3-26-2020

A Comparative Evaluation of the Detection and Tracking Capability Between Novel Event-Based and Conventional Frame-Based Sensors

James P. Boettiger

A COMPARATIVE EVALUATION OF THE DETECTION AND TRACKING CAPABILITY BETWEEN NOVEL EVENT-BASED AND CONVENTIONAL FRAME-BASED SENSORS

THESIS

James P Boettiger, Flight Lieutenant, RAAF

AFIT-ENG-MS-20-M-007

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-20-M-007

A COMPARATIVE EVALUATION OF THE DETECTION AND TRACKING CAPABILITY BETWEEN NOVEL EVENT-BASED AND CONVENTIONAL FRAME-BASED SENSORS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

James P Boettiger, BE BSc MSc

Flight Lieutenant, RAAF

March 26, 2020

AFIT-ENG-MS-20-M-007

A COMPARATIVE EVALUATION OF THE DETECTION AND TRACKING CAPABILITY BETWEEN NOVEL EVENT-BASED AND CONVENTIONAL FRAME-BASED SENSORS

THESIS

James P Boettiger, BE BSc MSc
Flight Lieutenant, RAAF

Committee Membership:

Michael A Marciniak, Ph.D
Chair

Michael R Hawks, Ph.D
Member

Robert C Leishman, Ph.D
Member

Abstract

Event-based sensors are a novel type of visual sensor that operate entirely differently to their traditional frame-based counterparts. Not bound by a synchronous frame-rate, pixels from these sensors respond both independently and asynchronously to changes in log-light intensity. Because of this operating paradigm, these sensors afford unique characteristics such as low power consumption, reduced bandwidth requirements, high temporal resolution (1 μs) and high dynamic range (120 dB). They therefore threaten to disrupt the well-established frame-based community and offer potential for significant enhancement to existing and future Defense capability.

This research centers around object detection and tracking. Traditional frame-based algorithms remain tied to predefined frame-rates and therefore suffer from motion blur, low dynamic range, speed limitations and high data processing requirements. Event-based sensors therefore present a novel solution to these challenges.

This research establishes a fundamental understanding of the characteristics of event-based sensors and applies it to the development of a detection and tracking algorithm. In parallel, a basic frame-based detection and tracking algorithm has been developed and used to perform a comparative assessment against two different event-based detection and tracking algorithms. The first approach used event-based pseudo-frames parsed through standard frame-based tracking algorithms while the second used filtered events asynchronously to construct target tracks directly. While effective detection and tracking was demonstrated, overall this research did not find an ideal event-based solution that could be considered suitable to replace existing frame-based technology for this application. Nevertheless, the findings did show there is significant value in pursuing this technology further.

Table of Contents

Page

Page

List of Figures

List of Tables

A COMPARATIVE EVALUATION OF THE DETECTION AND TRACKING CAPABILITY BETWEEN NOVEL EVENT-BASED AND CONVENTIONAL FRAME-BASED SENSORS

I. Introduction

1.1 Problem Background

The detection and tracking of moving objects in free space is an area of computer vision that has benefited greatly from many years of research and development. To date there are many different algorithms available with new and improved revisions being developed on a regular basis. With such maturity, it is not surprising that such algorithms provide very effective solutions in a wide range of applications. There are however, select scenarios in which traditional detection and tracking algorithms break down where in many cases it is not attributable to the algorithm but rather the fundamental operation of a frame-based sensor. Despite extensive research, traditional frame-based algorithms remain tied to predefined frame-rates that lead to image artifacts such as motion blur and sensor characteristics such as low dynamic range, speed limitations and the requirement to process large data files often filled with copious amounts of redundant data.

Event-based sensors, also known as silicon retinas or neuromorphic sensors, are revolutionary optical sensors that operate fundamentally differently to traditional frame-based sensors and offer the potential of a novel solution to these challenges. Inspired by the functionality of a biological retina, these sensors are driven by changes in log-light intensity and not by artificial frame rates and control signals. Within these

sensors each pixel behaves both asynchronously and independently, enabling events to be generated with microsecond resolution in response to localized optical changes as they occur. The asynchronous nature of individual pixels within an event-based sensor implies there is no fixed frame rate. This in turn eliminates motion blur and enables a dynamic range in the order of 120 dB, a significant increase from 60 dB found in traditional frame-based technology. Additionally, as it is only events that generate information, these sensors require very minimal data storage and therefore have very low bandwidth requirements. It is these features that afford these sensors enormous potential on which to base a detection and tracking algorithm.

While originally developed with machine vision and autonomous vehicles in mind, the nature of these sensors has sparked interest in their ability to translate to the Defense environment. With the relative youth of this technology much is yet to be discovered and despite a variety of applications having been explored in recent years, to date the specific adaptation of these sensor to Defense has only been briefly explored. Given the unique characteristics of these sensors and their inherent advantages over traditional frame-based technology, the translation of these sensors to the Defense environment seems a natural progression.

1.2 Research Objectives

The intent of this research is to build a fundamental understanding of the characteristics of event-based sensors and apply that knowledge to the development of a detection and tracking algorithm that takes full advantage of its unique operation. In parallel, a fundamental frame-based detection and tracking algorithm will be developed and used to perform a comparative assessment as to the perceived benefits of event-based sensors and their overall potential for future development and incorporation into existing and future Defense capability.

For simplification, the scope of this research is limited solely to a stationary sensor in an effort to resemble a staring surveillance type function. Additionally, all algorithms produced as a result of this research have been developed using MATLAB® and so do not represent an optimized or streamlined product written in a faster computer language. While this could effect overall computational times, the impact on a comparative assessment is considered shared across both sensor types.

This research also acknowledges that event-based sensors are an immature technology in terms of design and production and as such they have low resolution and a significant vulnerability to noise. This research will therefore act as a proof of concept under the assumption that sensor quality will improve in time with a reciprocal improvement in capability.

This research will highlight the potential of event-based sensors to the Defense community and provide a launching point for further research and development. If the unique features of the event-based sensor can be harnessed to their full potential it is expected that a high speed, low power, high dynamic range detection and tracking product will be developed with comparable, if not better, performance than its frame-based counterpart. Challenges expected to arise along the way include the ability to quickly process and draw relationships between a large number of separate events, but perhaps more significant will be the ability to associate asynchronous and spatially separate event-based detections. Should these challenges be overcome and research with these sensors progress beyond this document, simulations and system trials are expected to follow with eventual incorporation into service equipment.

1.3 Document Overview

This document is broken down into seven chapters. Chapter I has first discussed limitations of frame-based detection and tracking and introduced the disruptive po-

tential of event-based sensors. Research objectives are also highlighted with attention drawn to scope, assumptions and anticipated challenges. Chapter II is a literature review that provides relevant background into the development of the event-based sensor and introduces a number of current research areas and applications. Chapter III characterizes the event-based sensor providing detail of circuit makeup and typical performance measures. In Chapters IV to VI the comparative analysis is formed. Chapter IV begins by detailing a fundamental frame-based detection and tracking algorithm providing insight into individual steps with support from specific examples. Similarly, Chapter V introduces event-based detection and tracking including the development and implementation of several event-based filters as a precursor to the development of two separate variants of event-based tracking. With the use of several different detection and tracking scenarios, the performance of algorithms developed in Chapters IV and V are compared in terms of tracking capability and computational load in Chapter VI. Chapter VII then concludes the document by providing a summary of all findings with an emphasis on the potential of event-based sensors and the need for further research.

II. Background and Literature Review

Event-based sensors are a recently developed technology that is gaining popularity in the field of artificial vision systems. Biologically inspired, the operation of these systems is fundamentally different than traditional frame-based imagers. Across a Focal Plane Array (FPA), each pixel responds asynchronously and independently to changes in log-light intensity, producing event-based output with high temporal resolution. The asynchronous nature of these sensors implies there are no fixed frame rates, and as such these devices offer a high dynamic range documented to be in the order of 120 dB [1]. Additionally, as it is only events that generate data, these devices require minimal storage and therefore communication bandwidth.

While originally developed with machine vision and autonomous vehicles in mind, the nature of event-based sensors has sparked interest in their ability to translate to the Defense environment. As they operate entirely differently than frame-based sensors, their output must also be interpreted entirely differently. To support the development of new technologies, it is crucial to characterize and predict the performance of these devices in addition to appreciating their limitations. With the successful characterization of these sensors it is anticipated that Defense will become motivated to optimize forthcoming technologies such as space imaging [2], autonomous navigation [3] or target detection and tracking [4, 5] in addition to developing unrealized technologies.

2.1 Biological Inspiration

A growing understanding in biological vision is leading the way to a more efficient and effective means of viewing one's environment. Beginning with the human eye ball, from an engineering perspective it is an incredibly capable organ. On one hand

it can function with limited photons under an overcast night sky emitting 10^{-6} cd/m^2, yet on the other hand it can function on snow covered slopes drenched in sunlight emitting up to 10^5 cd/m^2, a dynamic range of over 10 decades or 200 dB [6].

Event-based sensors specifically aim to mimic the biological retina and subsequent vision processing of the brain. While the retina is the photosensitive tissue of the eye, for it to work properly the entire eye ball is needed. Figure 1(a) shows a schematic cross section of a typical eye. Light entering the eye passes through the cornea and into the first of two humours. The aqueous humour is a clear mass that connects the cornea with the lens, helping to maintain the shape of the cornea. Between the aqueous humor and lens lies the iris, a colored ring of muscle fibres. The iris forms an adjustable aperture, called the pupil, which is actively adjusted to ensure a relatively constant amount of light enters the eye at all times. While the cornea has a fixed curvature, the shape of the lens can be actively moulded to adjust the eyes focal length as needed. On the back side of the lens is the vitreous humor that again helps to maintain the shape of the eye. Light entering the pupil then forms an inverted image on the retina.

The retina contains the photosensitive rods and cones and is a relatively smooth, curved layer with two distinct points; the fovea and optic disc. Densely populated with

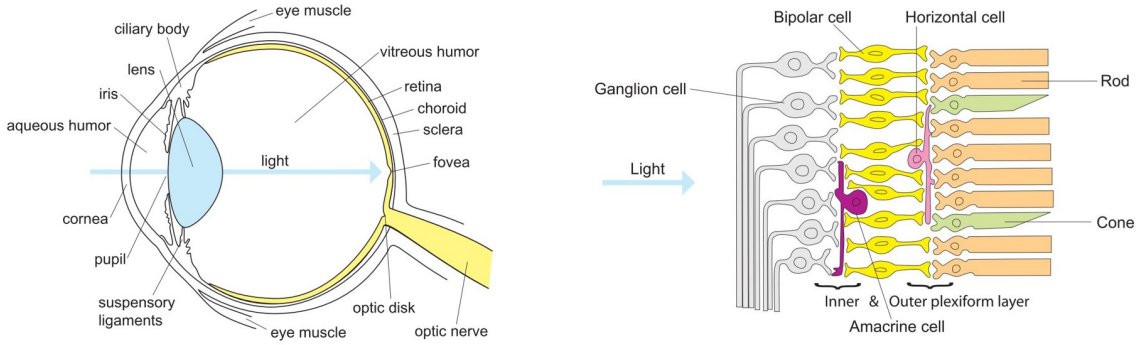


Figure 1: (a) Cross section of an eye ball (b) Schematic of the human retina [7].

cone cells, the fovea is positioned directly opposite the lens and is largely responsible for color vision. The optic disc, a blind spot in the eye, is where the axons of the ganglion cells leave the eye to form the optic nerve.

Although rods and cones are structurally and metabolically similar, their function is quite different. Rods are highly sensitive to light allowing them to respond to dim light and dark conditions, however they do not distinguish between colors and have low visual acuity. Conversely, cones need high light intensities to respond and have high visual acuity. Within the human eye, there are three different variants of cones each responsive to different bands in the visible spectrum, allowing the perception of color.

From the photoreceptors, neural responses pass through a series of linking cells, called bipolar, horizontal and amacrin cells (see Figure 1(b)). These cells combine and compare the responses from individual photoreceptors before transmitting the signals to the retinal ganglia cells. The linkage between neighboring cells provides a mechanism for spatial and temporal filtering, facilitating relative, rather than absolute, judgment of intensity, emphasizing edges and temporal changes in the visual field of view. It is the network of photoreceptors, horizontal cells, bipolar cells, amacrin cells and ganglion cells that can discriminate between useful information to be passed to the brain, and redundant information that is best discarded immediately.

Vision is therefore not so much about measuring the power and spectral composition of light, but more about using temporal changes in contrast to perceive one's environment. Objects within one's field of view, their form, their structure, their surface and their motion become the most important aspects of vision. While objects do not change under different lighting conditions, the light reflected from objects is largely dependent on their illumination source, implying that absolute illumination carries information about the light source and the object being illuminated. Replacing

absolute illumination information with local ratios of illumination differences removes information about the light source, and yet retains information about the illuminated object. It is this aspect that biological vision systems have taken advantage of and the key driver in the development of event-based sensors.

2.2 Event-Based Sensors

The first event-based vision sensor was developed when Misha Mahowald, a biology student, and Carver Mead, a transistor physicist, joined at the California Institute of Technology (Caltech) in 1991 [8]. As a demonstration device it incorporated adaptive photoreceptors, a spatial smoothing network and self-timed communication, however it suffered from several short-comings rendering it essentially unusable for real world applications. In 2004 Zaghoul and Boahen then incorporated both sustained and transient types of cells with adaptive spatial and temporal filtering to create a device that closely captured key adaptive features of biological retinas [9]. However, as this circuit design was intended more as a biological model rather than a practical device, it suffered poor response uniformity and limited dynamic range.

Up until the early 2000s, the development of event-based sensors focused largely on mimicking their biological counterparts as precisely as possible. Since then however, many different sensor types have been developed that cooperate with conventional processors and can be used in practical applications as alternate vision devices. In principle, any imaging sensor with an architecture that emulates biology's sparse data-driven signalling can be classified as event-based sensor, and as such a number of different sensing modalities have been proposed. One such example is a contrast sensor that uses intensity ratios between the local light intensity of a single pixel and a spatially weighted average of its nearest neighbors to reduce redundancy [10, 11]. A more common sensor however is based on absolute, or relative, intensity changes over

time [12, 13]. While more high-level abstraction sensors include gradient and optical flow sensors [14, 15], the most commercially available are contrast sensors that reduce temporal redundancy based on the change in absolute intensity over time.

2.3 Event Versus Frame-Based Sensors

To build an intuitive understanding of the output of event-based sensors, it is useful to look at the difference between it and a conventional frame-based sensors. While each pixel in a standard frame-based sensor captures absolute brightness over a fixed time interval (i.e. frame rate) across its FPA, pixels from an event-based sensor behave both independently and asynchronously to capture changes in log intensity. What this means practically is highlighted in Figure 2 where the difference in the output of a frame-based versus event-based sensor is compared when imaging a black dot on a rotating disk. The top sequence from a frame-based sensor shows a series of snapshots taken in time where, due to the frame-based nature of this sensor, it is evident there is considerable blind time between frames. Also, as the only change from one frame to another is the location of the black dot, a lot of redundant information is captured and stored, even when the disc is not rotating. Furthermore, due to the inherent need for an exposure time to capture a frame, if the scene is changing too fast, motion blur can become a significant artifact as seen in the last three frames of this sequence. It is also this exposure time that leads to the limited dynamic range of a frame-based sensor.

In contrast, the bottom sequence of Figure 2 shows the output from an event-based sensor. Here an almost continuous (μs resolution) stream of asynchronous events is captured as the black dot rotates (red and blue dots represent an increase or decrease in log-light intensity respectively). In contrast to the frame-based sensor, there is no missing information in between frames, no redundancy as only changes

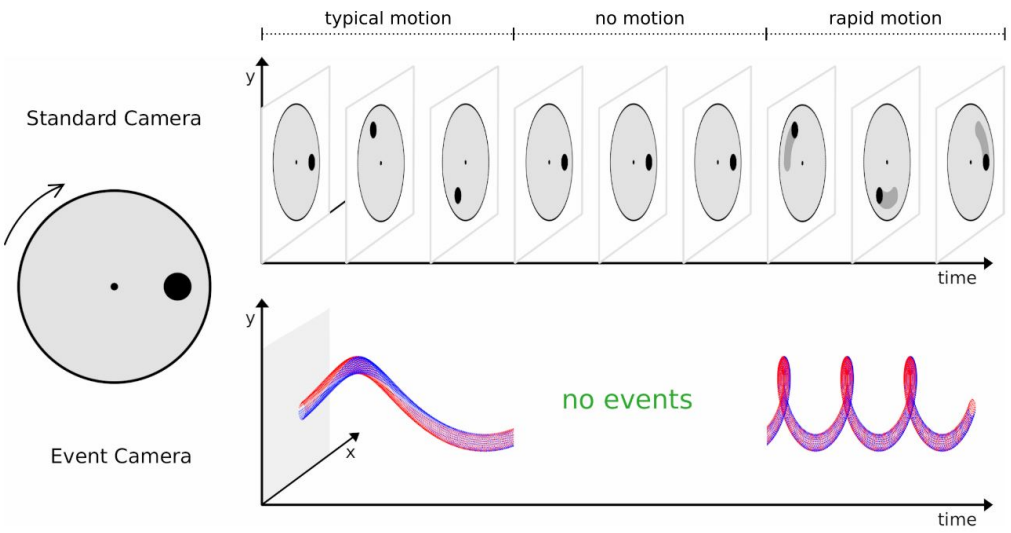


Figure 2: Comparison of the output from a conventional frame-based sensor versus a novel event-based sensor where red and blue dots represent an increase or decrease in log-light intensity respectively [16, 17].

are captured, and no motion blur even at a high rotational speed. Additionally, by encoding only scene changes, the bandwidth requirements needed to process and store a stream of events such as this is much less than for a frame-based sensor. Event-based sensors therefore have the potential to combat the low frame rate, high latency, low dynamic range and high power consumption of a traditional frame-based sensor. The drawback however is that the disruptive nature of these sensors presents a significant paradigm shift as engineers work to integrate their capabilities into new and existing applications.

2.4 Event-Based Sensor Types

With the progression of event-based sensors from laboratory prototypes to commercial products, there have been several generational models developed by multiple organizations. This section outlines the fundamental types of event-based sensors that are available today with a brief explanation as to their mode of operation and performance characteristics.

2.4.1 Dynamic Vision Sensor (DVS)

Widely considered the first practical event-based sensor, the Dynamic Vision Sensor (DVS) was developed in the mid-2000s by iniVation in conjunction with the Institute for Neuro-Informatics (INI) at the Swiss Federal Institute of Technology in Zurich (ETH-Zurich). Originally developed as a 64×64 pixel array in 2005, a full scale 128×128 pixel version was created the following year [13, 18, 19].

As the pioneering sensor, some key performance metrics were introduced and published in 2008 [13]. Beyond the standard performance metrics of conventional frame-based sensors (i.e. power consumption and pixel pitch), minimum attainable event threshold, threshold mismatch, pixel bandwidth, dynamic range and latency were all identified as unique and quantifiable metrics applicable to event-based sensors.

To introduce these metrics as they are applicable to all current event-based sensors, event threshold refers to the percentage change in scene illumination required to generate an event while mismatch is a measure of how this threshold changes across the FPA. It is essentially taken as the standard deviation from the mean threshold measured for each pixel, assuming normal distribution. For the original DVS, event threshold and mismatch were determined to be approximately 14% and 2.1% respectively, figures that would be improved on in later models.

Pixel bandwidth refers to the low pass filter characteristics of the DVS circuitry and is the maximum rate that a single pixel can accurately generate events in response to changes in brightness. Not to be confused with the microsecond temporal resolution of an event timestamp across the sensors FPA, pixel bandwidth for a single DVS pixel ranges from 300-3000 Hz and is heavily dependent on background illumination and sensor bias settings.

As with conventional frame-based sensors, dynamic range is defined as the ratio of maximum to minimum scene illumination at which events can be generated by high

contrast stimuli. For a DVS this has been demonstrated at over 120 dB.

Finally, latency refers to the time delay between sensing an illumination change and generating a corresponding event. Heavily dependent on the level of activity in a scene, the data bus reading out events can become a significant bottleneck whereby latency has been reported between the range of 15 μs to 4 ms.

2.4.2 Asynchronous Time-Based Imaging Sensor (ATIS)

Despite the purely event based nature of the DVS, many applications still require knowledge about absolute illumination levels. This led to the development of the Asynchronous Time-Based Imaging Sensor (ATIS) which provides both absolute brightness as well as events [20, 21]. Emerging in 2011 through the Austrian Institute of Technology and manufactured by Prophesee, the ATIS combines the DVS temporal contrast pixel with a new time-based intensity measurement pixel. Triggered by DVS circuitry, the ATIS leverages pulse-width modulation to generate an asynchronous exposure measurement for each pixel in response to a change in brightness.

In addition to providing absolute brightness information, the main advantages compared to the DVS are its higher resolution (304×240), higher dynamic range (143 dB), and lower latency (3 μs). The minimum contrast threshold and mismatch are also reported to be 13% and 0.25% respectively. Some drawbacks however include the increased pixel size and complexity due to the addition of a separate photodiode for intensity measurements, an increase in data output due to pulse-width modulation, and increased motion blur for dimly-lit, fast-moving dynamic scenes.

2.4.3 Dynamic and Active Pixel Vision Sensor (DAVIS)

To address some of the drawbacks of the ATIS, iniVation introduced the Dynamic and Active Vision Image Sensor (DAVIS) in 2014. Designed by Brandli et al. [22],

this sensor interleaves event data with conventional intensity frames rather than per-event intensity measurements. The main advantage of the DAVIS pixel design is that it shares the same photocurrent between the asynchronous detection of brightness changes and the synchronous readout of intensity frames, essentially using the same photodiode to generate both frame and event data. Reported values for dynamic range (130 dB for events and 55 dB for greyscale frames), minimum contrast threshold (11%) and mismatch (3.5%) are similar to earlier systems, whereas latency has been reduced to 3 μs. Additionally, a time-synchronized inertial measurement unit (IMU) has been embedded to provide gyro, accelerometer and magnetometer data for applications requiring electronic stabilization or motion compensation. Ultimately, through the combination of both event and frame-based data, to date the DAVIS represents the most viable sensor for a variety of applications.

2.4.4 Octopus Retina

Rather than modelling the redundancy reduction and local computation done in the retina, some imagers simply use Address Event Representation (AER) to directly communicate pixel intensity values. For example, the Octopus retina [23], named after the animal itself because it has photoreceptors on the outside of the eye rather than the inside, communicates the pixel intensity through the frequency or inter-spike interval of the AER event output. This biologically inspired readout method offers parallel pixel readout such that brighter pixels can be readout at a higher frequency than darker pixels as their well capacity fills sooner over a defined integration time. In contrast, serially scanned arrays allocate equal portions of bandwidth to all pixels independent of activity, and therefore continuously dissipate power as each pixel is readout inline with the frame rate.

The Octopus imager has the advantage of a small pixel size compared to other

event-based sensors, but has the disadvantage that the bus bandwidth is allocated according to the local scene illuminance. Because there is no reset mechanism and the event interval directly encodes intensity, a dark pixel can take a long time to emit an event, and a single highlight in the scene can saturate the readout data bus. Consequently, for an Octopus retina brighter pixels request the output bus more often than darker pixels and are therefore updated more frequently. Therefore, a moving object generating both dark and bright pixels will result in later intensity readings for dark pixels than for bright pixels, distorting the object's shape and creating motion artifacts similar to those seen from the ATIS. Additionally, mostly due to threshold variations in the individual pulse frequency modulation circuits, these sensors have a rather large fixed pattern noise (FPN) making then unsuitable for use as conventional imagers. It is largely for these reasons that Octopus retinas have not yet progressed to a commercial product and as such they will not be considered further in this research.

2.4.5 Color Sensitive Sensors

While some consider sensitivity to color not essential as proven by the many nocturnal animals that are monochromats, in some cases it can provide a discriminative cue and act as an information multiplier. The addition of color information to event-based vision has the potential to improve the performance of many tasks. For example, segmentation and recognition use color as an important source of visual information. Early attempts at color sensitive event-based sensors used the principle whereby the wavelength of different colors is separated depending on its level of penetration into a silicon photodiode. Unfortunately this approach suffered from noise, low sensitivity, and poor color separation.

A more common and widely available approach is to use a color filter array (CFA) which employs red, green and blue filters along with no filter for white. A simplified

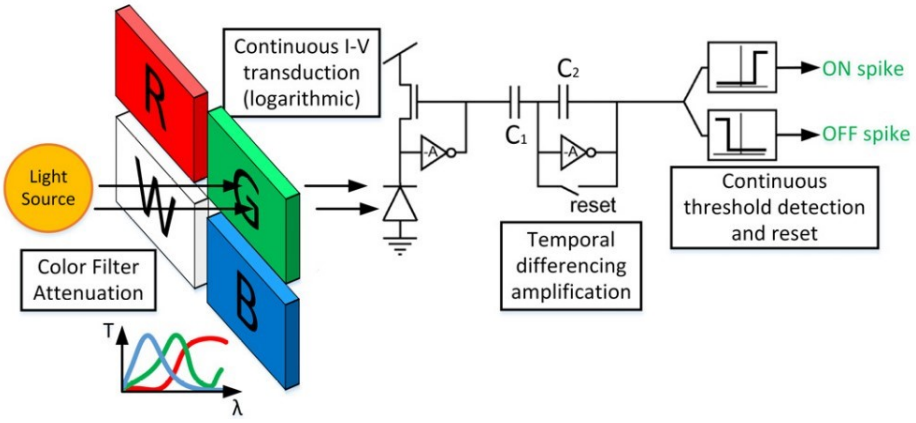


Figure 3: Simplified schematic of DVS pixels with RGBW color filters [24].

schematic of a DVS pixel with a red, green, blue and white (RGBW) color filter is shown in Figure 3 with the green path highlighted. Light incident on the sensor is focused onto the pixel array where its spectrum is modified depending on the absorption characteristics of the color filter. Once the light has reached the photodiode, the DVS pixel (described later in Section 3.3) is used to generate events depending on the level of logarithmic temporal contrast. The idea of a color event, is that if for example a blue bar moves over a red background, at the leading edge, red DVS pixels will respond with OFF events, while blue DVS pixels will respond with ON events. Visa versa for the trailing edge.

Today the DAVIS 346 offers arguably the best available color sensitive event-based sensor on the market, however to date little research has been done in order to realise its true potential. Fortunately datasets containing a variety of scenes are available as in [25] making future research more accessible.

2.4.6 Infrared Dynamic Vision Sensor

In many applications, in particular military, illumination of the target scene in the visible spectrum can be variable, and often make it difficult to discern objects of in-

terest. One way to combat this is to extend or shift the measurable spectrum towards the infrared. By taking advantage of the theory of blackbody radiation, any material or object above absolute zero will emit electromagnetic radiation dependent on its temperature. With typical temperatures found on Earth in the order of 300 K, the peak of Planck's blackbody radiation curve occurs at a wavelength of approximately 10 μm, aligning well with the Long Wave Infrared (LWIR) band ranging from 8 to 14 μm. Therefore when considering the detection and analysis of moving objects, their self-emitted radiation can be used as a distinguishing feature as long as the object's surface temperature differs sufficiently from the background ambient temperature.

A microbolometer is a thermal sensor whose function is based on the variation of its temperature-dependant electrical resistance due to its absorption of thermal infrared (IR) radiation. Because uncooled microbolometers are sensitive in the LWIR spectral range and can be readily integrated with complementary metal-oxide semiconductor (CMOS) readout circuitry, in 2009 Posch et al. worked to couple a microbolometer array with typical DVS readout circuitry to produce the first IR event-based sensor [26]. Its successful development would enable target tracking in low to zero visible light while taking advantage of the low data rates, low latency and high speed inherent to event-based sensors.

The initial design consisted of a 64×64 pixel microbolometer-based temporal contrast IR vision sensor sensitive in the 8-15μm wavelength range. Functionally very similar to the aforementioned visible event-based sensors, the design contains a microbolometer front end (vice a photodiode), transient amplifier and event generation circuitry. The biasing settings of the pixel readout circuit were set to yield a good trade-off between sensitivity and noise suppression.

The characterization test setup consisted of a rotating chopper disk intermittently blocking thermal radiation from a well defined heating element at a frequency f_c.

When considering the sensor's pixel sensitivity, events start to appear for a temperature difference ΔT between the target and background at approximately 35 K. For a ΔT of 300 K, the average event rate from pixels illuminated by the target reached 80% of that theoretically expected, however this rate dropped dramatically for ΔT below 150 K. A 50% response rate could be aligned with a ΔT of 110 K. The reason for these discrepancies is likely due to the non-ideal performance of the pixel circuits as well as internal and induced noise. This in effect causes additional and/or missing events in addition to time deviations. Despite this, Figure 4 shows an example of this prototype sensor viewing a soldering iron at different temperatures where both images clearly highlight the potential of this technology.

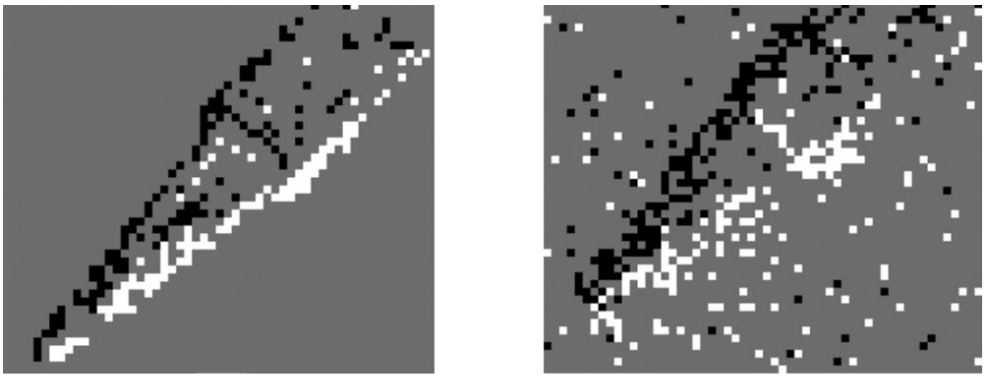


Figure 4: Example output from a IR microbolometer event-based sensor (a) Soldering tip at 400°C (b) 100°C. White and black pixels signify an increase and decrease in thermal brightness respectively [26]

.

Since this initial study in 2009 and a sequential paper in 2010 [27], little work has been published related to IR event-based sensors for unknown reasons. Regardless, when considering the applicability of such a sensor to the military, a sensor of this type could provide a practical, low-cost solution to many surveillance and motion detection problems under difficult illumination conditions.

2.4.7 Commercialized Event-Based Sensors

Table 1 captures the few commercialized event-based sensors available in today's market and compares their specifications and performance characteristics. Shown in chronological order, the DVS 128 was a first generation prototype produces by iniVation back in 2008, since then iniVation have produced a second (DAVIS 240) and now third generation sensor (DAVIS 346) each with improved resolution and performance characteristics. While initial sensors were first produced through university spin-offs in both iniVation and Prophesee, the arrival of tech-giant Samsung highlights how industry foresees the potential of event-based technology. By combining the strength of Samsung's research and development with its existing market presence, one would anticipate the advancements in this technology to escalate quickly in coming years. Due to availability, the DAVIS 240 will be used for all experiments and application demonstrations conducted throughout this research.

Table 1: Comparison of different commercialized event sensors [1]

	DVS128 [13]	**ATIS** [20]	**DAVIS240** [22]	**DAVIS346**	**DVS-Gen2** [28]	**CeleX-IV** [29]
Supplier	iniVation	Prophesee	iniVation	iniVation	Samsung	CelePixel
Year	2008	2011	2014	2017	2017	2017
Resolution (pixels)	128×128	304×240	240×180	346×260	640×480	768×640
Latency (μs)	12μs @ 1 klux	3	12μs @ 1 klux	20	65-410	-
Dynamic Range (dB)	120	143	120	120	90	100
Min. contrast sensitivity (%)	17	13	11	14.3-22.5	9	-
Die power consumption (mW)	23	50-175	5-14	10-170	27-50	-
Camera Max. Bandwidth (Meps)	1	-	12	12	300	200
Chip size (mm^2)	6.3×6	9.9×8.2	5×5	8×6	8×5.8	-
Pixel size (μm^2)	40×40	30×30	18.5×18.5	18.5×18.5	9×9	18×18
Fill Factor (%)	8.1	20	22	22	100	9
Supply voltage (V)	3.3	1.8 & 3.3	1.8 & 3.3	1.8 & 3.3	1.2 & 2.8	3.3
Stationary Noise (ev/pix/s) @ 25°C	0.05	NA	0.1	0.1	0.03	-
CMOS technology μm	0.35	0.18	0.18	0.18	0.09	0.18
	2P4M	1P6M	1P6M MIM	1P6M MIM	1P5M BSI	1P6M CIS
Grayscale output	no	yes	yes	yes	no	yes
Grayscale dynamic range (dB)	-	130	55	56.7	-	-
Max. framerate (fps)	-	NA	35	40	-	-
IMU Output	no	no	1 kHz	1 kHz	no	no

2.5 Demonstrated Applications

Current research into event-based sensors has explored their application to many engineering disciplines. By taking advantage of their small form factor, low power requirements, low data rates, high temporal resolution and high dynamic range, the use of these sensors has been demonstrated in a wide variety of applications. Originally developed with machine vision and autonomous vehicles in mind, their adaptability has been demonstrated in more conventional fields such as optical flow, image reconstruction, visual odometry and object tracking, but also in novel fields such as satellite tracking [2, 30–33]. While new applications are no doubt in development, the following section draws attention to some of these example applications with the potential for Defense utilization.

2.5.1 Optical Flow Estimation

Optical flow estimation refers to the problem of computing the velocity of objects within a sensor's field of view, without prior knowledge about the scene geometry or sensor motion. Despite years of optimization, even some of the most notable frame-based optical flow estimation methods are either computationally expensive [34], struggle under conditions of low or variable illumination, or cannot keep up with fast moving objects due to their inherent frame rate. While events provide neither absolute brightness nor spatially continuous data to exploit with neighboring pixels, should the challenge of manipulating the unfamiliar event format be overcome, the benefit of fine timing information, high dynamic range and low computational cost could prove fruitful.

Optical flow estimation using event-based sensors is typically categorized by differing methods of calculation. One example includes normal flow versus full flow. Normal flow considers only motion perpendicular to a brightness edge (i.e. parallel to

the brightness gradient), while full flow seeks to compute both the normal and tangential components of motion. Another example is sparse versus dense flow. Sparse flow considers optical flow in small localized regions while dense flow considers flow across all pixels. Since event-based sensors respond to moving edges, flow vectors computed at regions with no events generally stem from interpolation, and are therefore considered less reliable. On the other hand, regions of localized events are more likely to generate reliable optical flow vectors. Just with these two examples, event-based sensors have shown promise in alleviating some of the difficulties associated with traditional frame-based optical flow estimations. Comprehensive datasets with accurate ground truth in multiple scenarios would be essential steps to progress this technology [1].

2.5.2 Image Reconstruction

Events are triggered by changes in scene brightness, and so in ideal circumstances (noise free, perfect sensor response, etc), integration of such events can yield 'absolute' brightness. This is intuitive, since events are just a non-redundant per-pixel way of encoding the visual content in a scene. Moreover, due to the very high temporal resolution of events, brightness images can be reconstructed at a very high frame rate, or even continuously in time [35, 36].

By applying a message-passing algorithm between pixels in a network of visual maps, image reconstruction from events was first established in 2011 under rotational camera motion [37]. Also under rotational motion, high-resolution panoramas have been produced from event based data, essentially popularizing the idea of event-based High Dynamic Range (HDR) image reconstruction [38]. In this application each pixel of the panoramic image used a Kalman filter to estimate the brightness gradient, which is then integrated using Poisson reconstruction to yield absolute brightness.

Figure 5 highlights the effectiveness of HDR imagery using event-based sensors. The series of images shows a camera pointing at a traffic sign, heavily backlit by the sun. The left image is from a standard frame-based camera showing severe under-exposure in the foreground. The center image is from a DAVIS showing severe under and over exposed areas while the right image is an HDR image reconstructed from events. Clearly the dynamic range has been improved such that the background and foreground content of the image can be interpreted.

Figure 5: Example of image reconstruction to produce an HDR image [39]

An interesting aspect of reconstructing images from events is that it requires some form of regularization. Because of the independent nature of event-based pixels, per-pixel integration of brightness changes during a time interval only produces brightness increment images. To recover the absolute brightness at the end of a timing interval, an offset image is generally needed to be added to the start of the integration period [40, 41]. Surprisingly, techniques that use temporal smoothing or learned features from natural scenes have also been developed to avoid the need to know the initial conditions of an image [35, 42–44].

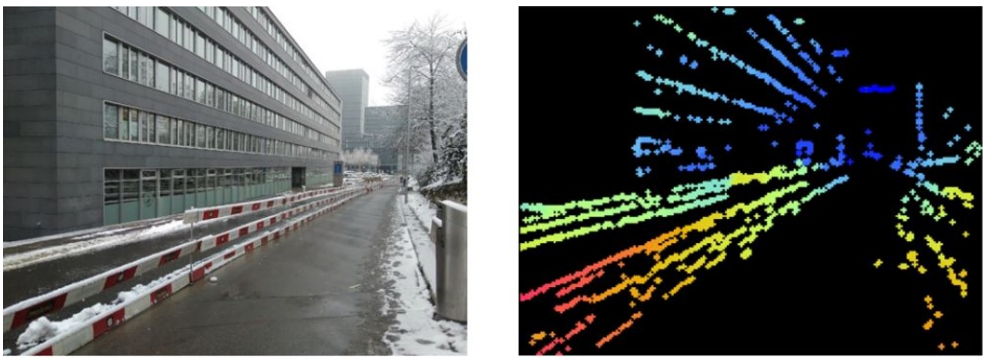
The development of image reconstruction techniques implies that it is possible to convert event-based data into brightness images such that mature computer vision algorithms can be applied with the potential to impact both event-based and

frame-based communities. The drawback is that while event-based reconstructed images capture high-speed motion and HDR scenes, it comes at the cost of increased processing, latency and power consumption. Despite this, the transfer of this technology to space-based staring sensors means the increased need for processing could be moved to a ground station, allowing the low bandwidth requirements of an event-based sensors to be maximized.

2.5.3 3D Reconstruction and Stereo Vision

The generation of high-speed real-time stereo images using traditional frame-based sensors remains computationally expensive and fundamentally difficult [45]. This is largely due to the high amounts of redundant data processed in every frame, additionally, frame-based sensors lack temporal dynamics as each frame requires a defined integration time, a feature incompatible with high-speed vision systems. Asynchronous event-based acquisition properties allow for simplification of the 3D matching process together with low computational cost. It has been shown that stereo matching can be computed in a manner that is not only faster and more efficient, but also more akin to the way in which natural systems process visual information.

3D reconstruction is usually performed using two or more synchronized sensors that are rigidly attached [47], however studies have shown that it can be performed with just a single sensor [46]. Figure 6 shows an example of 3D reconstruction using a single DAVIS where to generate the depth map from events at (b), the sensor head must be moved to acquire visual content. While this introduces additional complexity, it can be compensated for by precisely tracking the motion of the sensor with its inbuilt IMU.



(a) (b)

Figure 6: Example of monocular 3D reconstruction with a handheld DAVIS. (a) Reference frame (b) Semi-dense depth map pseudo-colored from red (close) to blue (far) [46]

.

2.5.4 Visual-Inertial Odometry

The task of visual-inertial odometry (VIO) is to recover the 6 degrees-of-freedom (DOF) pose of a camera from a set of visual measurements and inertial measurements from an IMU rigidly attached to the camera. While using conventional frame-based cameras to perform VIO is relatively mature, high-speed motion or high-dynamic range still presents a challenge due to the fundamental limitation of frame-based cameras. A camera with a low frame rate prevents the visibility of motion between frames, while a camera with a high frame rate draws more power and requires excessive processing, regardless of whether much of the captured data is duplicative or not. Frame-based cameras can also suffer from motion blur when exposure times are too long relative to the movement of the camera or scene. Lastly, with limited dynamic range, the ability to adequately capture very bright and very dim features in the same frame is severely limited. This has led to the recent motivation to develop event-based VIO (EVIO) [42].

Stemming from Zhu's implementation of EVIO in [48], AFIT's Autonomy and Navigation Technology (ANT) center has attempted to estimate a vehicle's 6-DOF

motion and pose by generating frames from event-based data before applying an extended Kalman Filter (EKF) to estimate the state of the system [3]. Within this research, the ANT center has also collected event-based data from an airborne fixed-wing unmanned aerial vehicle (UAV). To date this is the only time AFIT has collected event-based data during a flight test where the intent of future work is to interpret and utilize this data to its full extent.

2.5.5 Feature Detection and Tracking

Feature detection and tracking is a fundamental component of most machine vision applications and is a task well suited to event-based sensors. Because moving objects generate spatio-temporal energy that generates asynchronous events, trackers can take advantage of the low latency, high dynamic range and low power requirements of these sensors. The generated sequence of events can then be used as input to newly developed algorithms designed specifically for single and/or multiple object tracking applications.

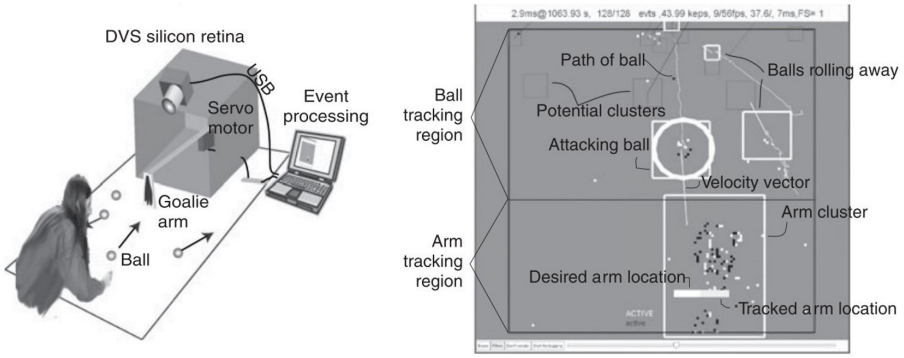
Dependent on the scenario, tracking algorithms typically assume either a static sensor or a moving sensor. Since event-based sensors respond to the apparent motion of edge patterns in a scene, for a static sensor events are mainly caused by moving objects, whereas for a moving sensor, events are caused by moving objects of interest as well as the motion of the background relative to the sensor. Therefore, when considering algorithm design, it becomes important to understand what to detect, how to represent it using events and how to associate detections into a track. For example, objects of interest are usually based on shapes (geometrically defined) or appearance (edge patterns), and so a tracking algorithm must be tailored to either of these assumptions dependent on its application. The following sections briefly describe select examples of object tracking approaches currently under development.

2.5.5.1 Cluster Tracker

Early event-based object trackers focused on demonstrating the low-latency and low-processing requirements of event sensors by assuming a static sensor and tracking objects as clusters of events. One example of this is the cluster tracker build into the Java Address Event Representation (jAER) project which is designed to track the motion of multiple moving compact objects based on a range of user inputs [49–51]. Examples of tracked targets include balls on a table, cars on a highway or trace particles in a 2D fluid. The tracker is based on modelling the object as a spatially connected and compact source of events. As the objects move, they generate events which are used to generate clusters. A cluster is described by a number of statistics including position, velocity, radius, aspect ratio, angle and event rate. Clusters typically form when events are registered in areas with no previous events and they are dropped when insufficient events are received to support them.

There are several advantages of cluster tracking compared with conventional frame-based trackers. First, because there are no frames, the events update clusters continuously in an asynchronous fashion. Second, only pixels that generate events need to be processed and the cost of this processing is dominated by the search for the nearest existing cluster. Lastly, the only memory required is to track cluster locations and analyze minor statistics, typically about 100 bytes per cluster.

A good example that highlights the effectiveness of this tracking approach when used in the right scenario is the robotic goalie [49]. Figure 7 shows the setup (a) and a screen shot (b) taken during operation. Simply, a person would shoot balls at a goal while the robotic goalie blocks the most threatening ball, determined to be the ball most likely to cross the line first. With a goalie hand size of 1.5 times the ball diameter, the robotic goalie was measured to block 95% of shots, clearly demonstrating that a fast, cheap tracking system can be used for fast response applications.



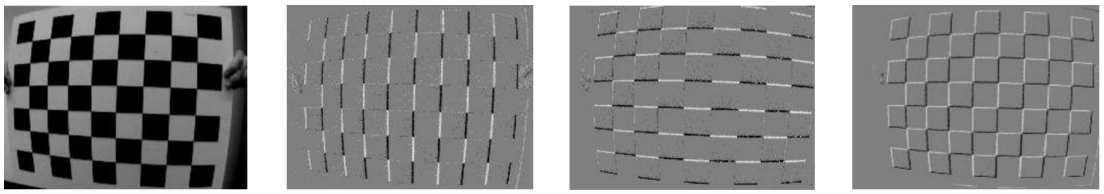
(a) Robotic goalie setup (b) Screen shot

Figure 7: Example of a robotic goalie using cluster tracking (a) shows the setup while (b) shows a screen shot of the system tracking multiple balls [52].

2.5.5.2 Combining Events and Frames

One significant challenge of using event-based sensors for feature detection and tracking is overcoming the change of scene appearance under different viewing scenarios. Figure 8 captures this effect when viewing a checkerboard pattern under different directions of motion. When moving diagonally with respect to the vertical and horizontal edges of a checkerboard, the checkerboard pattern is clearly visible. However, when moving only vertically or horizontally, the pattern appears vastly different. While event-based sensors offer significant advantages (i.e. high-dynamic range, no motion blur, microsecond latency), this example highlights that event correspondence between different directions of motion is not a simple task due simply to the changing appearance of edge patterns.

To combat this effect, if the absolute intensity of the imaged scene is available, the problem can be simplified as intensity frames do not depend on motion. By extracting features from intensity frames using well established methods, and subsequently tracking them asynchronously with optical flow estimates using events, the advantageous features of both sensor types can be exploited. Essentially, the frames provide a photometric representation while the events provide low latency updates.



(a) Single frame (b) Horizontal motion (c) Vertical motion (d) Diagonal motion

Figure 8: Challenge of scene appearance under different motion directions. (a) Shows a reference frame of a checkerboard while (b)-(d) show the output of an event-based sensor when viewing the same checkerboard under different directions of motion [53].

This method has been shown to produce feature tracks that are both more accurate (subpixel accuracy) and longer than state of the art methods across a wide variety of scenes [53].

2.5.5.3 Corner Detection and Tracking

When viewing a textured scene through an event-based sensor with rapid motion, millions of events can be generated per second. Therefore, current event-based algorithms either require massive parallel computation, or must depart from the event-based processing paradigm. With this in mind, some works draw inspiration from frame-based pre-processing techniques that aim to reduce an image to a set of keypoints such as corners. This has the advantage of significantly reducing the number of events that need processing, in addition to negating the impact of aperture problems in the imaging system. The method in [54] computes corners as the intersection of two moving edges, which are obtained by fitting planes in the space-time stream of events. More recently however, extensions of popular frame-based keypoint detectors, such as Harris [55] and FAST [56], have been developed for event sensors to reduce an image to a set of features. These corners are then typically fed into a high-level algorithm for object tracking [57, 58] where it has been shown to be capable of processing millions of events per second on a single core and reducing the event rate by

a factor of 10 to 20.

2.5.5.4 Space Situational Awareness

Determining the location of satellites is a challenging task typically performed from ground stations using techniques such as radar, signal Doppler and laser reflectors. Beyond this however remains the challenge of determining the orientation and rotational motion of such satellites. While arguably the most common technique uses telescopic frame-based imagery, significant limitations still arise largely due to the need for fixed exposure times per captured image. Satellites in Low Earth Orbit (LEO) have very high relative velocity and as such are very difficult to lock onto and track, making their imagery particularly vulnerable to motion blur. Furthermore, it has proven very difficult to image satellites during the day using this technique as exposure times must drop to avoid saturation while the noise floor is also significantly increased by photon noise from the bright sky, making it very difficult to discern whether a target is present or not.

By taking advantage of the high dynamic range and temporal resolution of event-based sensors, their application to improving Space Situational Awareness (SSA) by tracking satellites from LEO to Geostationary Earth Orbit (GEO) during the day or night has been demonstrated [2]. Figure 9 captures the output of an event-based sensor tracking both LEO and GEO satellites, highlighting the success of preliminary experiments and representing their potential to dramatically increase the capability of terrestrial optical sensors. While the results of this demonstration are shown using an unspecialized prototype and uncooled equipment, it is expected that further efforts to improve the setup will only lead to better results.

Taking this demonstration a step further, one logical progression of this technology is to incorporate the event-based sensor onto a space-based SSA platform. While not

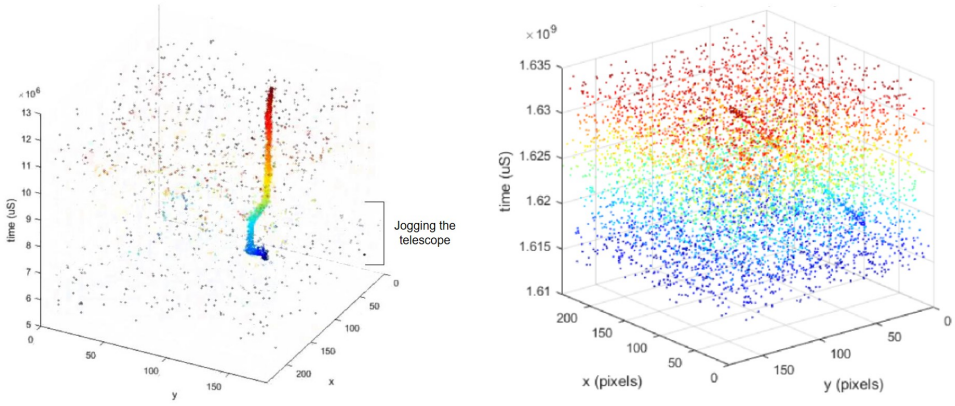


Figure 9: Example results captured from imaging LEO (left) and GEO (right) satellites using a DAVIS sensor. The LEO track deviates because the sensor was jogged to confirm the satellite was in the field of view. The colormap in both plots is linked to event timestamps where blue events have occurred first and red events last [2].

yet tested, this progression would take advantage of the low size, weight and power (SWaP) of the sensors in addition to their intrinsically low data rate requirements.

2.6 Possible Defense Applications

With low power and data requirements, microsecond temporal resolution and high dynamic range, event-based sensors present a significant opportunity to enhance the capability of several existing military applications and possibly form a essential component in many future systems. Here, a few examples have been selected for brief discussion.

For a common surveillance scenario such as monitoring the entrance to a secure building, a traditional frame-based sensor produces significant amounts of data, most of which is totally redundant. Regardless of whether a change in the scene has occurred or not, all frames must be processed continuously and in the same manner leading to a significant overhead in terms of processing requirements. Because an event-based sensor responds only to change, it offers an excellent low power and low data alternative with an equal, if not better, ability to alert significant activity at the

building entrance.

The high speed and high dynamic range of these sensors could also support the ability to target a fast moving object in challenging conditions. Interference can often occur in unfavourable lighting conditions however with the use of an event-based sensor this interference can be minimized to ensure the object remains tracked. Additionally, should the successful development of an infrared event-based sensor be achieved the scope of this scenario could extend to infrared bands.

With a higher resolution an event-based sensor could also make a significant improvement in the space domain as satellite staring sensors. Given event-based sensors only respond to change, most of the time their power consumption and data rate would be minimal offering a significant advantage to satellite SWaP considerations. Most likely applicable to LEO satellites, their relative motion to the surface of the earth would pose an issue however with further research, optical flow techniques could be used to filter constant motion and draw out instantaneous changes.

The scope of event-based sensors reaches far beyond the few examples mentioned here. Nevertheless, with these examples the military relevance of event-based sensors and their range of possible applications is quite apparent. With the disruptive nature of this new technology, the impact to Defense capability is as yet unknown, however expected to be significant in coming years and decades.

III. DAVIS Characterization

3.1 Preamble

This chapter introduces characteristics of the event-based Dynamic and Active Vision Image Sensor (DAVIS) by detailing its electronic circuitry and how it is used to emulate biological vision. As these sensors are uniquely different from frame-based sensors, newly defined performance characteristics are also introduced and explained. Through experimentation, the frequency response of these sensors is then explored with results analyzed and discussed with reference to the potential for these sensors to form the basis of a detection and tracking capability.

3.2 CMOS Sensor

All visual sensors, whether they be biological or man-made, begin with a photosensitive element that converts incoming photons into a well defined electrical signal. When considering conventional imaging systems, the Focal Plane Array (FPA) of most imagers can typically be classified as either a charge-coupled device (CCD) or a complementary metal-oxide semiconductor (CMOS).

Originally developed as a memory technology, CCDs allow for the storage and movement of charge in a semiconductor using electric fields. As they are photosensitive, they can also be used to accumulate charge during optical exposure which can then be moved to the device periphery and converted into an electrical signal. While originally dominant in the market of digital cameras as they were optimized for charge generation and transportation, fabrication methods made integration of this sensor into additional signal processing circuits particularly difficult.

CMOS sensors on the other hand use a fabrication process derived directly from that used for regular computer chips and therefore allow the integration of other

components such as fast, parallel analog to digital converters and synchronous logic. A feature particularly useful to the DAVIS pixel. Because charges cannot be moved easily through such chips, pixel signals must be amplified before they are read out. Initially transistors required for this amplification were subject to mismatch leading to CCDs being the preferred technology. With advancement in CMOS fabrication however, transistors have become smaller with less mismatch that can be further mitigated in software with techniques such as correlated double sampling (CDS). For this reason, CMOS technology now appears in the majority of today's imagers.

3.2.1 Active Pixel Sensor

In contrast to CCD, CMOS technology does not allow charge movement from each pixel to the sensor periphery without adding significant noise. To combat this the active pixel sensor (APS) was developed to buffer the signal within the pixel before being read out. Shown schematically in Figure 10, the reset transistor (MN1) charges the parasitic capacitance of the photodiode at the beginning of the readout cycle. The photocurrent generated by the photodiode then discharges this capacitance during optical exposure where the remaining charge is read out at the end of the readout cycle. This amplified read out is performed by the source follower circuit formed by MN2 which generates a current from the integrated voltage and MN4 which converts this current into a voltage again at 'readout' in Figure 10. MN3 is required to ensure that only one row can 'write' its current to the readout line shared by all pixels in a column.

3.2.2 Noise in CMOS sensors

Like any electronic device, CMOS image sensors are subject to various forms of noise. The following section explains some key noise contributors applicable to CMOS

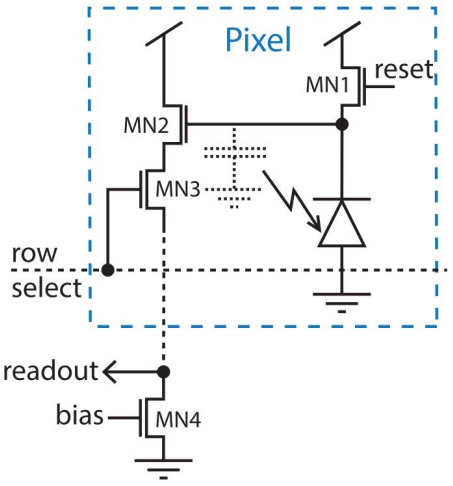


Figure 10: Three transistor active pixel sensor schematic [59]

sensors ranging from those that stem from the manufacturing process to more physics based characteristics.

3.2.2.1 Fixed Pattern Noise

One of the key problems with an APS is mismatch among pixel transistors which can lead to non-uniformities in photon response. Practically, this implies that the same amount of incident light integrated over a fixed time window can cause separate pixels to respond differently by reading out different voltages. These variations have both a temporal and spatial component, and while the temporal component can largely be removed by averaging a number of exposures, the spatial component, typically referred to as fixed pattern noise (FPN), often remains.

FPN is typically caused by two forms of non-uniformity. The first is dark signal non-uniformity which describes the offset component and is understood as the voltage offset from the average pixel response in the absence of external illumination. Before the introduction of correlated double sampling (explained further in Section 3.2.3), dark signal non-uniformity was the dominant noise source that prevented the success of CMOS technology. Secondly, photo response non-uniformity describes the gain

component and is caused by mismatch in readout transistors and integration capacitance which leads to different conversion gains from photons to read-out voltage. Since the readout of a CMOS array uses column shared transistors (MN4 of Figure 10), mismatch in these transistors leads to stripe patterns appearing in captured frames.

3.2.2.2 Shot Noise

Shot noise is a consequence of the particle nature of light and the fact that the number of electrons generated by incident light follows a Poisson distribution. The uncertainty for a given electron count following such a distribution scales with the square-root of average intensity as shown in (1).

$$(\Delta I)^2 \equiv \langle (I - \langle I \rangle)^2 \rangle \propto I \tag{1}$$

3.2.2.3 Thermal Noise

Thermal noise is caused by the random motion of electrons within an electrical circuit due to temperature. The effect is a fluctuating current that when sampled on a capacitance C, the according noise voltage is proportional to $\sqrt{\frac{kT}{C}}$ where k is Boltzmann's constant and T is temperature.

3.2.2.4 $1/f$ Noise

$1/f$ noise is caused by charges trapped in the readout transistor that lead to current fluctuations and a power density profile that drops off proportional to $1/f$ down to the point where fluctuations are dominated by thermal noise. f here refers to the response frequency of the transistor.

3.2.3 Correlated Double Sampling

One effective method to mitigate FPN as well as thermal noise is to use CDS. Using this method, when a pixel is reset after an exposure, the reset voltage is sampled and stored onto a capacitor. After exposure, the signal voltage is then subtracted from the stored reset voltage, significantly reducing the effect of dark signal non-uniformity and in effect removing the appearance of stripes in a captured image. Additionally, since the reset and signal voltage are stored on the same capacitor, this circuit also has the added benefit of suppressing thermal noise.

3.3 DVS Pixel Design and Operation

The fundamental properties of a Dynamic Vision Sensor (DVS) pixel are a direct consequence of abandoning common frame-based principles and modeling three key properties of biological vision: the event-based output, the representation of relative luminance change, and the rectification of positive and negative signals into separate output channels (ON/OFF). Figure 11 (a) and (b) capture the highly simplified mapping from a three-layer retina model (described earlier in Section 2.1) to a corresponding pixel circuit. Typical signal waveforms are then shown in (c) while (d) captures the response of an array of pixels to a natural scene.

The first practical event-based DVS pixel was developed by Lichtsteiner et al in 2008 [13]. The objective of this design was to achieve low mismatch, wide dynamic range (DR) and low latency in a reasonably small pixel area. The challenge was met with a fast logarithmic photoreceptor circuit, a high precision differencing circuit and two-transistor comparators. Figure 12 captures the architecture of how the three DVS components are connected.

Within the photoreceptor, light incident on the photodiode is converted into a current I_{ph} which is supplied by the transistor above. By feeding back the photore-

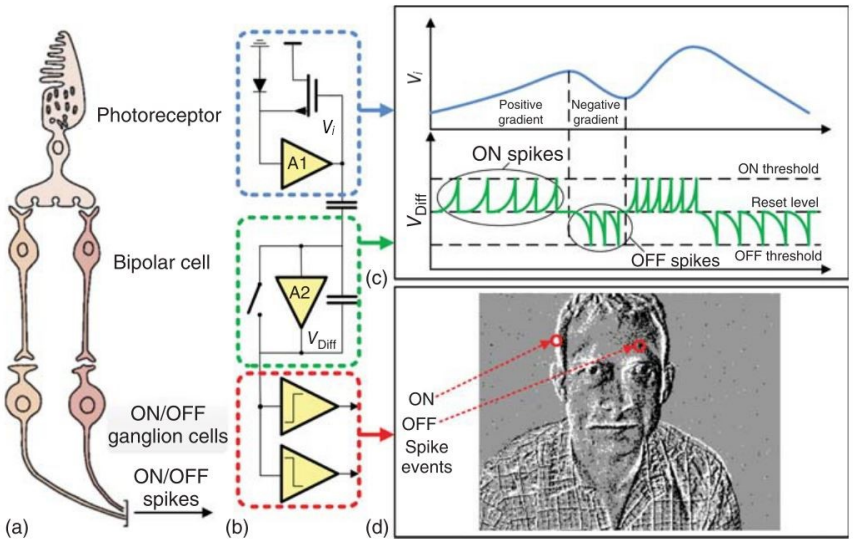


Figure 11: (a) Three-layer model of a human retina (b) Corresponding DVS pixel circuitry (c) Typical signal waveforms of the pixel circuit (d) The response of an array of DVS pixels to a natural scene. [60]

ceptor voltage to the gate, the photoreceptor voltage is clamped and the gate voltage becomes a logarithmic function of photocurrent with gain $-A$ [62]. This makes the contrast perception a logarithmic function of the stimulus intensity allowing the pixel to span a wide dynamic range of photocurrents without becoming saturated.

The output voltage of the photoreceptor, v_i, is then fed to a differencing circuit that computes and amplifies the temporal derivative, v_{diff}. Here the logarithmic voltage v_i gets buffered using a source follower circuit to charge the capacitor C_1. The other side of this capacitor, a floating node, is attached so that voltage deviations since the last reset get inversely amplified through capacitor C_2 by the well-matched ratio C_1/C_2. Because capacitors can be better matched than transistors, using capacitors for the differencing circuit allows mismatch, in particular gain mismatch, to be reduced.

The output of the differencing circuit v_{diff} is then fed into two current capacitors which compare it against a lower ON and upper OFF threshold (the OFF signal is inverted because the OFF comparator generates an active-low signal). If the transient

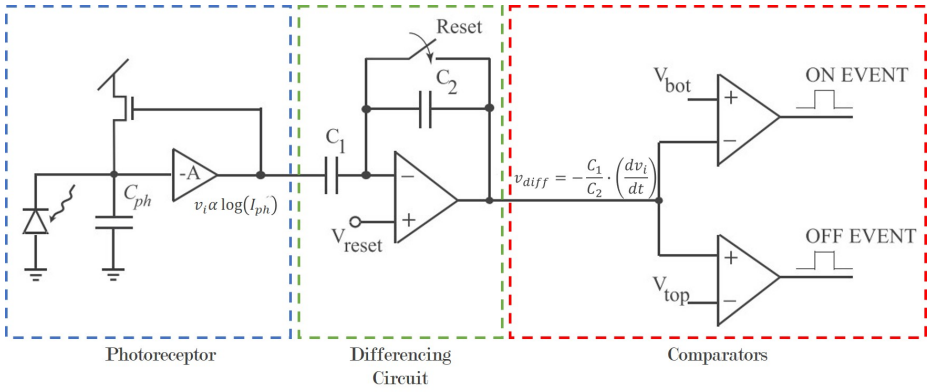


Figure 12: DVS pixel architecture [13, 61]

variation of v_{diff} exceeds one of the thresholds, an event is generated and sent off-chip via a shared data bus. After an event is generated, the pixel gets reset by shorting the differencing circuit feedback, thus balancing the amplifier at its reset level.

In terms of the generation of events, Figure 13 provides a visualization of pixel operation by displaying notional output signals v_i and v_{diff}, in response to an arbitrary time dependent input irradiance pattern. As v_i changes in response to incident light, v_{diff} moves away from its previous reset level until exceeding either the ON or OFF threshold value. When the signal reaches either level, a signal is sent off-chip to generate an event in Address Event Representation (AER) format, before resetting the pixel.

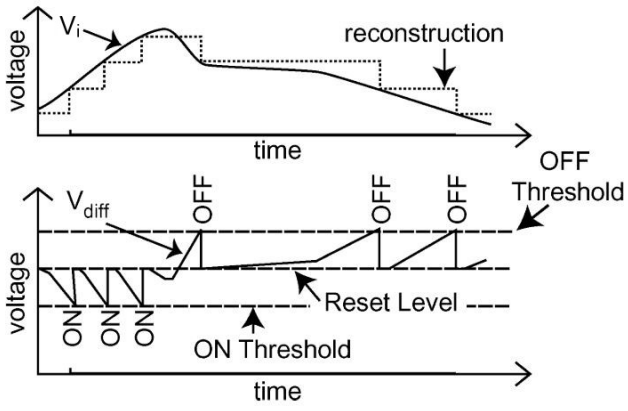


Figure 13: Schematic of DVS pixel converting incident light into events [13].

3.3.1 Address Event Representation

When an event is generated, the chip output from the DVS pixel is a 15-bit digital address containing the 7-bit x and y address of each pixel, as well as the ON/OFF polarity p of change. Each event is then time-stamped and buffered using USB First-In, First-Out (FIFO) technology and sent to a host PC for processing. Universally recognized as the standard format for event-based sensing, data is recorded in the format known as AER where pixel location (x, y), polarity p, and time t are represented in the form of (2).

$$\mathbf{e} = [x, y, p, t] \tag{2}$$

3.4 Dynamic and Active Vision Image Sensor

Realizing that the DVS circuit does not consume photocurrent and that it could be reused for intensity measurements, Berner et al. [63] and Brandli et al. [22] worked to combine the conventional APS pixel with the DVS pixel to create the DAVIS. This essentially combined the advantages of the DVS with the possibilities of conventional frame based imagers.

Figure 14 shows how both the APS and DVS circuits are connected. In order to avoid the APS readout affecting the readout of the DVS, the two circuits are separated by a cascode transistor MN5. Further to this, one of the crucial aspects in the layout of the DAVIS pixel is to avoid coupling of digital signals into the analog front end of the pixel. In previous models of the DVS pixel the acknowledge line ran over the photodiode and thereby coupled into the photoreceptor. By moving all digital lines away from the photodiode, this issue was largely avoided.

Another effect observed in the pixel is a feedback coupling from the output of the differencing circuit (v_{diff} of Figure 12) to the photodiode which leads to oscillations that can cause events. Similarly, there is some capacitive coupling between the trans-

fer gate signal TX and the V_{pr} node of Figure 14 such that the frame-based readout of the APS circuit can trigger events, a scenario seen regularly within the results of this thesis.

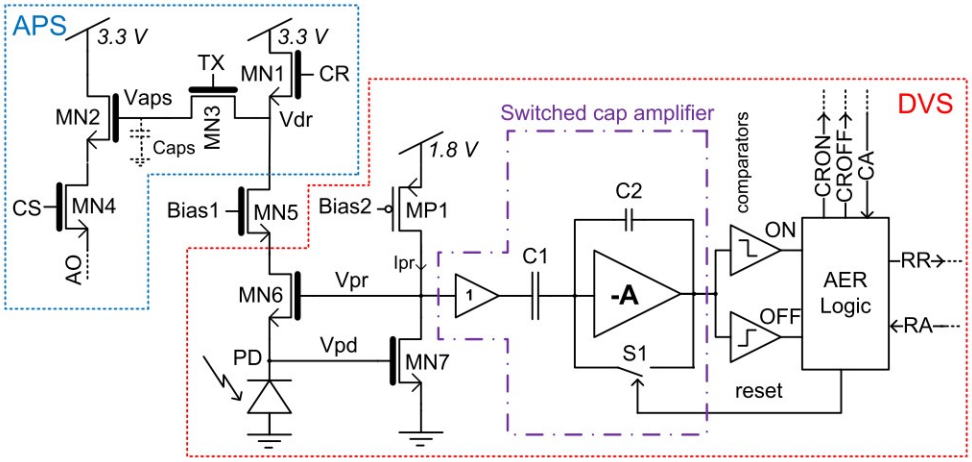


Figure 14: Simplified schematic of the DAVIS pixel showing the connection of separate APS and DVS circuits [22].

3.4.1 Rolling vs. Global Shutter

To simplify pixel design, improve readout speed, dampen thermal noise and allow for CDS, many CMOS imagers used a rolling shutter. Instead of exposing all pixels at the same time, with a rolling shutter pixels are exposed and read out row by row. This non-uniform exposure leads to confusing artifacts as highlighted in Figure 15 (a) and (c). Here exposure of the scene is spread in time such that the exposure plane is tilted where in (a), a continuous stream of DVS events is cut at multiple times making interpretation of the generated events difficult. Similarly in (c), the tilted exposure plane has lead to confusing image artifacts.

In contrast, the global shutter exposes all pixels at once and reads the entire array before resetting. As shown in Figure 15 (b) and (d), the exposure plane is not tilted thereby reducing motion artifacts. The drawback however, is that a global shutter

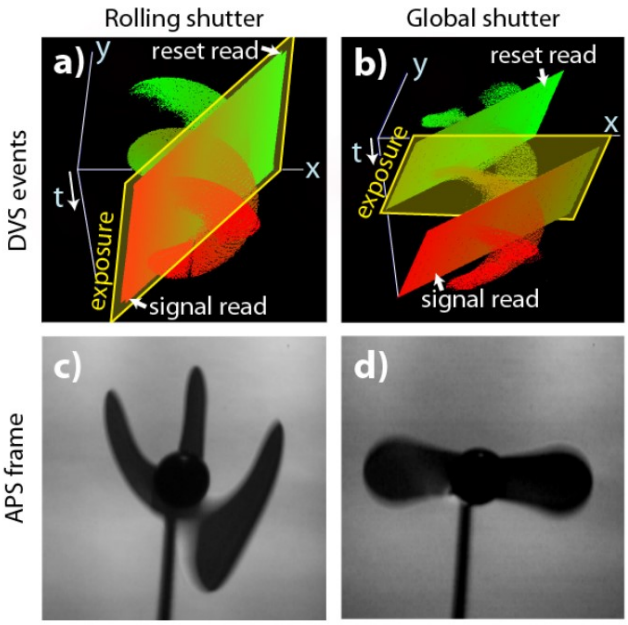


Figure 15: Rolling vs. Global Shutter: A 50 Hz rotating fan captured with a DAVIS using both a rolling and global shutter [22].

does not correct for thermal noise while a rolling shutter does, despite this, the global shutter will be used for all frame-based data captures throughout this thesis.

3.5 Characterization

Stemming from the nature of the logarithmic conversion in the photoreceptor, each DVS pixel is sensitive to temporal contrast C, defined in (3), where t is time and I is the photocurrent generated from incident light [13].

$$C = \frac{1}{I(t)} \frac{dI(t)}{dt} = \frac{d(\ln(I(t)))}{dt} \tag{3}$$

Here it can be seen that temporal contrast is directly proportional to the temporal change in log-light intensity.

In this process, events are timestamped with microsecond resolution and transmitted with sub-millisecond latency, meaning these sensors react very quickly to changes

in visual stimuli. Inherent to the design of a DVS, their output is heavily dependant on the amount of motion or brightness change in the scene. Since the 'change detection' process at each pixel adapts to the rate of change in the log-intensity of light that it monitors, the faster the motion, the more events that are generated per second. The output of an event-based camera is therefore a variable data-rate sequence of digital events, each representing a change in log-intensity of light compared with a predefined magnitude at each pixel at a particular time [13].

Each pixel of an event-based sensor responds independently to incident light as per (3). More specifically, in a noise free scenario an event $\mathbf{e}$ is triggered at pixel (x, y) at time t as soon as the brightness increment since the last event at that pixel reaches a predefined threshold θ defined in (4).

$$\theta = \Delta \ln(I) \tag{4}$$

Ideally θ is determined by the pixel bias currents which are digitally programmed into the sensor itself. In practice however, positive and negative events may be triggered according to different thresholds, θ_{ON} or θ_{OFF}, largely due to pixel-to-pixel mismatch in addition to noise. A primary example is in low-light conditions where small photocurrent values I are operating in the steepest portion of the log-curve in (4), therefore any small change in input irradiance can result in an event, making the sensor particularly vulnerable to noise. Fundamentally, each pixel must react to a small change in photocurrent in spite of any noise present in the signal. In general, it is this noise limitation that governs the relationship between temporal threshold and the speed of the DVS under various illumination and detection reliability scenarios [64].

3.5.1 Probabilistic Event Generation

Further to the concept of event generation is its probabilistic nature. In an ideal case, event generation occurs when either θ_{ON} or θ_{OFF} is breached. A more realistic model however needs to consider the effect of sensor noise and transistor mismatch known analogously as FPN. Due to the stochastic nature of local illumination and sensor operating parameters, event triggering conditions are essentially represented as a probabilistic function.

For an event-based sensor the equivalent measure of FPN is the standard deviation σ of the pixel-to-pixel variation in the event threshold θ. In an ideal case the response of each pixel would be a perfect step function whose binary output would change from zero to one or one to zero whenever the input photocurrent crossed a defined threshold. In practice however, the response is not a step function, and the behaviour of each pixel is highly individual. Figure 16 shows a notional example of the probability of registering an event at a single pixel with a nominal threshold value of 0.5 mV at the comparator input. The x-axis corresponds to an input voltage to the pixel comparator circuit while the y-axis represents the probability that the pixel will generate an event. This figure essentially highlights that for any one pixel, there is a

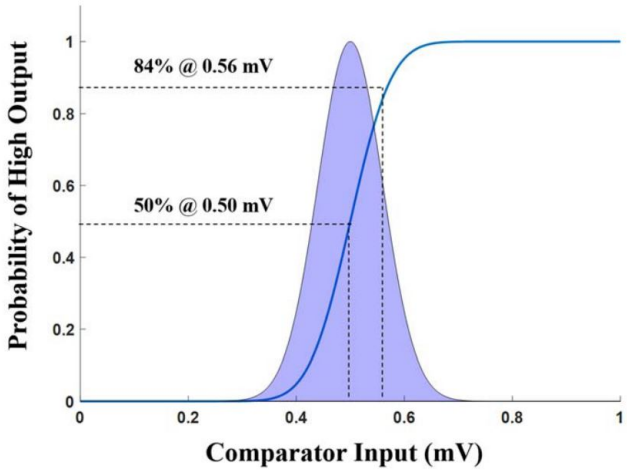


Figure 16: Gaussian noise model [65]

probabilistic dependence between the generation of events, and the pixel illumination. With a PDF such as this, noise statistics can be extracted and therefore utilized to estimate sensor behavior [13, 65].

3.6 Standard Event-Based Performance Metrics

Due to the unique operational construct of event-based sensors, performance metrics normally attributed to standard frame-based sensors simply do not apply. Since the commercialization of event-based sensors, it has therefore become important to define a range of performance metrics that aid in standardizing the production and performance of such sensors. Initially developed in 2008 at [13], Lichtsteiner introduced several key performance metrics examples of which include response uniformity, minimum threshold, dynamic range and pixel bandwidth. These metrics and more are introduced with their accompanying definition in the following sections.

3.6.1 Minimum Threshold and Threshold Mismatch

Defined in (3), contrast threshold refers to the percentage change in illumination required to generate an event. Assuming appropriate bias settings, minimum threshold quantifies this concept as the minimum average contrast threshold across an active sensor and is represented as a percentage. Extending this further, threshold mismatch (often referred to as response uniformity) refers to the standard deviation σ_θ of the event threshold from the mean, again represented as a percentage. Analogous to FPN, this figure characterizes the uniformity of the sensor where typically due to small manufacturing anomalies, each pixel does not have identical performance characteristics. For the case of the DAVIS 240C, minimum event threshold is quoted to be 11% with a threshold mismatch of 3.5% [66].

3.6.2 Latency and Jitter

Defined as the time delay between a change in illumination (e.g. blinking light emitting diode (LED)) and the generation of a corresponding event, latency is an important metric especially for near-real time applications such as autonomous navigation. Typically, latency depends firstly on the ability of the pixel to respond to changes in illuminations, and secondly on the peripheral AER circuitry to transmit an event along the data bus. In addition, when considering repeatable stimulation, latency jitter refers to the variance in the time of event generation within a pixel and the variance in the transmission time of the event.

Latency can be measured by illuminating a small region of the sensor with a low contrast blinking LED such that the difference between high and low is only enough to generate approximately a single event. Additionally, by varying the DC illumination, the change in latency in response to total scene brightness can also be measured. As shown in Figure 17 latencies are typically in the order of milliseconds and follow either inverse linear or inverse square characteristics with scene brightness depending on what photoreceptor biasing has been set.

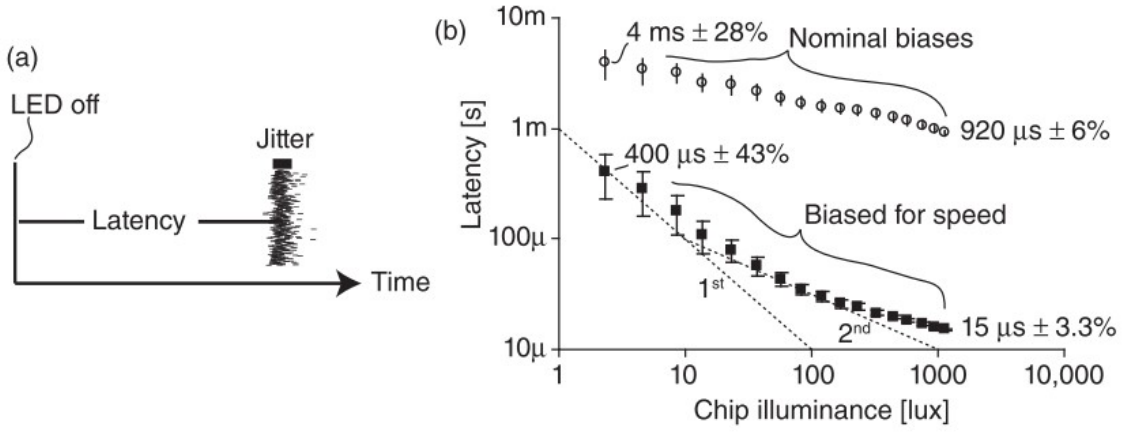


Figure 17: Example of DVS latency and latency jitter. (a) DVS pixel response to repeated OFF events. (b) Latency response for two bias settings as a function of sensor illumination [13].

As can be seen, the latency decreases with increasing scene brightness. The reason for this is that the capacitors in the front end comparator circuit of the DVS pixel can deplete faster when more photocurrent is available [59].

3.6.3 Dynamic Range

Dynamic range is defined as the ratio of maximum to minimum illumination within a scene whereby accurate events can still be generated [13]. For a DVS, given that pixels respond locally to relative changes in intensity, the dynamic range of the device is comparatively much higher than typical frame-based sensors with their fixed integration time. Using a lens with low f# (i.e. f/1.2), it has been shown that events can be reliably reproduced under moon light with illumination levels as low as 0.1 lux. At such low Signal-to-Noise Ratio (SNR), this is only possible because the low threshold mismatch allows setting a low threshold. At the other end of the scale, the sensor will also operate under direct sunlight with illumination levels as high as 100 klux. As this range of illumination can appear in a single scene and still be resolved, the dynamic range of the DVS therefore amounts to as much as 100 decades or 120 dB, significantly higher than a typical frame-based sensor noted to be in the order of 60 dB. Fundamentally, the reason for the high dynamic range of the DVS pixel arises from logarithmic compression in the front end photoreceptor circuit and local event-based quantization.

3.6.4 Data Bus Limitations

As previously discussed, when an event is generated a sequence of bits are sent off chip via the DVS data bus in AER format. As expected, inherent to this process is a bounding upper limit to the number of events that can be sent off chip with the correct time stamp as a function of time. This is known as sensor bandwidth and is

measured in units of events-per-second (EPS).

The DAVIS 240C has a specified bandwidth of 12 MEPS [66]. While in most scenarios this sensor will operate well within bounds, for a highly dynamic scene this bandwidth limitation may be tested. The consequence of encroaching on sensor bandwidth is that the data bus can become saturated, causing events generated after the specified limit to receive a delayed, and therefore incorrect time stamp, creating data integrity issues. Fortunately, data bus technology has advanced to the point where 12 MEPS is achievable, representing a generous limitation that is not likely to be approached during normal operation.

3.6.5 Pixel Bandwidth and Refractory Period

While an event-based sensor is capable of generating spatially resolved events with a resolution of 1 μs, this does not imply that changes in illumination incident on a single pixel can be resolved at the same rate. In fact, the response frequency of individual pixels is known to be far less. Known as pixel bandwidth, the term defines the maximum response rate of a single pixel. Conversely, refractory period is the inverse of pixel bandwidth and is defined as the minimum time detectable between consecutive events and is usually given for a single pixel or as an average across an array.

Fundamentally, pixel bandwidth results from the inherent frequency response limitations of the three stage DVS circuitry. Although DVS pixels are fast, like any physical transducer, they have finite bandwidth. If the incoming light intensity varies too quickly, the front end photoreceptor circuits filter out variations. The rise and fall time that is analogous to exposure time in standard image sensors, is therefore directly related to pixel bandwidth and refractory period.

3.6.5.1 Experiment Measurement of Pixel Bandwidth

As an exercise to become familiar with operating an event-based sensor, a simple experiment was conducted to measure the pixel bandwidth of the DAVIS 240C with a similar setup to that used by Lichtsteiner et al. in [13].

Shown schematically in Figure 18 and physically in Figure 19, a function generator was used to drive a green LED positioned inside an integrating sphere. Light exiting the integrating sphere then passed through a neutral density filter and variable aperture stop before entering the DAVIS 240C where data could be captured and recorded with a laptop running Java Address Event Representation (jAER) software. The function generator was used to drive the LED so that its brightness could be tightly controlled dependent on the variable voltage applied. Due to availability a green LED was used for this experiment, however the color is arbitrary as it is the ability to generate events that is important.

Placing the LED inside an integrating sphere ensured a uniform, homogeneous light source was presented to the DAVIS 240C. In addition to the function generator, by inserting a neutral density filter the baseline illumination level, or brightness, of the source could be further controlled. In order to prevent saturating the readout data bus with the excessive generation of events, the aperture stop was used to limit the spot size from the LED incident on the FPA of the DAVIS 240C. Of the DAVIS's 240×180 pixels, a circular area with diameter approximately 25 pixels was illuminated

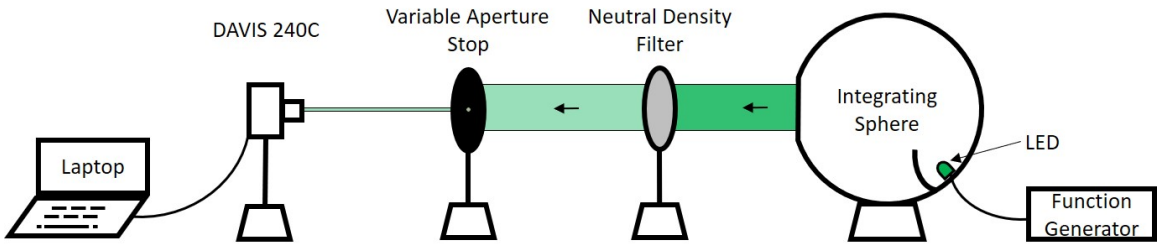


Figure 18: Schematic setup of characterization experiment.

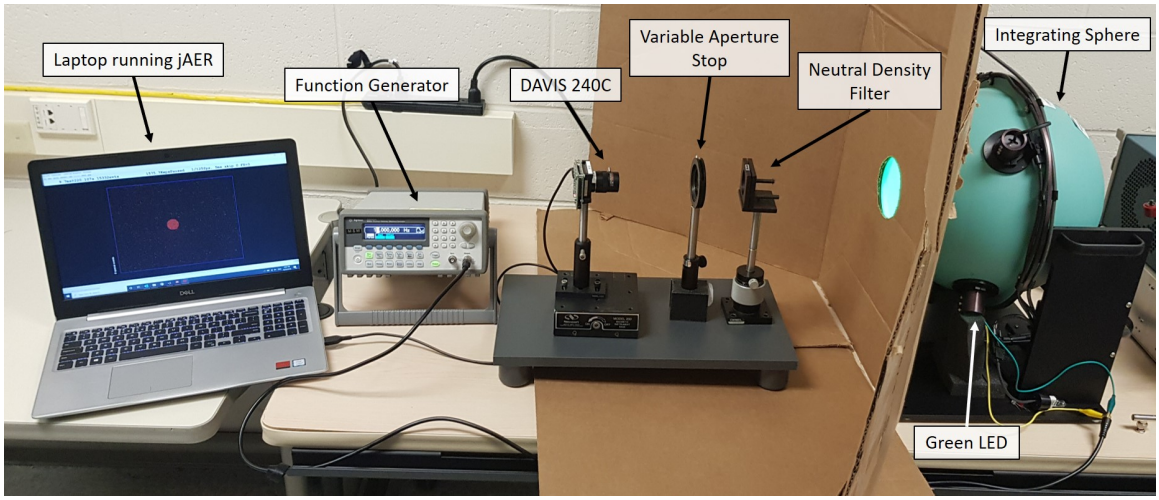


Figure 19: Physical setup of characterization experiment.

where the central 13×13 = 169 pixels were used for analysis. Operating in its linear region, the LED was driven with a biased sinusoidal voltage such that it was always emitting. Data was then captured for a number of stimulus frequencies ranging from 1 Hz to 3 kHz.

Results. In a dark room with the DAVIS 240C biased hard to maximize pixel bandwidth, the LED was driven by a sinusoidal voltage between 1.4 V and 5 V. By varying the stimulus frequency and transparency of the neutral density filter, the pixel bandwidth could be determined. As the response of each pixel to the same stimulus is not uniform, a probabilistic approach was adopted to determine pixel bandwidth in units of events per stimulus cycle. Here 'events' are calculated as the average number of events that occurred within the 13 × 13 pixel region of interest while 'stimulus cycle' is simply the total number of stimulus cycles presented to the sensor during a recording (i.e. total time of the recording divided by the known LED stimulus frequency). Using this approach and combining both ON and OFF events, the results for various brightness levels are shown in Figure 20.

The results highlight the similarity of the sensor to a first-order low-pass filter,

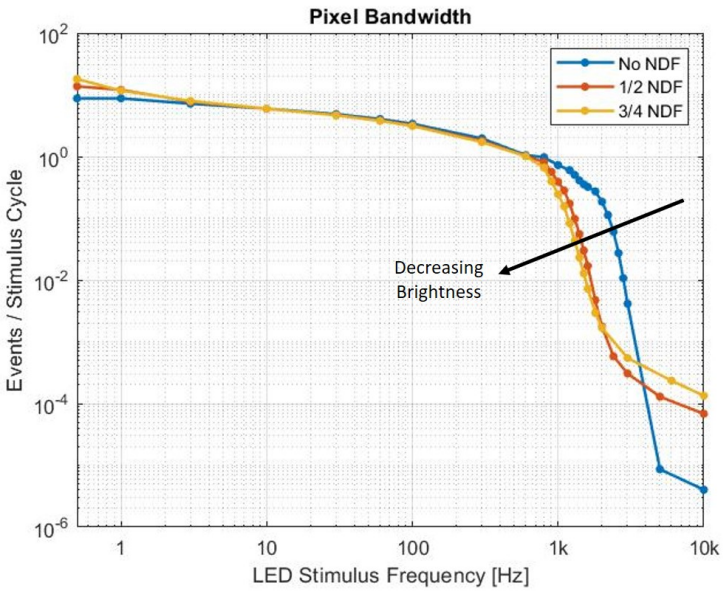


Figure 20: Measurement of pixel bandwidth.

demonstrating a cut-off at approximately 1 kHz. This drop off coincides with when the event rate drops to approximately one event per stimulus cycle. This is because once the comparator input signal drops below the event threshold, the normalized event rate rapidly falls to zero. It can also be seen that as the DC brightness decreases so too does the response frequency. The reason for this is that when incident photon flux is low, a smaller electric current is generated. Given that any circuit has a certain capacitive load, a smaller current implies a slower circuit response, resulting in a lower response frequency.

While typical published values are in the order of 3 kHz [13], the difference is likely due to variances in the experimental setup, in particular the brightness of the stimulating LED. One key point to note is that this measurement only determines the pixel bandwidth for a very specific experimental setup. Because the experiment requires selecting a stimulus amplitude and waveform, as well as specific DAVIS 240C bias settings, if any of these parameters were changed, the resulting pixel bandwidth could be significantly different. Nevertheless, one example of the sensor's response to

a rapidly oscillating stimulus has been captured and a better overall understanding gained.

Alternate approach - Fourier Analysis. An alternate method to more clearly show the frequency response of event based sensors was also developed using a Fourier analysis. In this approach a discrete time vector was created with microsecond resolution and populated with zeros. From the captured event stream, the time stamp of each event within the specified region of interest was mapped to its corresponding index location within the discrete time vector and populated with the polarity of that event. The mapping of events was done accumulatively where for example, if two ON events occurred at different pixels with the same time stamp, the number inserted into the discrete time vector at the corresponding time stamp would be two. The populated discrete time vector was then converted into the frequency domain using a Fast-Fourier Transform (FFT) with the results plotted and analyzed. Figure 21 steps through this process schematically.

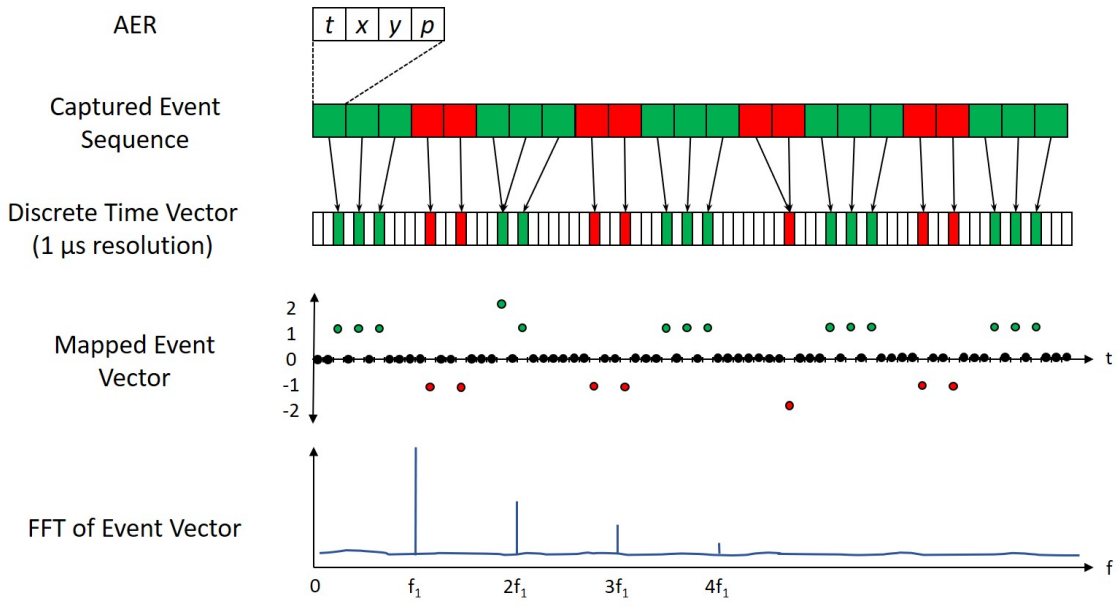


Figure 21: Explanatory diagram showing how the Fourier analysis of events is constructed.

Results. Using this Fourier analysis approach, the event data from the previous pixel bandwidth experiment was analyzed. For the various neutral density filter used, the results were comparable and so only the results where no neutral density filter was used are shown and discussed here.

Figure 22 shows several examples of the output from this Fourier analysis for multiple stimulus frequencies incident on the previously specified 13×13=169 pixel region of interest. From these results a clear peak can be seen to align with the specified stimulus frequency. From 600 - 3000 Hz there are also distinct harmonics that are expected given the fundamental nature of converting to the Fourier domain.

Despite the reduction in magnitude as the stimulus frequency increases, from these results it is evident that the event-based sensor is capable of detecting stimulus frequencies up to the order of 5000 Hz. This however does not imply that the pixel bandwidth is also in the order of 5000 Hz. Because all events generated from within the region of interest contribute to building the discrete time vector, this approach uses the contribution from each pixel to create an average frequency response, hence why its cut-off does not align with the 1 kHz cut-off measured using the previous approach. Additionally, because of the stochastic nature of each pixel, each pixel within the region of interest does not react consistently despite the same repeated stimulus. Essentially, for a single pixel and a given stimulus, there is a non-unity probability that an event will be generated. Similarly, this probability varies for each pixel and so while some pixels can respond to high frequencies, others cannot. This explains why the magnitude of the frequency response drops so significantly as the stimulus frequency increases. Within the region of interest, fewer and fewer pixels are able to response to higher and higher frequencies and so the magnitude of the response reduces accordingly.

Despite these challenges, this method is still useful for determining the frequency

of stimulus incident on an event-based sensor to a range far higher than the defined pixel bandwidth, a feature potentially useful for determining the rotation speed of a fan for example.

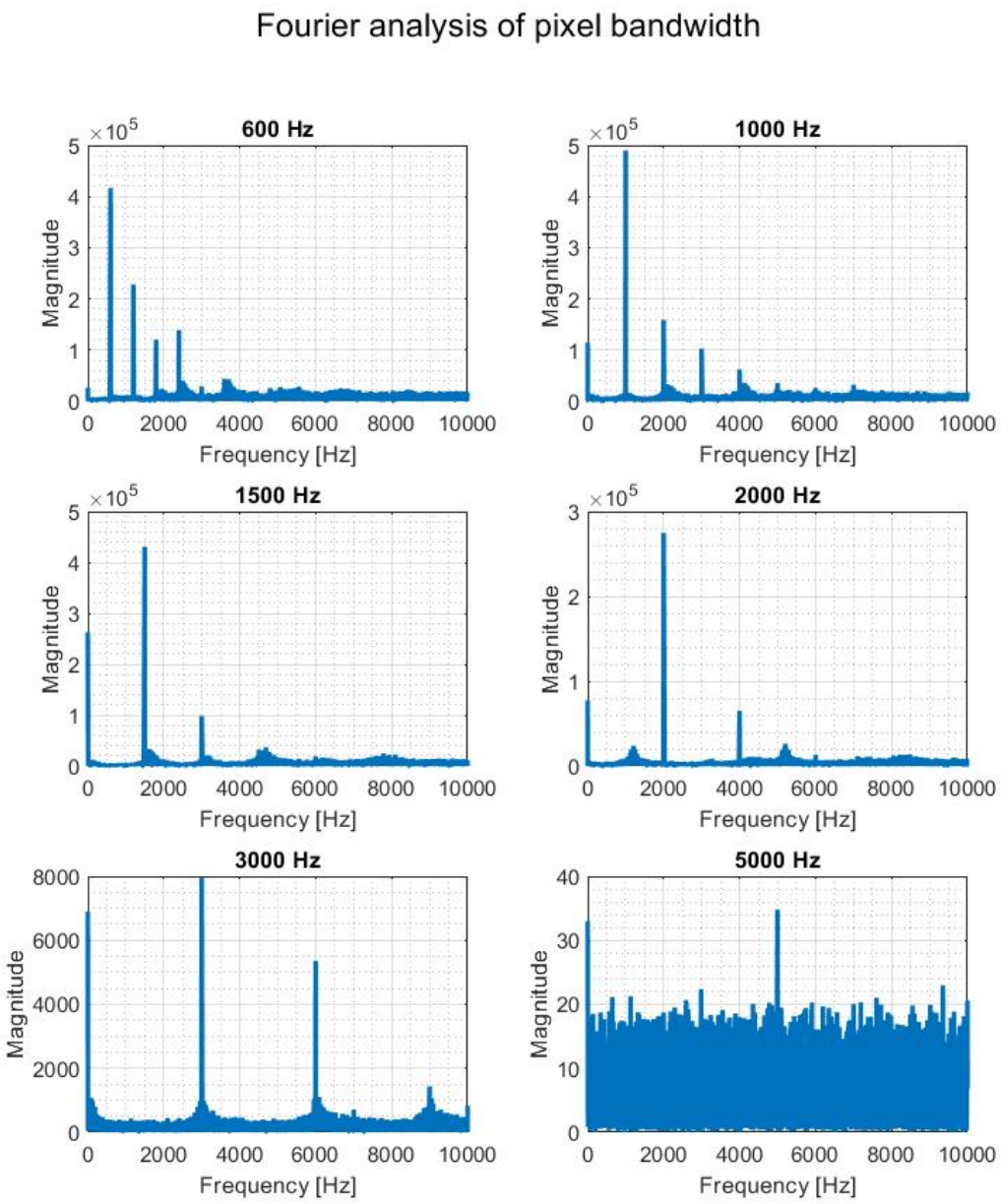


Figure 22: Select results from the Fourier analysis of pixel bandwidth experiments with changing stimulus frequency and no neutral density filter in place.

3.7 Closing Remarks

From this familiarization exercise, the complex dynamics and unique nature of event-based sensing was explored with reference to its response frequency. It became evident that depending on the sensor's bias settings and what is in its field of view (FOV), the response of the sensor can vary dramatically. From the results and analysis of this section, it has become clear that the sensor's frequency response becomes faster with brighter lighting conditions. In considering this, if in bright conditions pixel bandwidth is high, then the sensor will respond to faster oscillations in illumination. However, the by-product of this is that the sensor will also respond to higher frequency electronic noise, consequently producing more noise events. It is this phenomenon that is especially noticeable in low lighting conditions where a dark scene (or even regions of low light within a scene) will produce considerably more spurious events that a bright scene. Hence, it is very important to select bias settings that allow for adequate frequency response while minimizing unwanted noise as much as possible.

IV. Frame-Based Detection and Tracking

4.1 Preamble

Developed through many years of research, frame-based detection and tracking is a well established and mature technology. While there are many possible algorithms available, for the purposes of establishing a baseline to perform a comparison between frame-based and event-based tracking, a fundamental, relatively simple and effective approach to frame-based detection and tracking has been implemented for this research. This chapter outlines the approach taken for frame-based detection and tracking by introducing individual steps and discussing their composition in detail.

4.2 Frame-Based Detection and Tracking Process

In order to establish a mechanism to compare both frame and event-based methods, a fundamental frame-based approach was first selected to work with and base all future comparisons from. While many varied and unique detection and tracking algorithms have been developed over time, the approach adopted for this report uses a mature series of steps that, in general, are representative of common frame-based detection and tracking algorithms currently available.

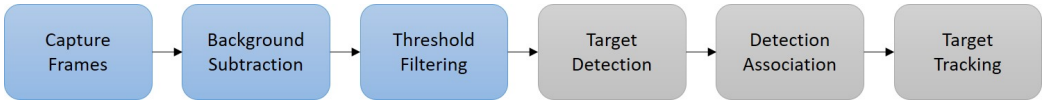


Figure 23: Frame-based target detection and tracking flowchart

The frame-based detection and tracking approach adopted for this report can be broken down into six primary steps. As shown in Figure 23, these include first capturing frames from the scene of interest before processing data via background subtraction and threshold filtering. These steps involve suppressing a stationary background

and filtering significant changes between frames with the effect of highlighting and extracting the motion of a moving target. Next, target detection takes filtered frames and identifies the spatial coordinates of possible targets. From there, detection association attempts to link detections across multiple frames to individual moving targets. Target tracking then uses associated detections to form target tracks that capture the current location and previous motion of a moving target. To explain how each of these steps was implemented in this research, the following sections provide further details in regards to their construction.

4.3 Background Subtraction

Change detection is the identification of changes in the state of a scene through the examination of pixel values between two or more video frames. Further to this, background subtraction is any technique which allows an image's foreground to be extracted for further processing. For many computer vision applications, background subtraction forms a critical step as it is the foundation of many object detection and tracking algorithms.

As with many computer vision algorithms, a wide variety of techniques are available, each tailored for unique and varied visual scenes as well as to meet user defined requirements (i.e. processing speed). Of the 29 different algorithms reviewed by Sobral et al. in [67], algorithms can be grouped according to their fundamental approaches. Examples include basic mean and variance methods, statistical methods using one or multiple Gaussians, statistical methods using color and texture features, fuzzy based methods, and methods that employ neural networks. The level of complexity in each of these methods varies considerably and is largely dependent on the nature of the scene for which it is used. Features such as dramatic changes in lighting (i.e. rising sun), casted shadows, fog, rain, shifting vegetation in wind or a panning

camera all pose significant challenges for effective background subtraction. While each of the above mentioned approaches perform well in simple, somewhat static scenes, as scene complexity increases the level of effectiveness can vary dramatically. Typically, the more complex the scene, the more complex the algorithm. Because of this, an algorithm must be chosen depending on the nature of the scene, its required processing speed and its final application.

One of the simplest approaches to background subtraction was chosen for this thesis where a temporal median is subtracted from each frame within a video sequence. Using the active pixel sensor (APS) frame based feature of the DAVIS 240C, the typical video sequences captured for this thesis could be considered simple. In all cases footage is captured from a stationary camera with either a stationary or slow moving background, constant lighting and either one or few small moving targets. Because of this, a simple approach could be adopted.

The approach implemented across most background subtraction algorithms uses a common three-step scheme. First is background initialization where the aim is to establish a background model from a fixed number of past frames. Second is foreground detection where a comparison is made between the current frame and the background model. This subtraction leads to computation of the scenes foreground. Third is background maintenance. During this process, more recent frames are used to update the background model developed in the initialization step helping to mitigate the effect of a slow moving background, or new objects that have been stationary for a long time.

Within the temporal median approach, the initial background model is created by calculating the median value of each pixel across a defined number of past and future frames (typically 10 frames either side of the current frame). By using the median pixel value from a series of sequential frames, the temporal impact of fixed pattern

noise (FPN) could be effectively averaged out. This background model was then subtracted from the current frame leaving behind any changes in the scene. To conduct the background maintenance step, a sliding window approach was adopted whereby the background model was repeatedly calculated using the same fixed number of past and future frames either side of the current frame. Figure 24 provides an example of the result where 10 past and 10 future frames were used.

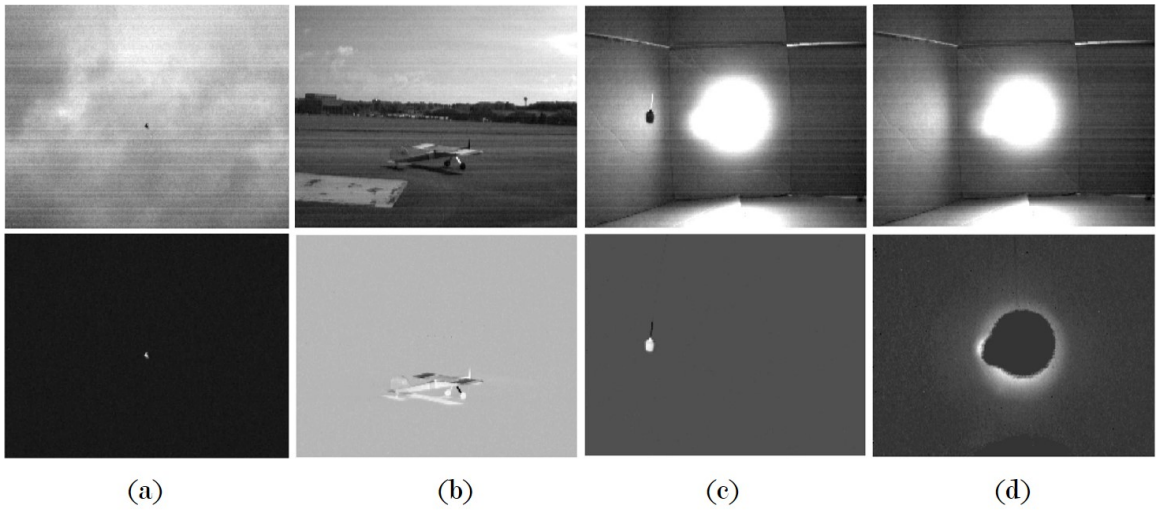


Figure 24: Background subtraction used under various circumstances. (a) A Skywalker x8 drone is effectively drawn out (b) A taxiing drone and its shadow. (c-d) A pendulum swinging in front of a bright light source. In (c) the pendulum is clearly visible while (d) highlights the impact of large changes in illumination due to shadowing.

Figure 24 (a) shows how effective background subtraction can be at pulling out a small airborne Skywalker x8 drone. In this case the background cloud is moving so slowly that it is effectively subtracted. Also notice that horizontal FPN is effectively removed using this method. Figure 24 (b) is simply a drone slowly taxiing. In this case the drone is clearly visible, but so too is its shadow. While this can disrupt interpretation of an object's true size and shape, because a shadow still represents a fundamental change to the background model, this artifact is difficult to avoid. Figure 24 (c) and (d) show a pendulum swinging in front of a bright light source.

In (c) the pendulum is far enough removed from the light source that background subtraction is effective. In (d) however, the pendulum is positioned directly in front of the light source slightly shadowing light from the camera. This drop in incident intensity results in a significant change from the background model causing the ring of the light source to be highlighted rather than the pendulum which has been effectively saturated out. This scenario represents an unfortunate limitation of background subtraction.

For the examples of Figure 24 it was found that by varying the number of frames to produce the background model from anywhere between 2 and 40 had little to no impact on the result. This is because in each case the background was either completely stationary or moving very slowly. The benefit of using more frames to produce the background model is reduction in the effect of noise through averaging. On the other hand, for a fast moving background (i.e. fast moving clouds) fewer frames are required as the background can only be considered stable for a short period. Fewer frames are also preferred for less, and closer to real-time, processing.

Despite shortfalls, use of the temporal median method for background subtraction is deemed suitable for this thesis. As all video sequences will be captured using a stationary camera with a stable, uncluttered background, this approach is expected to effectively identifying a moving object and enable further input into a threshold filter and eventually a detection and tracking algorithm.

4.4 Threshold Filtering

Assuming background subtraction has been completed successfully, before passing the resulting video sequence through a detection algorithm it is important to first filter out what is noise and what is signal. This is achieved using a simple process known as threshold filtering. Dependent on the camera used and the dynamics of a scene,

a suitable digital number is selected by the user to be a threshold. Each pixel from the background subtracted data is then compared against that threshold. Each pixel with a value less than the threshold will automatically be set to zero and considered background. Conversely, any pixel value above threshold will be considered signal and automatically set to one.

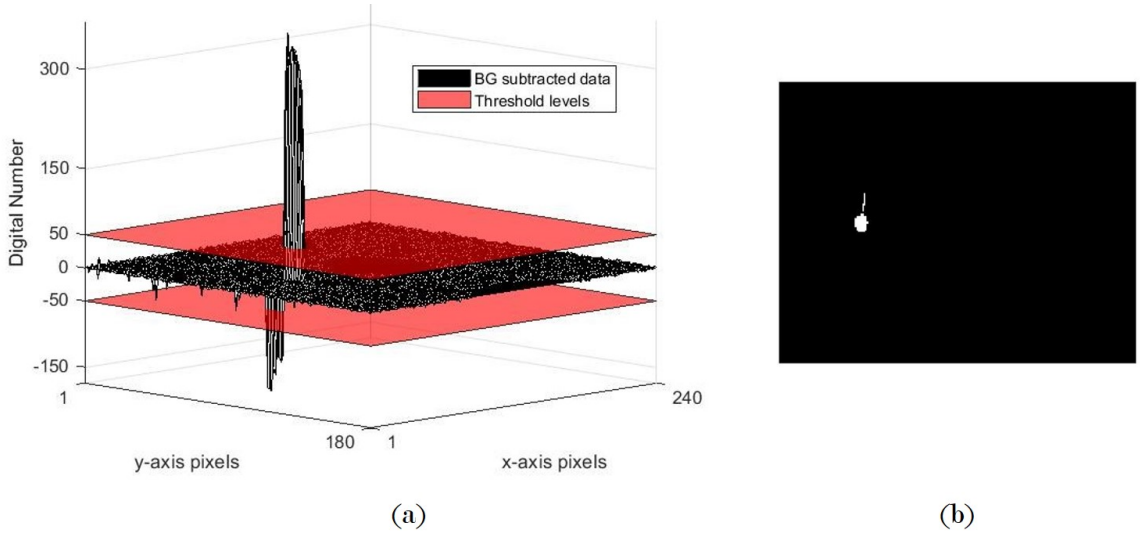


Figure 25: Example of threshold filtering. (a) A background subtracted frame (black mesh) with positive and negative threshold values (red planes). (b) Pixels above and below the red planes are set to one while all others are set to zero.

Figure 25 shows an example of threshold filtering where the pendulum scene from Figure 24 (c) has been used. As seen in (a) the threshold has been set to ± 50 as indicated by the red planes. All pixels with a digital number greater than +50 or less than -50 are considered signal and set to one while all other pixels are considered background and set to zero. This results in a frame as shown in (b). It is important to set both a positive and negative threshold to account for moving objects that impose either a positive or negative contrast on the background. In this example both cases are present. The main body of the pendulum is negatively contrast with respect to the background and so results in large positive digital numbers after background subtraction. Alternatively, the fishing line holding the pendulum reflects light from

the background light source creating a positive contrast that results in large negative numbers after background subtraction. After threshold filtering both features can be extracted effectively. It is then frames such as this that are passed through detection and tracking algorithms.

4.5 Target Detection

4.5.1 Single Frame

From threshold filtered frames, the next step now is to determine what pixels make up a target of interest. While background subtracted and threshold filtered frames draw out the detail of moving objects from background, it is not always accurate to assume that all moving objects constitute a target of interest. From the example of Figure 24 (b), the background subtracted frame is remarkably clean such that all non-zero pixels link to specific features (i.e. pendulum and pendulum string) while no random pixels have passed through the background filter to create false targets. Despite this, for this example the pendulum string is not considered a target of interest even with its clear presence in the background subtracted frame. In order to separate the two features and annotate the correct target of interest an algorithm must be employed that accurately discerns between wanted and unwanted signal.

In the first instance, an effective means to determine which pixels constitute an object detection is to take advantage of a combination of inbuilt MATLAB® functions. From the binary nature of a threshold filtered frame, pixels with a value of 1 are considered to have been generated by a moving target, they could therefore form a detection. Furthermore, when any of the 8 adjacent pixels surrounding that detection also have a value of 1 they are assumed to have been generated by a single larger target covering multiple pixels. By using MATLAB's® connected components function `bwconncomp`, the number of separate objects and the location of each pixel

inside each object can be grouped up and returned for further processing. It is important to note that at this stage the size of the object detection can range from 1 pixel to many.

The `'Area'`, `'Centroid'` location and `'Bounding Box'` of each detection can then be determined using the output of `bwconncomp` as input to the region properties function `regionprops`. A supporting algorithm can then be used to draw a bounding box around suspected detections from each frame in a video sequence.

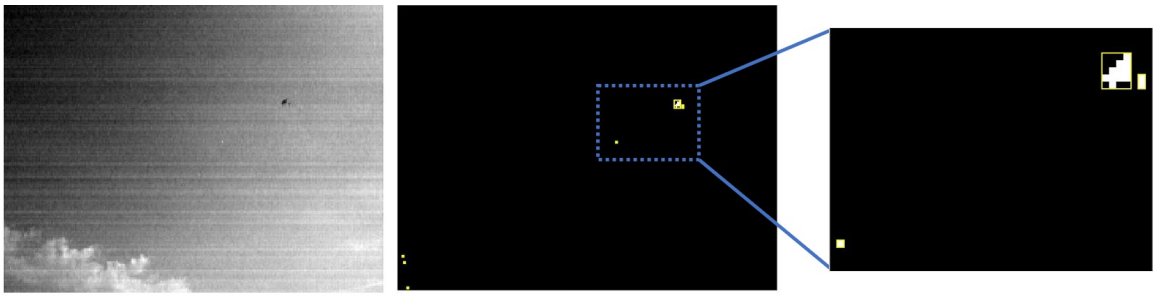


Figure 26: Example of target detection for an airborne drone in a single frame. (a) Reference APS frame (b) The result of background subtraction, `bwconncomp` and `regionprops` to draw yellow bounding boxes around suspected targets (c) Zoomed view of the actual target highlighting separation and the suspected presence of two targets.

Figure 26 shows a typical example of the outcome of this detection step starting with (a), an APS frame included for reference before (b) and (c), the background subtracted view with `bwconncomp` and `regionprops` aptly applied. In this figure, the white pixels are the direct result of background subtracted and threshold filtered frames using a sliding window of 20 frames and a threshold value of ± 20. The yellow boxes are drawn from the `'Bounding Box'` output of `regionprops`. Note the presence of two targets in the top right quarter at the actual location of the airborne drone in addition to four spurious single pixel detections in random locations throughout the frame.

The reason for the generation of two separate targets for the airborne drone is

simply due to the shape of the target. Difficult to see in the APS frame, but the narrow fuselage between the wings and vertical stabilizer is not big enough to be detected and so results in a gap between the main body of the aircraft and the tail. In order to mitigate this effect the function `bwmorph` in conjunction with the operator `'bridge'` can be used to bridge unconnected pixels with two non-zero neighbors. This way the detection of two targets from a single object can be consolidated into one.

In a similar fashion spurious single pixel detections can also be removed using `bwmorph` if desired. In some circumstances it is valid to assume a target object will encompass at least two or more pixels. In this example, the four spurious detections are the result of noisy pixels breaching the background subtracted threshold of ± 20. In most cases it is considered best to remove these noisy detections now in order to avoid putting excessive processing load on the target association stage. Therefore, by using the operator `'clean'`, isolated detections can simply be removed.

The outcome of using `bwmorph` with the operators `'bridge'` and `'clean'` is shown in Figure 27. By repeating this process for every frame in a video sequence the accuracy of the detection data can be improved and greatly simplified prior to passing it onto the target association and tracking stages.

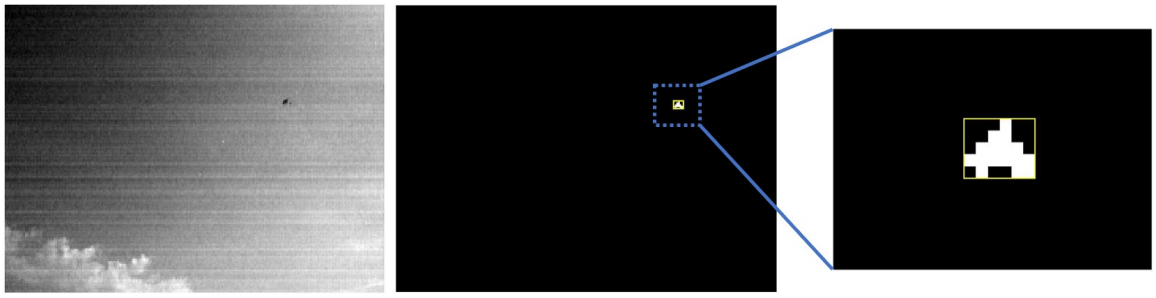


Figure 27: The effect of using MATLAB® function `bwmorph` with operators `'bridge'` and `'clean'` on the frames of Figure 26. Note the removal of single pixel detections and the consolidation of two separated detections into one for the airborne drone.

One of the effects of using `bwmorph` with the operator `'clean'` is that it allows the background subtracted threshold level to be lowered. The benefit of decreasing the threshold level is that it can draw out more detail in the actual target object. The drawback is that it comes at the expense of increasing noisy detections. Using `bwmorph` can dampen this effect while still retaining the benefit of improved detection sensitivity.

Considering the `'clean'` operator from a different angle, in some circumstances tracking a single or even sub-pixel target is the focus such that it is no longer valid to simply remove single pixel detections. In this case, the `'clean'` operator would be removed, transferring the noise filtering process to the association stage. Discussed in Section 4.6, by analysing detections across multiple frames, association can be used to discriminate between what is a valid detection and what is not. Unfortunately it comes at the cost of processing power when noisy detections are included.

4.5.2 Multiple Frames

As previously mentioned, detection using a frame-based video sequence must be performed on a frame-by-frame basis. To visually analyze these detections as a function of time, plots such as those shown in Figure 28 have been developed. Using the output of `regionprops`, these plots show the (x, y) location of each `'Centroid'` plotted against its corresponding frame timestamp. These 3-dimensional representations enable clear visualization of all detections for an entire video sequence. This in turn allows one to visualize the path of a moving object while clearly highlighting spurious uncorrelated detections.

In Figure 28, (a) shows detections of a banking Skywalker x8 drone starting in the far right center and ending in the far right bottom of the frame while (b) shows detections from a swinging pendulum. In (a) the `'clean'` operator of `bwmorph` was

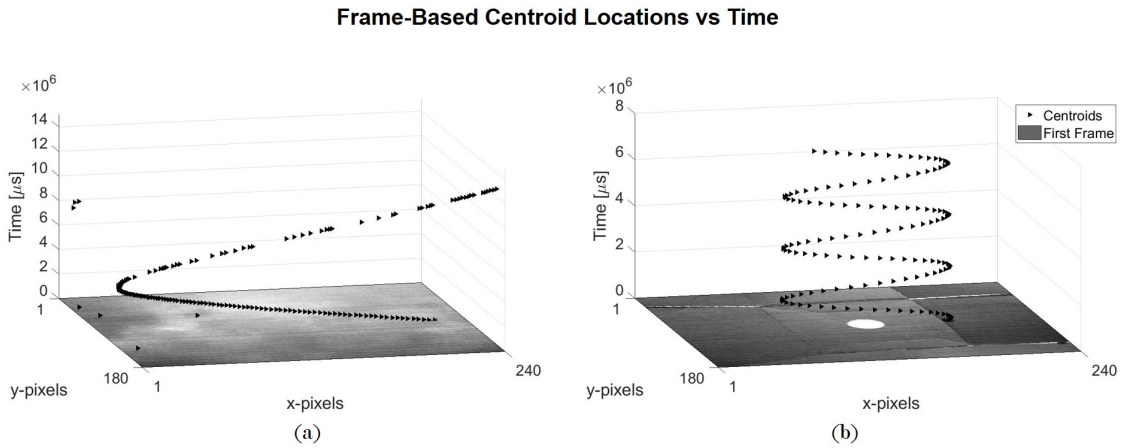


Figure 28: Frame-based centroids displayed as a function of time. (a) An airborne drone banking (b) A swinging pendulum.

used to remove single pixel detections, therefore spurious detections here consist of two or more pixels. In (b) however, because the pendulum covers up to 30 pixels, a size limitation was imposed such that only detections covering 15 pixels or more would be considered. With this approach, small noisy detections were removed leaving behind only valid detections from the swinging pendulum. When comparing frame and event-based tracking, these plots will prove useful for visually interpreting the capability of each detection method.

One of the advantages of these plots is that they quickly highlight and allow for visual identification of noisy events. Additionally, frames with missing detections are also clearly visible via gaps in the sequence of detections. To process nonconformity's such as this, fortunately filters like the Kalman filter can be used to both dismiss noise and interpolate missed detections. While not used in this comparative analysis, there is value in introducing Kalman filters in regards to their ability to enhance frame-based detection and tracking in addition to their relevance to event-based data.

4.5.3 Kalman Filter

One of the biggest challenges of target tracking is to provide accurate and precise estimations of unknown variables in the presence of uncertainty. In the case of visual-based target tracking, this uncertainty often arises from atmospheric effects, thermal noise or uncertain sensor behaviour for example. To help adjust to this uncertainty, the Kalman filter is one of the most important and commonly used tools in estimation algorithms. Fundamentally, implementing a Kalman filter algorithm produces estimates of hidden variables based on inaccurate and uncertain measurements which are often more accurate than those based on a single measurements alone. Additionally, it can provide a prediction of the future system state based on past estimations, a valuable trait in target tracking especially when missed detections and occlusions are a factor.

Taking a closer look at Figure 28 (a), the path of the airborne drone is clearly visible, yet within the sequence of detections small gaps appear. Considering the vertical time axis these gaps can range from one to several timestamps where they simply represent a missing detection from a corresponding frame. Missing detections occur in frames for several reasons but common causes include the target blending in with the background, occlusions, the target changing pose and becoming too small to resolve, or simply noise. Fortunately, as recordings such as this are of physical objects moving through space, one can assume they must therefore follow Newtonian physics and travel along a continuous path. Within the reasonable bounds of an object's maximum velocity and acceleration, it is therefore possible to predict a target's location for missing detections with considerable accuracy. Using similar logic, it is also possible to dismiss noisy detections based on whether or not a sequence of detections have violated the laws of physics.

The Kalman filter algorithm typically works in a two-step process and runs in

a recursive loop. In the prediction step, the Kalman filter produces estimates of uncertainty in the current state of a detection. Once the outcome of the next measurement (necessarily corrupted with uncertainty) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required. The advantage of this filter is that it can enhance the accuracy of actual detections by negating uncertainty, introduce synthetic detections that are otherwise lost and dismiss noisy detections that cannot be linked via the laws of physics. Additionally, due to its inherent ability to suppress noise, it also enables more sensitive detection by allowing the threshold setting on background subtracted frames to be lowered. As a result, this filter ensures an improved outcome for a frame-based approach by ensuring more accurate and continuous tracks.

With a constant frame-rate, using a consistent temporal separation to determine measures of uncertainty in an object's motion is both effective and relatively simple. For asynchronous events however, this is not the case as there is no defined temporal separation between events that combine to form a track. A Kalman filter can therefore not be used to interpolate missing events nor effectively dismiss noise. For this reason, to negate the influence of an interpretive algorithm and provide a strict comparison between sensor types, a Kalman filter has not been applied in this research as it would provide a disproportionate advantage to the frame-based approach.

4.6 Detection Association and Target Tracking

The aim of previous steps is to identify scene changes on a frame-by-frame basis and generate a series of stand alone detections. The next steps are to then link those detections to a common target and generate a track that captures the current

position and previous motion of that target. While to the human eye one can quickly and easily draw a link between separate detections and associate them to a single target, for a machine this task is quite challenging. In order to develop autonomous association between detections, the decision process must be extremely well defined in order to translate it into a functional algorithm.

This is an extremely challenging task where despite the maturity of this technology, no single algorithm has been developed that is effective across all possible scenarios. In general, a detection and tracking algorithm must be tailored to suit particular scenarios based on assumptions that predict aspects such as target size, target shape, number or targets, background dynamics, lighting or possible occlusions (e.g. smoke, cloud, sun). In fact, in many cases multiple algorithms will run in parallel in order to ensure an effective result. With this in mind and considering the primary goal of this research is to compare sensor types, a specific frame-based detection association and tracking algorithm has not been developed here. Instead, to establish an effective comparison between sensor types, the asynchronous event-based association algorithm, TriplClust [68], will be leveraged here. This algorithm simply takes in the (x, y, t) position of each detection and attempts to associate them to individual targets based on a series of geometric calculations. A detailed description of this algorithm is provided below in Section 5.6 with examples of its application to frame-based detections provided in Chapter VI.

4.7 Closing Remarks

This chapter has explored the individual steps that make up a standard frame-based detection and tracking algorithm. In adopting a simplistic approach, the steps of background subtraction, threshold filtering and target detection were discussed in detail with particular reference to how the algorithm is constructed. The impact of

Kalman filtering was then mentioned however not implemented due to the disproportionate advantage it would instill for frame-based tracking. Detection association and target tracking were then discussed briefly with specific detail deferred to Chapters V and VI.

Despite the simplicity of the approach discussed in this chapter, the basic concept of how frame-based data can be used to detect and track moving targets has been established. While this technique has proven capability, there remains considerable overhead in processing large amounts of redundant data in addition to the inherent challenge of tracking high speed targets and working with scenes with high dynamic range. In an attempt to overcome these shortfalls, the next chapter explores the implementation of detection and tracking using the unique capabilities of event-based sensors.

V. Event-Based Detection and Tracking

5.1 Preamble

Target detection and tracking is a complex problem and one that when using an event-based sensor presents a unique and novel challenge to overcome. In presenting potential solutions to this problem, this chapter discusses several filtering techniques required to pre-process raw events before detailing two fundamentally different approaches to detection association. One approach requires the use of pseudo-frames while the other performs association on filtered events directly. For both cases, relevant advantages and disadvantages are discussed with particular reference to their practical application.

5.2 Event-Based Detection and Tracking Process

Event-based detection and tracking diverges slightly from the equivalent process for frame-based sensors discussed earlier in Section 4.2. Shown in Figure 29, the process begins naturally with the capture of raw event data. Unwanted events must then be filtered appropriately to draw out relevant object motion. Unlike the frame-based process, the unique nature of event data must then be considered before target detection can occur. Here two fundamentally different approaches are considered where the first involves grouping events into defined temporal bins (i.e. pseudo-frames) while the second involves processing individual events asynchronously. For pseudo-frames, target detection, association and tracking take place in line with standard frame-based methods, while for an asynchronous approach, filtered events are treated as detections and so the process moves directly to association and tracking.

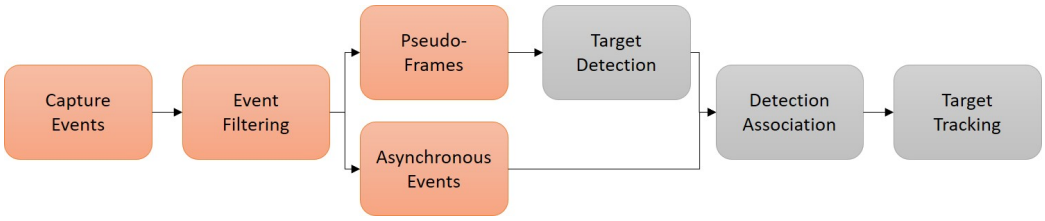


Figure 29: Event-based target detection and tracking flowchart

5.3 Event-Based Filtering

In order to take advantage of event-based sensors for the purpose of object detection and tracking, raw events must first be filtered to remove unwanted noise and draw out the necessary detail to capture an object's motion. Because event-based sensors are notoriously noisy, this step has become critical to ensuring their practicality. This section discusses several useful filters that can be used for detection purposes, capturing their implementation approach and highlighting their relevance through examples.

5.3.1 Refractory Period Filter

A Dynamic Vision Sensor (DVS) pixel does not exhibit an instantaneous response to stimulus, nor does it recover instantaneously to steady state after being excited. In effect, there is a finite time when a DVS pixel is considered unresponsive. Known as the refractory period, this period essentially limits how fast a pixel can respond to changing stimulus. For the case when events are generated within a known refractory period, they are most likely the result of spurious noise or current leakage between pixels and not from genuine log-light intensity changes. Because these events have occurred beyond the capability of the sensor it follows that a suitable filter to remove such events is essential.

In the refractory period filter, high frequency sensor noise is removed by enforcing a minimum time difference between two consecutive events of any, or of a single

polarity, at a particular pixel [69]. The algorithm is implemented by creating an empty 240×180 array to store the time stamp of the last valid event $t^{last}_{(x,y)}$ and using it to calculate the time difference $\Delta t_{(x,y)}$ between it and the current event $t_{(x,y)}$.

$$\Delta t_{(x,y)} = t_{(x,y)} - t^{last}_{(x,y)} \tag{5}$$

The result is then compared to a user defined refractory period Δt_{ref},

$$\Delta t_{(x,y)} > \Delta t_{ref} \tag{6}$$

where the current event is passed if and only if this condition is met, otherwise it is rejected. It should be noted that by using an array to store previous timestamps which are indexed using the (x, y) coordinates of an event, new events can be assessed as they arrive, implying this filter could be implemented in real-time.

As demonstrated in Section 3.6.5.1 by Figure 20, pixel bandwidth and therefore refractory period varies depending on background illumination. Despite this, it is not always best to set Δt_{ref} to its theoretical minimum, rather, it is better to tailor it to the expected nature of the scene. This is because there is often noise content that occurs within a longer period than Δt_{ref} but still with a frequency content higher than the fastest changing feature in a scene. It is therefore best to set Δt_{ref} to a value slightly less than the fastest changing feature in a scene.

If considering a scene with a slow moving drone in outdoor daylight conditions, values for Δt_{ref} could be as high as 10 ms, removing single pixel events that occur at or above 100 Hz of each other. Conversely, for a fast spinning black dot on a white disc in dim indoor light, Δt_{ref} could be as low as 1 ms or equivalently 1000 Hz. It is therefore important to tailor this refractory period filter in accordance with the expected nature of the scene in question.

In practice the effectiveness of this filter is quite limited as it only considers the response of a single pixel in time with no consideration to events occurring in neighboring pixels. Because of the stochastic nature of event generation (refer Section 3.5.1), even when optimized many unwanted noise events still pass through this filter. Because of this, it is therefore necessary to introduce additional stages of event filtering.

5.3.2 Nearest Neighbor Filter

Thermal noise and current leakage across pixels typically result in unwanted background noise events which both decay the quality of data collected as well as utilize unnecessary bandwidth and processing power. Fortunately there is a key difference that can be used to separate genuine scene activity from background noise events that can be implemented into a filtering algorithm.

Events that occur as a result of noise generally lack temporal correlation with events in their spatial neighborhood. In other words, noise events bear no spatial connection with their surrounding pixels. On the other hand, genuine events generated by changes in log-light intensity will have some form of temporal and spatial correlation as unique object and scene features transit pixels within a sensor's field of view. It is this difference that provides the means to filter out background noise by removing events that occur with no spatio-temporal correlation between themselves and their immediate neighboring pixels. This filter is known as the nearest neighbor filter.

In the implementation of this algorithm, again an empty 240×180 array is used to store previous event timestamps. To process an event at pixel (x, y), the nearest neighbor filter first determines whether one of the eight vertical, horizontal or diagonal neighboring pixels has had an event within a user defined time Δt_{NN}. If there exists a neighboring event with a time-stamp within Δt_{NN} of the event in question, that

event is considered supported and will pass through the filter. If it is unsupported, the event will be treated as noise and simply removed.

As with the refractory filter, an appropriate value for Δt_{NN} depends on the dynamics of the scene. If the scene is not expected to change quickly (i.e. a stationary sensor with a slow moving target) then a long enough Δt_{NN} must be used to ensure correlated events are not filtered out. The negative impact is that the longer Δt_{NN} is, the more likely a noise event will occur within that events nearest neighbor, leading to unwanted events passing through the filter.

Alternatively, in a similar scene but with a fast moving target, a shorter Δt_{NN} can be used to great effect. With a fast moving target, events in neighboring pixels are generated with a time difference typically much shorter than background noise, allowing genuine events to be picked out. Because a shorter Δt_{NN} reduces the likelihood of noise events inadvertently passing through the filter, much cleaner results can be produced.

As previously mentioned, the rate of spurious events depends greatly on the illumination level of a scene. It is generally understood that dark scenes generate far more events than brighter scenes, and because of this, filter optimization is particularly challenging. As a scene gets darker, because of the sheer volume of noise events and the rate of their occurrence, the effectiveness of this nearest neighbor filter diminishes. While the best way to remove high frequency noise is typically to reduce Δt_{NN} as much as possible, this can result in a significant number of genuine events also being removed. This becomes a important issue for scenes with a high dynamic range because of the distinct difference in the number of events from various spatial regions in a scene. Optimizing Δt_{NN} to be effective for both dark and bright regions presents a significant challenge and one that highlights a genuine limitation of this filter. That is, the challenge of drawing out signal data by removing noise within a dark

scene, or dark regions of a scene, is an issue not yet effectively solved. Despite these limitations, when used in bright conditions with relatively fast moving targets, its ability to reduce noise and retain genuine events for detection and tracking purposes is very effective.

5.3.3 Polarity Filter

Before applying refractory and nearest neighbor filters there is an important concept that must first be considered. Choosing whether or not to separate event polarity before applying said filters can have a dramatic impact on data usability from a detection and tracking point of view. At the most basic level a single event caused by changes in log-light intensity does not carry much information for detection and tracking purposes. Instead, it is the complex relationship between events from neighboring pixels that contain information about an object's position and motion. In a detection and tracking scenario, aside from noise, the primary cause of events is expected to be moving objects or targets (assuming the sensor is stationary). Therefore, as an object moves through space both its leading and trailing edge generate events of opposite polarity depending on whether the target has negative or positive contrast with its background. One can therefore conclude that both ON and OFF events contain reciprocal information such that only one polarity is enough to perform equivalent detection and tracking. While this is an ideal scenario, in practice bias settings and noise significantly disrupt the purity of this concept which typically leads to one polarity being favored.

To demonstrate this concept, event-based filters have been applied to event data from the same looping drone scene as that of Figures 26 to 28. In this scene, data was captured under bright daylight conditions using both active pixel sensor (APS) frames and DVS events simultaneously. Figure 30 shows the ON and OFF event data

from this capture while Table 2 contains the corresponding event statistics. Moving from left to right, (a) shows raw ON and OFF events as recorded by the DAVIS 240C, (b) and (c) then apply the refractory and nearest neighbor filters in series with Δt_{ref} and Δt_{NN} set to 4 seconds and 0.1 seconds respectively.

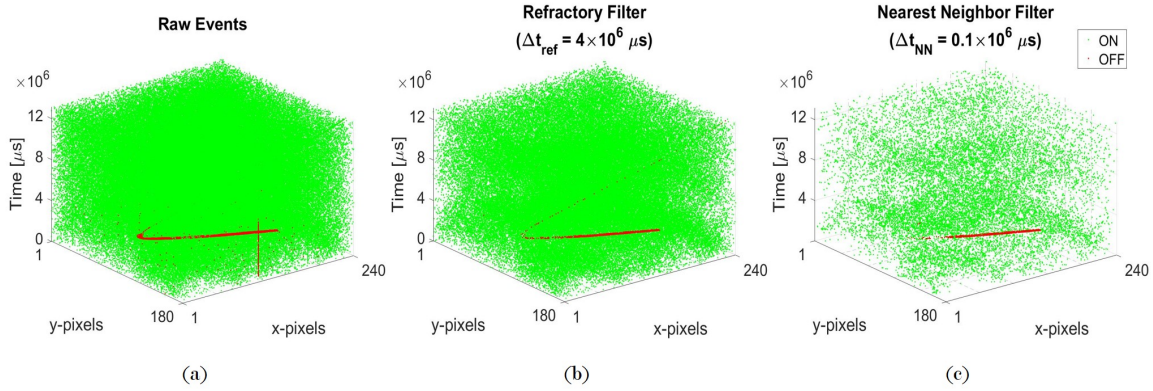


Figure 30: Event-based filtering - ON and OFF events

In (a), the sheer number of raw events essentially fills the entire (x, y, t) data-cube making visualization of the drone extremely difficult. Of the 208,629 raw events generated in this recording, 97.79% are ON while just 2.21% are OFF. In this case, the overwhelming majority of ON events is largely attributed to APS coupling where a repeating plane of random events occur at the APS frame rate of approximately 20 Hz. This effect is best visualized in Figure 31 where for both ON and OFF events, a histogram plots accumulated event count versus time for the first 0.5 seconds of this recording. Step-wise increases in accumulated event count are clearly visible where their temporal separation is approximately 20 Hz or a harmonic thereof, highlighting

Table 2: ON/OFF event statistics

Filter and Event Type	**Raw**		**Refractory**		**Nearest Neighbor**	
	ON	**OFF**	**ON**	**OFF**	**ON**	**OFF**
Number of Events	204,015	4,614	107,328	559	18,205	467
% of Filtered Events by Polarity	97.79	2.21	99.48	0.52	97.5	2.50
% of Raw Events by Polarity	97.79	2.21	52.61	12.12	8.92	10.12

the effect of APS coupling.

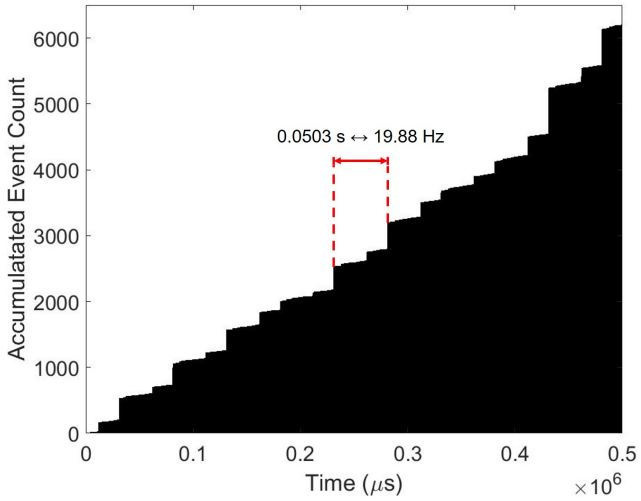


Figure 31: Event accumulation highlighting APS coupling at approximately 20 Hz

While the drones motion does generate both ON and OFF events, with raw events only it is not possible to draw out relevant detection and tracking information through noise. To remove spurious noise first the refractory filter was applied as shown in (b). In this case total events dropped by almost half to 107,887 with 99.48% ON events and 0.52% OFF events. Despite the drone's motion being marginally clearer, again the overwhelming presence of events makes it difficult to draw out relevant detection and tracking information. By then applying the nearest neighbor filter to the refractory filtered data the total events were reduced further to 18,672 with 97.50% ON events and 2.50% OFF events. As can be seen in (c), driving the filters this hard causes information of the drones path to be lost such that to the human eye only the first few seconds of flight can be associated with the drone while it covers multiple pixels in the near-field of the sensor. Therefore for this scene, the use of both ON and OFF events can be considered unfeasible for effective detection and tracking.

By separating event polarity it is possible to reveal information otherwise hidden amongst noise and excessive events. To explore this concept, Figure 32 plots only ON

events for the same looping drone scene while Table 3 captures the event statistics. Again the refractory and nearest neighbor filters are applied in series with Δt_{ref} and Δt_{NN} set to 4 seconds and 0.01 seconds respectively. Not surprisingly, because APS coupling generated so many spurious ON events the results are comparable to that of Figure 30. It is not until after the refractory and nearest neighbor filters are applied that the total number of events are reduced to 1.86% of the number of raw ON events that evidence of a moving target begins to emerge. Even so, ON events generated by the drone are so few that it is difficult to interpret its path.

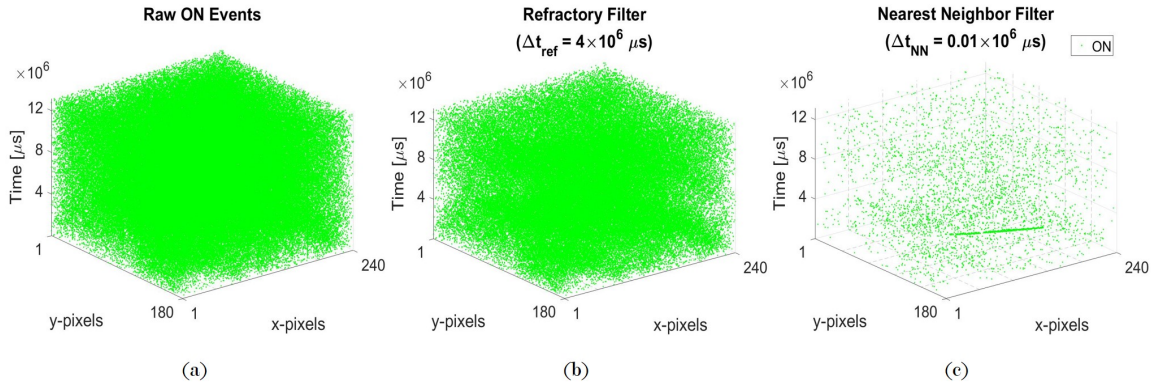


Figure 32: Event-based filtering - ON events only

Conversely, Figure 33 and Table 4 capture only OFF events with Δt_{ref} and Δt_{NN} set to 12 seconds and 0.64 seconds respectively. Quite dramatically, now the complete path of the drone is clearly visible with only minimal distraction. Even when viewing the raw OFF events of (a), the drone's path is clearly visible amongst the presence of numerous spurious events and a vertical line of events caused by a 'hot' pixel. Parsing raw OFF events through the refractory filter firstly removes evidence of the hot pixel

Table 3: ON event statistics

Filter and Event Type	**Raw**	**Refractory**	**Nearest Neighbor**
Number of ON Events	204,015	107,880	3,803
% of Raw ON Events	100	52.88	1.86

before the nearest neighbor filter cleans the event data completely, revealing only the path taken by the drone. In this process, the total number of OFF events has been reduced by 74.51% where although numerous events along the path of the drone have been filtered out, the drone's path remains clearly visible and highly usable for further detection and tracking algorithms.

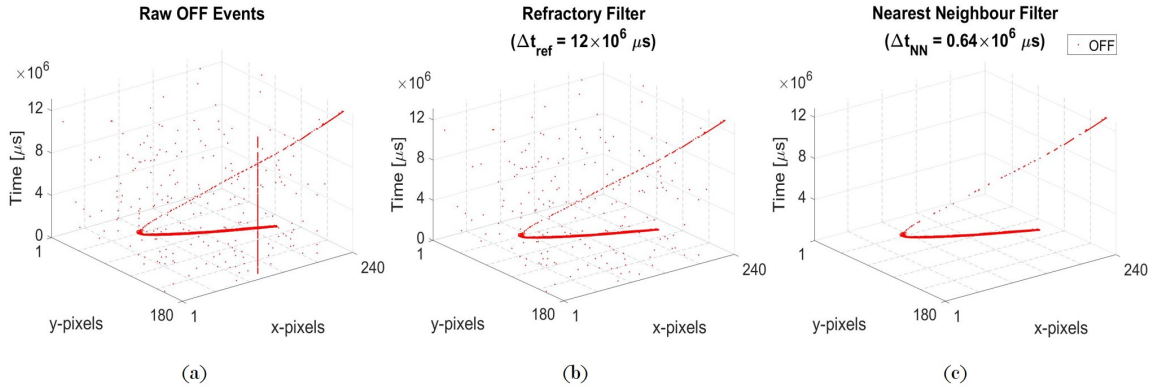


Figure 33: Event-based filtering - OFF events only

Through this scenario, the benefit of separating ON and OFF events prior to filtering has been demonstrated. As previously mentioned, because a moving target produces both ON and OFF events as it transits, equivalent information is contained in each polarity and therefore the polarity chosen is irrelevant. Instead, the best polarity to choose typically depends on bias settings, the nature of the scene and the quality of the recording. It is essentially situational where although for this scenario OFF events proved most effective, this will not always be the case.

Table 4: OFF event statistics

Filter and Event Type	**Raw**	**Refractory**	**Nearest Neighbor**
Number of OFF Events	4,614	1,418	1,176
% of Raw OFF Events	100	30.73	25.49

5.3.4 Event Pair Filter

Another very effective event-based filter draws on the presence of ON/OFF or OFF/ON event pairs that occur in the temporal domain at each individual pixel. By taking advantage of the transiting nature of a moving object, one can assume that as an object transits across a pixels field of view, its leading edge will generate events of one polarity while its trailing edge will generate events of the opposite polarity soon thereafter. Additionally, by imposing an upper limit on the temporal separation t_{EP} of these event pairs, those events linked directly to a moving target can be further localized. In effect, a recording's entire event stream can be reduced to just those event pairs caused by a transiting target of interest, consequently enabling the path of that target to be detected effectively. Furthermore, as this filter only seeks out event pairs, it operates completely independently of the number of targets in the sensors field of view, therefore becoming an enabler for multi-target tracking.

In looking at the functionality of this filter by considering the leading edge of a moving object, depending on the change in log-light intensity that a moving object projects, the threshold value in the comparator circuit of the DVS pixel may be breached several times. This in effect causes a string of events to occur in quick succession all with the same polarity. Similarly, as the object transits away the trailing edge causes a reciprocal effect with events of the opposite polarity. While the number of consecutive events of the same polarity can reveal detail on the absolute brightness of a target, in a detection and tracking scenario this information is considered secondary to simply identifying the target's spatial location. It is therefore deemed a valid approach to filter consecutive events of the same polarity and retain only those event pairs that contain a change in polarity.

To explain how this filter is implemented, firstly a $240 \times 180 \times 4$ matrix representing the (x, y) location of each pixel is pre-allocated to store four unique identifiers of

previous events at each pixel. In this case the four unique identifiers include the polarity, event index (i.e. its index number in the stream of all events), timestamp and whether or not it is the first event to occur at that pixel. As new events arrive and are processed sequentially they can be compared quickly to the previous event from that pixel with a decision made on whether to retain or filter the relevant event. With an efficient decision loop and minimal requirement for memory, this approach could therefore be implemented into real-time applications, availing these sensors to very high speed detection and tracking applications.

In considering this filter's ability to cut through noise, again the same assumption used for the nearest neighbor filter (Section 5.3.2) can be used. Because events that occur as a result of noise generally lack temporal correlation, the likelihood of noisy event pairs occurring with opposite polarity and within a user defined temporal separation is extremely low. Therefore, a single application of this filter has the ability to remove a significant percentage of spurious events. Although it is still possible for noisy event pairs to pass through this filter, their sparse occurrence generally lacks spatial correlation such that they can be removed with use of the nearest neighbor filter.

To demonstrate the effectiveness of this event pair filter when used in conjunction with the nearest neighbor filter, Figure 34 shows a typical outcome for the simple looping drone scenario used previously. By simply pulling out ON/OFF and OFF/ON pairs of events, the overloaded event point cloud of Figure 30(a) has been reduced to that shown in Figure 34(a). In this example APS coupling generated significantly more ON events which have been largely removed by the event pair filter. Only those ON events paired with the far more scarce OFF events remain. In (b), by then applying an upper limit Δt_{EP} of 0.1 seconds on the temporal separation of event pairs, only those events linked to a moving target are extracted, reducing the event point

cloud even further. At this point a number of disparate, noisy event pairs along with the previously identified hot pixels still remain. To remove these unwanted events the aforementioned nearest neighbor filter is then applied in (c), leaving behind a clean trace of the airborne drone built only from event pairs.

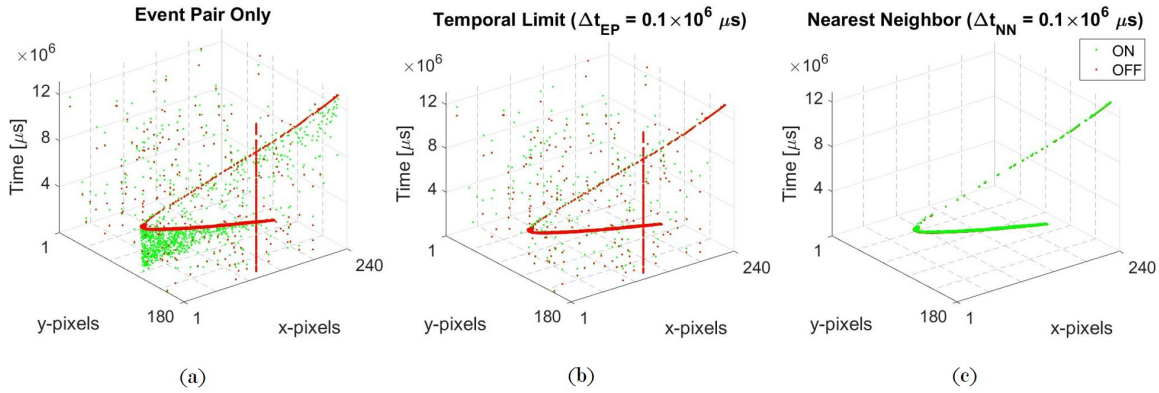


Figure 34: Example application of the event pair filter. For a looping drone, (a) draws out ON/OFF and OFF/ON pairs only while (b) enforces a maximum temporal separation between those event pairs. (c) then applies the nearest neighbor filter to produce a very clean result.

To indicate the reduction in the total number of events, Table 5 has been included showing the event count at each stage of the filter. It shows that by applying the event pair filter with a temporal limitation in conjunction with the nearest neighbor filter reduces the event count to just 1.35% of the total raw events while importantly retaining the spatial information of the moving target. One important point to note from this table is that ON/OFF and OFF/ON pairs can overlap meaning that the number of ON and OFF events do not necessarily need to be equal.

Table 5: Event pair filter statistics

Filter Type	**ON**	**OFF**	**Total**	**% of Raw Events**
Raw Events	204,015	4,614	208,629	100
Event Pair Only	2,797	2,389	5,186	2.49
Event Pair / Apply t_{EP}	2,029	2,330	4,359	2.09
Event Pair / Apply t_{EP} / NN	1,188	1,634	2,822	1.35

With this significant reduction in total events, the level of complexity required to perform target detection and association using only events is expected to be simplified. Despite forfeiting the ability to infer absolute brightness, reducing the total number of events while still retaining essential information to accurately capture the location of a moving target implies that fewer association decisions are required while the probability of accurately allocating events to particular targets is improved. By again employing the assumption that equivalent information is retained in both ON and OFF events, the total number of events can be reduced even further by using the polarity filter to truly maximize processing efficiency and overall accuracy.

In general the performance of this filter is very good, there are however some shortfalls. The fundamental assumption of this filter is that a moving object will create event pairs in quick succession as it transits over a pixel. Because of this, the filter is particularly effective for relatively fast moving objects moving perpendicular to the sensors line of sight. However, linked to the user-defined temporal limitation t_{EP} is an underlying assumption around the velocity of the moving target. For a fast moving object the temporal separation in event pairs is low allowing t_{EP} to be set low which reduces the probability of noise intrusion between event pairs. On the other hand, as the object velocity slows, t_{EP} must be increased to compensate, consequently increasing the probability of noise intrusion between event pairs which ultimately degrades the effectiveness of the filter as a whole. The scenario whereby this filter breaks down is therefore when an object's speed decreases or its orientation of motion aligns with the sensor's boresight, creating event pairs with a comparatively long temporal separation. Regardless of this shortfall, this filter has proved very efficient and effective at localizing only those events caused by a moving target such that the difficult issue of event association is made more feasible.

5.4 Detection and Tracking with Pseudo-Frames

After event filtering, the next logical step is to use those events for the detection and tracking of moving objects. Given the paradigm shift of event-based versus frame-based data, this presents a unique challenge to overcome and one that is discussed in the following sections.

5.4.1 Event-Based Pseudo-Frames

Perhaps the simplest and most intuitive means to process event data for target detection and tracking is through the use of pseudo-frames. As the name suggests, pseudo-frames are an event-based equivalent of standard video frames produced by reducing continuous event data into regularly spaced time intervals, or frames. Because of their equivalence to standard frames, pseudo-frames can be used as input to standard frame-based tracking algorithms with various levels of effectiveness.

The creation of pseudo-frames is dependent on a user defined frame rate, whereby events are grouped into frames in accordance with their timestamp. A defined frame rate implies there is a time period Δt_{frame} between each frame such that any event that occurs within that time will receive the same timestamp and be grouped into a single frame. Using a sliding window approach, a sequence of event-based pseudo-frames can then be created to emulate that of a standard video.

Figure 35 captures two examples of the transformation of continuous event-based data into pseudo-frames at various frame rates. The top sequence shows a 2-second snippet of a black dot on a white disc spinning at approximately 1.125 Hz. Here only OFF events are used before passing the events through a refractory and nearest neighbor filter with Δt_{ref} and Δt_{NN} set to 10 μs and 100 μs respectively. Similarly, the bottom sequence shows the Skywalker x8 drone performing a banking turn. Again only OFF events are used in conjunction with the refractory and nearest neighbor

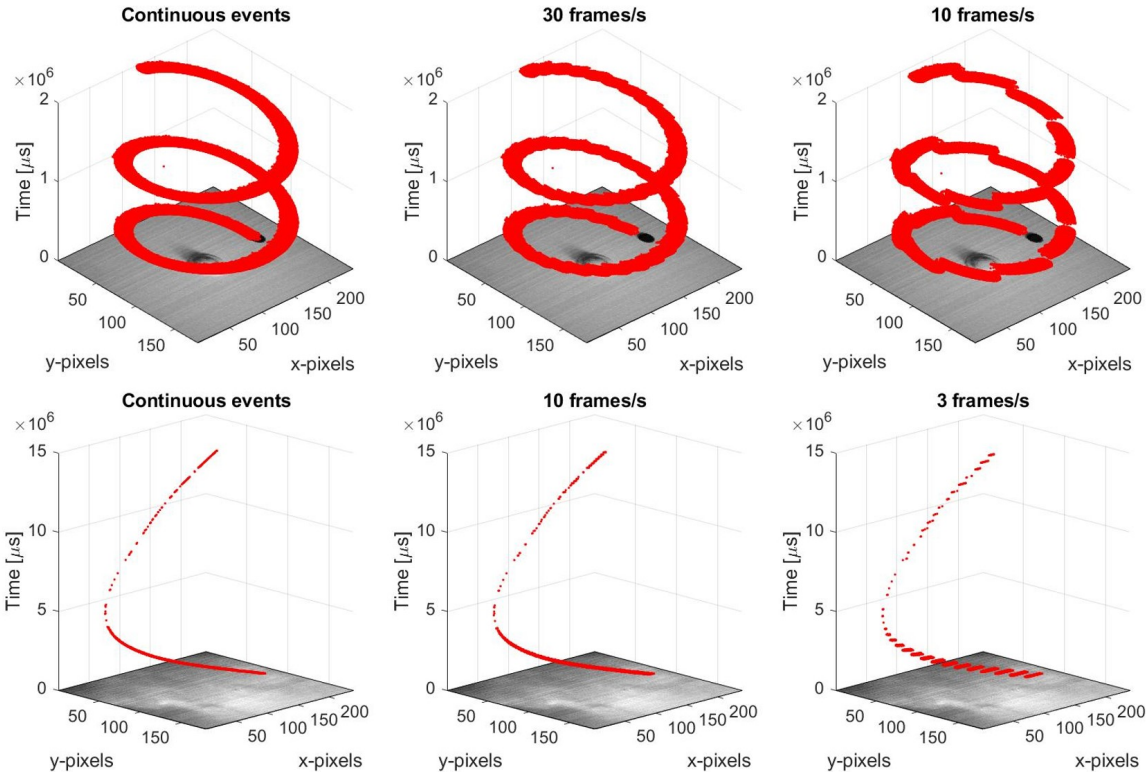


Figure 35: Two examples of event-based pseudo-frames displayed with various equivalent frame rates. Top sequence shows a spinning black dot while the bottom sequence shows an airborne drone performing a banking turn.

filters with Δt_{ref} and Δt_{NN} set to 350 μs and 1 s respectively.

While the temporal resolution of the DAVIS 240C is 1 μs, without applying pseudo-frames the stream of events is essentially continuous for both sequences. After applying pseudo-frames at various frame-rates the time step of events becomes visibly discrete. Using the top sequence as an example, because the black dot is spinning relatively slowly, a frame rate of 30 frames/s is fast enough to maintain a somewhat continuous stream of events and accurately capture the motion of the dot between pseudo-frames. For 10 frames/s however, the black dot rotates so far in one tenth of a second that the equivalent of motion blur is transferred to each pseudo-frame. In this case the location of the dot from frame to frame is indeterminate. For the bottom sequence there is a similar effect however the drone is moving slower than the

black dot and as such slower frame rates could be used. Figure 36 shows examples of single pseudo-frames from each sequence for various frame rates. This figure shows the validity of pseudo-frames when the appropriate frame rate is chosen, but also highlights faults in this approach when a frame rate is chosen that is either too low or too high.

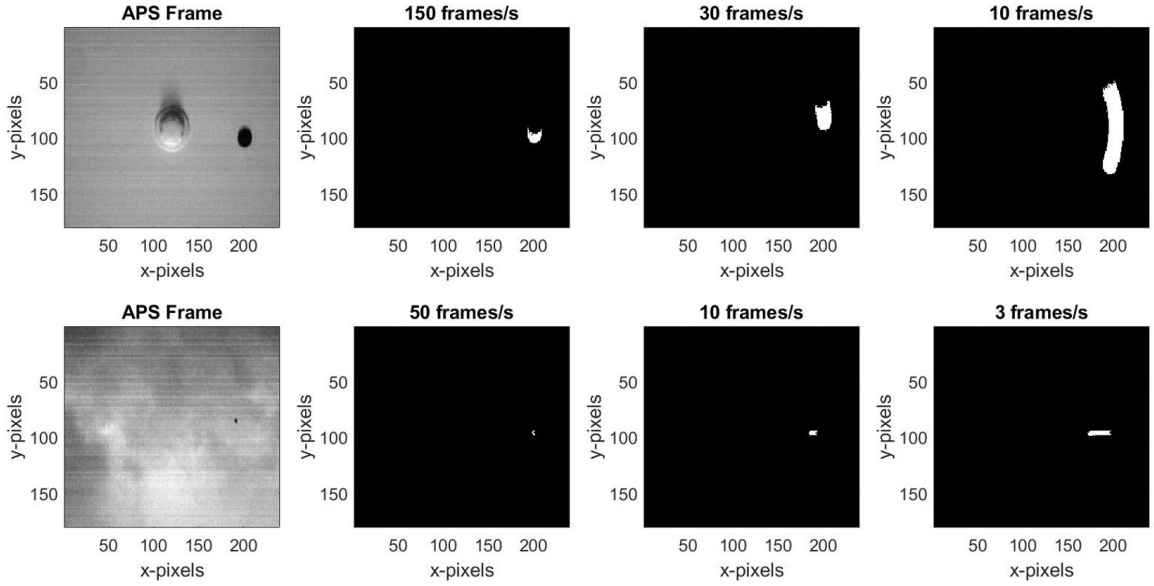


Figure 36: Two examples showing single event-based pseudo-frames produced by different frame rates. An extension of Figure 35, the top sequence shows a spinning black dot while the bottom sequence shows an airborne drone in a banking turn. The first APS frame in each sequence is shown for reference.

Moving from left to right through each sequence, an APS reference frame has been included as a visual reference, example pseudo-frames whereby the frame rate has been progressively stepped down then follow. Firstly it can be seen that as the frame rate decreases, more and more events become available to build each frame. Secondly, and perhaps more importantly however, is because events are generated by the motion of an object's leading or trailing edge, as the frame rate becomes high compared to an object's motion, pseudo-frames begins to resemble the edge of a moving object. This is highlighted in the second image of each sequence where, since

the target object is in negative contrast to the background and only OFF events have been used, these pseudo-frames show the leading edge of the black dot and airborne drone.

Comparing this to slow frame rates, in a similar fashion to how long exposure in frame-based sensors can create motion blur, so too is the effect in pseudo-frames. With the amalgamation of events over a relatively long period, the location of a target object is effectively smeared as shown in the far right image of each sequence. In this case the precise location of the target object is lost.

There must therefore be an ideal frame rate that encapsulates the shape of the target object together with its precise location, essentially resembling that of a standard background subtracted APS frame. To a point high frame rates will work for target detection until the number of events in each frame is simply not enough to distinguish any features from an object. The drawback is that higher frame rates require greater processing power. With this in mind, the best case is to use the slowest frame rate possible while still ensuring accurate object detection from frame to frame. To demonstrate this the following section discusses the ability of pseudo-frames to be used for accurate object detection.

5.4.2 Detection with Pseudo-Frames

Pseudo-frames are a binary image detailing the pixel location of illumination changes in a scene. In effect, they represent an equivalent form of background subtracted and threshold filtered APS frames. With that in mind, by inserting pseudo-frames directly into the frame-based detection and tracking flowchart of Figure 23, one can bypass the background subtraction and threshold filtering stages and move directly to target detection.

Figure 37 shows the outcome of using event-based pseudo-frames as input to the

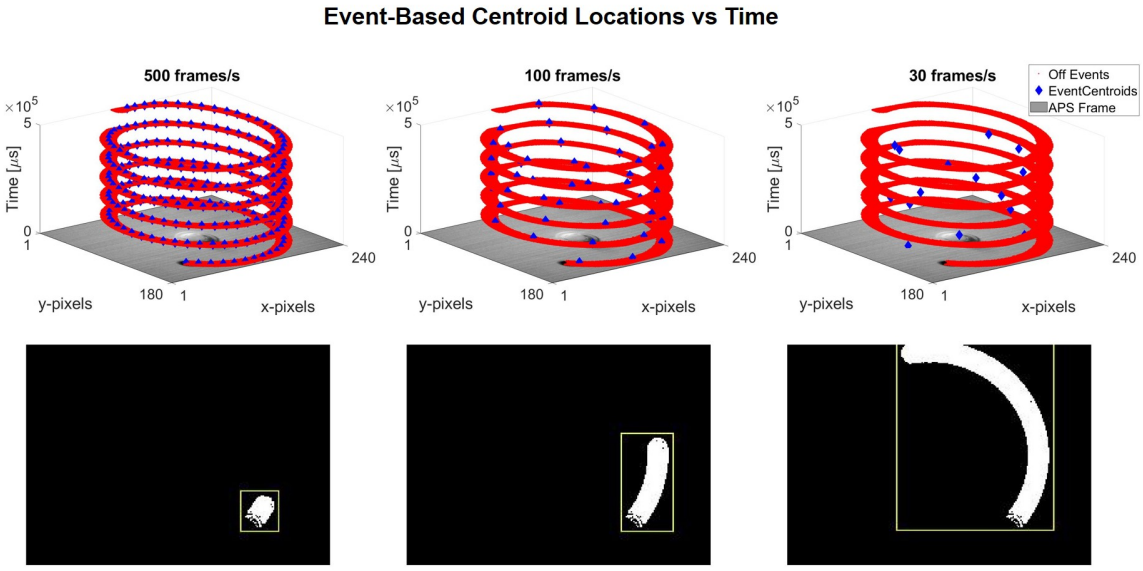


Figure 37: Event-based detection using pseudo-frames from a black dot on a spinning white disc. For various frame rates, the top sequence shows detection centroid locations versus time mapped onto filtered OFF events while and bottom sequence shows example pseudo-frames.

frame-based detection algorithm discussed earlier in Section 4.5. For this example again there is a single black dot painted on a spinning white disc, but in this case the rotational frequency has been increased to approximately 11.4 Hz. Using the same approach of Section 4.5.2, the (x, y) centroid location of each detection has been plotted against time for various pseudo-frame rates. Additionally the filtered OFF events are shown as a form of ground truth. The example pseudo-frames have been passed through the frame-based detection algorithm and consequently show detection bounding boxes. These provide a clear link to how centroid locations for each frame rate are located and mapped to the centroid versus time plots.

This figure highlights the importance of defining a suitable frame rate in order to ensure accurate detections. At 500 fps, while the time bin to generate pseudo-frames is short, it is still long enough to capture the detail of the black dot in a single frame. Upon applying the detection algorithm, accurate detections are made at high

frequency such that they follow the ground truth of raw events very well. Similarly for 100 fps, despite the effect of motion blur in each pseudo-frame, accurate detections can still be made at a sufficient rate to again follow ground truth events. Slowing the pseudo-frame rate further, at 30 fps, it becomes apparent that this approach begins to break down. The motion blur has become too extreme such that centroids begin to shift toward the center of rotation and away from the actual location of the black dot. The centroid location versus time therefore does not follow the event-based ground truth rendering this pseudo-frame rate inadequate for a scene such as this.

This example also highlights the versatility of pseudo-frames in that they can be adjusted through a wide range of frame rates in order to account for the nature of a scene. As demonstrated, generally no matter how fast an object is moving, the frame rate can always be increased such that effective detection, and therefore tracking, can be performed.

Some of the benefits of using pseudo-frames include taking advantage of the small bandwidth requirements of event-based sensors by eliminating the need to process redundant background pixels. Also, pseudo-frame rates are completely tailorable and as such can be adjusted to fit the dynamics of most scenes to ensure similar, if not better, detection performance when compared with conventional frame-based sensors. Despite this however, by grouping events into defined time bins, the asynchronous, microsecond resolution of event-based sensors has not been used to its full potential. For this reason, the true advantage of event-based technology cannot truly be unlocked using this approach. A detection and tracking algorithm that processes event data both independently and asynchronously is therefore highly sought after and discussed in the following section.

5.5 Asynchronous Event-Based Detection and Tracking

Despite its clear challenges, processing events independently and asynchronously is the ideal approach to ensure the full potential of event-based sensors are unlocked for target detection and tracking. This section discusses two different approaches to this problem where further explanation and detail is provided for the most promising.

5.5.1 Adaptive Time Surfaces

To view the independent, asynchronous and continuous output from an event-based sensor presents a significant challenge where to date only two valid methods have been presented in public literature. Discussed previously in Section 5.4.1, pseudo-frames represent the first and simplest method where a fixed time window is used to group and display events on a frame-by-frame basis. While shown to be effective in some scenarios, it is widely accepted that this approach forfeits the high temporal resolution of the sensor. In an attempt to overcome this, a second approach steps away from the concept of a global time window and instead applies an individualized time window to every single event. In this case, events are weighted with a temporally depreciating value typically beginning from 1 for ON or -1 for OFF events before decaying smoothly to zero over a user defined time constant τ. The decay type can be either linear or exponential where a linear approach is the simplest to implement but an exponential approach provides a more bio-inspired result [59]. Essentially, each method has its own advantages and disadvantages.

With this approach, because incoming events impact the visual surface both asynchronously and independently, the main advantage is that the high temporal resolution of the event-based sensor is utilized to full effect. To show the result, Figure 38 shows three variants of this form of time surfacing. The first uses a linear decay, the second an exponential decay and the third an exponential decay with accumulation.

From this result it can be seen that each variant displays the same event stream slightly differently where essentially τ, combined with the decay type determines the duration over which events remain visible and therefore how long they have an impact on the scene. From this point, it is typically application dependant as to which variant to use. For example, using an exponential decay feature extraction techniques such as Feature Extraction with Adaptive Selection Thresholds (FEAST) from [70] have been developed for object classification purposes forming an important building block for real-time machine learning applications.

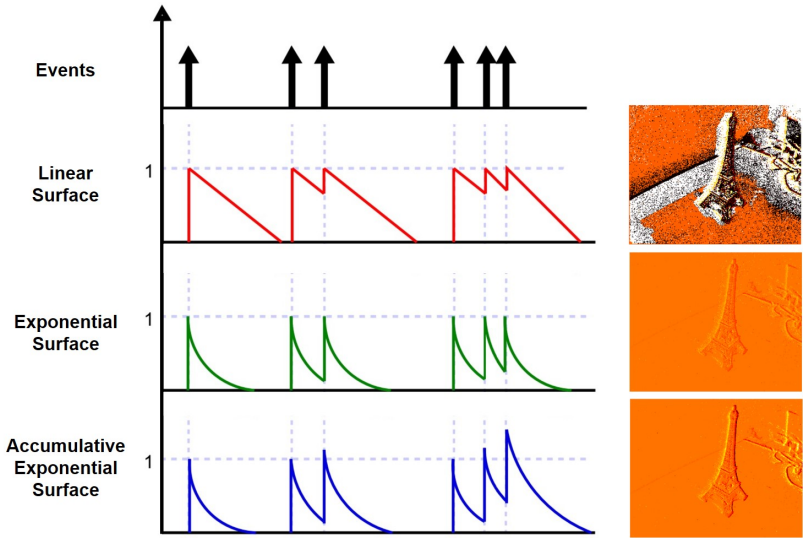


Figure 38: Example of adaptive time surfaces (modified from [71]).

While an adaptive time surface provides a means to display event-based data in an independent, asynchronous and continuous fashion, the result lends itself to feature extraction and object classification for cases where the target's size, orientation and trajectory can be accurately assumed and well understood. For the case of tracking objects where the size, orientation and trajectory are not only unknown, but can also change considerably with time, an adaptive time surface is not likely to be so effective. This is because it fundamentally relies on working in a 2D image plane, albeit a somewhat live image plane that is continuously updating. With that in

mind, morphing well established tracking algorithms from frame-based sensors to adaptive time surfaces remains a very difficult challenge due largely to the absence of a regular clock (i.e. frame rate). The issue of when to make a determination on the spatial location of a target is not well understood with this approach. Therefore because of this limitation, adaptive time surfaces although not tied to a traditional framed approach, have not been considered further for this research. Instead a novel approach considering 2-dimensional space with a third dimension, time, is suggested that takes advantage of associating events in a 3D point cloud to form tracks on an event-by-event basis.

5.5.2 Event Association in 3D Point Clouds

While more akin to a conventional frame-based approach, adaptive time surfaces provide arguably the most intuitive technique for event-based detection and tracking, yet for various reasons previously established, they considerably limit the ability to detect and track unknown targets travelling in unknown ways. Stepping away from this silo of thought by instead considering event-based data as a 3D point cloud, the problem of object detection and tracking likens itself more to curve detection in 3D space. Therefore, not only can comparatively effective object detection and tracking be achieved with this approach, but the full potential of the temporal resolution of the event-based sensor can be unlocked at the same time.

Curve detection in 3D point clouds is a relatively common problem that arises in many applications for a wide variety of sensor data. Some example use cases include curb detection and tracking from 3D Light Detection and Ranging (LIDAR) data [72], ball tracking in sport events [73] and the identification of particle tracks in Time Projection Chambers [74]. From these examples, and in general, 3D curve detection algorithms pursue three main aims in no particular order:

(a) Distinguish points that belong to a curve and those that are noise;

(b) Partition valid points into sets representing different curves; and

(c) Provide a description of each curve in parametric form (i.e. line segment).

With a parametric approach, curve detection starts from point (c) where a pre-defined parametric shape is used to describe a curve made up of many points in 3D space. Because the shape of the curve must be known or at least assumed prior, it leads to a voting scheme that can be achieved using a random sample of events [75] or more exhaustively for all points using a method such as the Hough Transform [76].

While in its generalized form the Hough Transform is designed to locate lines in 2D images, since being patented in 1962 it has found uses for many other shapes, in addition to being extended to 3D space. One of the first examples where the Hough Transform was applied to 3D point clouds was in [76] where data points captured from an Active Target Time Projection Chamber were associated to the straight line motion of radioactive particles. Influenced by this approach, the Hough Transform was then later applied to event-based data in [77] for the tracking of stars in Space Situational Awareness (SSA) applications. Because of the straight line motion of stars across the sky, event-based recordings of this motion make the ideal scenario to apply the 3D Hough Transform, which has consequently been achieved with great success. Unfortunately however, as this approach relies heavily on an object moving in a straight line, it will quickly breaks down when that is not the case. While this could be acceptable for an aircraft transiting straight through the field of view (FOV) of an event-based sensors, to generalize the tracking algorithm a more robust, non-parametric approach is needed.

To construct a generalized 3D point cloud tracking algorithm, the focus must shift to addressing only points (a) and (b) above while simply ignoring (c). The problem

is thus to partition events into a unknown number of possibly overlapping clusters to form curves, with an additional dislocated cluster representing noise. As a first step to solving this problem, the idea of grouping data points into triplets in 2D space was first suggested by Lezama et al. in [78]. Although the algorithm could be generalized to 3D space, it was optimized only for approximately equidistant points without much random perpendicular spread from the actual curve. As this approach does not hold in real-world 3D data, a change was suggested by Dalitz et al. in [68] where instead of looking for symmetric triplets, they instead looked for approximately collinear branches in addition to grouping triplets with a single link hierarchical clustering approach that uses an appropriately defined triplet distance measure. In doing so they produced the algorithm TriplClust, which due to its significance is detailed in the following section.

5.6 TriplClust Algorithm

Originally developed for particle track detection in Active Target Time Projection Chambers, TriplClust can be generalized to perform detection association in any 3D point cloud, including (x, y, t) point clouds generated by filtered event-based data. In that sense, it therefore provides a unique enabler for the low bandwidth detection capability of event-based sensors to perform association on an event-by-event basis, essentially allowing object tracks to be built with a microsecond temporal resolution.

5.6.1 TriplClust Description

The TriplClust algorithm operates by taking in points and grouping them into clusters representing either curves or noise. In a nutshell, the algorithm consists of the following four steps:

1. Smoothing by position averaging of neighboring points.

2. Using smoothed points to find triplets that are approximately collinear.

3. Single link hierarchical clustering of triplets.

4. Pruning by removal of small clusters and (optionally) by splitting clusters with large gaps.

With these four steps, each event within the point cloud is given the appropriate cluster label of the triplet to which it has been assigned in step (2). Should points not be assigned to a triplet or only be assigned to triplets that do not belong to clusters, then they will be labelled as noise. Additionally, as a single event can sometimes represent the interception of more than one target, that event can belong to more than one triplet. Resulting clusters can therefore overlap such that some events can be allocated to more than one cluster label.

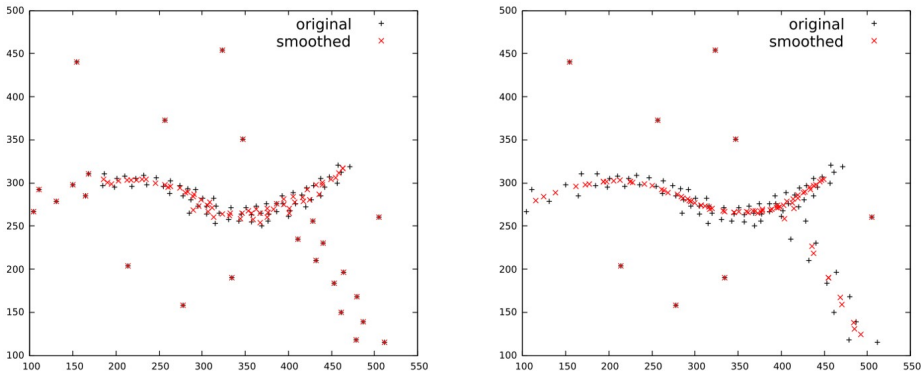
While a detailed description of the individual steps and the specific meaning of input parameters can be found in [68], concepts relevant to the adoption of this algorithm to event-based data are discussed in the following subsections.

5.6.1.1 Position Smoothing

A spread of events perpendicular to an object's direction of motion can have detrimental effects on the formation of triplets. Essentially, triplets formed by adjacent points can create branch angles that do not align with the object's motion. It is therefore desirable to reduce the spread of events around a main curve with the use of a smoothing operator.

Within TriplClust, the smoothing operator temporarily replaces the coordinates of each point with the mean value of all points in its immediate neighborhood before conducting future triplet operations. In this case, one point is considered to belong to the neighborhood of another point if the distance between those points is less than a

user defined threshold r_{smooth}. Figure 39 shows the effect of this smoothing operation with two different values of r_{smooth}.



(a) $r_{smooth} = 15$ (b) $r_{smooth} = 50$

Figure 39: 2D example of the effect of position smoothing with two different smoothing constants r_{smooth} [68].

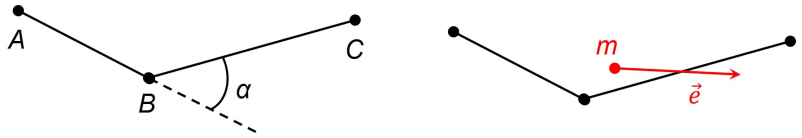
In this example it can be seen that for regions of low point density, the position of single points with no neighbors remains unchanged. Alternatively, for regions with high point density, several points fall within a points neighborhood, therefore influencing the position of points close to a curve by moving them closer to the curve's center axis.

The difference in Figure 39 (a) and (b) highlights the importance of choosing an appropriate r_{smooth}. In (a), r_{smooth} is too small causing insufficient smoothing where many points have not been smoothed at all, and for those that have, they do not converge well to a single curve. In (b), while a higher r_{smooth} causes more points to move to a curve's center axis, the downside is that curves are made shorter at the ends and that multiple curves can be fused at their vertices. Choosing an appropriate r_{smooth} is therefore vital in getting valid output from this algorithm where for the case of event-based data, r_{smooth} must be chosen to align with the broadest spatial dimension perpendicular to an object's direction of motion (i.e. an aircraft's wingspan in the sensor's FOV in units of pixels).

5.6.1.2 Triplet Grouping

The next step in the algorithm consists of taking smoothed points and building groups of three approximately collinear points called triplets. As an example, let the smoothed points A, B and C make up the triplet shown in Figure 40. From these points, a triplet is validated by taking the cosine of the angle α between each triplet branch and comparing it to a user defined threshold $a_{triplet}$ as shown in (7). For completeness, the variables m and $\vec{e}$ represent the midpoint and direction of the triplet which are later used as descriptors in the hierarchical clustering step that follows.

$$\cos(\alpha) = \frac{\langle \overline{AB}, \overline{BC} \rangle}{\|\overline{AB}\| \cdot \|\overline{BC}\|} < 1 - a_{triplet} \tag{7}$$



(a) Nodes A, B, C and angle α (b) Midpoint m and direction $\vec{e}$

Figure 40: Schematic description of triplet properties [68].

In running this algorithm, triplets are computed from a smoothed point cloud in a loop over all points. Essentially, each point is considered as a possible midpoint B while all points among its $k_{triplet}$ nearest neighbors are tried as candidate points A and C. A triplet is discarded if it meets the condition of (7) where in the context of event-based sensing, this essentially limits the maximum acceptable turning rate of an object. When an object travels in a straight line triplet points A, B and C will align such that the angle α will approach zero. When an object undergoes a turning maneuver however, the points A, B and C no longer align whereby α becomes larger and larger. In the case of an airborne object moving through space, it therefore makes

sense to set $a_{triplet}$ in such a way that it encapsulates the maximum anticipated turning rate of the object to be tracked.

5.6.1.3 Input Parameters

There is a number of input parameters that play an important role in the successful application of TriplClust. These parameters are explained briefly below with context provided as to their influence on real-world event-based data.

Characteristic Length (d_{NN}). The 3-dimensional distance between neighboring points changes depending on the spatial units and resolution of points that make up a data set. It is therefore useful to define a characteristic length that can be used as an indicator for the spread of points around the middle axis of a curve. To calculate this length, the distance to the nearest neighbor for each point is computed where the first quartile of this distance defines the characteristic length d_{NN}. The first quartile is chosen because typically noise points have a greater distance than valid points on a curve.

Smoothing Radius (r_{smooth}). As previously discussed, neighborhood smoothing has the effect of moving data points towards the medial axis of a curve. In most cases, a moving object will generate a spread of events perpendicular to its direction of motion due simply to its size in the sensor's FOV. To smooth this effect, it is therefore best to set r_{smooth} to the size of that spread where for TriplClust, the default setting is $2d_{NN}$.

Triplet Thresholds ($a_{triplet}$, $k_{triplet}$, $n_{triplet}$). These parameters control which groups of three points are acceptable triplets possibly belonging to the same curve. The default value $a_{triplet} = 0.03$ corresponds to a threshold triplet angle of $|\alpha| = 14.07°$

in accordance with (7). Depending on the density and spread of points around a central curve, a threshold value of $|\alpha| = 14.07°$ might rule out triplets of adjacent points. To compensate, the parameter $k_{triplet}$ has been introduced to allow for a wider spread of smoothed points to be considered in building a triplet as shown in Figure 41. Triplets with a greater separation subtend a smaller angle α and are therefore more likely to be accepted as a triplet. The default value of $k_{triplet}$ is 19.

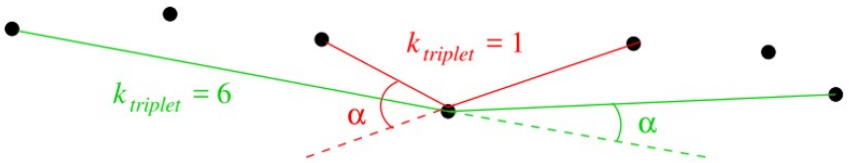


Figure 41: Schematic demonstration of the influence of $k_{triplet}$ on the determination of α [68].

For performance reasons, $n_{triplet}$ was introduced to limit the maximum number of triplets that can be matched to a single midpoint. Its default value is 2.

Clustering Control ($s_{cluster}$, $t_{cluster}$, d_{max}). The parameter $s_{cluster}$ controls the weight of the spatial distance with respect to the directional difference between two triplets and must be scaled by d_{NN}. The default value for $s_{cluster}$ is $d_{NN}/3$. The cutoff threshold $t_{cluster}$ sets a limit on the maximum distance between triplet pairs to form clusters while d_{max} sets a limit on the maximum translational gap within a cluster. A large d_{max} enables large gaps in curves to be overcome provided continuation of triplets after a gap forms an extrapolation of the curve before the gap.

5.6.2 Application of TriplClust

While TriplClust was originally developed with other applications in mind, the potential translation to event-based data is significant. If raw event-based data can

be pre-processed and presented to TriplClust, than it can be shown to associate a stream of events to particular objects, effectively performing single and multi-target tracking on an event-by-event basis.

Raw event-based data is typically very noisy where a significant percentage of spurious events can be attributed to noise within the sensor itself. Presenting this raw event-based data to TriplClust would therefore result in the requirement for significant computing power to process an exhaustive number of triplets, most of which would be irrelevant while any valid triplets would likely be corrupted with noise. Because of this, it makes sense to filter event-based data appropriately before parsing to TriplClust.

For event-based target detection and tracking, in general it is best to capture the motion of a moving target with as few events as possible. For the presentation of events to TriplClust, this is no exception. In order to reduce a stream of raw events such that it can be presented to TriplClust, filtering must first take place. While simply separating event polarity (Section 5.3.3) and applying the nearest neighbor filter (Section 5.3.2) was successful in some cases, it was found experimentally that in the majority of cases the event pair filter (Section 5.3.4) paired with the nearest neighbor and polarity filter proved most effective.

In the first instance, the event pair filter combined with the nearest neighbor filter reduces events generated by object motion to two somewhat identical planes of events with opposite polarity. For clarity, an example of these planes is shown in Figure 42 where a zoomed-in section of the banking drone event data is shown after the event pair and nearest neighbor filters have been applied. This figure shows a layer of OFF events (red dots) underneath a layer of ON events (green dots) where their vertical separation is temporal indicating that this was a negative contrast object.

To the human eye, a stream of events like that shown earlier in Figure 34(c)

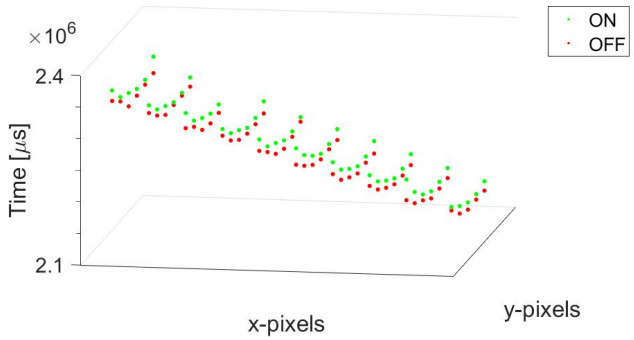
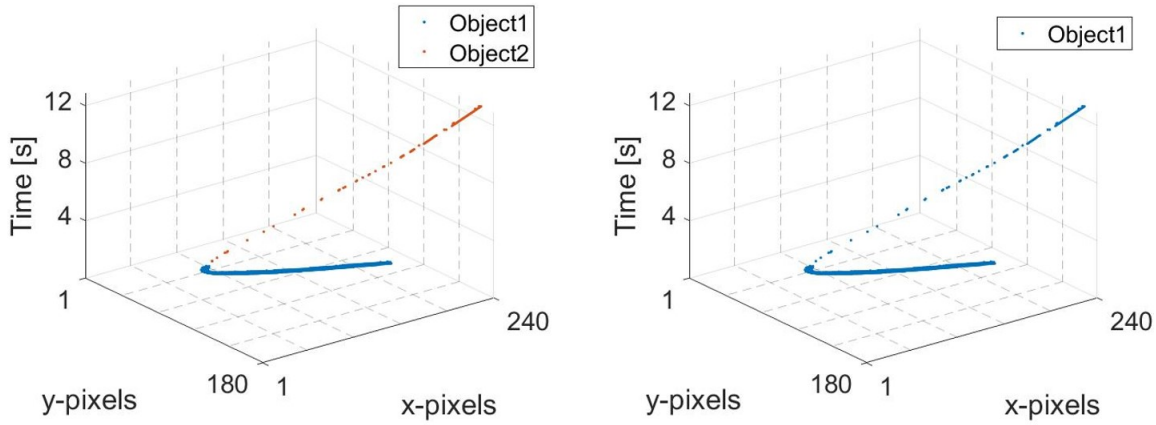


Figure 42: The output of the event pair filters can produce two almost identical layers of events with opposite polarity.

clearly shows the path taken by the drone. For input into TriplClust however, due to its insensitivity to polarity, presenting two identical planes will only generate additional complexity with no added benefit. While the initial smoothing stage can be completed effectively, the issue arises at triplet grouping. Not only does the number of possible triplets require significant processing, but due to the close proximity of smoothed points across planes, in many cases the user defined triplet angle α can be easily breached, causing TriplClust to allocate a large majority of events to noise. To overcome this issue it is simply a matter of separating the event polarity and presenting only a single plane of ON or OFF events to TriplClust. This way the total number of events is essentially halved and triplets are only calculated from smoothed points created along a single plane.

In revisiting the banking drone event data again, raw events were passed through the event pair and nearest neighbor before taking OFF events only and passing them through TriplClust. For two different $a_{triplet}$ values, Figure 43 shows a comparison of the event association performed by TriplClust. All other parameters were kept at their default values.

From these results it can be seen in (a) that when $a_{triplet}$ is low, the association of events through a tight turn breaks down resulting in the creation of two separate



(a) $a_{triplet} = 0.03$ (b) $a_{triplet} = 0.1$

Figure 43: Output of TriplClust acting on banking drone event data. In (a) $a_{triplet}$ is too low to handle the drones change in direction and so creates two separate tracks.

object tracks. When $a_{triplet}$ is increased to accommodate for a significant change of velocity, triplets with higher α are now accepted leading to a continuous stream of events all associated to the same object as shown in (b).

In two other examples, Figure 44 (a) and (b) show a series of plots moving from raw events data, to filtered data and finally to the association of events to object tracks using TriplClust. The events from (a) were captured from a single small SS40 quadcopter with flashing LEDs flying at night in and out of the sensor's FOV. The events from (b) are of the Skywalker x8 drone flying during the day with the addition of a small birds passing through the FOV at the same time. These sequences of plots not only highlight the remarkable effect that filtering has on drawing object motion out, but also the benefit of TriplClust in allocating each event to specific object tracks or noise. In (a) it is worth noting that over a 25-second recording, 92 million raw events were reduced to just 1,600 to produce these object tracks.

In moving from raw events to filtered events for Figure 44 (a), the event pair and nearest neighbor filters were used in a similar fashion to that discussed in Section 5.3.4 before separating only OFF events. This produced a clean event stream containing

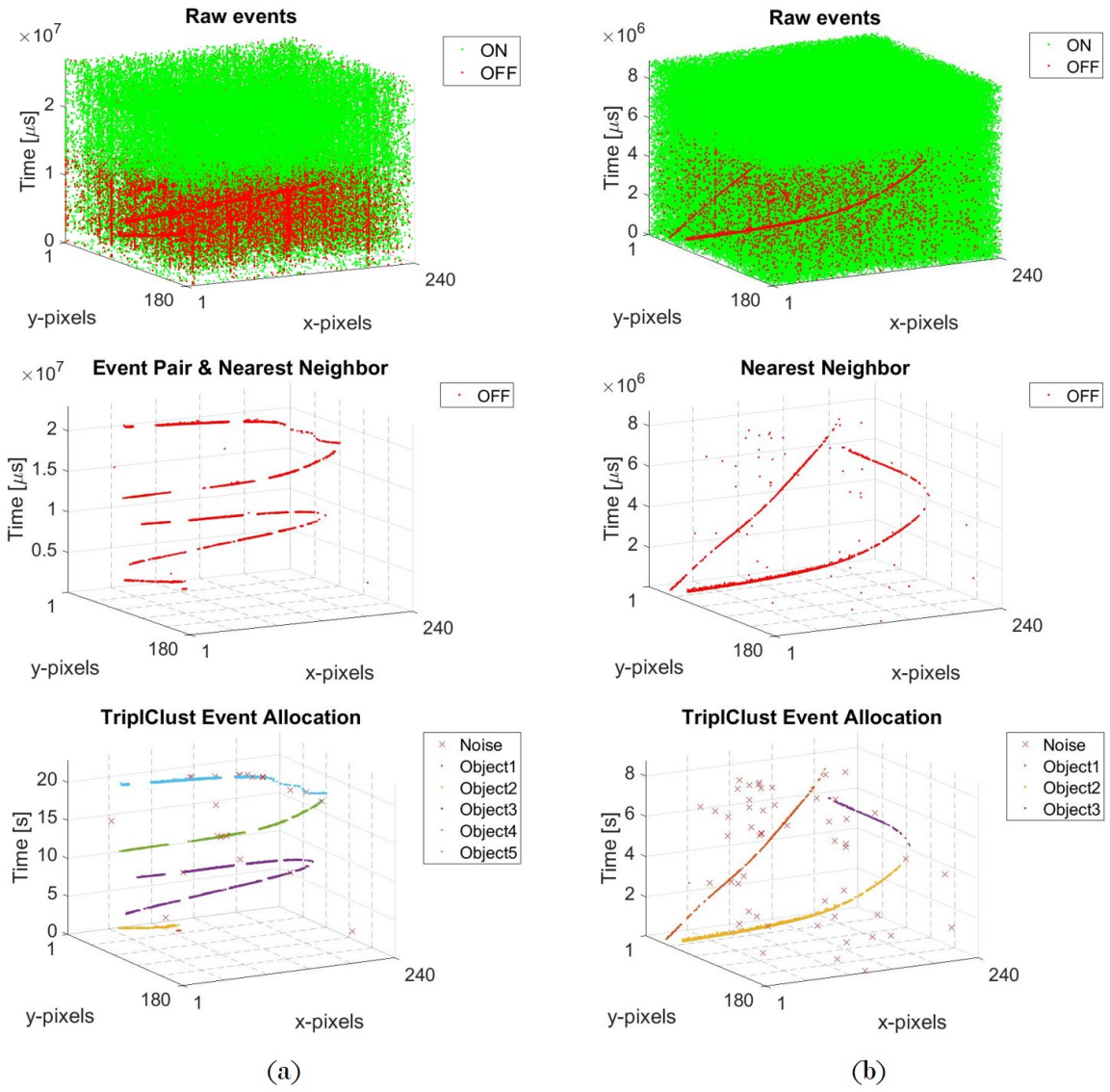


Figure 44: Example output of TriplClust acting on (a) an airborne quadcopter flying at night with a flashing LED and (b) an airborne drone flying during the day with a bird also flying through the FOV.

only the motion of the SS40 drone to be used as input to the TriplClust algorithm.

From the output of TriplClust, despite there being only one drone the identification of multiple objects is due simply to the drone moving in and out of the sensor's FOV. Because of this, it cannot be expected that TriplClust will associate events through such a break in continuity. Therefore, five separate objects have been de-

tected with the first ('Object1') being a very small group of events caused by the drone's LED being on only for a short period at the very beginning of the recording. Objects 2, 4 and 5 in yellow, green and blue respectively show a single pass of the drone where in each case the drone moves out of the sensor's FOV before re-entering sometime later. For Object3 in purple however, the drone remains in the sensor's FOV as it moves from left to right to left again. Impressively, TriplClust retains the association of events for this single track.

Another point to note is that flying the drone at night with a flashing LED results in the event-based sensor only generating events when the LED is on and the drone is moving. For that reason, the filtered data resembles that of a dashed line moving through space where the gaps between event streams are the result of the drone's LED being off, essentially flying undetected by the sensor. Fortunately, the application of TriplClust is able to link these gaps when d_{max} is unrestricted (i.e. set to 'none') such that as long as a group of events is positioned along a continuous curve a triplet will be formed that will enable them to be associated accordingly. The result in the final plot demonstrates this effect for all object tracks.

For the case of Figure 44 (b), raw events were filtered using only the nearest neighbor filter before separating OFF events. Through trial and error it was discovered that by using this filter only, events would not be over-filtered and a more continuous stream of events could be retained to capture the motion of the drone and bird. This became particularly important at the point where the drone began to bank directly towards the sensor. The drawback however was that this approach retained additional unfiltered noise.

With the use of TriplClust, the filtered events were then segmented into object tracks as shown in the final figure of series (b). In this case the bird that transits the FOV is captured as Object1. Simultaneously the banking drone is captured as

Objects 2 and 3. For the case of the drone, the reason two separate objects have been identified is due partially to the drone's change in direction causing a breach of the acceptable triplet angle α, but perhaps more so due simply to a limitation of the sensor itself. In this particular example, as the drone begins to bank it starts to track directly towards the sensor. Because of this, pixels on target see no change in log-light intensity for a period of time before the drone's direction of motion begins to move off boresight. The result is that while the drone is tracking directly towards the sensor, no events are generated leading to a significant gap. Unlike the previous example in (a), TriplClust cannot bridge this gap because the drone's change in direction no longer represents a continuous curve within the bounds of the algorithm.

Further to the association of events to object tracks, the effective allocation of disparate events to noise within TriplClust is also noteworthy and an important component of this algorithm. In (b), the use of the nearest neighbor filter left behind a substantial number of uncorrelated residual events that could not be combined in any way to form a continuous 3D curve. In that sense, they were aptly partitioned as noise and could be simply removed if so desired.

While Figures 43 and 44 demonstrate the effectiveness of TriplClust in three relatively simple examples, due to the unique and complex nature of event-based data there are some specific scenarios that render TriplClust less effective. One shortfall arises when a moving object covers a large number of pixels (i.e. large target at range or small target up close). In this case the first concern is the significant increase of event numbers and therefore possible triplet pairs requiring additional processing power. More importantly however, is that in order to associate events from a large target covering a significant portion of the FOV, r_{smooth} must be set suitably high. While this ensures optimum smoothing, the drawback is that any smaller targets could be grouped together, in effect limiting the resolution of the algorithm. Addi-

tionally, uncorrelated noise events are more likely to be encompassed by r_{smooth} and considered in viable object clusters. With that in mind, TriplClust works best with event-based data when objects cover only a few pixels.

Another issue linked to target size is the influence of sensor resolution. With future generations of event-based sensors expected to have improved resolution, there will consequently be an equivalent increase in the total number of events. In regards to asynchronous detection and tracking using TriplClust, the impact will be that for the same size object, more events will need processing into triplets and clusters, introducing the need for additional processing power. Not dissimilar to frame-based sensors, this issue becomes a typical engineering problem whereby increased resolution offers the ability to detect and track small targets. The drawback however is that for large targets the increase in event counts introduces significant processing overhead with little discernible difference in terms of tracking the spatial location of a moving object. The choice of sensor resolution must therefore be application specific.

Despite this shortfall, when used correctly and within the bounds of its capability, TriplClust exhibits significant potential in its current state as highlighted by the previous few examples. If the algorithm were tailored specifically for event-based sensors however, its usability would only be amplified. One simple adjustment could be treating event pairs within a defined temporal separation as a single point. This would simplify event filtering and improve the algorithm's ability to associate event pairs to a single object. Perhaps a more significant improvement could be integrating TriplClust into a real-time system. Currently the algorithm is written to operate on a complete data set and must therefore be applied in post-processing. Because the aforementioned event-based filters have been written in such way that events can be filtered as they arrive, those events could be passed to TriplClust in real-time. It therefore makes sense to develop TriplClust in such a way that it can perform smooth-

ing, calculate relevant triplets and allocate filtered events to identified object clusters all in real-time. Of course in this case, the measure of 'real-time' is dictated by the input parameters, in particular r_{smooth} and $k_{triplet}$, as they influence the number of past and future events required for triplet calculations, clustering and pruning. Despite this, in the military environment the practical relevance of a real-time detection and tracking system using an event-based sensor is significant.

5.7 Closing Remarks

Within this chapter the process of event-based target detection and tracking has been explored. The value of effective filtering was first discussed and demonstrated in terms of its importance to partitioning only those events generated by object motion. It was then established that object detection and tracking using event-based sensors is a unique and challenging problem whose solution is not restricted to just one approach. Despite being effective in some scenarios, it was established that pseudo-frames forfeited the asynchronous advantage of event-based sensors. Stepping away from a standard frame-based approach, this issue was overcome through the use of TriplClust where in effect, the backbone of an optical tracker with microsecond temporal resolution and comparatively low bandwidth requirement was suggested.

VI. Compare Frame and Event-Based Target Tracking

6.1 Preamble

In this chapter, a select series of scenarios has been chosen to highlight differences in the detection and tracking capabilities of the frame, pseudo-frame and asynchronous event-based approaches discussed above. Initial comparisons were made using laboratory-based scenarios with a simple swinging pendulum, a black dot on a spinning white disc and moving tennis balls. Comparisons were then made in more realistic scenarios with a series of small airborne drones flown in both day and night conditions. In each case the outcome is analyzed and discussed in regards to the advantages and disadvantages of each approach to specific scenarios.

6.2 Comparison

In reference to the frame and event-based detection and tracking flowcharts of Figures 23 and 29, while there is overlap between individual steps, on their own each approach is uniquely different. To compare these approaches, scenarios have been chosen to challenge sensor differences in dynamic range and temporal resolution, in addition to the fundamental operation of each sensor. To do so, as individual steps have been discussed in previous chapters, this comparison only considers application of the entire process for each approach and performs an assessment on their overall detection and tracking capability. Note that in order to conduct a fair comparison in the detection association and target tracking steps for each approach, TriplClust has been used in conjunction with detection centroid locations captured from frame and pseudo-frame based data in addition to applying it to filtered asynchronous event-based data.

6.2.1 Swinging Pendulum

It has been established that the dynamic range of an event-based sensor (∼120 dB) is much higher than its frame-based counterpart (∼60 dB). For target detection and tracking this implies an event-based sensor has the potential to detect targets that transit regions of extreme brightness or darkness. Essentially, where a frame-based sensor would normally be over or under exposed, an event-based sensor is expected to function within its operational bounds.

From a practical point of view, nowhere is the impact of limited dynamic range more cumbersome than when a target transits between the sun and the sensor. While the exposure times for localized regions of a frame-based sensor can be implemented, traditionally, frame-based sensors have struggled with this scenario as the sun simply saturates the sensor. To overcome this issue, an event-based sensor presents an excellent alternative.

In an attempt to emulate the performance of both sensor types to detect and track a target moving through the sun, a simple laboratory experiment was conducted. Using a pendulum constructed from clear fishing line and a small weight suspended from the ceiling, the weight was aligned to swing between the DAVIS 240C and an integrating sphere set up to mimic the sun by emitting light intensity up to 10,000 lux. Detection and tracking of the pendulum's weight was then attempted using both sensor types and the techniques described in Chapters IV and V.

In the first instance, a baseline was set whereby detection and tracking was performed using all three approaches with the integrating sphere off. This was a calibration exercise used to test the detection and tracking capability of each approach in the simplest case. The results are captured in Figure 45.

With a frame-based approach using background subtraction and threshold filtering, the detected centroids are plotted in (a). At 20 fps, this approached produced

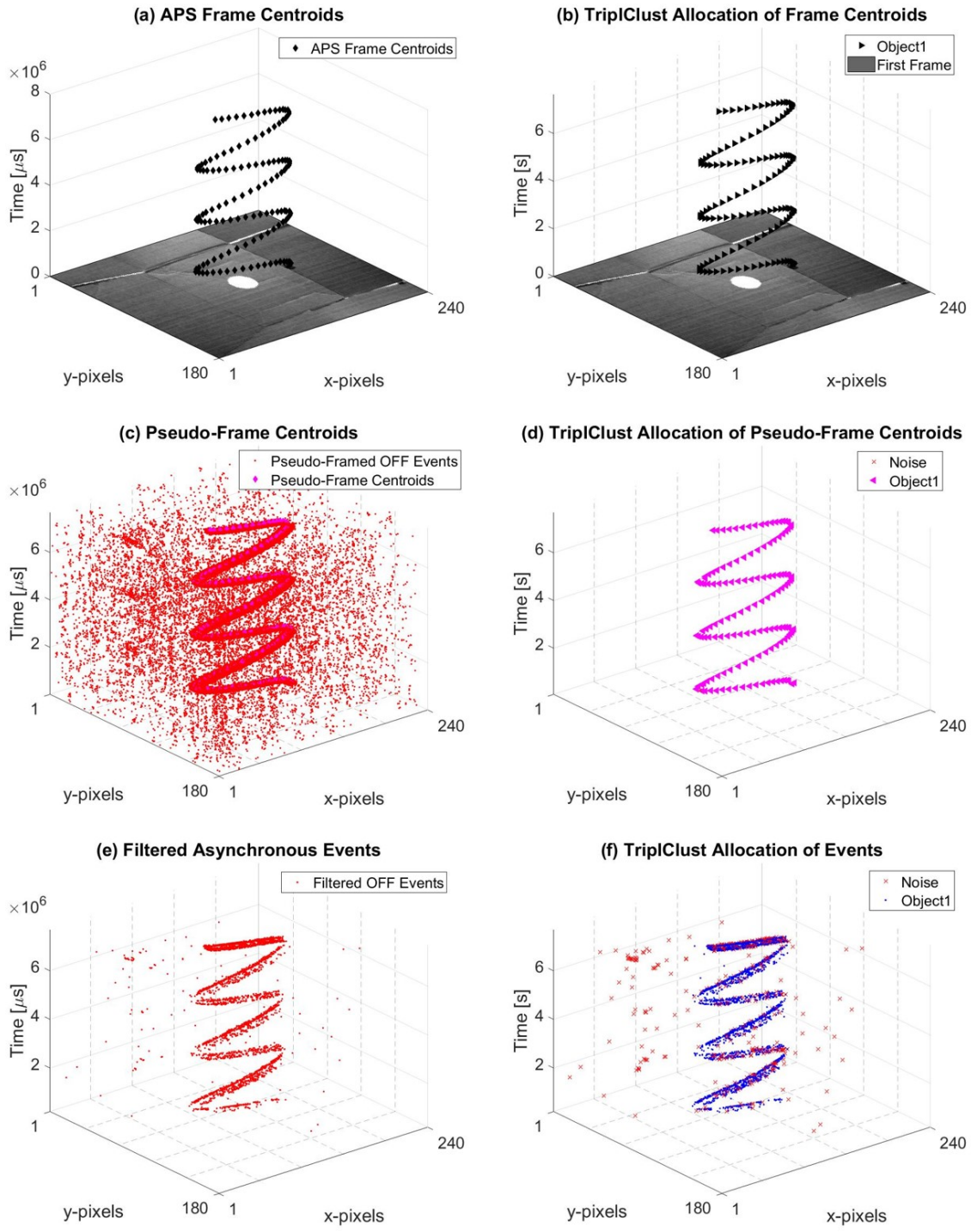


Figure 45: Tracking demonstration of a swinging pendulum.

136 detections that essentially capture the continuous path of the pendulum with no noticeable noise. When parsing these points through TriplClust as shown in (b), the

algorithm has no trouble associating all points to the same object indicating successful target detection and tracking using only frames.

Moving to the use of pseudo-frames, in this case raw events were filtered using the refractory filter and nearest neighbor filters only. This was because while the event pair filter is generally considered more effective at cleaning up event data, in order to produce pseudo-frames a sufficient number of events is required. Hence the rather noisy point cloud shown in (c). Also shown in (c) are the 144 detection centroids determined using 20 pseudo frames-per-second with the same frame-based techniques discussed earlier. Again, when parsing these centroids through TriplClust the algorithm had no trouble associating all detection as shown in (d).

When using events asynchronously, more rigorous filtering is required to minimize the event count while still retaining the motion of the pendulum. In this case, both the event pair and nearest neighbor filters were used leaving behind 3,097 of 274,521 events as shown in (e). In (f), TriplClust has then successfully associated relevant events to the motion of the pendulum while accurately partitioning noise.

From this example it can be seen that each approach is capable of tracking the swinging pendulum where Figure 46 has been provided to validate their accuracy and highlight the comparison between each method. Here it can be seen that there is no discernible difference in the path of the swinging pendulum across each approach.

In a much more challenging scenario, the integrating sphere was next setup to emit 1,000 lux which for the frame-based sensor created a large saturated spot in the center of the frame that the pendulum would pass through during each period. As shown in Figure 47 (a), this caused a gap in frame-based detections when the pendulum passed through the bright spot that translated to the TriplClust association shown in (b). Fortunately TriplClust was able to bridge the detection gap and generate one main object track with the need to separate only a few noisy detections.

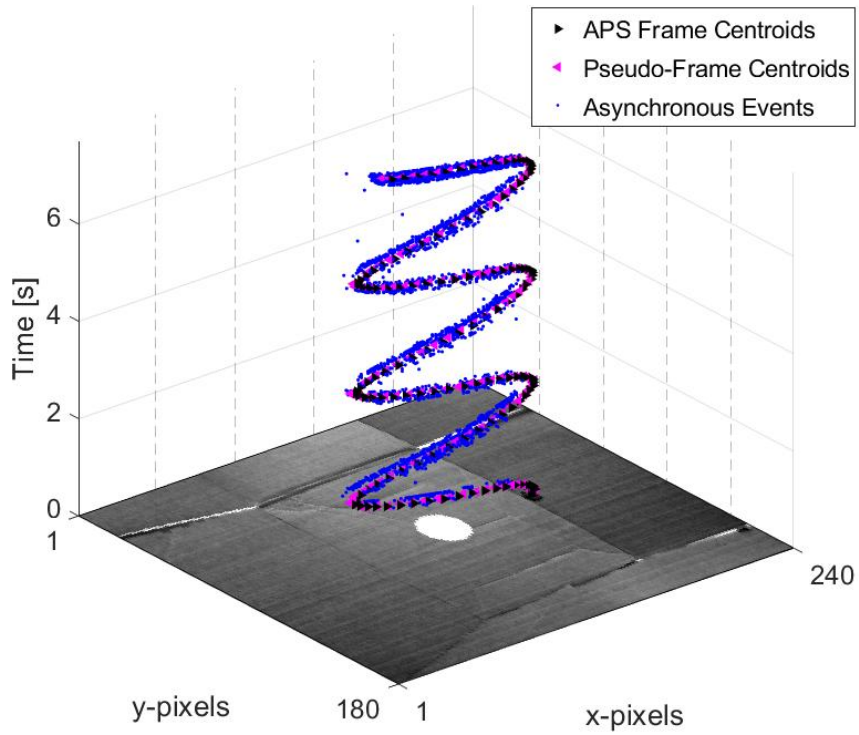


Figure 46: Tracking comparison for a swinging pendulum.

Using pseudo-frames for the same scenario, events were first filtered using the event pair and nearest neighbor filters before binning them into frames and running frame-based detection algorithms to produce (c). As can be seen, due to the high dynamic range of the event-based sensor the stream of events, and therefore detections, remain continuous while the pendulum passes through the path of the bright integrating sphere. This then translates to (d) where TriplClust has associated those detections to accurately capture the motion of the pendulum. The reason the object track is not as smooth as Figure 45 (d) in the previous example is due to the fishing line being backlit by the integrating sphere. This caused the line to generate events which made the target size larger and pushed the detection centroid up. When the pendulum passed through the bright spot of the integrating sphere the effect was less dramatic

and so the detection centroid was lowered to align more with actual position of the pendulum weight.

In using events asynchronously, (e) shows the result of filtering raw events using three application of the event pair filter interlaced with the nearest neighbor filter before separating only OFF events. This resulted in the reduction of 251,306 events to 8,514 events that were then associated using TriplClust as shown in (f). In this case the effect of the backlit fishing line is more evident where a notch in the stream of events is visible as the pendulum passes the center of its swing. Nevertheless, TriplClust has associated the filtered event stream to create a continuous object track that accurately captures the motion of the pendulum through the bright region created by the integrating sphere.

In comparing the output of each approach as shown in Figure 48, the limited dynamic range of the frame-based sensor caused detection gaps as the pendulum passed through the saturated region created by the integrating sphere. On the other hand, for both event-based methods the pendulum was detected continuously irrespective of the saturated region. When considering the alignment of the centroid locations from frames and pseudo-frames the slight offset was caused by the backlit fishing line, an unfortunate byproduct of the experimental setup and one that would not occur in a real-world scenario with an airborne aircraft for example.

This simple experiment has demonstrated the advantage that a sensor's high dynamic range offers to a detection and tracking scenario. In this case, the event-based sensor was able to track an object continuously where as the frame-based sensor was hindered by saturation. When considering application to the military environment, being able to utilise the high dynamic range of the event-based sensor in conjunction with pseudo-frames or asynchronous-events implies the detection and tracking of an airborne target could be achieved irrespective of the sun position or intensity. Ad-

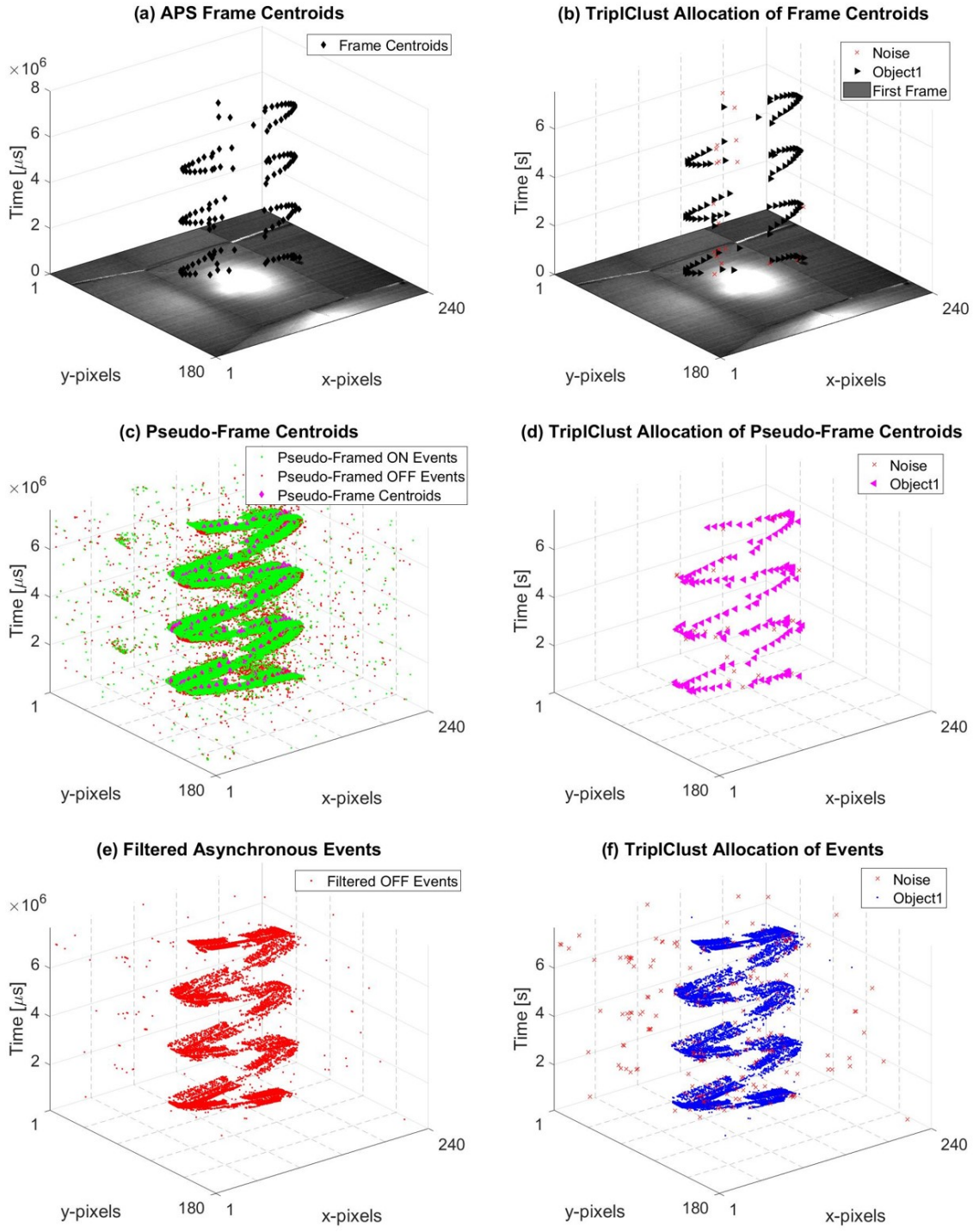


Figure 47: Tracking demonstration of a swinging pendulum (bright background).

ditionally, if in the future, infrared event-based sensors are available this too could apply to counteracting the deployment of flares.

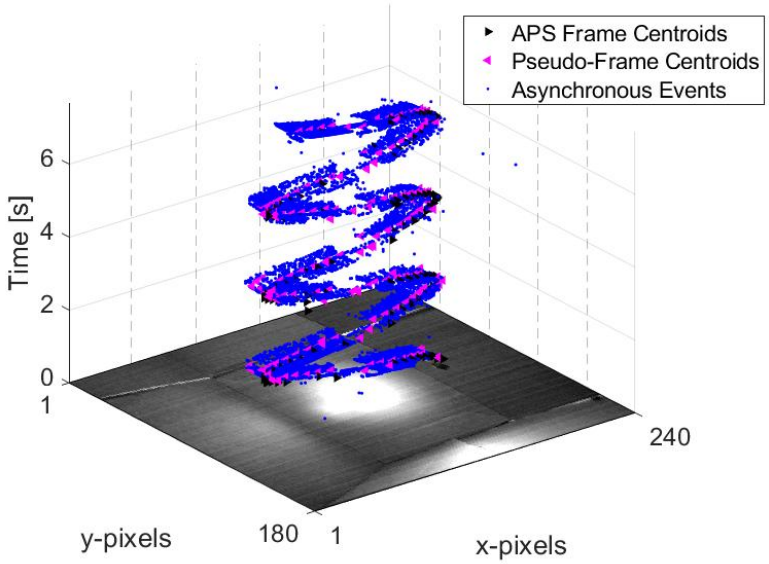


Figure 48: Tracking comparison for a swinging pendulum (bright background). Note the gap in frame-based detections when the pendulum crosses the bright point in the background.

6.2.2 Spinning Dot

In this comparison the influence of temporal resolution for each sensor type was assessed. To do so, an experiment was set up using a black dot on a white disc attached to a pedestal fan set to a fan-speed of low (∼8.53 Hz). Recordings were taken with each sensor type before parsing through relevant detection and tracking algorithms with the results captured in Figure 49. Of note, in this case the frame-based and event-based data were not captured in the same recording as per all other comparisons in this report. Due to excessive active pixel sensor (APS) coupling, a combined recording made event-based data too noisy to filter effectively, hence a stand alone event-based recording was captured ensuring far cleaner results. Due to the repetitive nature of the spinning dot, applying detection and tracking algorithms for each sensor type to different data sets was considered valid.

With the frame-based approach, a frame rate of 20 Hz resulted in the black dot be-

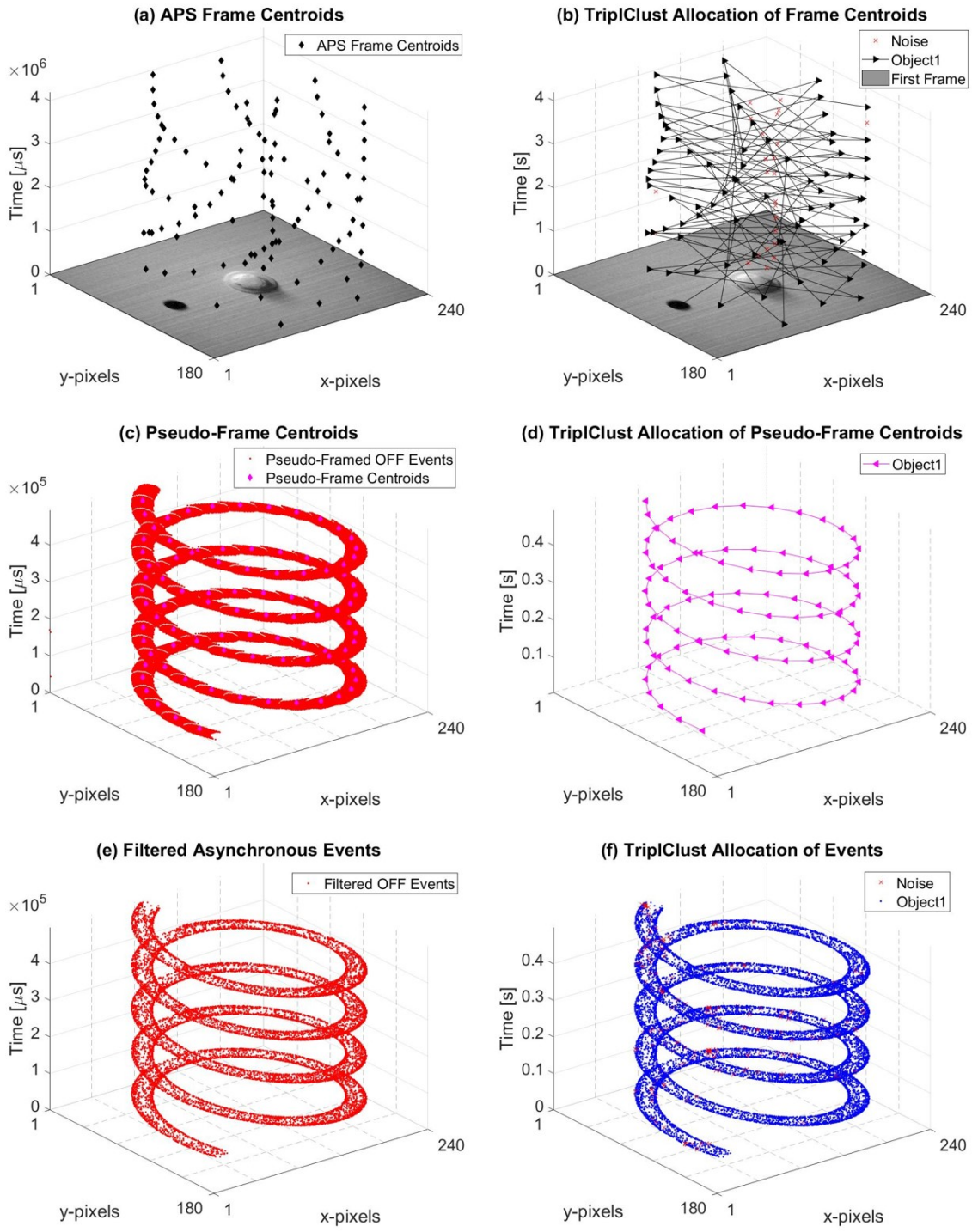


Figure 49: Demonstration of tracking a dot on a spinning disc.

ing detected only 2.34 times per rotation. While background subtraction and threshold filtering resulted in accurate detections as shown in (a), the low sampling rate

did not enable accurate tracking of the dot. Ignoring noisy detections generated from the central fan hub, TriplClust associated most detections to the spinning dot where connecting lines have been used in (b) to highlight the poor track this approach has produced.

In a pseudo-frame approach, the raw event count was reduced from 257,219 to 34,890 by first using the refractory, nearest neighbor and polarity filter. These events were then binned into pseudo-frames where as discussed in Section 5.4, the frame rate is completely tailorable to the scene in question. In this case it was set to 200 pseudo-frames per second to enable 23.44 detections per rotation. The result of parsing these pseudo-frames through the relevant detection algorithm and TriplClust are shown in (c) and (d). For comparison, the 99 detections in (d) have again been connected where in this case the circular motion of the dot is clearly visible indicating successful tracking.

The results of asynchronous event-based detection and tracking are shown in (e) and (f) respectively. Here events have been reduced to just 13,746 again using three applications of the event-pair filter interlaced with the nearest neighbor filter. As expected, with such clean data TriplClust has accurately associated filtered events to a single object track correctly capturing the path taken by the spinning dot.

From the results of this experiment it is clear that while a frame-based approach has the ability to detect the spinning dot in a single frame, with a frame rate of just 20 Hz there is no way of tracking the dot successfully. While it is possible to increase this frame rate to suit the scene, the dramatic increase in memory requirements would need to be considered where the copious amount of redundant background data in this scenario would be a significant drawback.

In comparing the two event-based approaches, the primary delineator is the resolution of pseudo-frames versus asynchronous tracking. While both approaches are

considered to have tracked the spinning dot successfully, the microsecond resolution of asynchronous tracking versus the 5 ms resolution at 200 pseudo-frames per second creates a distinct difference. Clearly asynchronous tracking has much higher fidelity and so the path of the dot can be followed very closely. The drawback however, is that to associate 13,746 events requires significant processing time. On the other hand, pseudo-frames generate significantly more data yet only 99 detections which require significantly less processing. In this scene the difference between each approach is more dramatic because of the size of the dot in the sensor's field of view (FOV). A smaller dot size would generate fewer events and therefore be faster to associate with TriplClust. Therefore in general, asynchronous tracking is considered better suited to small, fast moving targets while pseudo-frames are more suited to larger targets.

6.2.3 Moving Tennis Balls

To demonstrate the tracking ability of all three methods in a more uncontrolled scenario, tennis balls acting as target objects were rolled and bounced around inside the laboratory environment. This steps away from the controlled and predictable pendulum and spinning disc examples used previously and provides a more realistic demonstration of the tracking ability of each method.

The example chosen for this comparison captures three tennis balls being rolled one after the other from one brightly lit laboratory room through wide double doors into an adjacent dark laboratory room. In the first room the ball's path is flanked by a dark metal shelving unit on the left while on the right is a large white board leaning against a similar dark shelving unit. The adjacent room is relatively open with only a few office chairs positioned at the back. The first and third ball roll directly away from the DAVIS 240C and into the second room while the second ball rebounds off an obstacle near the right hand double door frame and travels from right to left across

the sensor's field-of-view.

Figure 50 (a) and (b) show the frame-based detection and target association respectively. At 20 fps, the results in this example are very poor with no clear distinction between the three tennis balls or the paths they have taken. The primary hurdle for the frame-based approach to overcome is the large dynamic range of the scene. In trying to image both a bright and dark room simultaneously, as can be seen by the first frame, shine from the floor in the first room saturates the sensor hiding the whereabouts of the tennis balls. The few detections that occur in this saturated region are caused by the shadow of the tennis balls however as lights were evenly distributed across the ceiling this shadow was only intermittent leading to a broken sequence of detections. Unfortunately these shadows were not enough to enable detection of the second rebounding tennis ball. As the first and third balls moved into the second room the exposure time of each frame is better suited leading to a more continuous stream of detections before the balls either stop or become too small to resolve. In (b), TriplClust has attempted to associate all frame-based detections where five separate objects have been identified. Here Objects 1 and 2 belong to the first tennis ball, Object 3 is the very beginning of the second tennis ball, Object 4 is a brief reflection of the second ball from the right hand shelving unit and Object 5 follows the path of the third ball reasonably well. From these results it is clear the path of each tennis ball is not well understood and that effective object detection and tracking has not been achieved.

In an attempt to detect and track using pseudo-frames, Figure 50 (c) and (d) capture the outcome. In (c) raw events were filtered using the event pair and nearest neighbor filters with Δt_{EP} and Δt_{NN} set to 0.1 and 0.01 seconds respectively, before binning into pseudo-frames at 20 fps. From (c) the paths of all three balls have become more apparent where not only is the presence of the second ball visible, but so too

is its rebounding deviation. In addition, the dark shelving unit on the left of the scene results in an abundance of noise in that region that is clearly visible. From the filtered events, detections were generated and passed to TriplClust as shown in (d). In this case four objects were identified whereby Object 1 and 3 accurately capture the path of the first and third balls while Objects 2 and 4 capture the path of the second ball. For the second ball, due to its sudden change in motion (i.e. rebound), TriplClust was unable to associate relevant detections to a single target. From this result it can be seen that the increased dynamic range of the event-based sensor has enabled detection of each tennis ball through the different lighting conditions.

In tracking the tennis balls with asynchronous events, Figure 50 (e) shows raw events again filtered using three applications of the event pair filter interleaved with the nearest neighbor filter before separating only OFF events. In this case raw events needed to be filtered heavily to remove as much noise as possible and ultimately keep the total event count down to reduce the load on TriplClust. The consequence is that the grouping of events along the path taken by the tennis balls has been visibly weakened and also significantly shortened. Essentially, heavy filtering has thinned out valid events and cut off those more disparate events produced as the tennis balls become smaller in the sensor's field-of-view. When parsing these filtered events through TriplClust, as shown in Figure 50 (f) the events have been associated to show the paths of the tennis balls in line with that shown for pseudo-frames in (d). The primary difference however is that the tracks are far less pronounced and significantly shorter. This result indicates the heavy filtering required prior to applying TriplClust has in effect been counter productive and removed a significant portion of events needed to accurately track the tennis balls, therefore demonstrating a significant drawback in asynchronous event-based tracking.

In an interesting comparison, Figure 51 (a) shows the output of TriplClust for each

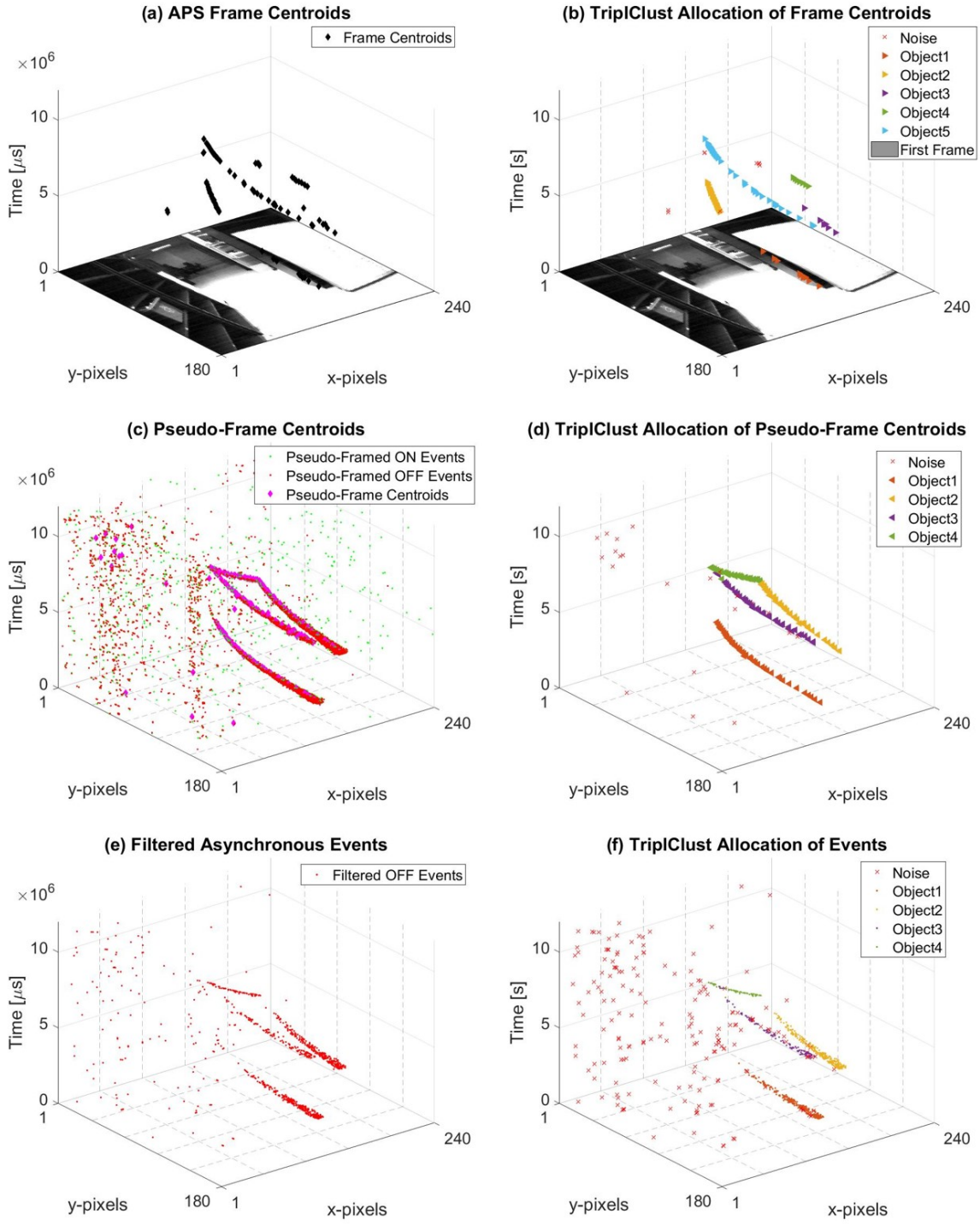


Figure 50: Demonstration of tracking multiple tennis balls.

detection and tracking method with noise removed and without segregating detections to specific objects. It appears the frame-based and pseudo-frame detections actually

complement each other in a sense that pseudo-frames are effective for detecting the balls in the saturated region of the scene while standard frames are effective for detecting the slow, subtle movement of the balls at the final moments of the recording in the more suitably exposed region. The reason pseudo-frame detections appear to drop targets early in this scenario is because at the point where the scene moves from the first bright room to the second dark room the balls have not only become smaller in the sensor's field-of-view, but they also track much slower across pixels. This causes the event stream in the (x, y, t) domain to curve up sharply as the time between events becomes greater and their occurrence more sparse. Compounded by filtering, this causes the pseudo-frame target tracks to be dropped earlier than expected despite the event-based sensor actually responding to the motion of the tennis balls.

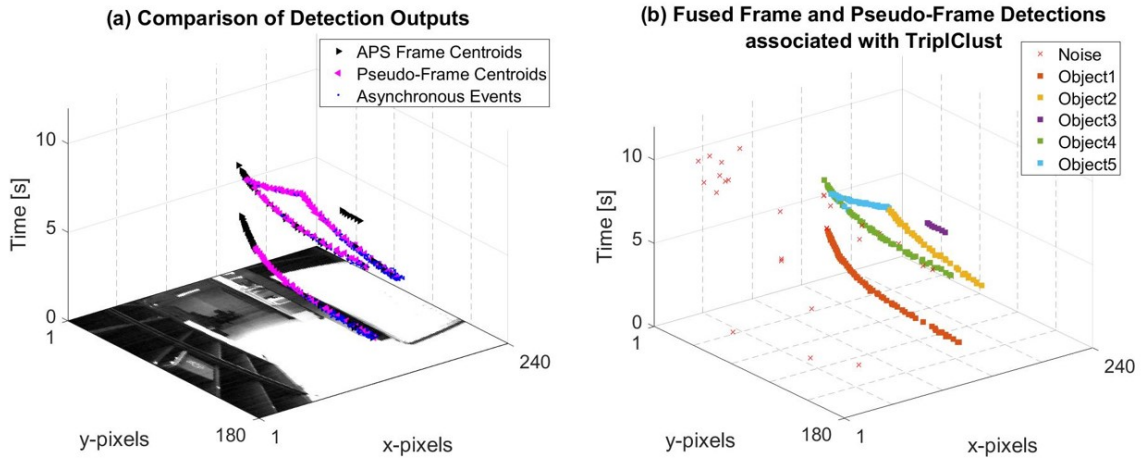


Figure 51: (a) Tracking comparison for multiple tennis balls where frames and pseudo-frame detections complement each other. (b) These detections are then fused to produce longer, more accurate tracks.

In Figure 51 (b), both frame-based and pseudo-frame detections have been fused and parsed to TriplClust to produce longer, more accurate object tracks demonstrating that the advantages of both sensor types can be combined to produce superior results in the right circumstances. The drawback however is that fusing output from

the two methods results in the accumulation of processing requirements which is a consequence that must be considered.

In considering the performance of each approach separately in this demonstration, it was shown that frames by themselves were not able to produce any effective tracks while asynchronous events produced weak and short tracks. Pseudo-frames on the other hand successfully produced effective object tracks. By taking advantage of the high dynamic range of an event-based sensor and not having to filter raw events so heavily, moving tennis balls could be detected and consequently tracked effectively using TriplClust. Furthermore, specific to this data set it became apparent that both frame-based and pseudo-frame detections could be fused to produce a improved tracking result than for each method individually.

6.2.4 Airborne Drones

To compare the performance of each detection and tracking method in a realistic and highly applicable military scenario, the DAVIS 240C was used to record airborne targets flying above Wright-Patterson Air Force Base. Short of using full size aircraft, inexpensive and highly maneuverable radio controlled (RC) aircraft (referred to as drones in this report) were used as substitutes. Despite their smaller size, this enabled repeated flybys and the ability to capture a large data set for analysis. The primary drawback is that the flying altitude of a drone is only a few hundred feet and therefore does not match that of a full size aircraft, reducing the effect of cloud occlusion and atmospheric turbulence. Nevertheless, using drones provides a means to challenge each approach in a realistic scenario cluttered by cloud, sun glare and atmospheric turbulence that could not simply be recreated in the laboratory.

Three different drone variants were used in this comparison where Figure 52 captures their design. With two fixed wing aircraft and one rotary quadcopter it enabled

Figure 52: (Left) Equipment setup to track drones at Wright-Patterson Air Force Base with the fixed wing drone in the background. (Top right) The Skywalker x8. (Bottom right) The SS40 quadcopter with front and rear LEDs.

the ability to observe a continuously moving target as well as a stop-and-go platform in order to draw out the effectiveness of both frame-based and event-based approaches. In addition, recordings were taken both at approximately midday and also in the dark of night. Examples of the both scenarios are discussed below.

6.2.4.1 Daytime Flying

In a similar recording to the previously discussed banking drone, the example chosen for this comparison is of the drone flying from right to left across the field of view of the DAVIS 240C. At the same time, six small birds travelling quickly enter the scene randomly from the top of the frame moving towards the right. With the sensor pointed towards an overcast sky, the background is entirely cloud.

As with previous comparisons, Figure 53 (a) captures computed detections from the frame-based recording taken at 20 fps. While the size and contrast of the drone made for simple detection, in order to capture the small, fast moving birds the minimum target size was reduced to a single pixel and the threshold value after background subtraction was reduced significantly. This increased the sensitivity of the detection

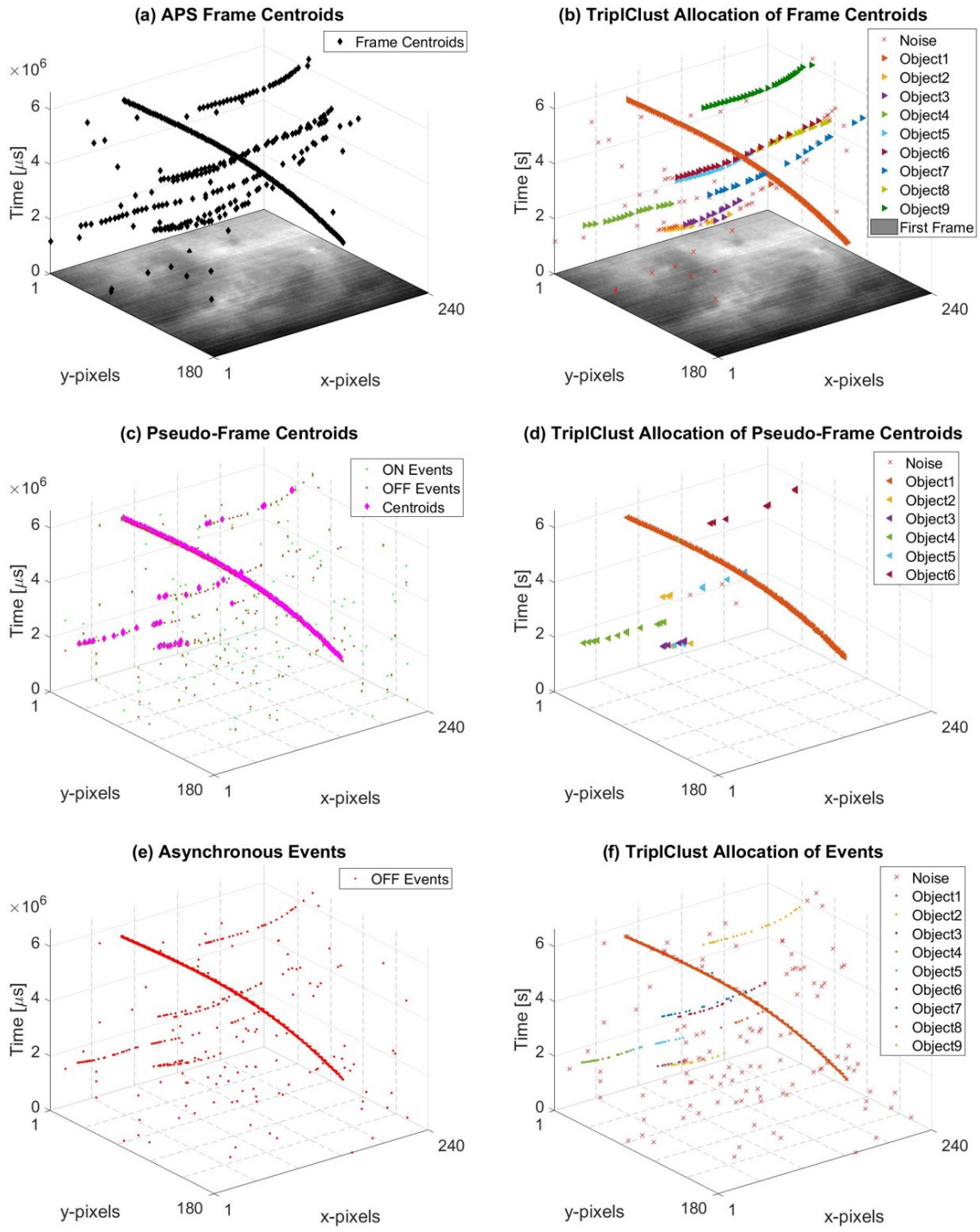


Figure 53: Demonstration of tracking drones with background birds.

at the cost of including additional false detection (i.e. noise). Fortunately the data from this recording was very clean and so did not cause a significant problem. Fig-

ure 53 (b) then captures the association of detections using TriplClust where Object1 captures the dominating track of the drone without issue. Surprisingly the six small birds are also tracked albeit with some flaws. At any one time each bird only covered a single pixel, additionally, with the overcast background at times the contrast between background and bird was non-existent which led to broken sequences of detections. While TriplClust was able to associate the majority of detection streams correctly, due to large breaks in detections and random changes in direction from the birds, multiple tracks have been identified from single targets. With that in mind, Objects 4 and 7 belong to bird 3 while Objects 5 and 8 to bird 4 (Note: birds numbered in the order they appear). Despite the small errors in association, the results demonstrate that frame-based detection and tracking can not only track the primary drone target, but at the same time also detect and track small, fast, random moving birds that were difficult to follow with the naked eye.

In performing the same task with pseudo-frames, Figure 53 (c) and (d) capture the result. First the event pair filter with Δt_{EP} set to 0.1 seconds was applied to raw events which enabled one to recognize the path of the drone and birds alike. In passing these filtered events through the pseudo-frame detection algorithm, due to the sparse nature of events and the single pixel bird size, at 20 fps pseudo-frames could not group enough events together to generate a detection and so while the drone is detected and tracked effectively, events from the birds have not been converted into detection and therefore tracks. Highlighted in (d), the drone has been clearly tracked at Object 1 however no clear tracks appear for the six birds demonstrating that in this scenario, the use of pseudo-frames is not sufficient.

Figure 53 (e) and (f) capture the result of using events asynchronously to detect and track all targets. Despite usual APS coupling generating excessive ON events, in this case OFF events were relatively clean and so in order to retain enough target

information, only the polarity filter was used to separate OFF events from raw data. This also had the benefit of ensuring a single plane of events were passed to TriplClust at the cost of needing to segregate a larger number of uncorrelated, isolated events (i.e. noise). The output of TriplClust shown in (f) demonstrates the drone has been clearly tracked while detections from the birds have been associated to multiple objects in a similar way to the frame-based approach. From the results of (f), for this example it can be seen that asynchronous tracking has been effective not only for a large primary target but also at the same time for smaller, more random moving targets.

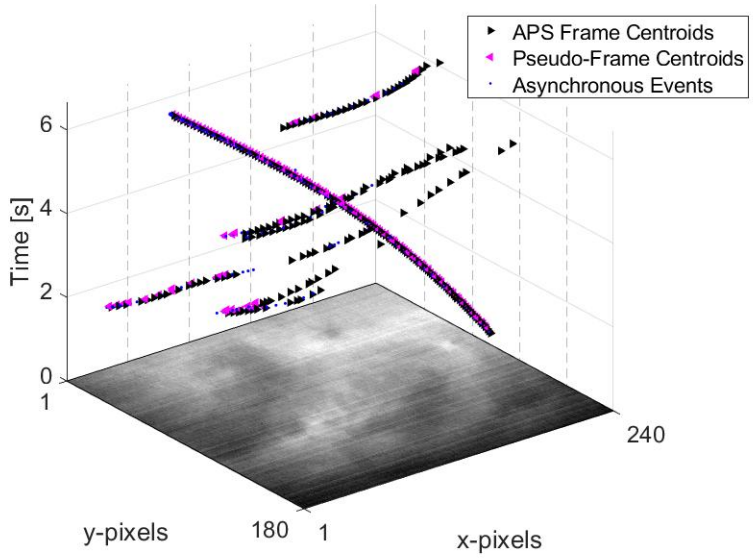


Figure 54: Tracking comparison of tracking drones with background birds.

To compare the performance of each approach Figure 54 provides a visual reference. From these results it is clear that a pseudo-frame approach compares poorly, only tracking the drone with certainty while essentially missing the presence of the six birds. The frame-based and asynchronous event approaches produced results of very similar effect with some subtle differences. For the drone both tracks are comparable however while each approach detected and tracked each bird to a certain extent, the

frame-based approach actually produced much longer tracks for four out of the six birds. This is an unusual result as it appears not to be caused by the association of TriplClust but rather the event-based sensor not detecting the birds in the first place. Considering a change in illumination has been detected with a frame-based approach so too should the event-based sensor create events for the same stimulation. It is therefore not well understood why the asynchronous event-based approach drops targets earlier that the frame-based approach in this case.

6.2.4.2 Nighttime Flying

To compare the performance of each approach in a nighttime scenario, the SS40 quadcopter from Figure 52 was flown in an open outdoor area with a low hedge and numerous tall trees in the background. On an overcast night, only minimal lighting was provided by distant housing and street lights. As previously discussed, the SS40 has two constant white LEDs on the nose and a red flashing LED on the tail. For this example the white LED lights were pointed towards the sensor at all times negating the impact of a flashing LED on the event-based sensor.

In low light conditions in order to image the environment, frame-based exposure time is generally increased significantly. In this example, to resolve the background hedge and trees against the night sky the frame rate was dropped to 0.34 Hz, equivalent to an exposure time of approximately three seconds. Consequently this led to significant motion blur when capturing the quadcopter's movement which is evident in Figure 55 (a) and (b). While normally the first frame is shown as a reference, in this case an accumulation of all frames has been shown to capture the complete path taken by the quadcopter. In the detection step, for a single frame the quadcopter's motion was captured by a long streak. While detecting the streak is simple, its centroid location is usually not representative of the target's actual location. In addition,

a low frame rate led to a low detection count making it difficult to piece together the motion of the quadcopter. As shown in (a), there are only 10 detections that even to the human eye do not capture the motion of the quadcopter. TriplClust reaffirms this in (b) whereby each detection is determined to be noise with no feasible geometric relationship identified between each detection. From this example, it is therefore not feasible to detect and track an airborne target using a frame-based approach in low light condition.

For the event-based sensor the effect of low light has far less impact and requires no adjustment to the sensor's bias settings at all. Also, despite low light conditions being found to produce considerably more noise events, surprisingly in this example noise is no more prevalent than in daylight recordings and therefore standard filtering techniques remained effective.

Figure 55 (c) and (d) show the result of detection and tracking using pseudo-frames. Here the event pair and nearest neighbor filters were used with Δt_{EP} and Δt_{NN} set to 0.1 and 0.01 seconds respectively. It can be seen that filtered events clearly capture the path taken by the quadcopter. Interestingly, there are several discontinuities in the event stream which are the result of the quadcopter becoming stationary in flight. In this case events are not created in the absence of an illumination change therefore providing a prime example of an event-based sensor's ability to reduce redundant data. In using pseudo-frames again at 20 fps, clear detections were produced and parsed to TriplClust with the results shown in (d). With the exception of only a few outliers, all detections were effectively associated to the path of the quadcopter demonstrating that pseudo-frames are an effective means to detect and track a moving target in low light, nighttime conditions.

In performing the same detection and tracking task with asynchronous events, the event pair and nearest neighbor filters were applied as per that for pseudo-frames

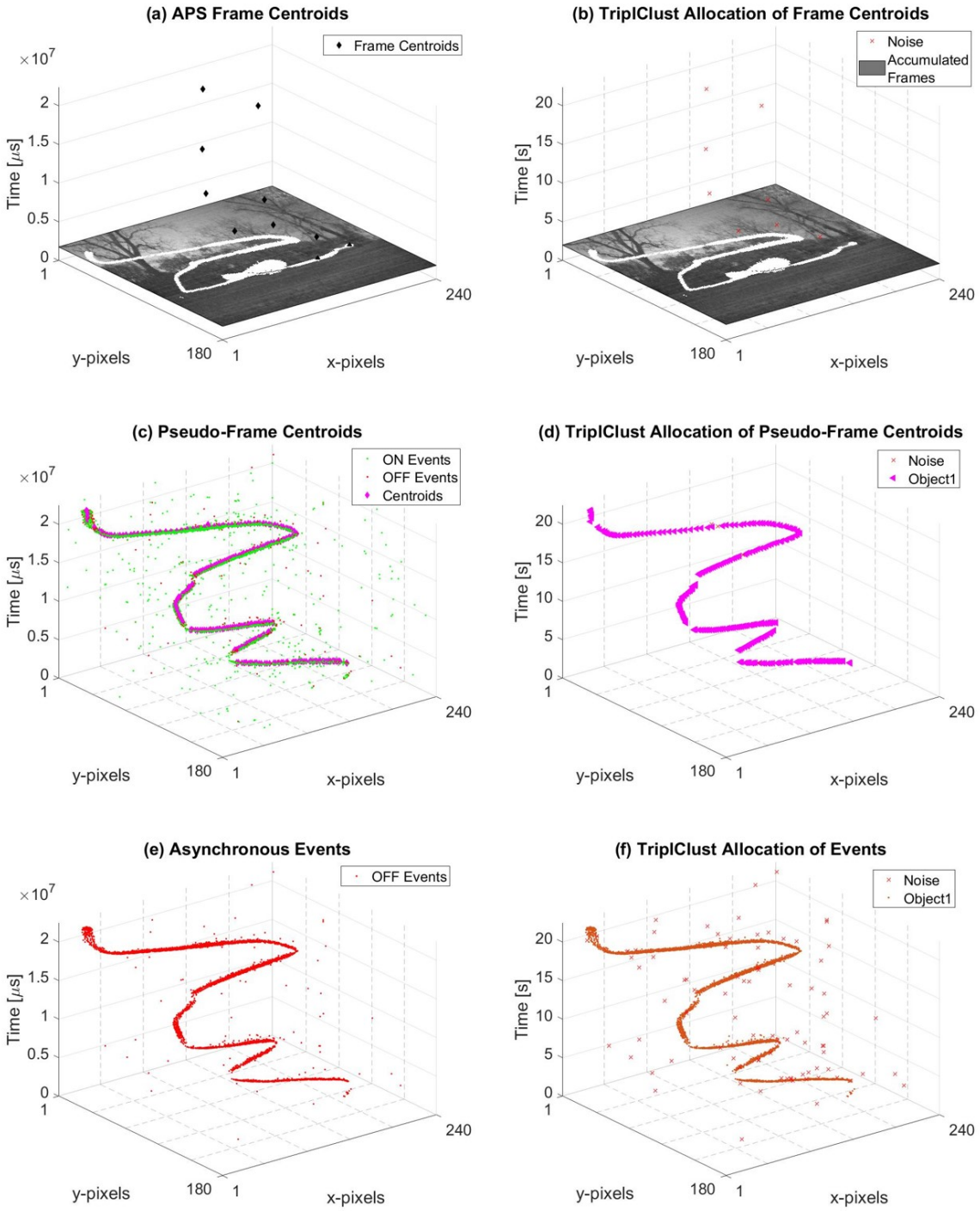


Figure 55: Demonstration of tracking the SS40 quadcopter at night.

before separating out only OFF events as shown in (e). This reduced 209,780 raw events to just 3,018 and allowed TriplClust to operate efficiently. In parsing filtered

events to TriplClust the output is shown in (f) where it can be seen that again TriplClust has accurately associated events to form the track taken by the quadcopter while effectively discriminating noise. As with pseudo-frames, asynchronous detection and tracking has provided comparable results and established that an event-based approach provides an effective means to track moving objects continuously in low light.

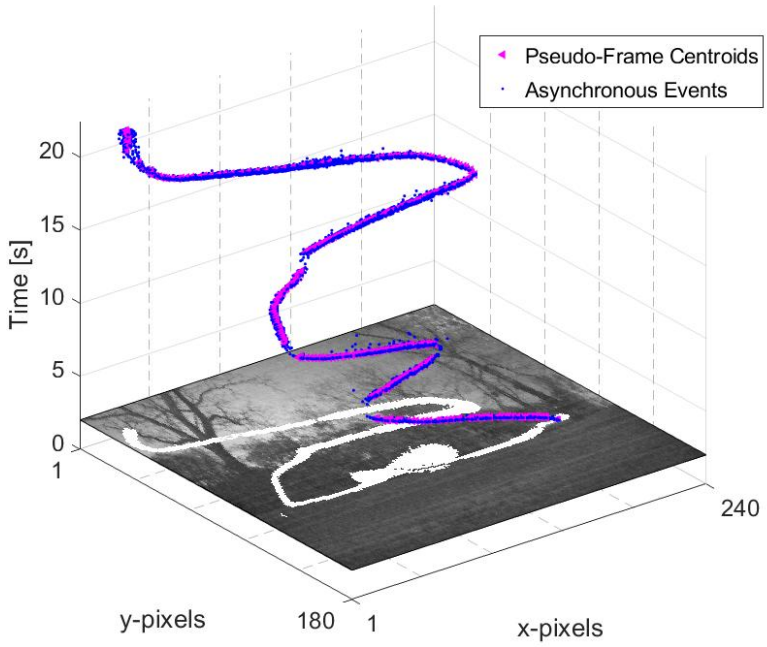


Figure 56: Comparison of tracking the SS40 quadcopter at night.

In specifically comparing the performance of pseudo-frames versus asynchronous events, Figure 56 captures the output of TriplClust on the same plot. Note frame-based detection and tracking was omitted in this case due to its poor performance. In this case both tracks align very well with the only noteworthy difference being the temporal resolution of each track. At 20 fps the stream of pseudo-frame detections is temporally separated by much more than asynchronous events with their microsecond resolution. The asynchronous event-based approach therefore produces a far more

continuous track, negating any uncertainty in target location between detections. While with a slow moving target the impact is minimal in this example, if at anytime target velocity increases then this effect becomes more important and the advantage of asynchronous detection and tracking becomes more valuable.

6.2.5 Computational Comparison

Generally the performance of a detection and tracking algorithm is not solely judged by its ability to follow a moving target. Given the reliance of these algorithms on computation power, both memory and computational time are also important considerations when evaluating different algorithms. In this section, both the computational load and the processing time required to deliver a tracking output are discussed for each of the comparison scenarios above. To support this comparison, Tables 6 and 7 paired with Figure 57 have been included. First Table 6 details total frame and event counts to provide a link to the overall file size shown in Table 7. Additionally, detection counts and filtered event counts are also provided as they tie to the overall processing time. Table 7 then details the processing time for event filtering, detection phases and association with TriplClust. The total processing time and file size then capture the overall computational requirements of each algorithm. Please note that for this comparison each algorithm was optimized to return the best tracking result possible without consideration of computation load. For a more thorough comparison both the detection capability and computation load would be optimized in unison.

For these calculations, a Dell Inspiron 15 5000 Series laptop was used with an 8th Gen Intel Core i7 processor running Windows 10. Also, file sizes here are determined by the size of the Matlab® .mat file used to store specific frame-based data cubes and Address Event Representation (AER) event streams. Processing speeds were timed

using the `tic` and `toc` commands in Matlab®.

Because each scenario is uniquely different in terms of its length of recording, frame-based exposure time and level of activity (i.e. target size and amount of motion), there is only value in considering overarching trends in the data. Beginning with file size, as expected frame-based data is stored in considerably larger file sizes due to the fact that information from every pixel is stored regardless of whether a change in the scene occurred or not. For event-based data with its asynchronous, independent

Table 6: Compare computational load

Scenario	Approach	Frame Count	Event Count	Detection Count	Filtered Event Count
Pendulum (0 lux)	Frames	136	-	136	-
	Pseudo-Frames	154	274,521	144	27,727
	Events	-	274,521	-	3,097
Pendulum (1000 lux)	Frames	153	-	162	-
	Pseudo-Frames	153	251,307	197	59,495
	Events	-	251,307	-	8,514
Spinning Disc	Frames	85	-	105	-
	Pseudo-Frames	99	1,546,241	99	34,890
	Events	-	1,546,241	-	13,746
Tennis Balls	Frames	232	-	97	-
	Pseudo-Frame	240	115,919	193	9,992
	Events	-	115,919	-	861
Daytime Drone	Frames	133	-	341	-
	Pseudo-Frames	131	145,161	163	1,564
	Events	-	145,161	-	1,100
Nighttime Drone	Frames	8	-	10	-
	Pseudo-Frames	448	209,780	289	6,456
	Events	-	209,780	-	3,018

Table 7: Compare computational time

Scenario	Approach	Event Filters (s)	Detection (s)	TriplClust (s)	Total Time (s)	File Size (KB)
Pendulum (0 lux)	Frames	-	0.52 ± 0.01	0.79 ± 0.02	**1.31 ± 0.02**	**6,664**
	Pseudo-Frames	0.85 ± 0.01	0.28 ± 0.01	0.57 ± 0.01	**1.69 ± 0.02**	**840**
	Events	1.88 ± 0.03	-	45.52 ± 0.43	**47.4 ± 0.43**	**776**
Pendulum (1000 lux)	Frames	-	0.57 ± 0.01	0.49 ± 0.01	**1.05 ± 0.01**	**7,572**
	Pseudo-Frames	1.27 ± 0.01	0.30 ± 0.01	0.73 ± 0.01	**2.30 ± 0.02**	**857**
	Events	1.55 ± 0.02	-	566.6 ± 12.2	**568.1 ± 12.2**	**810**
Spinning Disc	Frames	-	0.36 ± 0.01	0.20 ± 0.003	**0.55 ± 0.01**	**3,883**
	Pseudo-Frames	1.74 ± 0.03	0.18 ± 0.004	0.25 ± 0.01	**2.16 ± 0.03**	**2,589**
	Events	11.74 ± 0.52	-	534.2 ± 17.6	**545.9 ± 17.6**	**2,563**
Tennis Balls	Frames	-	0.81 ± 0.03	0.45 ± 0.005	**1.25 ± 0.03**	**7,272**
	Pseudo-Frames	0.73 ± 0.05	0.40 ± 0.02	1.44 ± 0.01	**2.57 ± 0.05**	**460**
	Events	0.77 ± 0.005	-	5.54 ± 0.11	**6.31 ± 0.11**	**412**
Daytime Drone	Frames	-	0.51 ± 0.02	2.31 ± 0.002	**2.83 ± 0.02**	**6,819**
	Pseudo-Frames	0.09 ± 0.004	0.21 ± 0.01	0.94 ± 0.01	**1.24 ± 0.02**	**363**
	Events	0.001 ± 0.0005	-	10.5 ± 0.2	**10.5 ± 0.2**	**339**
Nighttime Drone	Frames	-	0.03 ± 0.005	0.10 ± 0.006	**0.13 ± 0.01**	**435**
	Pseudo-Frames	0.65 ± 0.01	0.74 ± 0.03	3.24 ± 0.06	**4.63 ± 0.07**	**981**
	Events	0.64 ± 0.01	-	260.1 ± 7.3	**260.7 ± 7.3**	**904**

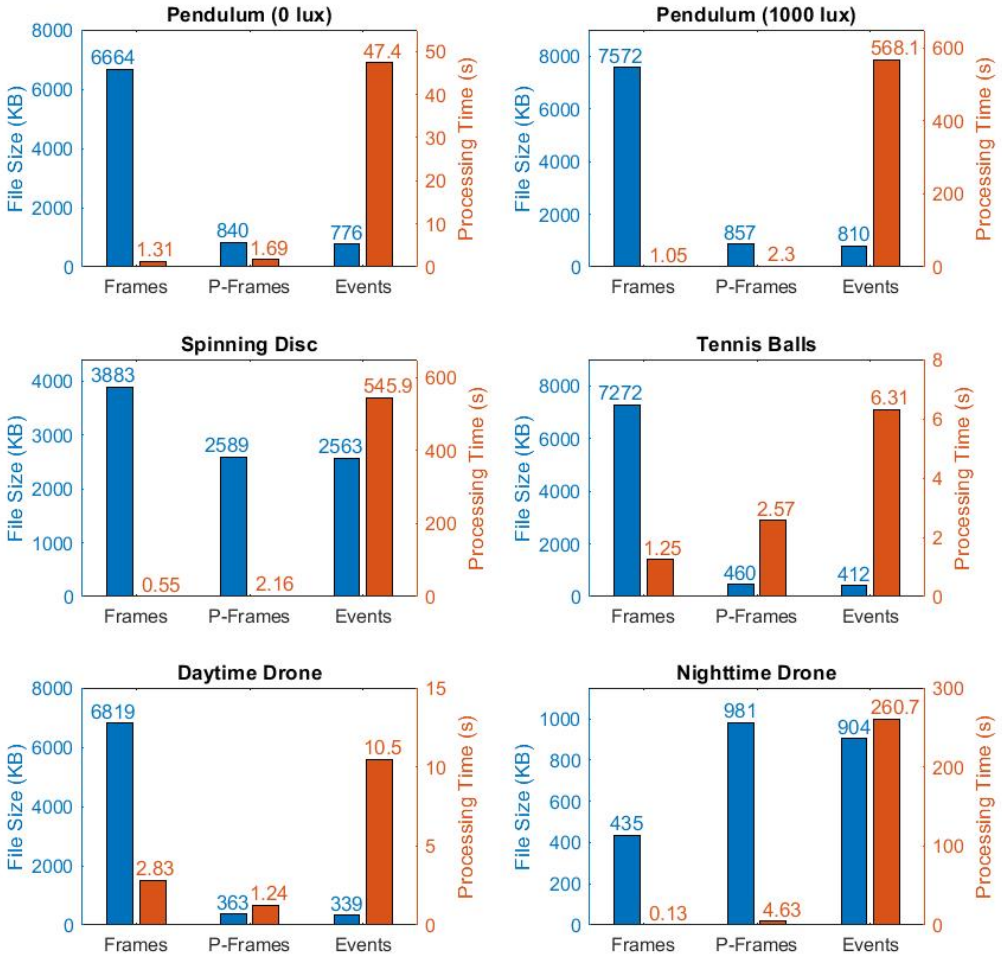


Figure 57: Comparison of computational load

nature, file size is dependent on the level of change in a scene. With a stationary sensor and a relatively stable background, it is sensor noise and object motion that generally make up the total event count. In each comparison scenario, both the APS and Dynamic Vision Sensor (DVS) were operating simultaneously within the DAVIS 240C. While in most cases this introduced significant APS coupling, the total event count, and therefore overall file size remained significantly lower than its frame-based counterpart. Two exceptions were the spinning disc and nighttime drone sequences.

For the spinning disc a much longer recording was used for the frame-based data for reasons discussed in Section 6.2.2. If the two recordings were of the same length the event-based file sizes would be considerably larger due to the sheer number of events generated by a large target moving at high speed. For the nighttime scene, because the frame-based exposure time was so long only eight frames were captured which led to a much smaller file size, albeit with not enough information to effectively track the target.

For event-based pseudo-frames both the raw events and post-processed pseudo-frames make up the overall file size. Although in most cases a comparable count of 240×180 frames are created for both frame and pseudo-frame-based approaches, because pseudo-frames are filled with a large proportion of zeros, Matlab® is able to store these frames very efficiently resulting in files sizes ranging in the tens of kilobytes. For this reason there is very little difference in the memory requirements of pseudo-frames and raw events.

In regards to the event-based filtering, pseudo-frames require less filtering as they are not so sensitive to noise and actually work better with a more dense accumulation of events. In most cases asynchronous events must be filtered heavily to minimize noise as much as possible to leave behind only the trail of a moving target even if it greatly dissipates the density of a track. For this reason event filtering is typically faster to produce pseudo-frames rather than asynchronous events, nevertheless, with the efficient construction of these filters the difference in processing time is minimal.

In comparing the detection processing time for frames and pseudo-frames, because pseudo-frames bypass the frame-based background subtraction and threshold filtering steps, detection is comparably much faster for pseudo-frames. Bearing in mind however, that because events must first be filtered to build pseudo-frames, this advantage is usually negated by event filtering which in most cases results in the advantage

shifting to the frame-based approach. Another thing to bear in mind is that for this comparison the detection step used for both frame-based approaches apply in-built Matlab® commands and so arguably have an advantage over the authors personally developed event-based scripts.

From this analysis it is clear that the true bottleneck in each of these scenarios lies with the TriplClust algorithm where regardless of the approach, as the point cloud parsed to TriplClust increases in size, so too does the processing time. In most cases the frame and pseudo-frame-based approaches parse several hundred data points to TriplClust. However, for the asynchronous event approach this number increases to several thousand. The one exception was in the daytime drone scenario where the frame-based approach generated more detections than the pseudo-frame-based approach and therefore had a longer processing time. Notable in this case was that the frame-based approach actually tracked better in this scenario.

While the asynchronous event-based approach offers the potential for more continuous tracking and an ability to track high speed targets, it comes at the cost of heavily increased processing time. If TriplClust could be optimized for event-based data and integrated into these detection and tracking algorithms, it is anticipated that processing times for large point clouds would reduce such that the processing times for asynchronous tracking could become more comparable to the other two approaches used here.

6.3 Closing Remarks

From the result of this comparison it appears there is no clear outlier in terms of which approach is best. In consideration of its ability to track, processing speed and memory requirements, no single approach offers a superior solution in all scenarios. It has been shown that the ideal solution depends largely on the scene in question

and the anticipated moving target of interest.

The frame-based approach offers relatively high speed and an effective means to detect and track a moving target but it comes at the cost of high memory requirements and remains hindered by scenes with high dynamic range and fast moving targets. While its performance can be enhanced through additional algorithms (i.e. Kalman filtering), these concerns link directly to fundamental limitation in the sensor and therefore cannot be overcome.

Asynchronous event-based detection and tracking capitalized on the primary advantages of event data with very low data rates, high dynamic range and the potential to continuously track high speed targets with no prior knowledge as to their speed. This is not the case for frames or pseudo-frames where their frame-rate must first be tailored based on prior assumptions. Unfortunately, the byproduct of associating asynchronous events comes at the expense of computational time. However future research could help to improve this. At this stage with the analysis developed here, it is apparent that asynchronous event-based tracking is best suited for the detection and tracking of small targets that generate fewer events. This is particularly relevant to military applications where it is often the desire to establish an autonomous staring sensor capable of detecting and tracking small airborne targets moving at high speed in a wide range of outdoor lighting conditions.

If the best approach had to be chosen it would be pseudo-frames. As a hybrid frame and event-based approach, it capitalized on the advantages of both in terms of low memory requirements, fast processing speeds, high dynamic range and effective tracking capability. The primary flaw with this approach is that is does not take advantage of the event-based sensor's asynchronous nature and relies on prior knowledge of a target's speed to determine a suitable frame-rate. Nevertheless, from the results of this comparison this approach offers an excellent solution for detection and

tracking in the widest range of unique scenarios.

VII. Conclusions

A traditional frame-based approach to object detection and tracking is inherently limited by the operational characteristics of its imaging sensor where despite extensive research, it remains a significant challenge to detect and track high speed targets in scenes with high dynamic range while minimizing the load on data storage and computational power. In an effort to overcome these fundamental challenges, this thesis has explored the application of a novel event-based sensor to the same detection and tracking problem and performed a comparative assessment in regards to overall tracking performance and computational load.

With the paradigm shift that comes with moving to event-based sensors, their operational characteristics were first explored in an effort to become familiar with the sensor and establish techniques for processing raw sensor output. As build up to a comparative assessment a fundamental frame-based detection and tracking algorithm using background subtraction and threshold filtering was first developed before significant focus shifted to the event-based sensor. Here several event-based filters were developed to pull out specific target motion, each with varying degrees of effectiveness. It was found in most cases that when used in series, both the event pair and nearest neighbor filters removed the greatest amount of noise and produced the cleanest data for further processing.

At this point the concept of pseudo-frames was introduced and shown to be an effective means for processing event data before parsing into existing frame-based tracking algorithms. The fundamental flaw however was that this approach did not take full advantage of the asynchronous nature of raw event data. In pursuit of a better solution, an asynchronous event-based detection and tracking approach was developed that relied on event-based filtering to remove as much noise as possible before applying a geometric association algorithm known as TriplClust. While proven

to be effective at associating filtered events, as soon as the event count reached the order of thousands the processing load, and therefore time, increased dramatically.

As a result of this research it was found that the use of a frame-based sensor for detection and tracking provides an excellent solution, albeit one with the overhead of limited dynamic range, a dependence on frame rate and the need for significant data storage. Asynchronous event-based detection and tracking on the other hand offers a low power, low data, high dynamic range almost continuous object tracking solution with the unfortunate overhead of significant processing time limiting its ability to be integrated into a real-time system.

As a hybrid of both approaches, pseudo-frames offer an excellent compromise between both sensor paradigms. It has been shown that pseudo-frames have low computation load and produce very effective detection and tracking results. They take advantage of the event-based sensor's high dynamic range and high speed temporal resolution when their frame-rate is tailored appropriately. While they do not utilise the asynchronous nature of event-based sensors to their full extent, the results have shown they remain a very effective tool for object detection and tracking.

Overall, this research has not found an ideal event-based detection and tracking solution that could be considered suitable to replace existing frame-based technology. Despite the fundamental advantages that event-based sensors afford, an algorithm could not be developed that fully capitalizes on the sensor's low power, low data, high speed and high dynamic range characteristics. While pseudo-frames have produced arguably the best result in this research, the analysis shows there is significant value in pursuing a computationally efficient asynchronous event-based detection and tracking algorithm for the future.

7.1 Future Work

Given the potential of event-based sensors there is significant opportunity for further research and development. While their applicability is broad, possible extensions to the work presented here may include:

- Broaden the scope of this research to consider a panning or non-stationary sensor. Could detection and tracking from an airborne unmanned aerial vehicle (UAV) be possible with event-based technology?
- Continue to develop and optimize different event-based filters both in terms of their effectiveness and processing speed.
- Explore the possibility of applying a Kalman filter to asynchronous event-based data to fill in detection gaps in heavily filtered data.
- Optimize the TriplClust algorithm for event-based data and explore the possibility of making it real-time.
- Integrate a real-time event-based detection and tracking algorithm into hardware and test.
- In this research all event-based sensor bias levels were kept at their default setting. There is value in conducting a systematic analysis of these bias settings in an attempt to optimize the ability of an event-based sensor to detect a moving target under various operational scenarios.
- Asynchronous event-based tracking using TriplClust is hamstrung by excessive processing time. Consider exploring alternative methods to asynchronous event-based tracking.

- Develop a comparative assessment where both tracking and computational load are optimized in unison rather than in isolation.

- Extend this research to include newer, more capable sensors. With less noise and higher resolution, question whether the algorithms presented in this research would work with next generation sensors. What could be the challenges?

- Infrared event-based sensors are anticipated to become available in the near future. Consider applying similar detection and tracking algorithms developed in this research to the infrared frequency band.

- Objects can rapidly change size within a sensor's field-of-view like in the case of a missile seeker homing-in on its objective. Consider the impact of this effect on asynchronous event-based detection and tracking. Assess the impact of a changing target size in terms of computation load and real-time tracking.

- Assess the possibility of event-based sensors being used for fire control applications. High temporal resolution affords a significant advantage, however consider whether the precise location of a target can be determined with enough certainty to deploy munitions based solely on the data from an event-based sensor.

7.2 Military Potential of Event-Based Sensors

At the conclusion of this research it is the hope that this work will bring light to the enormous potential that event-based sensors present to the Defense community. Despite the immaturity of this technology, it is not often that a new sensor is developed and so with a shift in operational paradigm, comes a shift in thinking. Beyond the concept of detection and tracking, the scope of possibilities for event-based sensors is limited only by one's imagination. Given enough time and appropriate resources

for further research and development, it is expected that the disruptive nature of this technology will soon be realized.

Bibliography

1. Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *Computing Research Repository*, pages 1–25, 2019.

2. Gregory Cohen, Saeed Afshar, Andre van Schaik, Andrew Wabnitz, Travis Bessell, Mark Rutten, and Brittany Morreale. Event-based sensing for space situational awareness. *Advanced Maui Optical and Space Surveillance Technologies Conference*, pages 1–13, 2017.

3. Kaleb Nelson. *Event-based visual-inertial odometry on a fixed-wing unmanned aerial vehicle (MSc Thesis)*. Air Force Institute of Technology, 2019.

4. Anjie Gao, Zheng Mao, and Legong Sun. Research on target detection algorithms in optical-electronic imaging. *International Conference on Multimedia Technology*, pages 5056–5059, 2011.

5. Deepak Khosla, Yang Chen, Kyungnam Kim, Shinko Y. Cheng, Alexander L. Honda, and Lei Zhang. A neuromorphic system for object detection and classification. *Signal Processing, Sensor Fusion, and Target Recognition XXII*, 8745(May 2013):87450X, 2013.

6. R. W. Rodieck. *The first steps in seeing*. Sinauer Associates Inc, Sunderland MA, 1998.

7. Patrick Lichtsteiner. *An AER temporal contrast vision sensor*. PhD thesis, ETH Zurich, 2006.

8. Misha Mahowald. *An analog VLSI system for stereoscopic vision.* Kluwer Academic Publishers, Boston, MA, 1994.

9. Kareem A. Zaghloul and Kwabena Boahen. Optic nerve signals in a neuromorphic chip I: Outer and inner retina models. *IEEE Transactions on Biomedical Engineering*, 51(4):657–666, 2004.

10. Dongsoo Kim and Eugenio Culurciello. A compact-pixel tri-mode vision sensor. In *IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, pages 2434–2437. IEEE, 2010.

11. Juan A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco. A signed spatial contrast event spike retina chip. In *IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, pages 2438–2441, 2010.

12. Yu M. Chi, Udayan Mallik, Matthew A. Clapp, Edward Choi, Gert Cauwenberghs, and Ralph Etienne-Cummings. CMOS camera with in-pixel temporal change detection and ADC. *IEEE Journal of Solid-State Circuits*, 42(10):2187–2196, 2007.

13. Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 dB 15 micosecond latency asynchronous temporal contrast vision sensor. *Solid State Circuit*, 43(2):566–576, 2008.

14. Jack Tumblin, Amit Agrawal, and Ramesh Raskar. Why I want a gradient camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 103–110, 2005.

15. Alan A Stocker. An improved 2D optical flow sensor for motion segmentation. In *IEEE International Symposium on Circuits and Systems*, pages 332–335, 2002.

16. Hanme Kim. *Real-time visual SLAM with an event camera*. PhD thesis, Imperial College London, 2018.

17. Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-DOF pose tracking for high-speed maneuvers. *IEEE International Conference on Intelligent Robots and Systems*, pages 2761–2768, 2014.

18. Patrick Lichtsteiner and Tobi Delbruck. A 64x64 AER logarithmic temporal derivative silicon retina. *Research in Microelectronics and Electronics*, II:406–409, 2005.

19. Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 dB 30 mW asynchronous vision sensor that responds to relative intensity change. In *IEEE International Solid State Circuits Conference*, pages 2060–2069, 2006.

20. Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. 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(1):259–275, 2011.

21. Daniel Matolin and Christoph Posch. Dynamic, single photodiode pixel circuit and operating method thereof, 2018.

22. Christian Brandli, Raphael Berner, Minhao Yang, Shih Chii Liu, and Tobi Delbruck. A 240x180 130 dB 3 microsecond latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

23. Eugenio Culurciello, Ralph Etienne-Cummings, and Kwabena A. Boahen. A biomorphic digital image sensor. *IEEE Journal of Solid-State Circuits*, 38(2):281–294, 2003.

24. Diederik Paul Moeys, Chenghan Li, Julien N.P. Martel, Simeon Bamford, Luca Longinotti, Vasyl Motsnyi, David San Segundo Bello, and Tobi Delbruck. Color temporal contrast sensitivity in dynamic vision sensors. In *IEEE International Symposium on Circuits and Systems*, 2017.

25. Cedric Scheerlinck, Henri Rebecq, Timo Stoffregen, Nick Barnes, Robert Mahony, and Davide Scaramuzza. CED: Color event camera dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

26. Christoph Posch, Daniel Matolin, Rainer Wohlgenannt, Thomas Maier, and Martin Litzenberger. A microbolometer asynchronous dynamic vision sensor for LWIR. *IEEE Sensors Journal*, 9(6):654–664, 2009.

27. Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A two-stage capacitive-feedback differencing amplifier for temporal contrast IR sensors. *Analog Integrated Circuits and Signal Processing*, 64(1):45–54, 2010.

28. Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, Jooyeon Woo, Yohan Roh, Hyunku Lee, Yibing Wang, Ilia Ovsiannikov, and Hyunsurk Ryu. A 640x480 dynamic vision sensor with a 9 micrometer pixel and 300 Meps address-event representation. In *IEEE International Solid-State Circuits Conference*, volume 60, pages 66–67. IEEE, 2017.

29. Menghan Guo, Jing Huang, and Shoushun Chen. Live demonstration: A 768x640 pixels 200 Meps dynamic vision sensor. In *IEEE International Symposium on Circuits and Systems*, page 6853, 2017.

30. Gregory Cohen, Saeed Afshar, and Andr van Schaik. Approaches for astrometry using event-based sensors. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, pages 1–7, 2018.

31. Tat-Jun Chin, Samya Bagchi, Anders Eriksson, and Andre van Schaik. Star tracking using an event camera. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

32. Viviane S. Ghaderi, Marcello Mulas, Vinicius Felisberto Santos Pereira, Lukas Everding, David Weikersdorfer, and Jorg Conradt. A wearable mobility device for the blind using retina-inspired dynamic vision sensors. In *IEEE Engineering in Medicine and Biology Society*, pages 3371–3374, 2015.

33. Lukas Everding, Lennart Walger, Viviane S. Ghaderi, and Joerg Conradt. A mobility device for the blind with improved vertical resolution using dynamic vision sensors. In *International Conference on e-Health Networking, Applications and Services*, pages 1–5, 2016.

34. Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1647–1655, 2017.

35. Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Continuous-time intensity estimation using event cameras. In *Asian Conference on Computer Vision*, 2018.

36. Souptik Barua, Yoshitaka Miyatani, and Ashok Veeraraghavan. Direct face detection and video reconstruction from event cameras. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1–9, 2016.

37. Matthew Cook, Luca Gugelmann, Florian Jug, Christoph Krautz, and Angelika Steger. Interacting maps for fast visual interpretation. In *International Joint Conference on Neural Networks*, pages 770–776, 2011.

38. Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew Davison. Simultaneous mosaicing and tracking with an event camera. In *British Machine Vision Conference*, 2014.

39. Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017.

40. Christian Brandli, Lorenz Muller, and Tobi Delbruck. Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor. In *IEEE International Symposium on Circuits and Systems*, pages 686–689, 2014.

41. Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *International Journal of Robotics Research*, 36(2):142–149, 2017.

42. Henri Rebecq, Ren Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3857–3866, 2019.

43. Gottfried Munda, Christian Reinbacher, and Thomas Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. *International Journal of Computer Vision*, 126(12):1381–1393, 2018.

44. S Mohammad Mostafavi I., Lin Wang, Yo-Sung Ho, and Kuk-Jin Yoon. Event-based high dynamic range image and very high frame rate video generation using

conditional generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10081–10090, 2019.

45. Paul Rogister, Ryad Benosman, Sio Hoi Ieng, Patrick Lichtsteiner, and Tobi Delbruck. Asynchronous event-based binocular stereo matching. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):347–353, 2012.

46. Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. EMVS: Event-based multi-view stereo 3D reconstruction with an event camera in real-time. *International Journal of Computer Vision*, 126(12):1394–1414, 2018.

47. Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. EMVS: Event-based multi-view stereo. In *British Machine Vision Conference*, pages 1–11, York, 2016.

48. Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based visual inertial odometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5816–5824, 2017.

49. Tobi Delbruck and Patrick Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *IEEE International Symposium on Circuits and Systems*, pages 845–848, 2007.

50. M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schön, B. Kohn, and H. Garn. Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. In *Digital Signal Processing Workshop*, pages 173–178, 2006.

51. Tobias Bolten, Regina Pohle-Fröhlich, and Klaus D. Tönnies. Application of hierarchical clustering for object tracking with a dynamic vision sensor. In *International Conference on Computational Science*, pages 164–176, 2019.

52. Shih Chii Liu, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas. *Event-based neuromorphic systems*. Wiley, 2015.

53. Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Asynchronous, photometric feature tracking using events and frames. In *European Conference of Computer Vision*, pages 766–781, 2018.

54. Xavier Clady, Sio Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. *Neural Networks*, 66:91–106, 2015.

55. Chris Harris and Mike Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, volume 15, pages 147–151, 1988.

56. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *European Conference of Computer Vision*, pages 430–443, 2006.

57. Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *British Machine Vision Conference*, pages 1–11, 2017.

58. Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. *IEEE International Conference on Intelligent Robots and Systems*, pages 4144–4149, 2016.

59. Christian Brändli. *Event-based machine vision*. PhD thesis, ETH Zurich, 2015.

60. Gabriel Cristóbal, Laurent Perrinet, and Matthias S Keil. *Biologically inspired computer vision: Fundamentals and applications*. Wiley-VCH, 2016.

61. Juan A. Leñero-Bardallo, Ricardo Carmona-Galán, and Angel Rodríguez-Vázquez. Applications of event-based image sensors - Review and analysis. *International Journal of Circuit Theory and Applications*, 46(9):1620–1630, 2018.

62. Tobi Delbruck and Carver A. Mead. Adaptive photoreceptor with wide dynamic range. In *IEEE International Symposium on Circuits and Systems*, volume 4, pages 339–342, 1994.

63. Raphael Berner, Christian Peter Brändli, Minhao Yang, Shih-Chii Liu, and Tobi Delbrück. A 240x180 120 dB 10 mW 12 microsecond latency sparse output vision sensor for mobile applications. *International Image Sensor Workshop*, pages 186–187, 2013.

64. Diederik Paul Moeys, Federico Corradi, Chenghan Li, Simeon A Bamford, Luca Longinotti, Fabian F Voigt, Stewart Berry, Gemma Taverni, Fritjof Helmchen, and Tobi Delbruck. A sensitive dynamic and active pixel vision sensor for color or neural imaging applications. *IEEE Transactions on Biomedical Circuits and Systems*, 12(1):123–136, 2018.

65. Brian McReynolds. *A comprehensive test methodology and physics-based camera model for characterizing neuromorphic imagers (MSc Thesis)*. Air Force Institute of Technology, 2019.

66. iniVation. DAVIS Specifications Sheet.

67. Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, 2014.

68. Christoph Dalitz, Jens Wilberg, and Lukas Aymans. Triplclust: An algorithm for curve detection in 3D point clouds. *Image Processing On Line*, 9:26–46, 2019.

69. Vandana Padala, Arindam Basu, and Garrick Orchard. A noise filtering algorithm for event-based asynchronous change detection image sensors on TrueNorth and its implementation on TrueNorth. *Frontiers in Neuroscience*, 12:1–14, 2018.

70. Saeed Afshar, Ying Xu, Jonathan Tapson, Andr van Schaik, and Gregory Cohen. Event-based feature extraction using adaptive selection thresholds. 2019.

71. Gregory Cohen. *Event-based feature detection, recognition and classification.* PhD thesis, Western Sydney University, 2017.

72. G. Zhao and J. Yuan. Curb detection and tracking using 3D-LIDAR scanner. In *IEEE International Conference on Image Processing*, pages 437–440, 2012.

73. V. Renò, N. Mosca, M. Nitti, C. Guaragnella, T. D'Orazio, and E. Stella. Real-time tracking of a tennis ball by combining 3D data and domain knowledge. In *International Conference on Technology and Innovation in Sports, Health and Wellbeing*, pages 1–7, 2016.

74. Christoph Dalitz, Yassid Ayyad, Jens Wilberg, Lukas Aymans, Daniel Bazin, and Wolfgang Mittig. Automatic trajectory recognition in active target time projection chambers data by means of hierarchical clustering. *Computer Physics Communications*, 235:159–168, 2019.

75. Yassid Ayyad, Wolfgang Mittig, Daniel Bazin, Saul Beceiro-Novo, and Marco Cortesi. Novel particle tracking algorithm based on the Random Sample Consensus Model for the Active Target Time Projection Chamber (AT-TPC). *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 880:166–173, 2 2018.

76. Christoph Dalitz, Tilman Schramke, and Manuel Jeltsch. Iterative Hough transform for line detection in 3D point clouds. *Image Processing On Line*, 7:184–196, 2017.

77. Tat-Jun Chin and Samya Bagchi. Event-based star tracking via multiresolution progressive Hough transforms. In *IEEE Winter Conference Applications of Computer Vision*, pages 1–10, 2020.

78. Jos Lezama, Gregory Randall, Jean Michel Morel, and Rafael Grompone Von Gioi. An unsupervised algorithm for detecting good continuation in dot patterns. *Image Processing On Line*, 7:81–92, 2017.

Acronyms

AER Address Event Representation. 13, 38, 44, 46, 132

ANT Autonomy and Navigation Technology. 24

APS active pixel sensor. 33, 34, 39, 40, 57, 62, 75, 76, 77, 81, 86, 87, 114, 126, 133

ATIS Asynchronous Time-Based Imaging Sensor. 12, 13

CCD charge-coupled device. 32, 33

CDS correlated double sampling. 32, 35, 40

CFA color filter array. 14

CMOS complementary metal-oxide semiconductor. 16, 32, 33, 34, 40

DAVIS Dynamic and Active Vision Image Sensor. 12, 21, 23, 29, 32, 39, 40

DOF degrees-of-freedom. 23, 24

DR dynamic range. 36

DVS Dynamic Vision Sensor. 10, 11, 12, 14, 16, 36, 38, 39, 40, 41, 42, 45, 46, 47, 71, 75, 80, 133

EKF extended Kalman Filter. 24

EPS events-per-second. 46

EVIO event-based VIO. 23, 24

FEAST Feature Extraction with Adaptive Selection Thresholds. 90

FFT Fast-Fourier Transform. 51

FIFO First-In, First-Out. 38

FOV field of view. 54, 93, 96, 98, 102, 103, 104, 105, 117

FPA Focal Plane Array. 5, 9, 11, 32, 48

FPN fixed pattern noise. 13, 34, 35, 42, 43, 44, 57, 58

GEO Geostationary Earth Orbit. 29

HDR High Dynamic Range. 21, 22

IMU inertial measurement unit. 12, 23

IR infrared. 16, 17

jAER Java Address Event Representation. 25, 48

LED light emitting diode. 44, 45, 48, 49, 50

LEO Low Earth Orbit. 29, 31

LIDAR Light Detection and Ranging. 92

LWIR Long Wave Infrared. 15, 16

RC radio controlled. 123

RGBW red, green, blue and white. 14

SNR Signal-to-Noise Ratio. 46

SSA Space Situational Awareness. 29, 93

SWaP size, weight and power. 29, 31

UAV unmanned aerial vehicle. 24, 141

VIO visual-inertial odometry. 23

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2020	Master's Thesis	Sept 2018 — Mar 2020

4. TITLE AND SUBTITLE

A Comparative Evaluation of the Detection and Tracking Capability Between Novel Event-Based and Conventional Frame-Based Sensors

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Boettiger, James P, FLTLT

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-20-M-007

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Not bound by a synchronous frame-rate, pixels from an event-based sensor respond independently and asynchronously to changes in log-light intensity. They therefore afford unique characteristics like low power consumption, reduced bandwidth requirements, high temporal resolution and high dynamic range. Traditional frame-based technology continues to suffer from motion blur, low dynamic range, speed limitations and high data storage requirements. Event-based sensors therefore offer a potential solution to these challenges. This research centers around a comparative assessment of frame and event-based object detection and tracking. A basic frame-based algorithm was developed and used to compare against two different event-based algorithms. First event-based pseudo-frames were parsed through standard frame-based algorithms while secondly, target tracks were constructed directly from filtered events. Effective detection and tracking was demonstrated however this research did not find an ideal event-based solution to replace existing frame-based technology. The findings do however show there is significant value in pursuing this technology further.

15. SUBJECT TERMS

Event-based sensors, neuromorphic sensors, target detection, target tracking

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			FLTLT James P. Boettiger, AFIT/ENG
U	U	U	UU	170	19b. TELEPHONE NUMBER *(include area code)* (937) 344-3250, james.boettiger.au@afit.edu