i, p_i) \Leftarrow t_i$
      if mod$(i, I)$=0 then
          $S_i^0 \Leftarrow (t - T_i^0 < \tau_0)$
      end if
  end for
```

activation binarization methods to be used, however, the use of XNOR gates incurs additional hardware resources compared to AND gates, motivating our use of the latter.

After training, the finalized binary features can be used for inference on the input event stream as an event-based convolutional layer. This results in a first layer feature map S_i^1 which can be sampled and processed by the classifier in an event-based manner. In this work, in order to isolate the gains provided by the event-based convolution layer, the input events stream is also converted to an input event surface S_i^0 to be sampled and processed by the classifier in an identical manner to the feature map. The generation of the input surface S_i^0 and feature layer surface S_i^1 a detailed in Algorithm 5.1 and 5.2 respectively.

5.2.6 Alternative Event-based Methods: On-Off Bi-polar and Uni-polar Events

In this section, we introduce two alternative methods for comparison to the proposed and implemented First-AND method. In the first method, the difference between consecutive SPAD frames is converted into a sparse On-Off event stream using a simple thresholding operation analogous to those used in other event-based sensors. In the second method, we then augment these On-Off events by introducing uni-polar and bi-polar events. Event-based sensors generally operate by converting an analog signal (typically pixel illumination in vision) to a sequence of events via processing through a change detection and thresholding

ALGORITHM 5.2
Event-based Binary Feature Convolution

```
Require: $\boldsymbol{e}_i = [x_i, y_i, p_i, t_i]^T, i \in \mathbb{N}$
Ensure: $\boldsymbol{S}_i^1, i \in \mathbb{N}$
  $\boldsymbol{S}_i^1 \Leftarrow \mathbf{0}$, $\boldsymbol{T}_i^0 \Leftarrow \mathbf{0}$, $\boldsymbol{T}_i^1 \Leftarrow \mathbf{0}$, $\boldsymbol{ROI}_i \Leftarrow \mathbf{0}$
  for each event $\boldsymbol{e}_i$ do
      $\boldsymbol{T}_i^0(x_i, y_i, p_i) \Leftarrow t_i$
      $\boldsymbol{ROI}_i(x, y) \Leftarrow (t - \boldsymbol{T}_i^0(x + x_i, y + y_i, 1) < \tau_0)$,
       where $x = [-r_1 : r_1]$ and $y = [-r_1 : r_1]$
      for each neuron $n \in 1..N_1$ do
          $d(n) \Leftarrow \sum_{x,y} \boldsymbol{ROI}_i(x, y) \wedge \ddot{\boldsymbol{w}}_1(x, y, n)$
      end for
      $q_i \Leftarrow \mathrm{argmax}_n d(n)$
      $\boldsymbol{T}_i^1(x_i, y_i, q_i) \Leftarrow t_i$
      if $\mathrm{mod}(i, I)$=0 then
          $\boldsymbol{S}_i^1 \Leftarrow (t - \boldsymbol{T}_i^1 < \tau_1)$
      end if
  end for
```

circuit as described in chapter 1. For the DTOF SPAD imager, this analog signal is the photon time of flight information Z_k which encodes detected depth at the kth laser pulse. Algorithm 5.3 details the generation of On-Off events from the photon time of flight data.

This approach to event generation is more straightforward than the First-AND approach and has the advantage of providing single pixel resolution which is missing in the receptive field based First-AND method. The trade-off however, is the need for measurement and storage of high-resolution timing data Z_k at each pixel that the First-And approach avoided and which increases the per pixel hardware resource cost. The hardware requirements for implementing On-Off events and derived solutions from SPAD DTOF data will be the investigated in future work and is beyond the scope of this work.

Having generated the On-Off events used in standard event-based sensors, we now augment these with two additional event polarities. These two event polarities encode shape invariant local change information via a novel approach. The two event polarities which we name uni-polar and bi-polar events are used to generating a combined On-Off-Bi-polar and Uni-polar (OOBU) event stream.

ALGORITHM 5.3 On-Off Event Generation

```
Require: $\boldsymbol{Z}_k, k \in \mathbb{N}$
Ensure: $\boldsymbol{f}_i = [x_i, y_i, p_i, t_i]^T, j \in \mathbb{N}$
  $\boldsymbol{\Sigma}_k \Leftarrow \boldsymbol{0}$
  $\Delta \boldsymbol{Z}_k \Leftarrow \boldsymbol{0}$
  for each frame $k > 1$ do
      $\Delta \boldsymbol{Z}_k \Leftarrow \boldsymbol{Z}_k - \boldsymbol{Z}_{k-1}$
      $\boldsymbol{\Sigma}_k \Leftarrow \boldsymbol{\Sigma}_{k-1} + \Delta \boldsymbol{Z}_k$
      for each pixel at (x,y) do
          if $|\boldsymbol{\Sigma}_k(x, y)| > \theta$ then
              $i \Leftarrow i + 1$
              $p_j \Leftarrow \mathrm{sgn}(\boldsymbol{\Sigma}_k(x, y))$ , (ON-OFF Events)
              $\boldsymbol{f}_i \Leftarrow [x_i, y_i, p_i, t_i]^T$
              $\boldsymbol{\Sigma}_k(x, y) \Leftarrow 0$
          end if
      end for
  end for
```

The bi-polar and uni-polar events are generated using a simple recent event counting operation over the surrounding 8 pixels around the current event. In this approach, which is described in Algorithm 5.4, if the recent events in the neighboring 8 pixels are all On or all Off and their number exceeds a threshold ϕ_1, then a uni-polar event is generated. If however, both On and Off polarities are present, then if both the On and Off counts are above a threshold ϕ_2 then a bi-polar event is generated. In this work the values $\phi_1 = 2$ and $\phi_2 = 1$ were selected through observation of the data. A detailed investigation of different threshold selections will be the subject of future work.

Since only the local event count is considered, the orientation or structure of recent events does not matter. This results in a unique local feature that is invariant to feature shape, allowing a wide range of different shapes to generate the same event types while still capturing critical local activation information.

An example of the On-Off and the uni-polar and bi-polar event streams are shown in Figure 5.19 demonstrating that On-Off event streams faithfully capture the salient spatio-temporal features of the target while the uni-polar and bi-polar events combine local features in a manner that provides distinct, higher scale information to the down-stream processor.

ALGORITHM 5.4
On-Off, Bi-polar and Uni-polar (OOBU) Event Generation

```
Require: $\boldsymbol{f}_i = [x_i, y_i, p_i, t_i]^T, i \in \mathbb{N}$
Ensure: $\boldsymbol{g}_j = [x_j, y_j, p_j, t_j]^T, j \in \mathbb{N}$
  $\boldsymbol{S}_i = \boldsymbol{0}$
  for each event $\boldsymbol{f}_i$ do
      $\boldsymbol{T}_i(x_i, y_i, p_i) = t_i$
      $\sigma^+ \Leftarrow \sum_{x,y} (t - \boldsymbol{T}_i(x, y, 1) < \tau)$,
      $\sigma^- \Leftarrow \sum_{x,y} (t - \boldsymbol{T}_i(x, y, -1) < \tau)$,
       where $x \in \{x_i - 1, x_i, x_i + 1\}$ and $y \in \{y_i - 1, y_i, y_i + 1\}$
      if $(\sigma^+ > \phi_2) \wedge (\sigma^- > \phi_2)$ then
          $j \Leftarrow j + 1$
          $p_j \Leftarrow 2$ , (Bi-polar Event)
          $\boldsymbol{g}_j \Leftarrow [x_j, y_j, p_j, t_i]^T$
      else if $\big((\sigma^+ > \phi_1) \wedge (\sigma^- = 0)\big) \vee \big((\sigma^- > \phi_1) \wedge (\sigma^+ = 0)\big)$ then
          $j \Leftarrow j + 1$
          $p_j \Leftarrow 3$ , (Uni-polar Event)
          $\boldsymbol{g}_j \Leftarrow [x_j, y_j, p_j, t_i]^T$
      end if
  end for
```

To illustrate the power of uni-polar and bi-polar events in capturing high-level salient feature information, a simple three-class classification problem is shown in Figure 5.20. Here two linear thresholds are combined to separate the three classes using only the event polarity count information. Figure 5.20(a) shows that simple examination of the On and Off event counts is not a useful method of discriminating the three example classes. For this example, the best accuracy achievable using two separating On-Off event count ratio thresholds is 44.13% accuracy which is only slightly above chance 33.33%. In contrast, as shown in Figure 5.20(b), when bi-polar and uni-polar events are generated from the On-Off event stream, a simple bi-polar uni-polar event count test results in 92.83% accuracy when two event count ratio thresholds are used in combination. While this extremely simple event counting method does not extend to more challenging tasks (such as the full 15 class SPAD dataset) this simple example illustrates the significant discriminatory power of OOBU events. In this work, we show that when OOBU events streams are processed in a more sophisticated manner, they

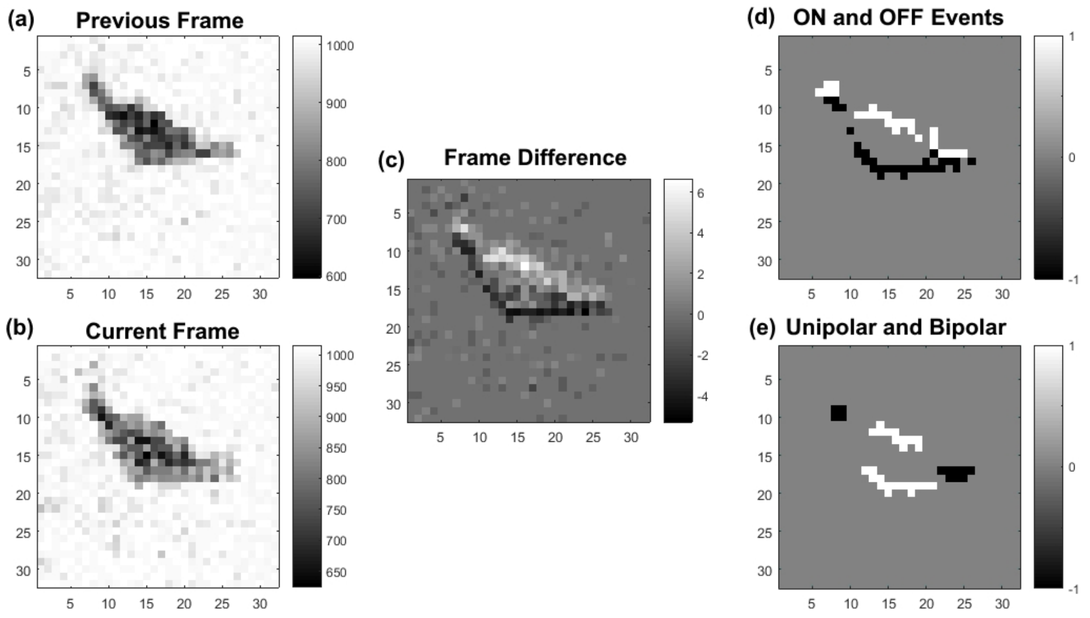


FIGURE 5.19: **Generating On-Off-Bi-polar and uni-polar events from SPAD sensor data.** Panels (a) and (b) show the previous and current captured frames from the SPAD sensor respectively. (c) Shows the frame difference between the current and previous frames. (d) Shows the On and Off events produced via thresholding of the events at $\theta = +/-2$. (e) Shows the uni-polar and bi-polar events generated via Algorithm 5.4.

outperform other types of event streams and produce the highest performing results of the dataset.

We now combine the OOBU events generated from the frame-based SPAD dataset into a single event stream and process it through the same surface generation and feature extraction algorithm described in Algorithm 5.1 and Algorithm 5.2 using identical training architecture and learning parameters. In doing so we are able to combine the information from the On-Off, uni-polar and bi-polar event streams into a single feature extractor network. Figure 5.21 shows the remarkable spatio-temporal patterns extracted at each network size from the SPAD OOBU event-based data stream. Figure 5.21 demonstrates how the FEAST algorithm extracts the dominant patterns in the dataset for any given network size and how the information contained in the On, Off uni-polar and bi-polar event streams can be combined to provide powerful discriminatory features.

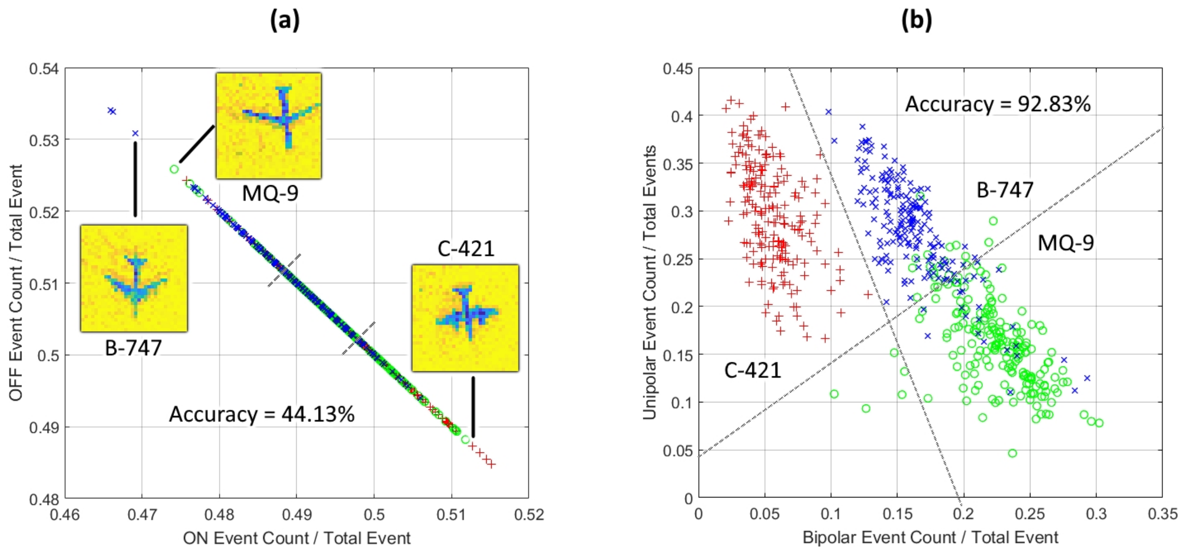


FIGURE 5.20: **Separability of an example three class problem using only event polarity counts.** (a) Plots the ratio of Off event vs On events for each recording for three example airplane classes. The classes are separated using a combination of two thresholds resulting in 44.13% accuracy. (b) The same test is performed using bi-polar and uni-polar event ratios. Again two thresholds are used to separate the three classes this time resulting in 92.83% accuracy.

5.2.7 Pooling, Surface Sampling and Classification

After the feature extraction operation performed by FEAST is complete, the target region of size $A_x \times A_y$ on the time surface is selected via a surface summation and thresholding operation described in Chapter 3 and implemented for real-time GPU based platforms in [132].

After the $A_x \times A_y$ target region is selected, the variable-sized 2D area from the surface must be mapped to the statically sized classifier input layer. To perform this mapping we explore two alternative methods. In the first method, which we call 1D pooling, the $A_x \times A_y$ region is summed across rows and columns resulting in two one-dimensional vectors $\boldsymbol{V}_x$ of size $A_x \times 1$ and $\boldsymbol{V}_y$ of size $A_y \times 1$. These two vectors are then re-sampled to two $L \times 1$ vectors. To speed up and simplify these re-sampling operations, the input data was first cropped or zero buffered and resampled using a zero-order-hold operation which was implemented via pre-calculated Look Up Table. This hardware optimized method was introduced in [132]. In the second method, which we call 2D pooling, the $A_x \times A_y$ target region is re-sized to a

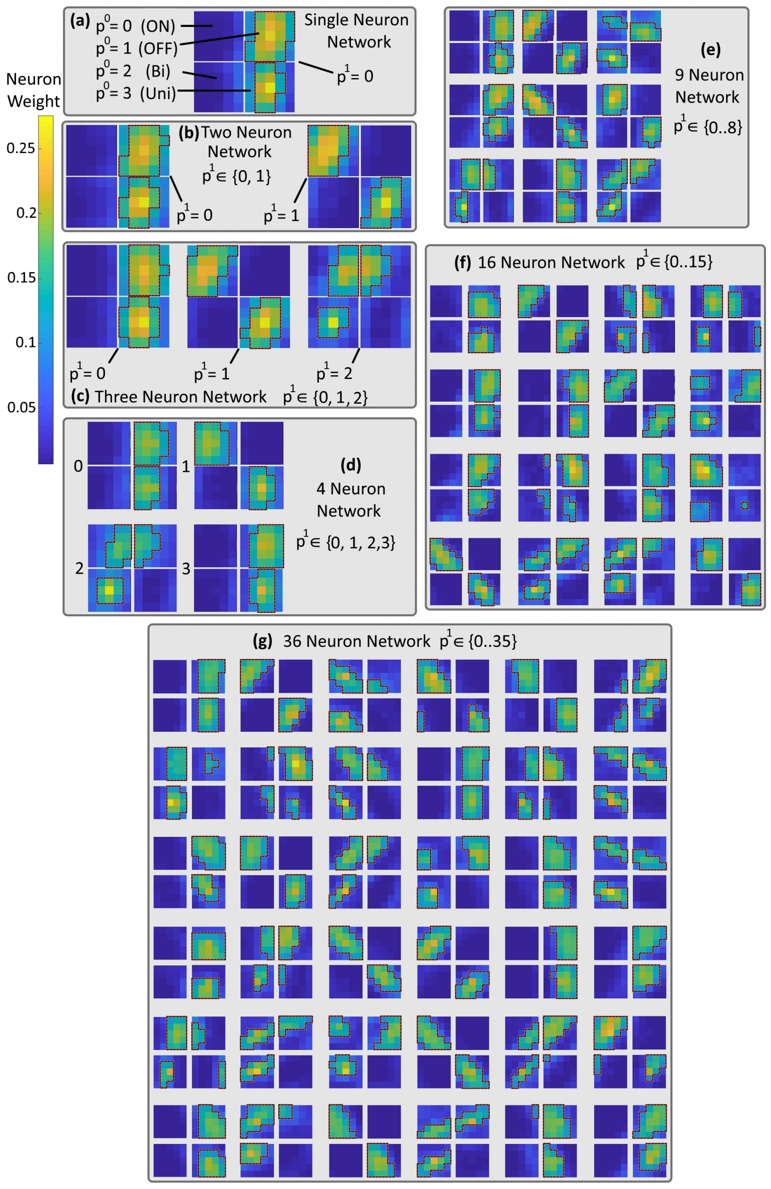


FIGURE 5.21: **Trained OOBU neuron weights across network sizes.** (a) Shows a trivial single-neuron network. The four images represent the On ($p^0 = 0$), Off ($p^0 = 1$), bi-polar ($p^0 = 2$) and uni-polar ($p^0 = 3$) feature weights of the single neuron ($p^1 = 1$). This single neuron network simply serves to show the dominant spatio-temporal pattern present in the dataset which is Off events ($p^0 = 1$) occurring alone ($p^0 = 3$). (b) A two neuron network ($p^2 = 0, 1$) shows the next most dominant observed pattern is the On event ($p^0 = 1$) occurring alone ($p^0 = 3$). Note that the first neuron in this network is nearly identical to that in (a). (c) The third most dominant pattern is the On and Off events occurring together in a diagonal pattern. (d) The fourth most dominant pattern is a second variant of the Off event occurring alone on the right-hand side of the ROI. Note that the presence of this variant results in the first neuron being trained in a complementary manner with the Off events occurring on the left-hand side of ROI such that the summation of neuron 1 and 4 would approximately equal the single dominant neuron in (a). Panels (e), (f) and (g) show networks of 9, 16 and 36 neurons with increasingly complex features.

two-dimensional image of size $L \times L$. For this method a 2D resample function using linear interpolation between adjacent values was used.

The 1D pooling method significantly simplifies the implementation of the resampling and also provides the added benefit of a classifier with a smaller input layer. However the 2D pooling method captures significantly more information and as we show in this work, results in higher recognition performance in most cases, creating design trade-offs that require investigation. In a similar way, the size of the classifier input layer which is $m = L \times 2$ in 1D case, and $m = L \times L$ in the 2D case, can also affect the accuracy. Extreme pooling of the target region into a pool size, of say $L = 1$, results in significant spatial information loss while also greatly simplifying hardware implementation. Conversely, less pooling with a larger L captures greater spatial information while increasing hardware resource requirements. For this reason, in this work, we perform all trials over a range of pool sizes ($L \in \{1, 2, 3, 4, 6, 8, 12, 16, 24\}$) to investigate these effects and provide useful design guidelines for hardware implementation.

Another important hyper-parameter in the operation of the feature extraction and classification system is the frequency of surface or image sampling for the processing by the classifier.

For the frame-based data, the time interval between classification operations was selected as 80 microseconds or every 8th laser pulse. This time interval resulted in a total of 1.22 million classification operations over the entire dataset or approximately $50.51 +/- 8.07$ classification operations per recording. This total number of operations was then used to normalize the number of classification operations on the event-based dataset to provide an approximately equal number of input samples to the classifier enabling an unbiased comparison of the frame-based and event-based systems. For the First-AND event streams, keeping the total number of classification operations constant results in an inter classification event interval of 51 events. For the On-Off and OOBU events, the interval between classification operations becomes 74 and 201 events respectively. By operating the classifier in this event-based manner, the rate of processing becomes dependent on the level of salient change in the field of view as opposed to constant in the frame-based approach.

All classification tests in this work were performed on the full 15 airplane, 24000 recording augmented dataset. The dataset was split randomly into a 21600 recording (90%) training set and 2400 (10%) test set. All tests were repeated over n = 20 trials. The original frame-based dataset was converted to the equivalent event-based datasets via the methods described in Section 5.2. All tests were performed using a simple linear classifier. The input to the linear classifier consists of a resized target region from a sampled time surface ($\boldsymbol{S}_i^0$ or $\boldsymbol{S}_i^1$) which is vectorized into a $1 \times m$ input vector $\boldsymbol{u}$. where, $m = L^2$ for the 2D pooling case and $m = 2L$ for the 1D pooling case. The output of the linear classifier is a $1 \times n$ predicted output vector $\hat{\boldsymbol{v}}$ with n being the number of output classes which in this case is 15. The predicted output vector $\hat{\boldsymbol{v}}$ is calculated via:

$$\boldsymbol{v} = \boldsymbol{W}\hat{\boldsymbol{u}} \tag{5.1}$$

where $\boldsymbol{W}$ is the trained $m \times n$ weight matrix.

To determine the winning output class j during inference, an argmax operation is performed on the predicted output vector $\boldsymbol{v}_i$:

$$j = \text{argmax}_{i \in [1..n]}(\hat{\boldsymbol{v}}_i) \tag{5.2}$$

For the process of training and testing, all sampled input vectors u are concatenated to form the $o \times m$ input matrix $\boldsymbol{U}$ where o is the number of samples over the entire dataset. A corresponding $o \times n$ ground truth output matrix $\boldsymbol{V}$ is also generated using a 'one-hot coding' scheme with the actual class for each sample set to 1 and all others set to zero.

The input matrix is then split into the 21600 recording training input matrix, $\boldsymbol{U}_a$ and 2400 recording testing input matrix $\boldsymbol{U}_b$. The corresponding output matrix $\boldsymbol{V}$ is similarly split into $\boldsymbol{V}_a$ and $\boldsymbol{V}_b$. The classifier is trained via a calculation of the pseudoinverse solution to the weights mapping the input activation layer or feature layer to the output classes as given by:

$$\boldsymbol{W} = (\boldsymbol{U}_a^T \boldsymbol{V}_a)^T (\boldsymbol{U}_a^T \boldsymbol{U}_a + \lambda \boldsymbol{I}) \tag{5.3}$$

where $\boldsymbol{I}$ is the identity matrix and $\lambda = 0.1$ is the regularization factor used for applying ridge regression [133]. In cases where the pseudoinverse operation could not be performed in a single pass (due to memory constraints) the equivalent online method was used [118]. In either approach, the training operation is deterministic and repeatable such that the source of variance in classification accuracy is exclusively due to the feature extraction operation and the random splitting of the dataset.

5.2.8 Event-based Processor Implementation on FPGA

The event-based binary feature convolution and classifier system was implemented in FPGA hardware on a Cyclone IV E platform using Quartus Prime v16.0.0 Lite Edition. The input to the system is defined as the four-polarity event streams (here referred to as the 4 RFs for Receptive Fields). This system was implemented in hardware by Langdon Davis based on the design developed by this student.

The top-level subsystem called NEURO_NET, incorporates a 5×5 ROI patch sizes with 16 feature neurons. The AER_FIFO component shown in Figure 5.22, contains the encoded AER events received via the event-based SPAD sensor for processing. The FIFO holds 29 words (512) of 32-bits in length. The FIFO is accessed via an FSM (Finite State Machine) which sequences the reading and processing of events and is used to control access to the DDR2 controller and synchronization logic between the 133 MHz clock domain of the DDR2 Controller and the clock domain in which the access originates. The NEURO_REGS component contains the local registers that hold the binary features $\ddot{\boldsymbol{w}}^1$ used by the NEURO_RF_CONV components. The registers are programmed via software during initialization. The NEURO_RF_CONV components also contain the binary convolution logic of the feature set and the ROI patch loaded from the recent time surface $\boldsymbol{S}^0$. Four NEURO_RF_CONV components are instantiated, one for each event polarity of the event stream. The output of this component is the convolved feature map. The ADD_4x8CITS_1CLK components contain the logic for the addition of the convolved features maps of each of the four polarities. The NEURO_CLASSIFICATION component contains the local registers that hold the classifier weights W (pre-calculated offline), the

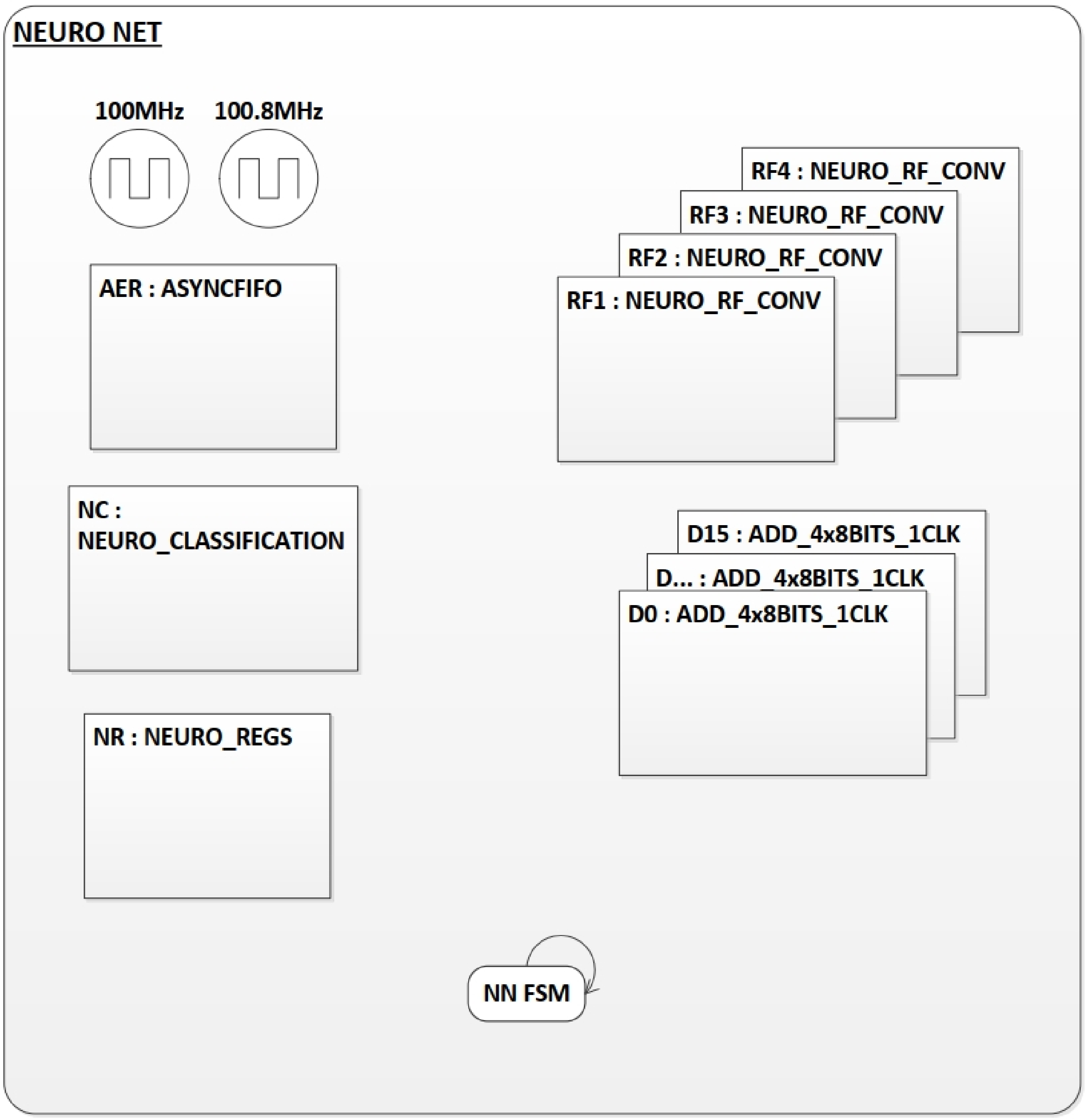


FIGURE 5.22: **Neuro Net Entity Clock Domains and Instantiated Components.**

logic to determine the winning neuron from the sum of the convolved feature maps and the logic to perform the classification.

Figure 5.23 displays the FSM for the NEURO_NET component. In state 'IDLE' the AER FIFO is interrogated and if an AER event exists the event is read and state transition to 'UPDATE_TIME_MATRICES' occurs. In state 'UPDATE_TIME_MATRICES' the AER event is decoded to the row/column addresses along with the neuron triggering the event and the event time. The corresponding time address in the DDR SDRAM is updated with the event time and state transition to 'UPDATE_PATCH_MATRICES' occurs. In state 'UPDATE_PATCH_MATRICES' the ROI patch is read from the DDR2 SDRAM starting with the first polarity and one pixel at a time. With an ROI patch size of 5×5 and 4 polarities, a total of 100 DDR2 SDRAM addresses are read in series. As each pixel is read, the corresponding NEURO_RF_CONV component is signaled and the read pixel patch time convolution occurs. Once all patches have been read and processed, the state transition to 'SUM_RF_D' occurs. In state 'SUM_RF_D' the outputs of the 4 NEURO_RF_CONV components is summed and state transition to 'FIND_WINNER' occurs. In state 'FIND_WINNER' the NEURO_CLASSIFICATION component iterates through the summed convolution components to determine the max count and determine the winner neuron(s). State transition to 'UPDATE_HIST' then occurs and in this state the feature output or histogram is updated from the winning neuron(s). The state then transitions to 'IDLE' through state 'DONE'. In state 'IDLE' if the classification time has expired and a prescribed number of events have been processed, state transition to 'CLASSIFY' occurs. In state 'CLASSIFY' the NEURO_RF_CONV component performs the vector dot product operations on the feature maps. The result of this state is the classification and state transition to 'LOG'. The state 'LOG' samples the current feature maps for each neuron along with the resultant dot product output values which are used for classification. The log is added to a FIFO and can be retrieved by the software for post-processing. A state transition to 'IDLE' then occurs through state 'DONE'. This process repeats itself so long as AER events are available.

5.2.9 Real-time Frame-based Feature Extraction Network Implemented on GPU

To provide a comparison for the performance of the event-based feature extraction networks, equivalent frame-based systems with identical architectures and training methodologies were developed and tested. Here the event-based feature extractors are replaced with convolution and max pooling operations with the same feature sizes. In this way the frame-based networks precisely replicate the event-based operations with the only difference between the two

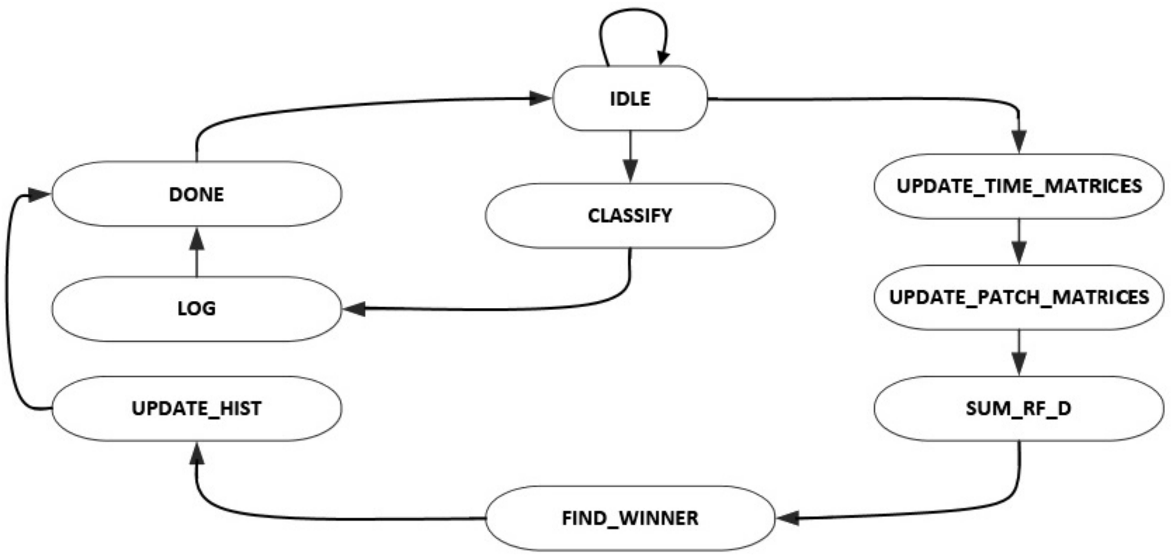


FIGURE 5.23: **NEURO_NET Finite State Machine (FSM).**

methods being the extra, arguably unnecessary, convolution operations performed in the frame-based system on the parts of the image exhibiting no significant change i.e. those with no events. Following the convolutional layer, the same pooling methods and linear classification operations were performed for all tests providing an unbiased comparison between the frame-based and event-based systems.

We implemented a subset of these frame-based feature extraction networks on an embedded NVIDIA Jetson TX2 board [132]. This hardware implementation aimed to demonstrate the feasibility of realizing a high-speed classifier for noisy low-resolution SPAD imagers. The real-time performance of the implemented frame-based system was demonstrated on a four airplane subset of the SPAD dataset presented in Section 5.2.1.

A range of feature extraction network sizes were examined consisting of 4, 8, 16, 32 and 64 random feature extractors. In addition, the 1D pooling method described in 5.2.7 was used to simplify implementation. In the next section, we compare the classification performance results of the frame-based systems to a range of equivalent event-based systems.

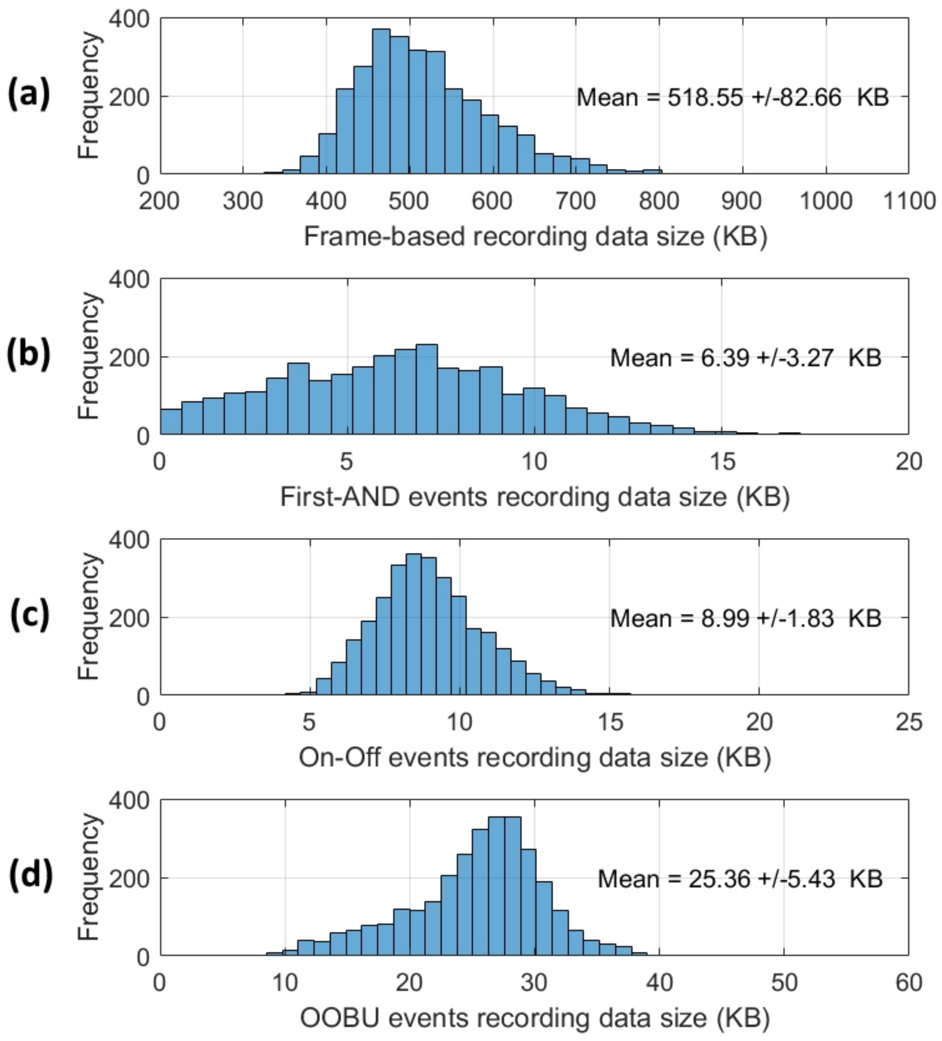


FIGURE 5.24: **Reduction in data size by event-based conversion.** (a) shows the distribution of size of the frame-based SPAD recordings. Panels (b), (c) and (d) show the size distributions of the First-AND, On-Off and OOBU event streams respectively. As detailed in Section 5.2.6, the threshold values $\theta = +/-2$ were used for the On-Off events shown in (c). The threshold values $\phi_1 = 2$ and $\phi_2 = 1$ were used to generate OOBU events in (d).

5.3 Results

5.3.1 Data Generation Rates

The recorded frame-based SPAD imaging dataset was converted to a First-AND event-based data stream via simulation of the implemented First-AND circuit described in Section 5.2.4. In addition, the dataset was processed using of Algorithms 1 and 2 to generate the On-Off and

OOBU event-based data streams. As shown in Figure 5.24, the conversion of frame-based SPAD data to an event stream significantly reduces the recording size and thus the data-rate of the processor with associated savings in processing power and improved response time. The First-AND conversion method results in an 81 fold reduction in data-rate whereas the On-Off and OOBU methods result in 57 and 25 fold reductions respectively. Having examined the data-rates generated from the different methods, we now compare the classification performance of the First-AND, On-Off and OOBU event streams to the original frame-based SPAD imaging dataset.

5.3.2 Classification

The tests presented in this section cover the raw frame-based dataset as well as the event-based methods described. We further test the effect of different pooling methods as well as processing by random and trained feature extraction networks. The tests also examine in detail the effect of pooling window sizes and network sizes on performance. All test were performed using a linear classifier mapping frames-based images and event-based samples of time surfaces to the output classes in a repeatable manner. Fixed-point precision using 8-bits was used for simulation parameters mirroring the fixed-point FPGA implementation described in Section 5.2.8 and the ASIC implementation in Section 5.2.4.

Figure 5.25 shows the classification results when using frame-based processing on the SPAD dataset. The results are organized in per frame and per recording accuracy results. For the per recording accuracy measure, the class with the highest number of winning frames is selected as the correct class. Unsurprisingly, since this process effectively performs a pooling and max operation over the information in all the frames of a recording, the per recording classification accuracy is consistently above that of the per-frame accuracy measure.

It is clear from the accuracy results in Figure 5.25, that the 2D pooling almost always outperforms 1D pooling. However, the 1D pooling method is significantly simpler to implement in hardware and faster to compute in software motivating its investigation and comparison to the

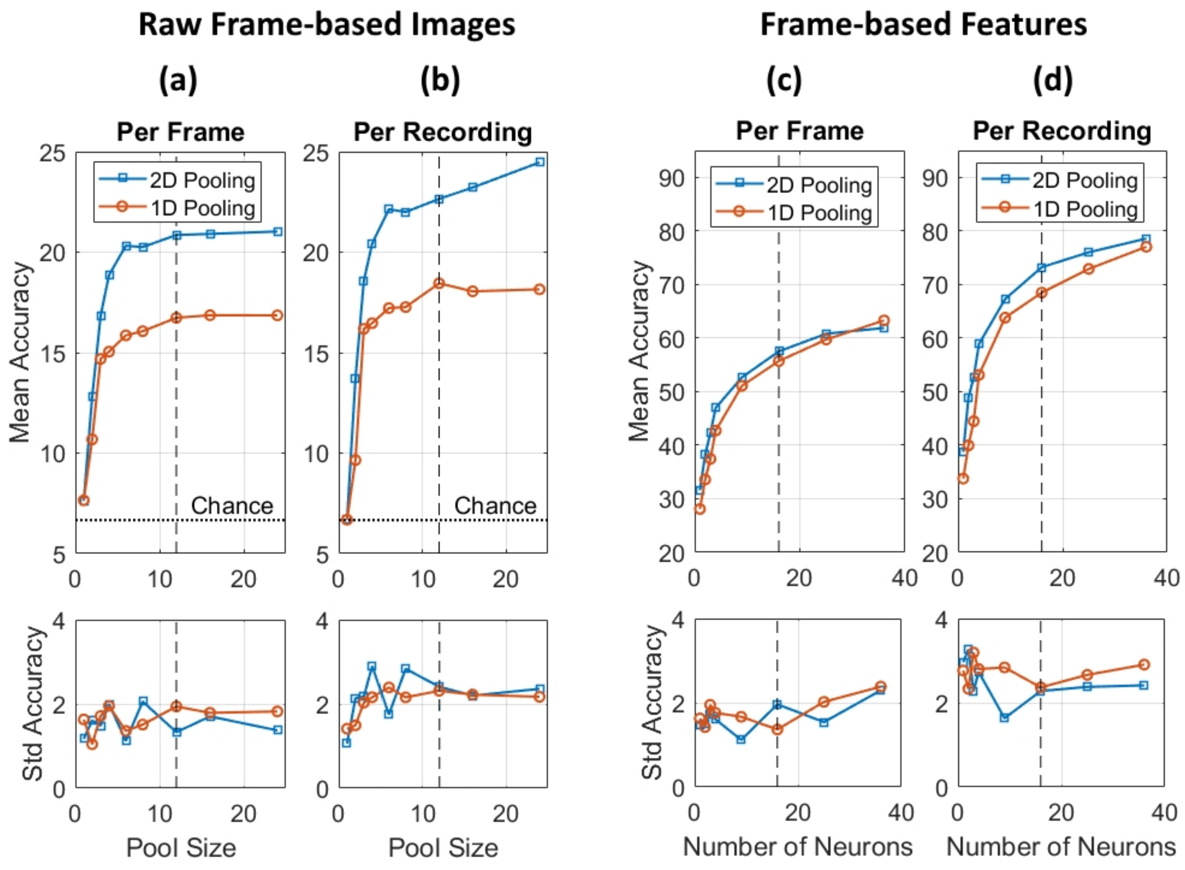


FIGURE 5.25: **Classification accuracy on the frame-based SPAD airplane dataset across a range of parameters.** Panels (a) and (b) show per frame and per recording accuracy respectively where classification is performed directly on the raw SPAD images via a linear classifier. The top and bottom two panels show the mean and standard deviation of classification accuracy respectively. Results for both two-dimensional pooling and one-dimensional pooling are plotted as a function of pool size and compared to chance 1/15 = 6.7%. The vertical dashed line at $L = 12$ indicates the pooling window size chosen in the subsequent tests. Panels (c) and (d) show per frame and per recording accuracy respectively after feature extraction as a function of the number of feature extracting neurons. The dashed vertical line at $N = 16$ marks the network size chosen for FPGA implementation. All results presented are over n = 20 independent trials.

2D pooling method. Given hardware constraints, these comparisons provide valuable information on the resource versus performance trade-offs which are critical during the hardware design stage.

The first point in Figure 5.25(a) at $L = 1$, collapses all information in each frame to a single number. As expected, this global pooling of the entire raw image produces an identical accuracy for both the 1D and 2D pooling methods that is close to chance. This result effectively demonstrates that, as expected, the mean value across the pixels of the image provides approximately zero information about the target class. As the size of the pooling window L increases, the classification accuracy rises sharply before stabilizing above $L = 12$ pixels. Since little additional information can be generated by increasing the pooling window resolution to or above the original $A_x \times A_y$. Thus the best results achievable using the raw frame-based SPAD data, a pooling layer and a linear classifier is accuracy that is below 25%. The per recording accuracy measure shown in 5.25(b) is similarly poor providing only slightly higher accuracy at the larger pool sizes.

Figure 5.25(c) and (d) show the classification accuracy of trained frame-based feature extraction networks with identical architecture and training parameters as the event-based networks discussed in Section 5.2.5. The first point in Figure 5.25(c), at $N = 1$, $L = 12$ and accuracy of approximately 30%, represents a trivial convolution of the SPAD frames by the single commonest feature in the dataset. Capturing slightly more spatial information than the raw image, this trivial solution performs only slightly better than the raw frame results of (a) with pool size $L = 12$. Here, the additional information derived from the incorporation of local spatial information in the convolution operation provides approximately 10% improvement in accuracy. As the number of feature extractors is increased, the classification accuracy increases to slightly above 60% and below 80% for the per frame and per recording measures respectively at $N = 36$ neurons. While every increase in network size improves system accuracy, there are diminishing returns with each layer size increase, a pattern that we see consistently in all tests.

As a point of comparison, the classification accuracy results achieved in the real-time embedded GPU-based platform on the reduced four airplane dataset are detailed in Table 5.1. These results show the same pattern of increased accuracy with network size albeit from a higher accuracy floor due to the greater simplicity of the reduced dataset. In these results, we see mean accuracies of between 86.98% and 98.7%. Note that the comparison of the 1D pooling and 2D pooling methods shown in Figure 5.25(c) and (d) demonstrate only a slight

TABLE 5.1: **Accuracy and execution time as a function of feature layer size for the real-time frame-based feature extractor networks implemented on the TX2 GPU platform.** Modified from [132].

No. Features	**4**	**8**	**16**	**32**	**64**
% Accuracy	86.98	92.19	95.31	96.88	98.70
Exec Time (ms)	30.18	31.40	38.61	47.81	65.07

advantage in favor of the 2D pooling method thus validating the hardware design choice of implementing the simpler 1D pooling method in [132].

Figure 5.26 shows the classification performance of the proposed event generation methods. The classification results show a large increase in accuracy for all event-based methods relative to the original frame-based SPAD dataset shown in Figure 5.25(a). The event stream with the lowest accuracy is that of the On-Off events shown in Figure 5.26(a). The first point plotted is at pool size $L = 1$ which shows per accuracy slightly above 10%. This is equivalent to only using the event polarity count for classification which unsurprisingly results in the lowest accuracy. As the pool size is increased above $L > 8$ the per frame accuracy rises reaching approximately 55% and 65% for the 1D and 2D pooling methods respectively. The results in (b) follow a similar pattern with per recording accuracies reaching 62% and 76% respectively for the 1D and 2D pooling methods. The First-AND event results are shown in panels (c) and (d). Here the 2D pooling accuracies are slightly above those of On-Off events. The relative improvement accuracy is even greater for the 1D pooling method again motivating its use in hardware. The OOBU event accuracy results shown in (e) and (f) are consistently higher than those of both the On-Off and First-AND events making OOBU events the clear winner of the three approaches.

In the simple three-class example shown in Figure 5.20, the accuracy achieved using only event counts jumped from 44% for the On-Off event to 92% for the OOBU events. Here, for the full 15 class problem, the improvement from the first points plotted in Figure 5.26(a) to (e) is more modest. This shows when only using the event count for classification we can achieve an improvement of slightly below 20% for the OOBU from slightly above 10% for the On-Off events. While this more modest improvement is unsurprising given the much

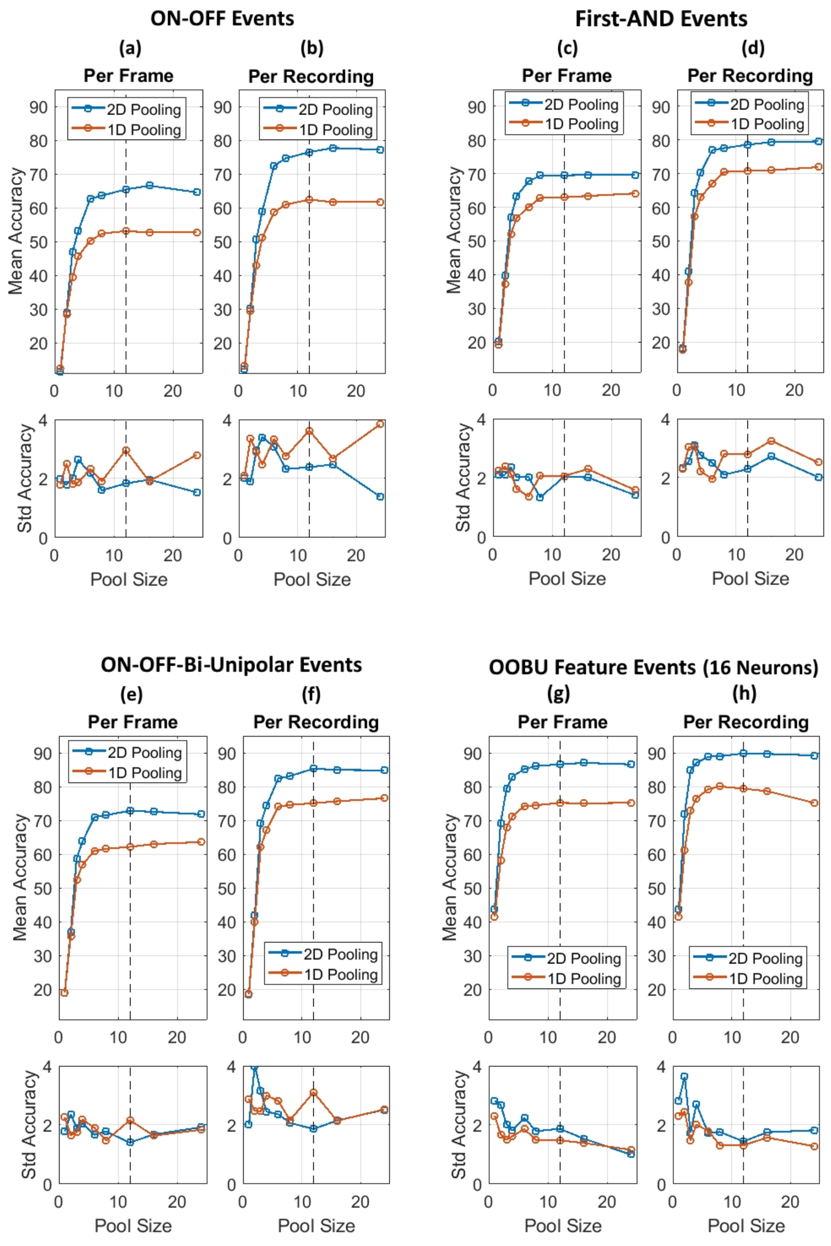


FIGURE 5.26: **Classification accuracy on event-based data streams generated from the SPAD the dataset.** Panels (a) and (b) show the per frame and per recording accuracy results of the On-Off event streams. Panel (c) and (d) show the same for First-And events, (e) and (f) for OOBU events and (g) and (h) show the same for feature events generated from OOBU events using 16 FEAST neurons. All results are shown as a function of the pool window size L. The vertical dashed line indicates the chosen window size $L = 12$ used in subsequent tests.

more challenging 15 class problem, the results from this and the larger pool sizes still support a consistent improvement in accuracy from On-Off events to OOBU events.

Finally panels (g) and (h) of Figure 5.26 show the accuracy achieved via the addition of an event-based feature extraction layer. This configuration is used in the hardware-implemented system described in 5.2.8 with OOBU events serving as inputs to 16 trained features. The performance of the full system is examined as a function of different pooling methods and pool sizes. The per frame and per recording accuracies of the feature extraction layer are the best of all the event streams tested, starting from slightly above 40% accuracy and reaching 75% and 87% per frame accuracy for the 1D and 2D pooling methods respectively. The per recording accuracies are even higher at 79% and 90% for the 1D and 2D pooling methods respectively. Here the $L = 1$ result at above 40% accuracy and the $L = 12$ result at 90% accuracy are both remarkable given the complexity of the 15 class view-invariant airplane classification dataset, the simplicity of the applied methods and the significantly lower performance on the frame-based dataset using identically structured and trained classification architectures. These accuracy results together with the clear data-rate advantages detailed in Section 5.3.1, highlight the suitability of the use of event-based sensing approach to the high noise SPAD time of flight imaging data.

Having investigated the effects of different pooling methods on the various event-based data streams, we now investigate the effect of the feature extractor network size on classification accuracy. Since the size of the feature extractor network affects hardware resource consumption and/or processing speed, we seek to determine the smallest network which provides an acceptable level of performance given the resource constraints and speed requirements. Figure 5.27 shows accuracy results for the First-AND and OOBU event streams which provided the highest performance in the pooling test experiments. Figure 5.27(a) and (b) shows the per frame and per recording accuracies of the First-AND event stream processed by random binary-valued feature extraction networks. The first points on panels (a) and (b) represent trivial single neuron feature extractors. As the layer size increases the mean classification accuracy increases rapidly reaching slightly above 70% at the highest network size tested which is $N = 36$. These random feature results provide a baseline for evaluating the trained binary

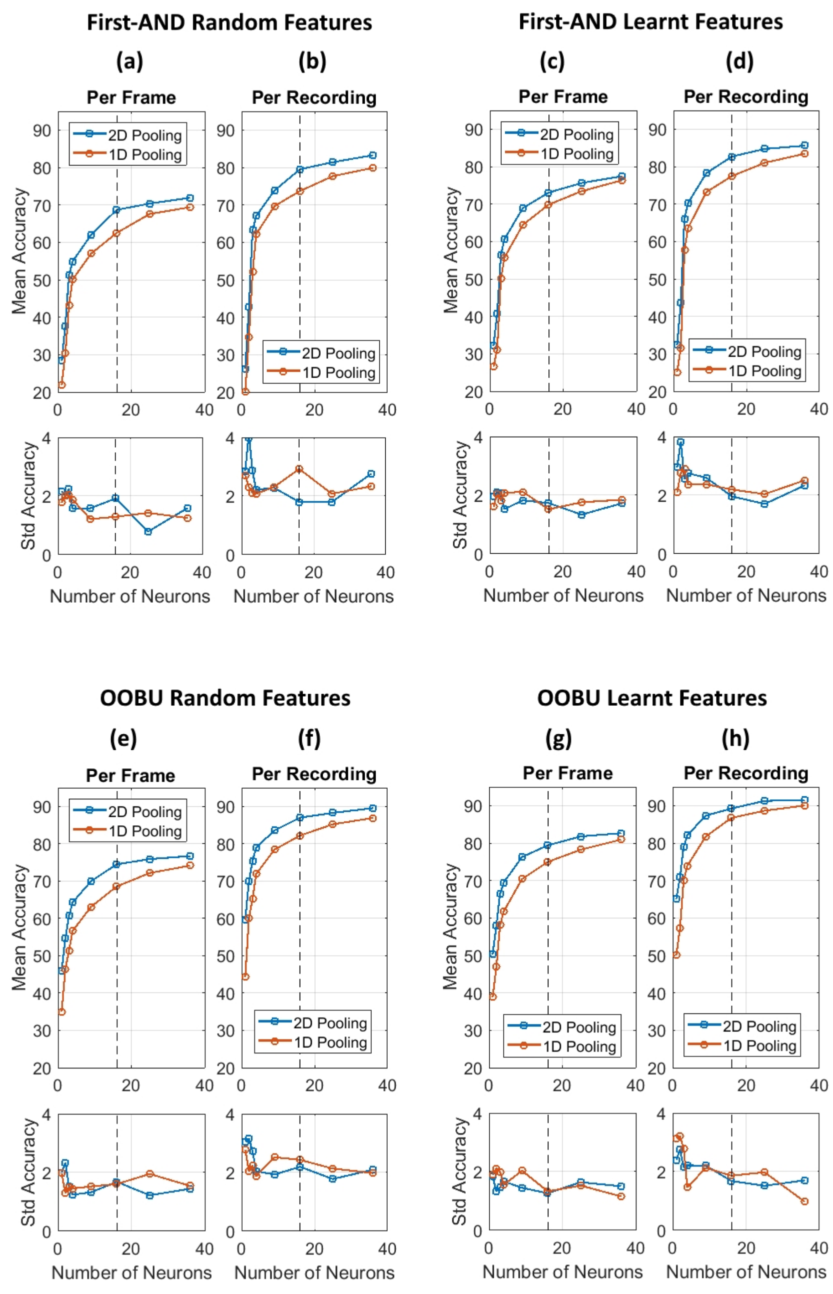


FIGURE 5.27: **Classification accuracy using feature event streams generated from random binary networks and trained binary networks operating on the proposed First-AND and OOBU event streams** (a) and (b) show per frame and per recording classification accuracy using First-AND events processed through a random binary feature extraction layer of varying size and (c) and (d) show the same with the random binary features replaced by trained binary features. (e) and (f) show per frame and per recording accuracy results of random binary features operating on OOBU events and (g) and (h) show the same for trained binary features.

networks whose results are shown in panels (c) and (d). The identical network architectures and processing methods used, isolate the improvements gained via feature training alone. The results show that trained features consistently outperform random features. Here as in the raw event stream tests, 2D pooling still consistently outperforms 1D pooling, here however, the margin is smaller. This result is expected since the feature extraction operation projects the raw event stream onto a large number of sparsely populated feature surfaces such that when the surfaces are pooled via the 1D method, less information is lost in comparison to the 2D pooling method. In other words, as the size of the feature extraction layer expands, the effect of information loss due to pooling becomes less significant.

In panels (e) to (h) the same comparison between random features and trained features is performed, this time on the OOBU events. We again see that as with the First-AND case, trained OOBU features outperform random ones. And again we see that OOBU events consistently outperform First-AND events this time when processed through an event-based feature extraction network. The results in panel (g) and (h) represent the highest accuracy achieved on the dataset where at a layer size of $N = 36$, a per-frame accuracy of 82.64% and a per recording accuracy 91.5% is achieved. The vertical dashed line in Figure 5.27 represents the network size $N = 16$ chosen for the implementation of the event-based FPGA processor described in Section 5.2.8. This network layer size was chosen to provide a reasonable trade-off between accuracy and hardware resource requirements.

5.3.3 FPGA Implementation Results

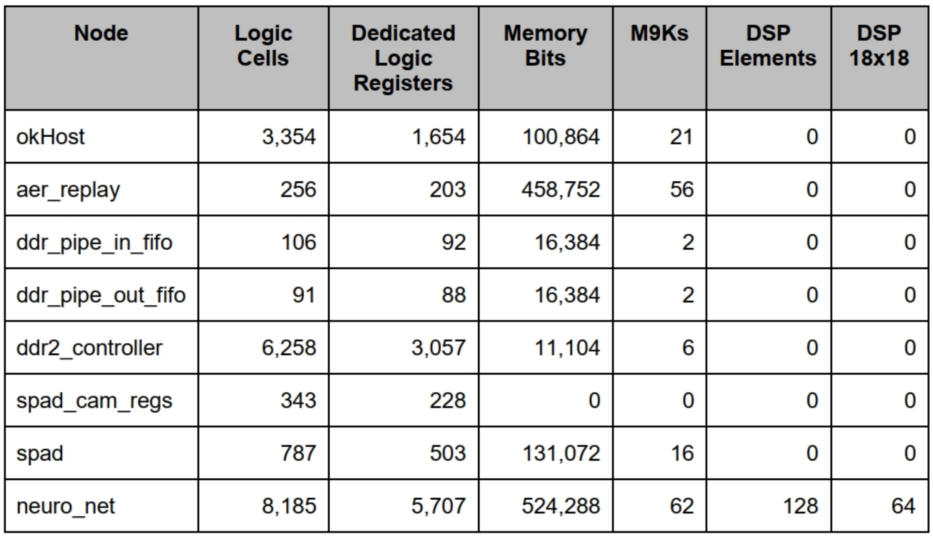
With the accuracy results provided in the preceding section, we now look at the FPGA implementation results for an instance of the event-based processor whose classification accuracy results were detailed in Figure 5.27(c) and (d) for the First-AND event streams and (g) and (h) for the OOBU event streams. Since identical fixed-point implementations were used in both the software and FPGA implementations, the classification accuracy results from the preceding sections apply directly to the end-to-end FPGA implemented system which here is referred to as the NEURO_NET.

As can be observed from Table 5.2, a substantial amount of the AER_REPLAY and NEURO_NET nodes each consume a significant amount of memory. Ultimately the AER_REPLAY node would be removed when the SPAD imager is interfacing the event-based processor directly. The memory consumed by the NEURO_NET node is primarily due to the FIFO that is used for logging purposes. The logging node provides a means of both testing/debugging the neural network design but more importantly allows retrieval of the classification data from the FPGA and into the software.

Besides the memory resources consumed, the NEURO_NET also consumes 128 DSP elements. These DSP elements exist in the NEURO_CLASSIFICATION node where the vector dot product operation on the output feature map or histogram and the classifier weights occur. Currently, the classifier weights are restricted to a 12-bit signed notation, hence each neuron consumes two embedded 9-bit multipliers (multiplication is pipelined over two clock cycles), for a total of 16 neurons 32 multiplications and with four event polarities the results in a total of 128 multipliers.

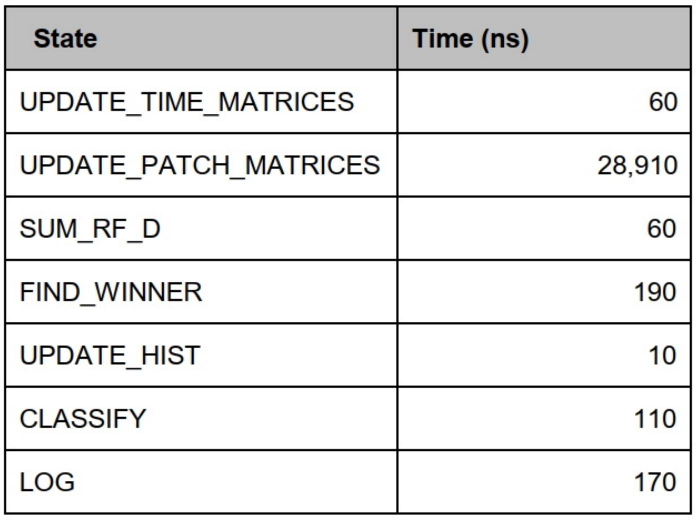
Table 5.3 lists the execution times for the Neuro Net states, and the timing information has been generated from various Signal tap captures. The total time to process a single AER event and update the feature map or histogram is 29.24μs. The vast majority of this time is consumed in the 'UPDATE_PATCH_MATRICES' state where the ROI from the time surface (or patch data) is read from the DDR2 SDRAM. In this state, a total of 100 memory addresses are read, with a patch size of 5, and with 4 RF, $5\times5\times4=100$. On average to synchronize and read a single address of the DDR2 SDRAM from the 100 MHz clock domain takes approximately 290 ns, 29 clock cycles. Any future implementations of this event-based processor must take into account this bottleneck. Solutions to this bottleneck include refinement of the logic, faster RAM and finally the development of a cache system outlined in the discussion section.

TABLE 5.2: **NEURO_NET hardware resource costs.** Here, Logic Cells refer to the fundamental building blocks of an FPGA system. Dedicate Logic Registers are used to store variables locally on the FPGA. M9K memories are Altera's embedded high-density memory arrays and DSP elements refer to Digital Signal Processing elements which are specialized circuits that perform predefined mathematical operations in a highly optimized manner. Courtesy of BAE Systems. FPGA hardware implemented by Langdon Davis based on designs developed by this student.



Node	Logic Cells	Dedicated Logic Registers	Memory Bits	M9Ks	DSP Elements	DSP 18x18
okHost	3,354	1,654	100,864	21	0	0
aer_replay	256	203	458,752	56	0	0
ddr_pipe_in_fifo	106	92	16,384	2	0	0
ddr_pipe_out_fifo	91	88	16,384	2	0	0
ddr2_controller	6,258	3,057	11,104	6	0	0
spad_cam_regs	343	228	0	0	0	0
spad	787	503	131,072	16	0	0
neuro_net	8,185	5,707	524,288	62	128	64

TABLE 5.3: **NEURO_NET States Execution Time.** Courtesy of BAE Systems. FPGA hardware implemented by Langdon Davis based on designs developed by this student.



State	Time (ns)
UPDATE_TIME_MATRICES	60
UPDATE_PATCH_MATRICES	28,910
SUM_RF_D	60
FIND_WINNER	190
UPDATE_HIST	10
CLASSIFY	110
LOG	170

5.4 Discussion and Future Work

A potential weakness in the First-AND event generation method is the all or none behavior of the AND gates whereby even a single faulty, inactive pixel can disable the entire receptive field. While such 'dead pixels' were not observed in the recorded dataset, their potential presence in imagers with higher numbers of pixels is more likely. To protect against the effect of such non-idealities, the replacement of the AND gates with thresholded current summers will be investigated in future designs of the First-AND system.

In order to reduce the potential effect of inherent internal delays in the SPAD array, the global clock may be slowed such that the SPAD pixels are effectively sampled at a lower temporal resolution. In this way, small errors in timing measurement (but also timing measurements of nearby objects) are effectively quantized at the same low-resolution clock cycle. The utility or otherwise of this approach will be investigated in future work.

Given the simple method of their generation and orientation and shape invariant informational properties, OOBU events provide a promising approach to increasing event information in a wide range of event-based sensor systems. The feasibility of implementing these higher-level features in local sensor networks will be investigated in future work.

As detailed in 5.3.3 a major bottleneck operation of the event-based FPGA processor is the loading of a local ROI from the time surface surrounding a current event. Furthermore, this bottleneck becomes more significant as event polarities and ROI sizes increase. More event polarities are needed when implementing deep event-based convolutional networks and the use of larger ROI sizes can often be beneficial in applications where the underlying signal SNR is low as in event-based space imaging as discussed in Chapter 2.

While the direct approach to speeding up the ROI read operation is to use faster RAM, other cache-like architectures may provide a solution to the memory loading bottleneck by taking advantage of the likely proximity in space of new events relative to previous ones. In this approach, a slightly larger local region than the ROI may be loaded from RAM to local registers and with each new event, this locally stored address space is first interrogated and if the newest event is close to a previous one, its ROI will have been stored locally and can be fetched at speed. If we assume spatial proximity between temporally proximal events, such an approach is likely to significantly reduce the memory bottleneck associated with the ROI retrieval. In future work, we will investigate potential design solutions to this problem with the aim of providing better timing performance for larger, deeper event-based networks in hardware.

5.5 Conclusion

In this work, a challenging SPAD imaging recognition dataset was presented. Three event-based processing approaches were proposed. First-AND, On-Off and On-Off-Bi-polar-Uni-Polar (OOBU) events. The classification accuracy of these event-based methods was investigated and compared to the original frame-based dataset using either linear classifiers or feature extractor networks followed by a linear classifier as a processor. Across all tests, the event-based methods outperform their frame-based counterparts in terms of accuracy and reduced data-rate. This is because the event generation methods involve the pooling of raw sensor data over either time or space or both, significantly increasing the information content of each event in comparison to the raw pixel range values in the frame-based data. Within the event-based approaches, the OOBU events resulted in the highest recognition accuracy followed by First-And and On-Off events. In terms of data-rates, the First-AND events result in the lowest data-rate followed by the On-Off and OOBU events resulting in 81, 57 and 25 fold reduction in data-rate respectively. In addition to the event-based generation methods, a range of different network parameters were investigated with larger networks shown to outperform smaller ones, trained networks outperforming random networks and two-dimensional spatial pooling outperforming one-dimensional pooling.

The systems investigated in this work provide a range of well-performing points in the event-based design space which can be integrated with Direct Time of Flight SPAD sensor hardware to provide event-based processing that not only drastically reduces data-rate coming off the sensor, but also the quality of the output data as it relates to challenging tasks such as a view-invariant classification of large complex datasets. By using the same learning methodology and the same single-layer network structure and by testing across multiple design dimensions such as pooling and network size, we demonstrate exhaustively that the event-based methods outperform the frame-based system across all parameters while serving as a guide for the design of such networks in hardware. The FPGA implementation of the event-based processor utilized the tests demonstrates the hardware efficiency and processing speed of the design for real-time applications of SPAD sensor technology.

CHAPTER 6

Conclusion

In this work a range of event-based detection, tracking and classification algorithms were investigated on several novel high-noise high-speed datasets. The focus on speed and noise both in the design of the datasets and the approach to processing, resulted in unique challenges which are absent in controlled data collection environments. These challenges in turn motivated the design of distinct solutions each with their accompanying insights. As we see in this thesis, working with imperfect high-noise real-world data, attempting to process information in high-speed environments and a fixation on algorithm simplicity for ease of hardware implementation even at the expense of performance, often results in the development of distinct approaches and methods which are quite different to the conventional machine vision and machine learning approaches where optimal performance on relatively controlled datasets is most valued.

Examples of such distinct approaches developed in this work include the following:

When investigating event-based memory surfaces of differing complexity (for future use in FPGA hardware) we found that index surfaces provided great robustness to variance in target velocity on our uncontrolled airplane dropping dataset. Again the motivation for examining index surfaces in the first place was their potential for simplifying implementation in hardware relative to time surfaces.

When investigating the FEAST algorithm for use on noisy datasets we found that learnt noise features could be used to provide automatic noise filtering providing the rest of the network with a noise free event stream. We also found another use for the noise feature in selection of the network layer size. We found that by forgoing some information during feature learning (through the use of adaptive selection thresholds) we were rewarded with the missed-event-rate and imbalance-in-neuron-activation signals. These two readily available signals were found to be highly useful for estimating network convergence during learning. And the later signal was found to also be useful during inference by predicting classification performance through the Gini coefficient. All these novel ideas and mechanisms were found as the result of trying to minimize hardware complexity through the use of adaptive selection thresholds to balance network activation and use of this solution on a noisy dataset.

When developing detection and tracking solutions for real-world space imaging data, the default focus on event-based noise filtering resulted in a cascaded filter design where noise suppression, feature detection and tracking all operated as an ever more selective noise filter. The developed system used large ROIs for maximizing information capture. By using an event-based template matching method and determining the unimodality of each ROI's angular activation, an orientation, size and velocity invariant feature detector was developed. This feature was found to outperform other high-speed algorithms in terms of accuracy and speed. The focus on maximizing processing speed motivated a cascaded filter design where at each layer the increased complexity and response time of each filter was counteracted by the reduction in the number of events processed such that the over-all system's processing time was only slightly greater than the first, fastest layer even though the last layer was slower by orders of magnitude.

When attempting to minimize the output data-rate and processing requirements of largely redundant, noisy SPAD DTOF data, we found that a simple local AND-gate based spatio-temporal on-chip detector was able to reduce sensor data by orders of magnitude while still providing an increase in classification performance. By completely discarding the conventional measurement aspect of the sensor we were able to convert a high-speed high-noise frame-based dataset to a stream of highly informative feature events. After the implementation of our binary event-based FPGA back-end processor (which was designed for a four polarity First-AND input event stream) the drive to make full use of all four polarities led to the development of On-Off-Bipolar and Unipolar (OOBU) events (in addition to standard On-Off events). Investigation of Uni-polar and Bi-polar events was found to significantly enhance classification performance by providing highly useful local activation information that was quite distinct from what a conventional convolutional network can extract.

An important question arising in this thesis that requires further detailed investigation is the scalability of event-based processing systems. As discussed in Section 5.4, a critical issue in event-based processing is the memory access requirements which are substantial and likely to limit scalability. Event-based processing essentially swaps computation with memory access which given current computing technologies is not a good trade. Solutions to the memory access issue in event-based processing could involve new technologies that allow high-speed local memory access as well as alternative architectures. Established memory management techniques such as cache hierarchies go against the event-based processing creed which holds that event-based data invariably generates un-ordered highly informative sparse streams of events from much more ordered less informative data sources making caches unsuitable for event-based data. While the assumption that event-based data is almost always sparser and more informative than frame-based data is certainly true, the amount of residual order remaining in an event-based data stream is likely to be application-dependent and cannot be discounted without investigation. Thus, the utility of event-based caches for overcoming the memory access issues discussed in this work remains an open area for investigation.

6.1 Future Work

In future work we will investigate the many open lines of inquiry indicated in each chapter. In addition to these, we aim to test the methods developed in this thesis in neuromorphic hardware across a range of real-world environments and in applications where high-speed processing of noisy event-based data is appropriate. Finally we aim to combine the developed event-based feature detection, target tracking and object recognition methods in dedicated system-on-chip hardware to provide noise-robust real-time event-based processing for autonomous platforms.

Bibliography

1. Monk, T. & Paulin, M. G. Predation and the origin of neurones. *Brain, behavior and evolution* **84,** 246–261 (2014).
2. Monk, T. *The evolutionary origin of nervous systems and implications for neural computation* PhD thesis (University of Otago, 2014).
3. Mead, C. A. & Mahowald, M. A. A silicon model of early visual processing. *Neural networks* **1,** 91–97 (1988).
4. Le, Q. V. *et al.* Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv:1112.6209* (2011).
5. Orban, G. A., Van Essen, D. & Vanduffel, W. Comparative mapping of higher visual areas in monkeys and humans. *Trends in cognitive sciences* **8,** 315–324 (2004).
6. Nauhaus, I., Nielsen, K. J., Disney, A. A. & Callaway, E. M. Orthogonal micro-organization of orientation and spatial frequency in primate primary visual cortex. *Nature neuroscience* **15,** 1683 (2012).
7. Rullen, R. V. & Thorpe, S. J. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural computation* **13,** 1255–1283 (2001).
8. Levy, W. B. & Baxter, R. A. Energy efficient neural codes. *Neural computation* **8,** 531–543 (1996).
9. Patterson, D. A. & Hennessy, J. L. *Computer organization and design MIPS edition: the hardware/software interface* (Newnes, 2013).
10. Hamilton, T. J., Afshar, S., van Schaik, A. & Tapson, J. Stochastic electronics: A neuro-inspired design paradigm for integrated circuits. *Proceedings of the IEEE* **102,** 843 (2014).
11. Rasche, C. *The making of a neuromorphic visual system* (Springer Science & Business Media, 2005).

12. Afshar, S. *et al. Emergence of competitive control in a memristor-based neuromorphic circuit* in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (2012), 1–8.
13. Tapson, J. C. *et al.* Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Frontiers in neuroscience* **7,** 153 (2013).
14. Hussain, S., Basu, A., Wang, R. M. & Hamilton, T. J. Delay learning architectures for memory and classification. *Neurocomputing* **138,** 14–26 (2014).
15. Gütig, R. & Sompolinsky, H. The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience* **9,** 420 (2006).
16. Afshar, S., George, L., Tapson, J., van Schaik, A. & Hamilton, T. J. Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels. *Frontiers in neuroscience* **8,** 377 (2014).
17. Boahen, K. A. in *Neuromorphic systems engineering* 229–259 (Springer, 1998).
18. Culurciello, E., Etienne-Cummings, R. & Boahen, K. *High dynamic range, arbitrated address event representation digital imager* in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No. 01CH37196)* **3** (2001), 505–508.
19. Furber, S. B., Galluppi, F., Temple, S. & Plana, L. A. The spinnaker project. *Proceedings of the IEEE* **102,** 652–665 (2014).
20. Merolla, P. A. *et al.* A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345,** 668–673 (2014).
21. Chan, V., Liu, S.-C. & van Schaik, A. AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems I: Regular Papers* **54,** 48–59 (2007).
22. Serrano-Gotarredona, T., Andreou, A. G. & Linares-Barranco, B. AER image filtering architecture for vision-processing systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* **46,** 1064–1071 (1999).

23. Posch, C., Matolin, D. & Wohlgenannt, R. A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS. *IEEE Journal of Solid-State Circuits* **46,** 259. ISSN: 0018-9200 (Jan. 2011).

24. Zaghloul, K. A. & Boahen, K. Optic nerve signals in a neuromorphic chip I: Outer and inner retina models. *IEEE Transactions on Biomedical Engineering* **51,** 657–666 (2004).

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

26. Ruedi, P.-F. *et al.* A 128/spl times/128 pixel 120-db dynamic-range vision-sensor chip for image contrast and orientation extraction. *IEEE Journal of Solid-State Circuits* **38,** 2325–2333 (2003).

27. Schanz, M., Nitta, C., Bußmann, A., Hosticka, B. J. & Wertheimer, R. K. A high-dynamic-range CMOS image sensor for automotive applications. *IEEE Journal of Solid-State Circuits* **35,** 932–938 (2000).

28. Mallik, U., Clapp, M., Choi, E., Cauwenberghs, G. & Etienne-Cummings, R. *Temporal change threshold detection imager* in *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.* (2005), 362–603.

29. Kramer, J. *An on/off transient imager with event-driven, asynchronous read-out* in *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353)* **2** (2002), II–II.

30. Lichtsteiner, P., Delbruck, T. & Kramer, J. *Improved ON/OFF temporally differentiating address-event imager* in *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004.* (2004), 211–214.

31. Lichtsteiner, P., Posch, C. & Delbruck, T. *A 128 X 128 120db 30mw asynchronous vision sensor that responds to relative intensity change* in *2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers* (2006), 2060–2069.

32. Lichtsteiner, P., Posch, C. & Delbruck, T. A 128x128 120 dB 15mus Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE journal of solid-state circuits* **43,** 566–576 (2008).

33. Culurciello, E. & Andreou, A. G. CMOS image sensors for sensor networks. *Analog Integrated Circuits and Signal Processing* **49,** 39–51 (2006).

34. Guo, X., Qi, X. & Harris, J. G. A time-to-first-spike CMOS image sensor. *IEEE Sensors Journal* **7,** 1165–1175 (2007).

35. Shoushun, C. & Bermak, A. *A low power CMOS imager based on time-to-first-spike encoding and fair AER* in *2005 IEEE International Symposium on Circuits and Systems* (2005), 5306–5309.

36. Luo, Q. & Harris, J. G. *A time-based CMOS image sensor* in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512)* **4** (2004), IV–840.

37. Chen, S., Bermak, A. & Boussaid, F. A compact reconfigurable counter memory for spiking pixels. *IEEE Electron Device Letters* **27,** 255–257 (2006).

38. Gottardi, M., Massari, N. & Jawed, S. A. A 100μW 128×64 Pixels Contrast-Based Asynchronous Binary Vision Sensor for Sensor Networks Applications. *IEEE Journal of Solid-State Circuits* **44,** 1582–1592 (2009).

39. Chin, Y.-J. & Berger, T. A software-only videocodec using pixelwise conditional differential replenishment and perceptual enhancements. *IEEE Transactions on Circuits and Systems for Video Technology* **9,** 438–450 (1999).

40. Hamamoto, T. *et al. Computational image sensors for on-sensor-compression* in *Proceedings of Fifth International Conference on Microelectronics for Neural Networks* (1996), 297–304.

41. Gruev, V. & Etienne-Cummings, R. A pipelined temporal difference imager. *IEEE Journal of Solid-State Circuits* **39,** 538–543 (2004).

42. Lenero-Bardallo, J. A., Serrano-Gotarredona, T. & Linares-Barranco, B. *A mismatch calibrated bipolar spatial contrast AER retina with adjustable contrast threshold* in *2009 IEEE International Symposium on Circuits and Systems* (2009), 1493–1496.

43. Arreguit, X., Van Schaik, F. A., Bauduin, F. V., Bidiville, M. & Raeber, E. A CMOS motion detector system for pointing devices. *IEEE Journal of solid-state circuits* **31,** 1916–1921 (1996).

44. Chan, V., Jin, C. & van Schaik, A. An address-event vision sensor for multiple transient object detection. *IEEE transactions on biomedical circuits and systems* **1,** 278–288 (2007).

45. Schiemenz, F., Utzmann, J. & Kayal, H. Survey of the operational state of the art in conjunction analysis. *CEAS Space Journal* **0.** ISSN: 1868-2502 (2019).

46. Kessler, D. J. & Cour-Palais, B. G. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research* **83,** 2637. ISSN: 0148-0227. http://doi.wiley.com/10.1029/JA083iA06p02637 (June 1978).

47. Bobrinsky, N. & Del Monte, L. The space situational awareness program of the European Space Agency. *Cosmic Research* **48,** 392. ISSN: 0010-9525. http://link.springer.com/10.1134/S0010952510050035 (Oct. 2010).

48. Lal, B. *et al. Global Trends in Space Volume 1: Background and Overall Findings* tech. rep. P-5242 (IDA Science and Technology Policy Institute, Washington DC, (2015)), 34.

49. National Research Council of the National Academies. *Continuing Kepler's Quest* 83. ISBN: 978-0-309-26142-5 (National Academies Press, Washington, D.C., Sept. (2012)).

50. Cohen, G. *et al.* Event-based Sensing for Space Situational Awareness. *The Journal of the Astronautical Sciences.* ISSN: 0021-9142 (2019).

51. Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B. & Delbruck, T. Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proceedings of the IEEE* **102,** 1470. ISSN: 00189219 (2014).

52. Cheung, B., Rutten, M., Davey, S. & Cohen, G. Probabilistic Multi Hypothesis Tracker for an Event Based Sensor. *2018 21st International Conference on Information Fusion, FUSION 2018,* 933 (2018).

53. Lambert, A. & Cohen, G. *Study of a spatio-temporal sensor for turbulence characterisation and wavefront sensing* (2018).

54. Palmer, D. M. & Holmes, R. M. Extremely Low Resource Optical Identifier: A License Plate for Your Satellite. *Journal of Spacecraft and Rockets* **55,** 1014. ISSN: 0022-4650. arXiv: 1802.04820 (2018).

55. Cohen, G., Afshar, S. & van Schaik, A. *Approaches for Astrometry using Event-Based Sensors* in *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS) (2018)* (), 1.

56. Burt, P. *et al.* Object tracking with a moving event camera. *Proceedings. Workshop on Visual Motion,* 2 (2016).

57. Drazen, D., Lichtsteiner, P., Häfliger, P., Delbrück, T. & Jensen, A. Toward real-time particle tracking using an event-based dynamic vision sensor. *Experiments in Fluids* **51,** 1465 (2011).

58. Ni, Z., Ieng, S., Posch, C., Régnier, S. & Benosman, R. Visual Tracking using Neuromorphic Asynchronous Event-based Cameras. *Neural Computation* **27,** 925 (2015).

59. Tedaldi, D., Gallego, G., Mueggler, E. & Scaramuzza, D. Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS). *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP),* 1. http://ieeexplore.ieee.org/document/7605086/ (2016).

60. Lagorce, X., Meyer, C., Ieng, S.-H., Filliat, D. & Benosman, R. Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE transactions on neural networks and learning systems,* 1–12. ISSN: 2162-2388 (Sept. 2014).

61. Chin, T.-J., Bagchi, S., Eriksson, A. & van Schaik, A. *Star tracking using an event camera* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2019)* (2019).

62. Aggarwal, A. *SL-8 R/B Satellite details 1968-040B NORAD 3230* https://www.n2yo.com/satellite/?s=3230. (2019).

63. Woodard, M., Lotspeich, N. & Vitale-Sullivan, M. *TheSkyX User Manual* College of Idaho (). 4 pp. January, 2019.

64. Afshar, S., van Schaik, A. & Cohen, G. *Space Imaging Dataset* https://www.westernsydney.edu.au/icns/reproducible_research/publication_support_materials/space_imaging. (2019).

65. Taverni, G. *et al.* Front and back illuminated dynamic and active pixel vision sensors comparison. *IEEE Transactions on Circuits and Systems II: Express Briefs* **65,** 677 (2018).

66. Żołnowski, M., Reszelewski, R., Moeys, D. P., Delbruck, T. & Kamiński, K. OBSERVATIONAL EVALUATION OF EVENT CAMERAS PERFORMANCE IN OPTICAL SPACE SURVEILLANCE (2019).
67. Youden, W. J. Index for rating diagnostic tests. *Cancer* **3,** 32 (1950).
68. Powers, D. M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation (2011).
69. Czech, D. & Orchard, G. *Evaluating noise filtering for event-based asynchronous change detection image sensors* in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)* **2016-July** (IEEE, June (2016)), 19. ISBN: 978-1-5090-3287-7.
70. Cox, D. NOTES ON THE ANALYSIS OF MIXED FREQUENCY 1 DISTRIBUTIONS. *British Journal of Mathematical and Statistical Psychology* **19,** 39 (1966).
71. Hartigan, J. A., Hartigan, P. M. *et al.* The dip test of unimodality. *The Annals of Statistics* **13,** 70 (1985).
72. Silverman, B. W. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society: Series B (Methodological)* **43,** 97 (1981).
73. Cheng, M.-Y., Gasser, T. & Hall, P. Nonparametric density estimation under unimodality and monotonicity constraints. *Journal of Computational and Graphical Statistics* **8,** 1 (1999).
74. Conradt, J. *et al. A pencil balancing robot using a pair of AER dynamic vision sensors* in *2009 IEEE International Symposium on Circuits and Systems* (2009), 781.
75. Seifozzakerini, S., Yau, W.-Y., Zhao, B. & Mao, K. *Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor.* in *BMVC* (2016).
76. Reverter Valeiras, D. *Event-based detection and tracking* PhD thesis (Paris 6, 2017).
77. Benosman, R., Ieng, S.-H., Clercq, C., Bartolozzi, C. & Srinivasan, M. Asynchronous frameless event-based optical flow. *Neural Networks* **27,** 32 (2012).
78. Orchard, G. *et al.* HFirst: A Temporal Approach to Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8828,** 1–1. ISSN: 0162-8828 (2015).

79. Giulioni, M., Corradi, F., Dante, V. & Del Giudice, P. Real time unsupervised learning of visual stimuli in neuromorphic VLSI systems. *Scientific reports* **5,** 14730 (2015).

80. Lagorce, X., Ieng, S. H., Clady, X., Pfeiffer, M. & Benosman, R. Spatiotemporal features for asynchronous event-based data. *Frontiers in Neuroscience* **9,** 1–13. ISSN: 1662-453X (2015).

81. Lagorce, X., Orchard, G., Gallupi, F., Shi, B. E. & Benosman, R. HOTS: A Hierarchy Of event-based Time-Surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8828,** 1–1. ISSN: 0162-8828 (2016).

82. Peng, X., Zhao, B., Yan, R., Tang, H. & Yi, Z. Bag of events: An efficient probability-based feature extraction method for aer image sensors. *IEEE transactions on neural networks and learning systems* **28,** 791–803 (2016).

83. Tapson, J., Cohen, G. & van Schaik, A. ELM solutions for event-based systems. *Neurocomputing* **149,** 435–442 (2015).

84. Ghosh, R., Mishra, A., Orchard, G. & Thakor, N. V. *Real-time object recognition and orientation estimation using an event-based camera and CNN* in *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings* (2014), 544–547.

85. Glover, A. & Bartolozzi, C. *Event-driven ball detection and gaze fixation in clutter* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), 2203–2208.

86. Glover, A. & Bartolozzi, C. *Robust visual tracking with a freely-moving event camera* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), 3769–3776.

87. Cohen, G. *et al. Event-based Sensing for Space Situational Awareness* in *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)* (2017), 1–13.

88. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **70,** 489–501. ISSN: 09252312 (Dec. 2006).

89. Orchard, G., Jayawant, A., Cohen, G. K. & Thakor, N. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience* **9,** 1–15. ISSN: 1662-453X (Nov. 2015).

90. Serrano-Gotarredona, T. & Linares-Barranco, B. *MNIST-DVS Database* 2015. `http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html` (2015).

91. Hu, Y., Liu, H., Pfeiffer, M. & Delbruck, T. DVS Benchmark Datasets for Object Tracking, Action Recognition, and Object Recognition. *Frontiers in Neuroscience* **10,** 1–5. ISSN: 1662-453X (Aug. 2016).

92. Barranco, F., Fermuller, C., Aloimonos, Y. & Delbruck, T. A dataset for visual navigation with neuromorphic methods. *Frontiers in Neuroscience* **10,** 1–9. ISSN: 1662453X (2016).

93. Afshar, S., Hamilton, T. J., Tapson, J. C., van Schaik, A. & Cohen, G. K. Investigation of event-based memory surfaces for high-speed detection, unsupervised feature extraction, and object recognition. *Frontiers in neuroscience* **12,** 1047 (2018).

94. Afshar, S. *et al.* Turn down that noise: synaptic encoding of afferent SNR in a single spiking neuron. *IEEE transactions on biomedical circuits and systems* **9,** 188–196 (2015).

95. Markram, H., Lübke, J., Frotscher, M. & Sakmann, B. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science* **275,** 213–215 (1997).

96. Sofatzis, R. J., Afshar, S. & Hamilton, T. J. *The Synaptic Kernel Adaptation Network* in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, June 2014), 2077–2080. ISBN: 978-1-4799-3432-4.

97. Afshar, S., George, L., Tapson, J., van Schaik, A. & Hamilton, T. J. Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels. *Frontiers in Neuroscience* **8,** 1–18. ISSN: 1662-453X (Nov. 2014).

98. Masquelier, T. & Thorpe, S. J. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology* **3,** 0247–0257. ISSN: 1553734X (2007).

99. Akolkar, H., Valeiras, D. R., Benosman, R. & Bartolozzi, C. Visual-Auditory saliency detection using event-driven visual sensors. *Proceedings of 1st International Conference on Event-Based Control, Communication and Signal Processing, EBCCSP 2015* (2015).

100. Yang, H., Shao, L., Zheng, F., Wang, L. & Song, Z. Recent advances and trends in visual tracking: A review. *Neurocomputing* **74,** 3823–3831. ISSN: 09252312 (2011).

101. Harris, C. & Stephens, M. *A Combined Corner and Edge Detector* in *Procedings of the Alvey Vision Conference 1988* (1988), 147–151.

102. Marĉelja, S. Mathematical description of the responses of simple cortical cells. *Journal of the Optical Society of America* **70,** 1297. ISSN: 0030-3941 (Nov. 1980).

103. Vasco, V., Glover, A. & Bartolozzi, C. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. *IEEE International Conference on Intelligent Robots and Systems* **2016-Novem,** 4144–4149. ISSN: 21530866 (2016).

104. Clady, X., Ieng, S. H. & Benosman, R. Asynchronous event-based corner detection and matching. *Neural Networks* **66,** 91–106. ISSN: 18792782 (2015).

105. Ieng, S.-H., Posch, C. & Benosman, R. Asynchronous Neuromorphic Event-Driven Image Filtering. *Proceedings of the IEEE* **102,** 1485–1499. ISSN: 0018-9219 (Oct. 2014).

106. Brandli, C., Strubel, J., Keller, S., Scaramuzza, D. & Delbruck, T. ELiSeD-An event-based line segment detector. *2016 2nd International Conference on Event-Based Control, Communication, and Signal Processing, EBCCSP 2016 - Proceedings* (2016).

107. Brandli, C., Berner, R., Minhao Yang, Shih-Chii Liu & Delbruck, T. A 240 x 180 130 dB 3 us Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits* **49,** 2333–2341. ISSN: 0018-9200 (Oct. 2014).

108. Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M. & Poggio, T. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29,** 411–426. ISSN: 01628828 (2007).

109. Chandrapala, T. N. & Shi, B. E. Invariant feature extraction from event based stimuli. *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob),* 1–6. http://ieeexplore.ieee.org/document/7523449/ (2016).

110. Chandrapala, T. N. & Shi, B. E. The generative Adaptive Subspace Self-Organizing Map. *Proceedings of the International Joint Conference on Neural Networks,* 3790–3797. ISSN: 2161-4393 (2014).

111. Ballard, D. *et al.* Dynamic coding of signed quantities in cortical feedback circuits. *Frontiers in psychology* **3,** 254 (2012).

112. Cohen, G. K. *Event-based feature detection, recognition and classification* PhD thesis (Paris 6, 2016).

113. Rueckauer, B. & Delbruck, T. Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor. *Frontiers in Neuroscience* **10,** 1–17. ISSN: 1662-453X (Apr. 2016).

114. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278–2323. ISSN: 00189219 (1998).

115. Cohen, G. K. *et al.* Skimming Digits: Neuromorphic Classification of Spike-Encoded Images. *Frontiers in Neuroscience* **10,** 1–11. ISSN: 1662-453X (Apr. 2016).

116. Lee, J. H., Delbruck, T. & Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience* **10.** ISSN: 1662453X. arXiv: 1608.08782 (Nov. 2016).

117. Cohen, G. *et al.* Spatial and Temporal Downsampling in Event-Based Visual Classification. *IEEE transactions on neural networks and learning systems,* 1–15 (2018).

118. Van Schaik, A. & Tapson, J. Online and adaptive pseudoinverse solutions for ELM weights. *Neurocomputing* **149,** 233–238. ISSN: 09252312 (Feb. 2015).

119. Kendall, M. G. *et al.* The advanced theory of statistics. *The advanced theory of statistics.* (1946).

120. Sen, A., Sen, M. A., Amartya, S., Foster, J. E., Foster, J. E. *et al. On economic inequality* (Oxford University Press, 1997).

121. Zappa, F., Tisa, S., Tosi, A. & Cova, S. Principles and features of single-photon avalanche diode arrays. *Sensors and Actuators A: Physical* **140,** 103–112 (2007).

122. Zappa, F., Gulinatti, A., Maccagnani, P., Tisa, S. & Cova, S. SPADA: Single-photon avalanche diode arrays. *IEEE Photonics Technology Letters* **17,** 657–659 (2005).

123. Bellisai, S. *et al.* Single-photon pulsed-light indirect time-of-flight 3D ranging. *Optics express* **21,** 5086–5098 (2013).

124. Berkovich, A., Datta-Chaudhuri, T. & Abshire, P. *A scalable 20× 20 fully asynchronous SPAD-based imaging sensor with AER readout* in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)* (2015), 1110–1113.

125. Shamim, M., Shawkat, A. & McFarlane, N. *A CMOS Perimeter Gated SPAD Based mini-Digital Silicon Photomultiplier* in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)* (2018), 302–305.

126. Villa, F. *et al.* CMOS imager with 1024 SPADs and TDCs for single-photon timing and 3-D time-of-flight. *IEEE journal of selected topics in quantum electronics* **20,** 364–373 (2014).

127. Afshar, S., Hamilton, T. J., van Schaik, A. & Delic, D. *SPAD AirPlane Dataset* https://www.westernsydney.edu.au/icns/reproducible_research/publication_support_materials/SPAD_planes. (2019).

128. Otsu, N. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* **9,** 62–66 (1979).

129. Kittler, J. & Illingworth, J. Minimum error thresholding. *Pattern recognition* **19,** 41–47 (1986).

130. Sauvola, J. & Pietikäinen, M. Adaptive document image binarization. *Pattern recognition* **33,** 225–236 (2000).

131. Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. *Xnor-net: Imagenet classification using binary convolutional neural networks* in *European Conference on Computer Vision* (2016), 525–542.

132. Mau, J. *et al. Embedded implementation of a random feature detecting network for real-time classification of time-of-flight SPAD array recordings* in *Proc. SPIE 11005, Laser Radar Technology and Applications XXIV* **11005** (2019).

133. Tapson, J. & van Schaik, A. Learning the pseudoinverse solution to network weights. *Neural Networks* **45,** 94–100 (2013).