**Event-Based Noise Filtration with Point-of-Interest Detection
and Tracking for Space Situational Awareness**

by

Nikolaus Salvatore

B.S. in Biological Sciences, Cornell University, 2016

B.S. in Electrical Engineering, University of Pittsburgh, 2018

Submitted to the Graduate Faculty of the

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical and Computer Engineering

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Nikolaus Salvatore

It was defended on

March 30, 2020

and approved by

Zhi-Hong Mao, Ph.D., Professor
Department of Electrical and Computer Engineering
Department of Bioengineering

Samuel Dickerson, Ph.D., Director and Assistant Professor
Department of Electrical and Computer Engineering

Thesis Advisor: Alan D. George, Ph.D., R&H Mickle Endowed Chair and Professor
Department of Electrical and Computer Engineering

**Event-Based Noise Filtration with Point-of-Interest Detection
and Tracking for Space Situational Awareness**

Nikolaus Salvatore, MS

University of Pittsburgh, 2020

This thesis explores an asynchronous noise-suppression technique to be used in conjunction with asynchronous, Gaussian-blob tracking on dynamic vision sensor (DVS) data. This type of sensor is a member of a relatively new class of neuromorphic sensing devices that emulate the change-based detection properties of the human eye. By leveraging a biologically inspired mode of operation, these sensors can achieve significantly higher sampling rates as compared to conventional cameras, while also eliminating redundant data generated by static backgrounds. The resulting high dynamic range and fast acquisition time of DVS recordings enables the imaging of high-velocity targets despite ordinarily problematic lighting conditions. The technique presented here relies on treating each pixel of the sensor as a spiking cell keeping track of its own activity over time, which in turn can be filtered out of the resulting sensor event stream by user-configurable threshold values that form a temporal bandpass filter. In addition, asynchronous blob-tracking is supplemented with double-exponential smoothing prediction and Bezier curve-fitting in order to smooth tracker movement and interpolate target trajectory respectively. This overall scheme is intended to achieve asynchronous point-source tracking using a DVS for space-based applications, particularly in tracking distant, dim satellites. In the space environment, radiation effects are expected to introduce transient, and possibly persistent, noise into the asynchronous event-stream of the DVS. Given the large distances between objects in space, targets of interest may be no larger than a single pixel and can therefore appear similar to

such noise-induced events. In this thesis, the asynchronous approach is experimentally compared to a more traditional approach applied to reconstructed frame data for both performance and accuracy metrics. The results of this research show that the asynchronous approach can produce comparable or even better tracking accuracy, while also drastically reducing the execution time of the process by seven times on average.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

The primary motivation behind the work presented within this thesis is to leverage the strengths of new dynamic vision sensor (DVS) technologies for space applications. These sensors' high dynamic range, under-sampling of redundant visual information, and exceptional power efficiency make them ideal for tracking dim, orbital objects on space platforms for space situational awareness. In addition to leveraging existing event-based object tracking algorithms, new methods of noise filtration are required to mitigate the radiation effects found within space environments. The goal of this work is to provide performance gains in terms of both speed and accuracy by exploiting the asynchronous nature of recorded DVS data.

Recent years have seen the development of a new class of imaging sensors capable of replicating basic properties of biological vision, namely its focus on detecting changes within scenes. The first of these new neuromorphic vision systems was the DVS proposed in [1], which details the sensor's architecture and relation to biological analogs. The DVS functions by detecting logarithmic intensity of luminance changes through a series of photoreceptors and integrating and comparative circuits associated with each individual pixel. Luminance changes are detected via conventional photodetectors such as those often found in active-pixel sensors. Once the luminance change at a given pixel induces a voltage beyond a certain predefined threshold, the cell will generate an event that encodes the (x, y) coordinates, polarity, and timestamp. Each of the pixel cells of the sensor monitors both positive and negative changes in luminance intensity, which is in turn reported in the positive or negative value of the polarity associated with each pixel event. It is also important to note that the events generated are not synchronized with the internal clock, creating the need for the timestamp recorded with each event. These events are then streamed to

the onboard processor through a multiplexing technique referred to as address-event representation (AER) that serves to maintain the absolute order in which events occurred. This system and addressing scheme have been shown to register events on the microsecond scale, making the DVS ideal for applications requiring extremely fast response times. The overall benefit of this asynchronous, change-based architecture is superior power efficiency, temporal resolution, and dynamic range as well as drastically reduced data rate as compared to conventional cameras [1,2]. After several iterations on similar biomimetic vision systems, the asynchronous time-based image sensor (ATIS) has emerged as one of the more well-developed variations on the base DVS design. While it retains the strengths of the DVS architecture, the ATIS also encodes the relative intensity of luminance in the timing of events, enabling it to reconstruct full gray-scale images in addition to the binary events generated by the DVS. The ability to reconstruct variable-intensity images allows for more traditional image-processing and computer-vision techniques, while still leveraging the high-speed data acquisition of the asynchronous sensor [3].

The primary focus of this research is to exploit the capabilities of DVSs for object tracking within the context of space-based observation, particularly in a low-Earth orbit (LEO) environment. The space environment imposes a unique set of challenges on computer vision that impact both the software and underlying hardware involved. First, and perhaps most fundamental, of these challenges is the size, weight, power, and cost (SWaP-C) constraints placed on hardware. DVSs excel in this regard as they are both lightweight and power-efficient, while also requiring fewer computing resources to perform event-based image processing. These qualities make them ideal for deployment on space platforms that make use of embedded architectures with strict power constraints. Another challenge faced by object tracking in space is the relative visibility of certain objects of interest. A significant percentage of space debris is made up of objects mere centimeters

in size, and which have exceptionally low reflectance while travelling at extremely high speeds. Despite their small size, the high speed of debris poses a serious danger to space platforms, which stand to benefit greatly from autonomous methods of avoidance [4,5]. Although some large-scale, ground-based optical solutions have been able to track exceptionally small space debris, the high dynamic range (HDR) and temporal resolution of DVSs may prove useful for autonomous collision avoidance onboard space platforms. Furthermore, the change-based detection properties of DVSs, coupled with their HDR and relatively low data rate, could be ideal for detecting aerial objects on Earth. Recent work using DVSs in conjunction with ground-based telescopes has demonstrated the potential of event-based approaches to tracking celestial objects. This research compared several bio-inspired vision sensors, including a DVS and ATIS, for the purposes of tracking objects in both low-Earth orbit and geosynchronous orbit (GEO) during daytime lighting conditions, displaying the efficacy of the HDR of bio-inspired sensors [6].

Another challenge of particular interest in this research is the effect of radiation on spaceborne hardware. Radiation effects are typically classified as either transient or cumulative, from which correct operation may or may not be recoverable. Both types of radiation effects can pose significant issues for space missions, though transient single-event effects (SEEs) can often be managed by simple power-cycling of hardware or through standard hardware and software dependability techniques [7]. For imaging equipment, SEEs can manifest as salt-and-pepper noise that can be mitigated via common noise-filtering techniques when processing captured images. However, not all SEEs are transient, and some can cause permanent damage to sensing equipment that results in affected pixels being latched in an excited or unexcited state. Cumulative total ionizing dose (TID) effects are of relatively greater concern due to the long-term deleterious effects on hardware [8]. TID studies with CMOS cameras have shown that proton and heavy-ion radiation

common in space can significantly impact photodiode responsivity and contribute to long-term degradation [9]. Other studies conducted on bipolar transistors, such as those that compose operational amplifiers, have shown significant impacts on voltage response with increasing TID [10]. Given the reliance of DVS upon CMOS components, they will likely experience many of the same types of degradation as other types of conventional CMOS cameras [11].

2.0 Event-Based Vision Techniques

The following section explores several approaches to object detection/tracking and noise filtration using asynchronous event stream data. These approaches differ in their interaction with the asynchronous DVS data, where some approaches reconstruct frames and apply conventional image processing techniques while others use the one-dimensional event data directly. These differing approaches form the basis of the comparison studies performed in this work. This section also introduces several existing tracker filtering methods and accuracy metrics that are used to supplement the event-based algorithms used in this work.

2.1 Object Detection and Tracking

The asynchronous nature of DVS data introduces new avenues for image processing given that the pixel data does not exist in conventional image frames. It is possible to use conventional image-processing and computer-vision techniques by integrating the events over a predefined period of time and then reconstructing conventional frames from these events based on the x and y pixel locations supplied. However, reconstructing frames from event data drastically slows the overall execution time of image processing algorithms, removing much of the benefit of the DVS versus conventional cameras. Furthermore, the high temporal resolution of event data is lost in reconstructed frames since events occurring at the same pixel location within the integration time will be reduced to a single event in the resulting frame. Nonetheless, recent work has also shown that a hybrid approach can be taken where conventional feature detection algorithms, such as the

Harris corner detector, are used to locate features within reconstructed frames which are then tracked in the asynchronous event stream [12]. However, processing the events entirely in an asynchronous manner can significantly reduce the execution time of various operations as well as decrease the overall computational load beyond even hybrid approaches. To date, several common image processing algorithms have been modified to work solely within an asynchronous context, while maintaining the accuracy derived from the original implementations. Event-based cluster trackers have proven to be quite effective at tracking large objects such as vehicles, which have clear positive and negative polarity edges, at extremely high framerate and with low computational resource usage [13]. Algorithms for asynchronous optical-flow calculation have exhibited superior speed and resource usage, while also retaining comparable performance to frame-based approaches [14]. An event-based Hough circle transform demonstrated the ability to perform high-speed, multi-object tracking within the context of microparticle tracking [15]. Additionally, spiking neural networks (SNNs) have been shown to naturally complement the spiking nature of DVS pixels. In recent work, SNNs have been used to asynchronously detect and track lines via clusters of spiking neurons representing line parameters in the Hough space. However, this approach scales poorly with increasing DVS pixel array size due to the large increase in the number of neurons required [16].

One technique of particular relevance used in this research is the event-based Gaussian blob tracker introduced in [17]. This approach allows for both object detection and tracking by instantiating a series of trackers whose position and shape in the visual field are defined by the parameters of their bivariate Gaussian distribution. Every new event generated by the sensor is evaluated for its probability of belonging to each of the existing trackers, spawning a new tracker if no tracker has a score beyond a predefined threshold. An exponentially decaying activity score

is also associated with each tracker in order for trackers to be deactivated or removed entirely after long periods of no excitation.

2.2 Noise Filtration

Several means of noise filtration have also been adapted for use with asynchronous event streams. One such method exploits the temporal aspect of the event stream by requiring each incoming event to be supported by neighboring events within a certain predefined time threshold. A two-dimensional array of the camera's visual field records the timestamp of the most recently generated event within a window. When a new event is generated, the event is only passed on if the timestamp at the corresponding pixel coordinate has a timestamp more recent than the chosen support time [13]. An issue with using this method for space applications is that objects of low reflectance may not always induce enough luminance change to generate an event. As a result, objects appearing as a single pixel may not necessarily excite every pixel in its path of motion and thus true events could be eliminated by the filter. Event-based optical flow has also been adapted for noise filtration by approximating the "lifetime" of events defined as the time required for adjacent pixels to be excited. Events with lifetimes close to zero are considered noise and omitted from the resulting event stream. While this method is effective, it requires events to be stored within a spatiotemporal window and can be computationally intensive with increasing window sizes [18].

Another technique relies on the supposition that actions and objects of interest in the foreground will generate relatively more events than the background. The visual field of the sensor is divided into an arbitrary number of cells that then maintain an exponentially decaying record of

the activity occurring within each of them. Only cells with activity scores higher than the average activity of all cells will pass their corresponding events through the filter [19]. A modified form of this noise suppression is employed in this research. An alternative approach has been developed that generates a sparse representation within the cells using the K-SVD algorithm, which has been used to denoise both individual frames and videos composed of reconstructed frames. Although this approach thoroughly denoises tested frames, it is computationally time-consuming and has not yet been shown to operate in real-time [20, 21].

2.3 Tracker Smoothing and Metrics

Even with the inclusion of noise filtration, event-based blob tracking solutions rely on position updates that pull the tracker in opposing directions despite accurately tracking objects as a whole. This non-uniform movement occurs because the events associated with an object do not necessarily have timestamps temporally ordered in the direction of the object's physical motion. These constant changes of direction necessitate some form of predictive filter in order to smoothly track object trajectories. Kalman and extended Kalman filters have been established as extremely effective for object tracking, but the overhead imposed can severely impact execution time [22]. Given the high-speed data acquisition of the DVS, event-based algorithms must be able to cope with potentially high-activity scenes generating large numbers of events. One alternative approach to Kalman filtering makes use of double exponential smoothing prediction (DESP), a common data-forecasting method, and was demonstrated in tracking head and hand movements. The experimental results showed comparable accuracy to the Kalman filtering approach, but with 135 times faster performance [23]. Assuming reduced or eliminated noise events and the generally

consistent motion expected for orbital objects of interest, this approach is well-suited for smoothing event-based tracker trajectories.

In addition to smoothing tracker trajectories, the error of the tracker position is assessed by several means adapted from previous works. One metric used to predict tracking failure is the forward-backward error typically employed with median flow tracking. This technique involves comparing the future and past trajectories of a given tracker and assigning an error value based on a chosen measure of distance. In an event-based context, this error can be used to suppress trackers activated by areas of constantly changing motion, which most likely do not correspond to objects of interest [24]. Lastly, a common metric for measuring tracker precision is the intersection over union (IoU) calculated between the given tracker and a ground-truth representation of the object of interest. This metric has been used extensively to compare the performance of different tracking algorithms and is used in this work to compare asynchronous and frame-based approaches with noise suppression [25].

3.0 Space Tracking Approach and Testing

The following section details the algorithm used for object tracking and noise suppression intended for space-based applications. The blob-tracking aspect of this research relies on the event-based Gaussian tracker technique established in [17] with slight modifications made to emphasize tracking single-point sources. This emphasis is necessitated by the large distances at which space objects are to be tracked in relation to the comparatively low resolution of the DVS (640×480). The event-based Gaussian trackers allow for both efficient object detection and tracking but would be susceptible to noise and attraction to non-target objects in space environments. The asynchronous approach attempts to mitigate this behavior by introducing tracker suppression in addition to noise suppression. Both asynchronous and conventional frame-based approaches are presented and then compared across a variety of metrics. The frame-based approach differs in that it accumulates events over a predefined interval of time and then constructs an image frame from the resulting array. The reconstruction of frames allows more conventional image-processing techniques to be applied, but it also increases the execution time and computational resources required.

3.1 Tracking Algorithm

The following section details the blob tracking algorithms adapted and analyzed in this work for object tracking. The section also elaborates on the trajectory projection and interpolation used in conjunction with blob tracking.

3.1.1 Event-Based Gaussian Blob Tracking

As stated previously, each event generated by the DVS is represented by a vector of the form $e_i = [x_i, y_i, t_i, p_i]$, where the x and y values correspond to the pixel location, t to the timestamp of the event and p to the polarity of the event. The polarity may only take on a value of $p \in \{1, -1\}$, indicating an increase or decrease in luminance respectively. The motion of objects can then be visualized as a point-cloud of events modeled by a bivariate Gaussian distribution, $N(\mu, \Sigma)$. The parameters of the Gaussian distribution used to model the position and shape of a tracked object are defined as

$$\boldsymbol{u} = [\boldsymbol{x}, \boldsymbol{y}]^T \qquad \text{3-1}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}. \qquad \text{3-2}$$

As each new event is generated, the probability of the event being associated with an active or inactive tracker is calculated as

$$\boldsymbol{p}_i(\boldsymbol{u}) = \frac{1}{2\pi} |\boldsymbol{\Sigma}_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{u}-\boldsymbol{u}_i)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}-\boldsymbol{u}_i)} \qquad \text{3-3}$$

where u_i denotes the corresponding locations of active and inactive trackers. The event is then associated with the tracker with the highest calculated p score and the parameters of the tracker are updated according to the weighted update calculation

11

$$u_t = \alpha_1 u_{t-1} + (1 - \alpha_1)u \qquad\qquad 3\text{-}4$$

$$\Sigma_t = \alpha_2 \Sigma_{t-1} + (1 - \alpha_2)\Delta\Sigma \qquad\qquad 3\text{-}5$$

where

$$\Delta\Sigma = \begin{bmatrix} (x - u_{tx})^2 & (x - u_{tx})(y - u_{ty}) \\ (x - u_{tx})(y - u_{ty}) & (y - u_{ty})^2 \end{bmatrix}. \qquad\qquad 3\text{-}6$$

If no tracker has a calculated p score above a certain predefined threshold δp, a new tracker is instantiated with mean centered on the new event's location and covariance matrix initialized to predefined values. In this work, a δp of 0.001 was used while the starting values of the covariance matrix were varied experimentally. In addition to the parameters of the bivariate Gaussian, each of the trackers has an activity score that is updated according to the equation

$$A_i(t) = \begin{cases} A_i(t - \Delta t)e^{-\Delta t/\tau_1} + 1, \text{if } p_i(u) > \delta p \\ A_i(t - \Delta t)e^{-\Delta t/\tau_1}, \text{otherwise} \end{cases} \qquad\qquad 3\text{-}7$$

where Δt is the time elapsed since the last tracker update and τ_1 is a constant chosen to tune the rate of tracker deactivation. If the activity A_i falls below a certain predefined threshold, the tracker will become inactive and no longer displayed. In the asynchronous tracking approach, every event is processed in the order in which it is received. However, while the frame-based approach retains the timestamps associated with each event, it evaluates all events in order at discrete time steps after noise suppression has taken place. In [17], two activity thresholds were used to differentiate

between deactivated and destroyed trackers. However, given that trackers in this research are intended for single-point sources that may have activities as low as one between updates, the activity threshold is chosen such that any activity is sufficient for the tracker to be considered active and no trackers are permanently destroyed. This can lead to large numbers of inactive trackers, but also ensures that small objects of interest will be tracked effectively.

3.1.2 Tracker Trajectory Tracking

In order to monitor the trajectory of the Gaussian blob tracker, its position is smoothed using DESP at regular intervals of time τ, which are chosen at runtime. With this method, the x and y positions of the blob tracker are treated as a time series of points modeled with a linear regression equation whose y-intercept and slope vary over time. At each multiple of time τ, two smoothing statistics are calculated as

$$\overrightarrow{Su_\tau} = \alpha_3 \overrightarrow{u_\tau} + (1 - \alpha_3)\overrightarrow{Su_{\tau-1}} \tag{3-8}$$

$$\overrightarrow{Su_\tau}^{[2]} = \alpha_3 \overrightarrow{u_\tau} + (1 - \alpha_3)\overrightarrow{Su_{\tau-1}}^{[2]} \tag{3-9}$$

where α_3 is an update parameter chosen to determine the degree of exponential decay and $\overrightarrow{u_\tau}$ is the vector representing the blob tracker's x and y location at time τ. The first smoothing statistic, $\overrightarrow{Su_\tau}$, represents the smoothed average value of event positions associated with the tracker, while the second, $\overrightarrow{Su_\tau}^{[2]}$, captures the smoothed trend in event positions, i.e. the tracker's motion. With these smoothing statistics, the tracker position at time $\tau + 1$ is then forecasted according to the equation

$$\overrightarrow{u_{\tau+1}} = \overrightarrow{b_o}(\tau) + \overrightarrow{b_1}(\tau+1) \qquad \text{3-10}$$

Where

$$\overrightarrow{b_1}(\tau) = \frac{\alpha}{(1-\alpha)}\left(\overrightarrow{Su_\tau} - \overrightarrow{Su_\tau}^{[2]}\right) \qquad \text{3-11}$$

$$\overrightarrow{b_0}(\tau) = 2\overrightarrow{Su_\tau} - \overrightarrow{Su_\tau}^{[2]} - \tau\,\overrightarrow{b_1}(\tau) \qquad \text{3-12}$$

The values calculated here, $\overrightarrow{b_1}(\tau)$ and $\overrightarrow{b_0}(\tau)$, represent the estimated slope and y-intercept respectively of the linear regression fitting the given tracker's position over time. Since these values vary over time, the α_3 parameter controls the extent to which new points affect the linear regression fit to the tracker's position. As a result, very small values on the order of 1×10^{-4} are chosen due to the large number of events involved in tracker updates. Finally, the smoothed trajectory of the tracked object is then interpolated between time steps with a cubic Bezier curve fit with the equation

$$B(t) = (1-t)^3 U_0 + 3(1-t)^2 t U_1 + 3(1-t)t^2 U_2 + t^3 U_3 \qquad \text{3-13}$$

where U_i indicates the $[x, y]^T$ position of points obtained from the smoothed tracker trajectory. In this equation, U_3 refers to the most recent tracker position in time obtained via smoothing, while the remaining points U_i refer to previous points in its trajectory.

Since new trackers are constantly spawned when an event has no tracker with p score above the given threshold, it is possible, and in practice quite likely, that multiple trackers will begin to overlap on the same object being tracked. To remedy this problem, the distances between and activities of active trackers are compared at each discrete timestep before curve-fitting takes place. Trackers that overlap with adjacent trackers of relatively higher activity are deactivated and omitted from curve-fitting.

3.2 Noise Filtration

This section describes both the frame-based and asynchronous noise suppression approaches compared in this work. Both filtration methods are applied to all DVS data streams received during experimental testing and used the same method of object and trajectory tracking outlined in the previous section.

3.2.1 Frame-Based Noise Filtration

The noise-suppression techniques employed differ between the asynchronous and frame-based approaches. For the frame-based approach, a conventional image frame is built by tallying the presence of events at each pixel location regardless of polarity or frequency. Once an event is received with timestamp greater than or equal to a multiple of the integration time chosen, a simple summation kernel is convolved with the integrated image according to

$$g(x,y) = \sum_{u=-a}^{a} \sum_{v=-b}^{b} K(u,v)I(x-u, y-v) \qquad \text{3-14}$$

15

where $I(x, y)$ is the integrated event frame and K is the kernel

$$K = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \qquad\qquad 3\text{-}15$$

Given that the outputs of this operation are fixed to integer values, $g(x, y)$ will only have values greater than zero when an event has at least three neighboring events within the integrated frame. This function therefore serves to mask any pixel location with fewer than three neighboring events occurring within the integration time τ. Since objects travelling within the DVS' view should ordinarily induce positive polarity events immediately followed by negative polarity events, the threshold of three neighboring events is chosen to ensure the events in question constitute an actual object in the sensor's view.

3.2.2 Asynchronous Noise Filtration

The asynchronous approach uses a version of the event-based dynamic background suppression introduced in [19], except that every pixel is associated with its own cell rather than a large number of pixels falling into the same cell. Reducing the number of events grouped for noise suppression is necessary for tracking targets that may be exceptionally small in size and induce small amounts of activity. In addition, cells are suppressed using a two-sided activity threshold rather than the average of all current cell activity. This change to the suppression technique serves to filter events occurring at pixels of both low and high activity, resulting in band-pass behavior

with respect to the frequency of events in question. As an event is received, the corresponding cell is updated in a similar fashion to the activity associated with trackers as

$$A_u(t) = A_u(t - \Delta t)e^{-\Delta t/\tau_2} + 1 \qquad\qquad 3\text{-}16$$

where Δt is the time elapsed since the last pixel update, A_u is the activity associated with the pixel at $u = [x, y]^T$ and τ_2 is a constant chosen to tune pixel activity decay. Thresholds δ_1 and δ_2 are chosen such that only events occurring at pixels with activity $\delta_1 < A_u(t) < \delta_2$ will be processed by the Gaussian blob tracking algorithm. Negative polarity events are ignored in this filtering since motion of exceptionally small objects is expected to generate positive polarity events immediately followed by negative polarity events on their trailing edges. This programmatic filtering of noise events also serves to filter large regions of change in the sensor's view such as sections of the Earth that might lie in the sensor's view. Since the sensitivity of the hardware itself would need to be maximized in order to track distant, dim objects, portions of the Earth would generate a significant number of spatially and temporally close events. These events will be filtered by the asynchronous approach and therefore significantly reduce computational load and lessen the number of false positives in tracking.

3.3 Tracker Suppression

Since the intent of this research is to track objects that may be as small as a single pixel in the DVS's view, parameters are chosen for the Gaussian blob tracker algorithm such that trackers will be spawned even for regions of exceptionally low activity. Setting the activation threshold so

low can in turn lead to some noise events behaving similarly to objects of interest and therefore escaping noise suppression. As a result, the trackers themselves are also suppressed by two separate metrics, where suppression indicates that the corresponding tracker is deactivated. First, the Gaussian tracker algorithm is modified such that negative polarity events are accumulated separately from negative polarity events as

$$N_i(t) = \begin{cases} N_i(t - \Delta t)e^{-\Delta t/\tau_1} + 1, if \ p_i(u) > \delta p \\ N_i(t - \Delta t)e^{-\Delta t/\tau_1}, otherwise \end{cases} \qquad \text{3-17}$$

where $N_i(t)$ is the negative polarity activity of tracker i at time t. A threshold is chosen at runtime such that trackers with negative polarity activity below this threshold will not be subject to DESP or trajectory curve fitting. These trackers are essentially omitted as not corresponding to an object of interest, but the trackers are still subject to being associated with new incoming events. This suppression serves to omit both trackers that may be associated with noise events as well as trackers that may be attracted by larger objects, such as sections of the Earth, which are not the intended target of this work. Second, a forward-backward error statistic is calculated at each multiple of the integration time τ when applying DESP. The Euclidean distance between the next predicted location $u_{\tau+1}$ of the smoothed trajectory and the last update location of the tracker is calculated and added to a running average of the tracker's forward-backward error as shown below.

$$E_t = E_{t-1} + \sqrt{(x_{\tau+1} - x_t)^2 + (y_{\tau+1} - y_t)^2} \qquad \text{3-18}$$

$$\overline{E}_t = \frac{E_t}{n_{events}} \qquad \text{3-19}$$

18

As with the negative polarity activity, an error threshold is chosen such that, if a tracker's forward-backward error exceeds the threshold, it will be deactivated and hidden from trajectory fitting. It should be noted that the frame-based approach does not employ tracker suppression due to spatially solitary events always being filtered. As a result, the frame-based method's ability to track point-source objects is limited, but the number of trackers instantiated is greatly reduced in comparison to the asynchronous method. The entirety of both algorithms is described with the pseudocode detailed in Algorithm 1 and 2.

Algorithm 1: Frame-based Tracking Algorithm

Initialize lastFrame and 2D image array;

for *each incoming event, e(x,y,t,p)* **do**

    Store e(x,y,t,p) at [x,y] position of 2D image array;

    if *(t-lastFrame)* $\geq$ *integrationTime* **then**

        Perform noise suppression with Eq. 14 and 15 on image array;

        for *each event in image array* **do**

            Determine closest tracker according to Eq. 3;

            Update all tracker activities according to Eq. 7;

            Promote(or demote) trackers to active layer (to hidden layer);

            Update position/shape of tracker with highest p according to Eq. 4 and 5;

            Apply double-exponential smoothing to updated tracker position (Eq. 8-12);

        end

        Fit Bezier curve to smoothed active tracker positions (Eq. 13);

        Draw trackers, trajectories and events on output image frame;

        Flush image frame and set lastFrame to last event's t;

    end

end

Algorithm 2: Asynchronous Tracking Algorithm

Initialize lastFrame and 2D array of pixel activity cells;

for *each incoming event, e(x,y,t,p)* **do**

    Update pixel cell at (x,y) according to Eq. 16;

    if $\delta_1 \leq A_u(t) \leq \delta_2$ *or event has negative polarity* **then**

        Determine closest tracker according to Eq. 3;

        Update all tracker activities according to Eq. 7 and 17;

        Promote(or demote) trackers to active layer (to hidden layer);

        Update position/shape of tracker with highest p according to Eq. 4 and 5;

        Apply double-exponential smoothing to updated tracker position (Eq. 8-12);

        Calculate tracker forward-backward error and add to average according to Eq. 18 and 19;

    end

    if *(t-lastFrame)* $\geq$ *integrationTime* **then**

        Draw events on output image frame;

        for *each active tracker* **do**

            if *tracker* $N_i(t) \geq$ *neg. polarity threshold and forward-backward error average* $\leq$ *error threshold* **then**

                Fit Bezier curve to smoothed active tracker positions (Eq. 13);

                Draw tracker and trajectory and events on output image frame;

            end

        end

    end

    Flush image frame and set lastFrame to last event's t;

end

3.4 Experimental Setup

In order to compare the asynchronous and frame-based approaches and assess their ability to track point source objects, experimental testing was conducted with varied velocity, luminance contrast, and distance to a point-source of interest. In this experiment, the point-source of interest consisted of a laser pointer projected onto a background illuminated by two direct current (DC), variable-intensity LED lamps. The lighting intensity of the lamps was varied at several discrete levels in order to assess tracking with different contrast levels. DC lighting was necessary, in addition to the room being darkened, due to the DVS's tendency to detect flickering from alternating current (AC) lighting. The laser pointer was mounted onto a stepper motor controlled via microcontroller board in order to control the angular velocity of the target at discrete micro-stepping levels.

3.4.1 Testing Conditions

Table 1 and 2 list the contrast ratios and speeds measured for each of the discrete levels used in testing. Tests were conducted using each combination of contrast ratio and speed mode for a total of 30 trials under base conditions. To study the effect of having a smaller target area, the distance from the camera to the target was then increased and additional tests were conducted with all contrast ratios and the Full, Quarter, and Slowest speed modes in order to cover the breath of speeds chosen. These 15 additional trials are denoted among the results with the corresponding contrast level and "Incr. Distance" in order to distinguish them. Tests were conducted at 2.5 meters and 5.0 meters to target respectively due to the constraints of the testing area.

Table 1: Contrast Conditions

Contrast #	Contrast Ratio
1	∞
2	**25.87**
3	**6.23**
4	**3.01**
5	**1.09**

Table 2: Speed Conditions

Speed Mode	Angular Velocity (Deg/s)
"Full"	**455.7**
"Half"	**232.26**
"Quarter"	**136.88**
"Eighth"	**63.16**
"Sixteenth"	**25.68**
"Slowest"	**7.4**

Since the targets in the 45 preceding trials follow a simple, horizontal and linear path, an additional 11 tests were conducted that included sudden and/or persistent changes in the direction of motion as well as multiple targets of interest. These additional tests are described in Table 3, several of which were repeated multiple times with different patterns of movement.

Table 3: Additional Tests

Experiment (# of Trials)	Description
Multiple Targets (3)	**Several targets with sudden movements and collisions between them**
High Contrast Non-Linear (3)	**Non-linear, varying-speed targets at the highest contrast, i.e. no backlighting**
Low Contrast Non-Linear (3)	**Non-linear, varying-speed targets at contrast level 4 (chosen due to accuracy loss at the lowest contrast level)**
Changing Direction (1)	**Target with persistent changes in motion, a spiraling motion, at highest contrast level**
Increased Distance (1)	**Non-linear, varying-speed target with 5m distance to target and highest contrast level**

3.4.2 Accuracy Metrics

The ground-truth for each test was generated using built-in OpenCV functions to locate contours and calculate minimum enclosing circles on noise-suppressed event frames [26]. Intersection-over-union and active tracking time metrics were then calculated by comparing the intersection of the event-based Gaussian blob trackers with the set of minimum enclosed circles. It should be noted that although the Gaussian trackers are elliptical in shape, the IoU metric was calculated using a circle circumscribed with the tracker's larger axis as its diameter. This approach results in a decrease in the IoU scores calculated but should not impact the active tracking time measured in each experiment since active tracking is determined by the tracker simply intersecting with the ground-truth circles across frames. Contrast levels were calculated as the ratio of laser pointer luminance to background lighting luminance as measured via luxmeter.

3.5 Event Simulation

While the experiments conducted in this work provide an analogue to the conditions that might be expected within a space environment, the scenarios posed during testing are obviously greatly simplified compared to those actually found in space. The experiments presented attempted to assess multiple aspects of relevance to object tracking in space, such as the sensors ability to discern small objects against backgrounds of varying illumination and the tracking algorithms' ability to track small objects moving at varying speeds. The additional testing also included multiple targets of interest in addition to non-linear movement patterns with sudden changes in direction to better emulate real world conditions. Even so, it is difficult for a ground-based

experimental setup to fully capture all of the conditions involved with object tracking on a real space platform. Furthermore, no DVS has yet been deployed on a space platform, so, to date, no publicly available event stream datasets exist. As a result, the ability to emulate an asynchronous event stream from conventional frame-based video recordings is extremely useful for assessing how a DVS might actually perform on the proposed task.

3.5.1 Previous Approaches to Event Stream Simulation

In order to produce a sufficiently accurate simulation of an event stream, several key characteristics of the DVS must be extrapolated from the available framed video data, chiefly, the change-based imaging and high temporal resolution. Several works have already explored creating event stream data from conventional videos, and the simulation used in this work adopts a similar approach. The earliest attempt at simulating asynchronous event streams simply uses the logarithm of the brightness sampled at each pixel stored in an array in memory. As each frame of conventional camera data is received, the difference between the new logarithmic brightness value and the previous is calculated, and if this difference surpasses the predefined threshold, positive or negative polarity events are generated accordingly. The simulator generates a number of spikes equal to the intensity difference divided by the chosen threshold, which are then assigned timestamps either all synchronous with the current image frame or linearly interpolated over the time period between frames. This linear interpolation would result in the generated spikes being equally distributed over the time period between frames [27]. This approach was later applied towards generating artificial, event-based datasets for visual odometry and SLAM algorithms hoping to leverage event-based techniques. Artificial scenes were created within the computer graphics program Blender, which were then converted to asynchronous event streams using the

method presented. In order to emulate DVS data using RGB pixel channels, the intensity at each pixel was calculated according to:

$$Y = 0.299R + 0.587G + 0.114B \qquad \text{3-20}$$

Given that the generated event stream should capture all intensity changes over the course of the simulated video, the event streams were validated by reconstructing the conventional video frames at time t according to:

$$log\ \hat{I}(u;t) = log\ I(u;0) + \sum_{0 < t_k \leq t} p_k C \delta(u - u_k) \delta(t - t_k) \qquad \text{3-21}$$

where $log\ I(u;0)$ represents the entire frame at time $t = 0$, p_k is the event polarity, C is the contrast threshold for event generation, and u_k is the x, y location of corresponding events [28].

In addition to this straight-forward event simulation, several extensions have been made to the basic algorithm with varying degrees of success. PIX2NVS is an event stream simulation framework that attempts to better capture the imaging properties of a DVS by using log-intensity, contrast-enhanced (LICE) brightness values. Luminance values at each pixel are first calculated using the 'perceptual' luminance of the pixel and its immediate neighborhood, followed by taking a weighted average of values in surrounding pixels. The spike trains are then generated according to similar manner as in [27], but with the threshold for event generation being compared to the minimum of a neighborhood of values rather than a single pixel. Upon validation testing, using LICE pixel values resulted in the simulated event stream having events temporally and spatially distributed more similarly to a true event stream, albeit with worse performance than log-intensity

approach [29]. Yet another framework, pyDVS, follows a similar approach to [27], but with several modifications that are also adapted in the simulation used in this work. Since most commercial cameras use gamma-encoded images whose brightness response does not vary significantly from a logarithmic response, intensity values from the conventional camera are used as-is rather than taking the logarithmic value. While event generation is determined in the same fashion as in [27], a number of spikes is generated up to a maximum number of spike 'bins' existing between each frame, where the number of spikes is calculated according to:

$$N_s = min\ (N_b, N_H) = min(N_b, \frac{\Delta B}{H})$$ 3-22

In this equation, N_b is the maximum number of bins, ΔB is the brightness difference between frames, and H is the chosen brightness threshold. However, the emulator uses spike-time encoding to reduce the overall number of spikes recorded in the event stream by only using the last bin in the spike chain generated. Timestamps are then assigned based on the previous frame and the time encoded by the generated spike:

$$R_{now} = R_{last} + N_sH.$$ 3-23

In addition to this technique, the emulator introduces several extensions that are also used in the emulator in this work. Firstly, adaptive thresholds for pixel intensity values are used such that pixels not being actively triggered become more sensitive time and, conversely, consistently stimulated pixels become less sensitive. This behavior is meant to emulate the manner in which physical pixels will slowly build up charge overtime in areas of slowly changing motion. Secondly,

lateral inhibition is a biological phenomenon in which neurons will inhibit the firing of neighboring neurons after firing themselves. This framework emulates this behavior by applying non-maximum suppression to small neighborhoods across the pixel array after calculating event generation in a new frame. As a result, only the pixels with the greatest change in intensity will generate events and thus drastically reduce the number of spikes present in the simulated event stream [30].

3.5.2 Event Stream Simulation Approach

The event simulation in this work draws heavily from the approach taken in [30] and is applied to freely available space platform footage in order to estimate performance within a space environment. Firstly, pixel luminance is calculated according to Equation 3-20 to emulate the DVS's response to RGB color data. With each new frame, the difference in luminance is calculated and the new value stored in the corresponding pixel array location. For difference values greater, positively or negatively, than a chosen threshold, a spike train is generated according to Equation 3-22. However, rather than encoding the magnitude of change in the timing of a single spike, a train of spikes is generated in each bin preceding the final bin calculated. This modification is made so that the blob tracking algorithm will still receive a sufficient number of update events for the trackers to be instantiated, moved, and reshaped smoothly. In order to emulate the makeup of event streams obtained from actual recordings, lateral inhibition is implemented to prevent large, dense cluster of events that would result from simple frame-by-frame intensity differencing. This was accomplished by applying non-maximum suppression across all pixel locations before spike trains were generated. Finally, the option of adaptive thresholding was implemented as follows:

$$H(u,t) = \begin{cases} H(u,t-1) \mathrel{+}= 5, & \frac{|\Delta B|}{H(u,t-1)} > 0 \\ H(u,t-1) \mathrel{-}= 5, & \frac{|\Delta B|}{H(u,t-1)} = 0 \\ H(u,0), & H(u,t-1) \le 0 \; or \; H(u,t-1) \ge 255 \end{cases} \qquad \text{3-24}$$

Since the threshold of each pixel is held separately, H(u, t) represents the threshold of the pixel at $u = [x\ y]^T$ at time t, which is incremented by 5 if spikes are generated, decremented by 5 if no spikes were generated, and returned to the original chosen threshold value if the threshold rises above the maximum of 255 or falls below the minimum of 0. Figures 1 and 2 show an example of space footage converted to an asynchronous event stream using the emulator detailed. The sample footage used was chosen due to the inclusion of both Earth and a section of space falling into the camera's field of view with the rocket being an object of interest that would be ideal for tracking with a DVS.
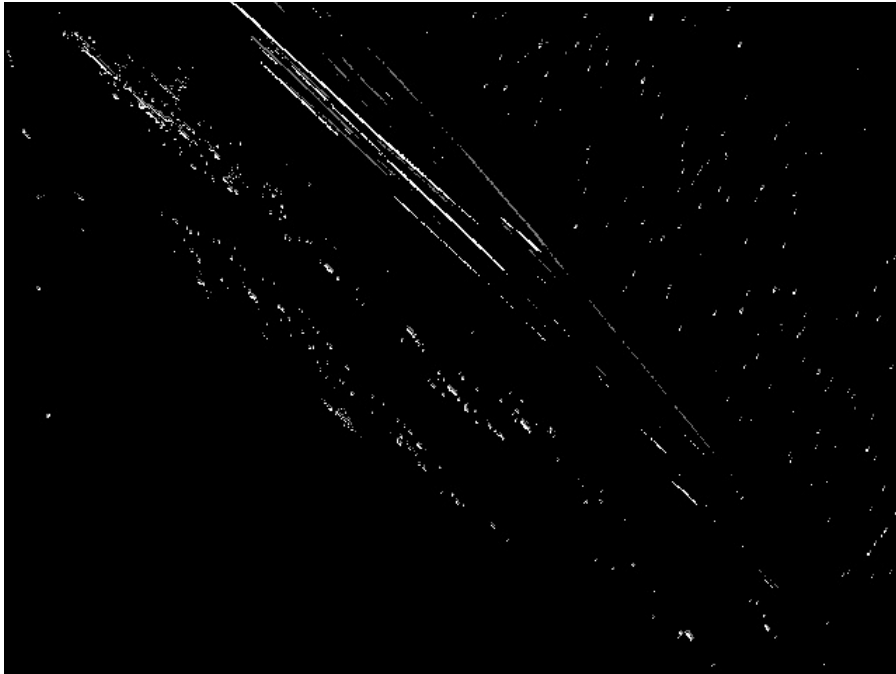
Figure 1: Sample Frame of Test Footage [31]

Figure 2: Reconstructed Frame from Simulated Event Stream

4.0 Experimental Results

The performance of the frame-based and asynchronous approaches was assessed with each combination of contrast, speed, and distance conditions previously presented as well as the additional 11 non-linear movement trials. Before evaluating accuracy across the experiments, threshold values used for noise-filtering in the asynchronous method were first tuned by inspection for each experiment. Filtering thresholds were chosen in order to maximize tracking accuracy in terms of percentage of actively tracked frames and with regard to the different speeds of the target. In general, slower targets necessitate a larger upper-bound threshold, δ_2, since they will repeatedly trigger events in a small range of pixels. In the same respect, the lower-bound threshold, δ_1, can be raised as well in order to filter additional non-target events without interfering with accurate target tracking. Due to the frame-based method filtering events according to the number of spatial and temporal neighbors, no tuning for the frame-based methods was required.

4.1 Noise Filtration

Figure 3 depicts the percentage of events that were filtered as noise by the asynchronous and frame-based approaches in each of the experiments. As target speed does not affect the number of events registered overall, filtered event percentages are averaged across all speed modes in experiments with the same contrast ratio.
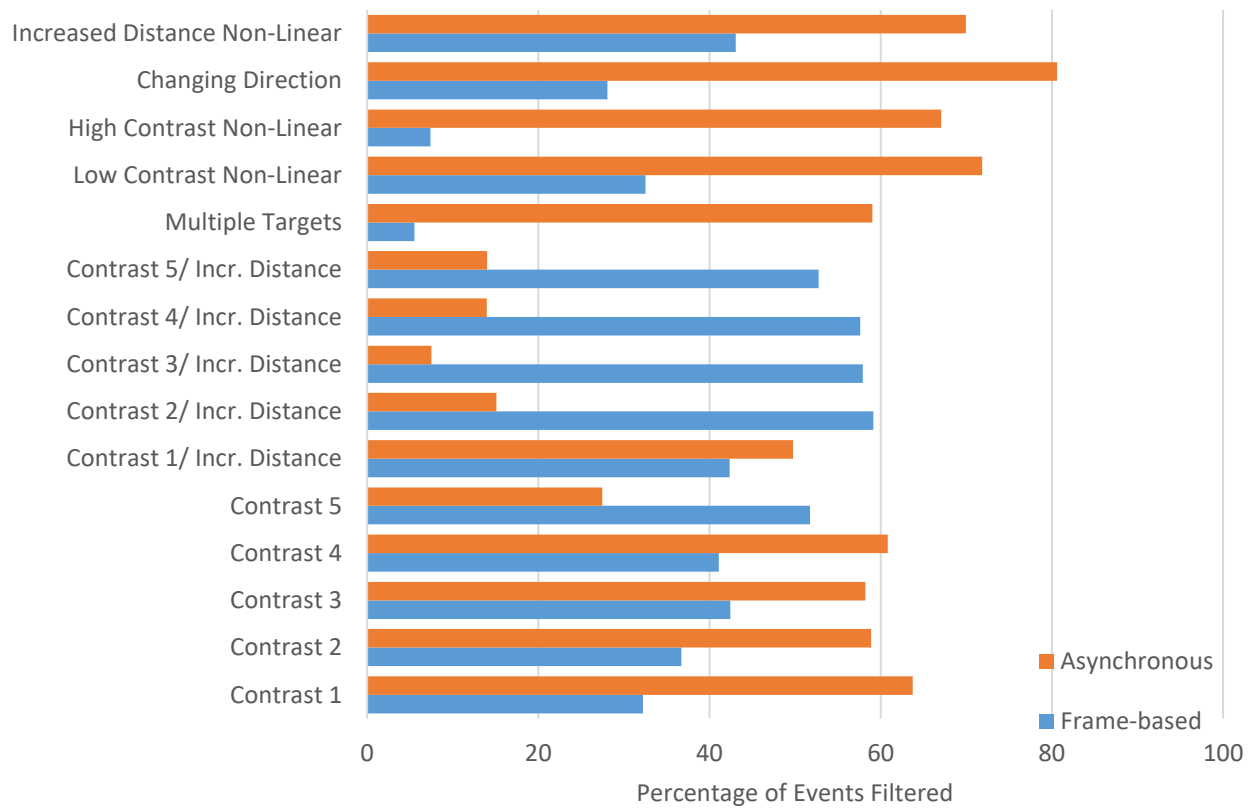
Figure 3: Noise Filtration by Experiment

Across most experiments, the asynchronous approach showed much larger percentages of filtered events than the frame-based approach. In some of the additional trials, the asynchronous approach reached up to approximately 80% average events filtered, such as in the case of the Changing Direction trial. Conversely, the frame-based approach had a maximum of only about 60% filtered events occurring in the trial with second contrast level and increased distance. The fact that the two approaches percentage of filtered events differed greatly between the different types of experiments also indicates that the two approaches differ in the context in which events are filtered. Since the frame-based approach filters events with few neighbors over time, this behavior suggests that the trials with increased distance exhibited far more solitary events as

reflected in the much larger percentage of filtered events. By contrast, the multiple target trials had many repetitive events occurring spatially and temporally close, resulting in extremely low filtering rates for the frame-based method, but much higher rates with the asynchronous method. While the total number of events observed in each experiment is largely determined by the length of the recording, variations in event numbers were observed with differing contrast levels as well. Although the 11 separate trials had varying recording lengths, the original contrast and velocity-controlled trials had roughly the same recording time across each. This fact indicates that lower contrast between the target area and background resulted in larger numbers of events that can be attributed to erroneous events occurring in the background. In ground applications, this phenomenon could be mitigated by significantly raising the sensitivity thresholds of the sensor itself, but this would not be possible on a space platform where maximum sensitivity is required to detect exceedingly dim and distant objects.

4.2 Performance

Figure 4 depicts the performance in average frames per second (FPS) for the two approaches as a function of the number of events generated. These performance numbers are drawn from each of the experiments previously detailed. It should be noted that the number of events shown is before noise filtration, which partially explains the difference in FPS for trials with approximately the same number of events. All trials were run using a four-core Intel Core i5-8250U 1.6 GHz processor under the Linux Ubuntu OS. The Bezier curve-fitting, frame drawing and frame-based noise-suppression portions of the algorithm were all parallelized using the C++ OpenMP API. Since the frame-based method involves applying a kernel convolution to the entirety

34

of reconstructed frames, the same number of calculations must be made regardless of the number of events in the event stream. As a result, the performance would be expected to be relatively constant across all experiments. Although execution times do appear to be consistent in the frame-based approach, some trials that exhibited larger or smaller numbers of unfiltered events, as well as more trackers being instantiated, resulted in small disparities in average execution time. Conversely, the asynchronous approach shows much greater variation in FPS versus the number of events, which is a result of the larger number of events filtered before tracking is performed. The number of inactive trackers instantiated also affects asynchronous performance, further adding to the FPS variation for experiments with approximately the same number of events.
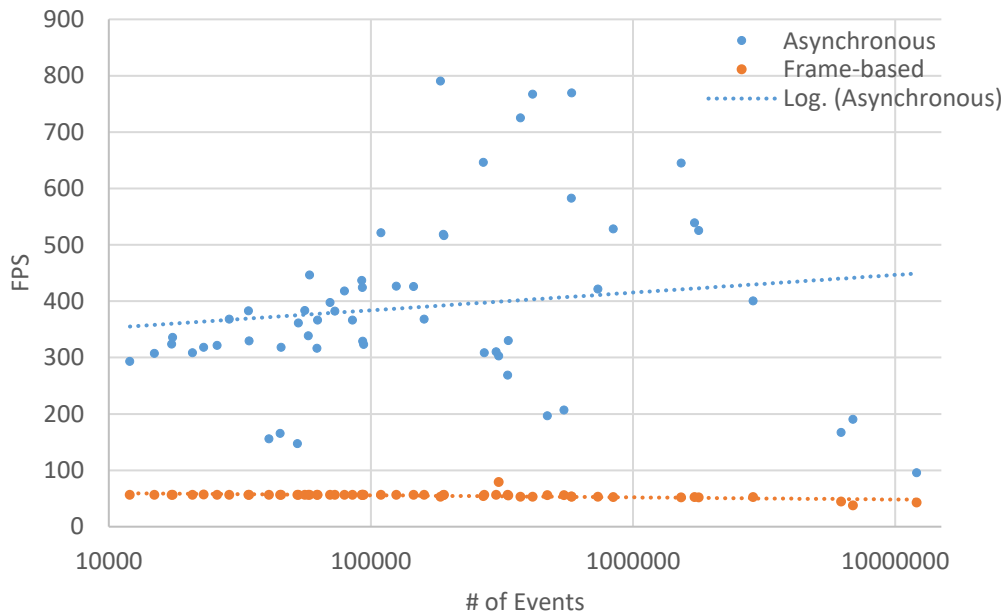
Figure 4: Performance in FPS vs. Number of Events

35

4.3 Tracker Instantiation and Suppression

Figures 5 and 6 display the number of trackers generated for each experiment and number of trackers suppressed in the asynchronous case respectively. In Figure 5, the number of trackers generated is reported as an average across all experiments with the same contrast level as the number of trackers instantiated is primarily dependent on the contrast. By contrast, Figure 6 shows the average number of trackers suppressed with experiments grouped by target speed since the asynchronous tracker suppression varies more with the speed of the target rather than brightness contrast. It should be noted that the vast majority of the trackers instantiated are inactive throughout most of the experiments and do not necessarily indicate a false positive in tracking. Full results can be seen in Figure 17 in Appendix.
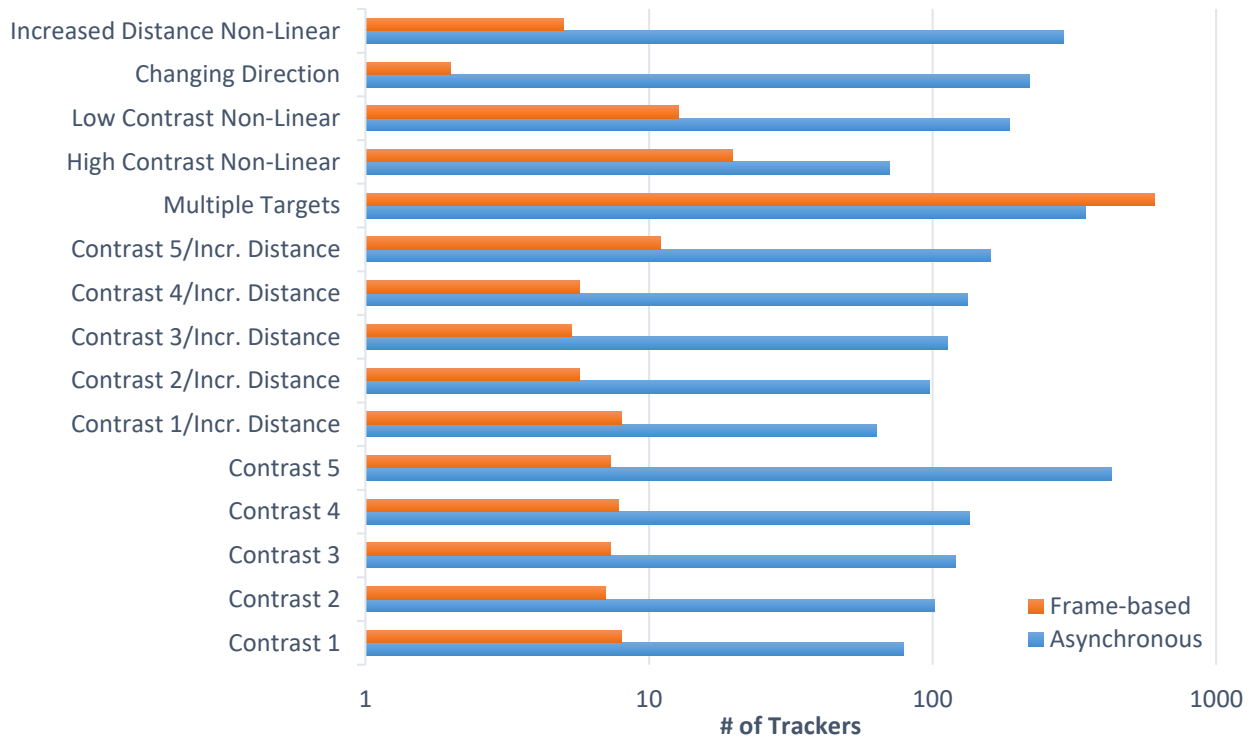

Figure 5: Average Number of Trackers Generated per Experiment Group

As expected, the asynchronous approach spawned considerably more trackers in each trial as a result of some noise events passing through filtration. However, there are a few notable exceptions, such as the multiple target trials, where the much higher percentage of events filtered by the asynchronous method resulted in comparatively fewer trackers being generated. Several trials at the slowest speed also exhibited no trackers instantiated by the frame-based method, which represented a complete failure to track the target. This failure was due primarily to the exceedingly small number of events generated by the target, which were in turn filtered by the frame-based noise suppression.
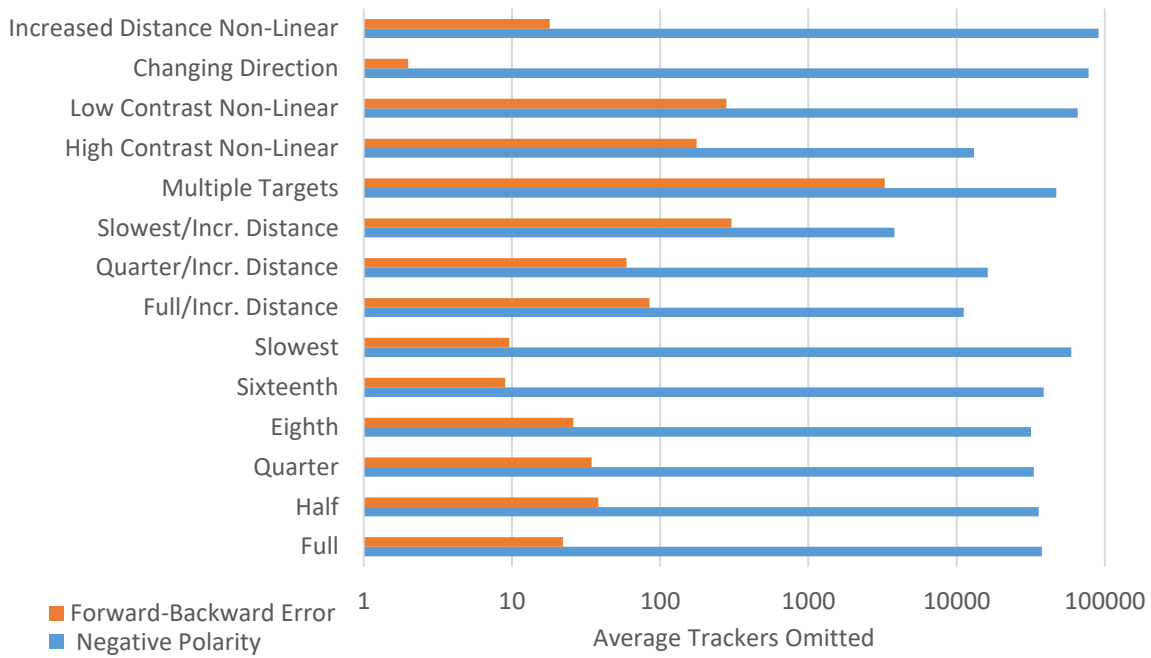
Figure 6: Number of Trackers Suppressed in Asynchronous Approach

It is important to note that the number of tracker omissions is cumulative across each trial, hence the number of omissions being greater than the number of trackers originally initialized. Furthermore, negative polarity thresholding is applied prior to forward-backward error

thresholding, leading to the majority of omissions being attributed to negative polarity. In some trials, the negative polarity threshold successfully suppressed all erroneous trackers, leaving none to be suppressed as a result of forward-backward error. Additionally, there were several instances in the trials with the slowest target speed and increased distance where no trackers were suppressed by negative polarity thresholding. During these trials, the exceptionally slow targets exhibited few or no negative polarity events during movement, which necessitated reducing the negative polarity threshold to zero. Full tracker suppression results can be seen in Figure 18 in Appendix.

4.4 Experimental Tracking Accuracy

Figures 7 and 8 show the general trends in tracking accuracy with averaged values across experiments with the same target speed. Tracking accuracy was measured as the intersection over union with the ground-truth position of the target per frame, while active tracking time refers to the percentage of frames where an active tracker intersected with the ground-truth target position.
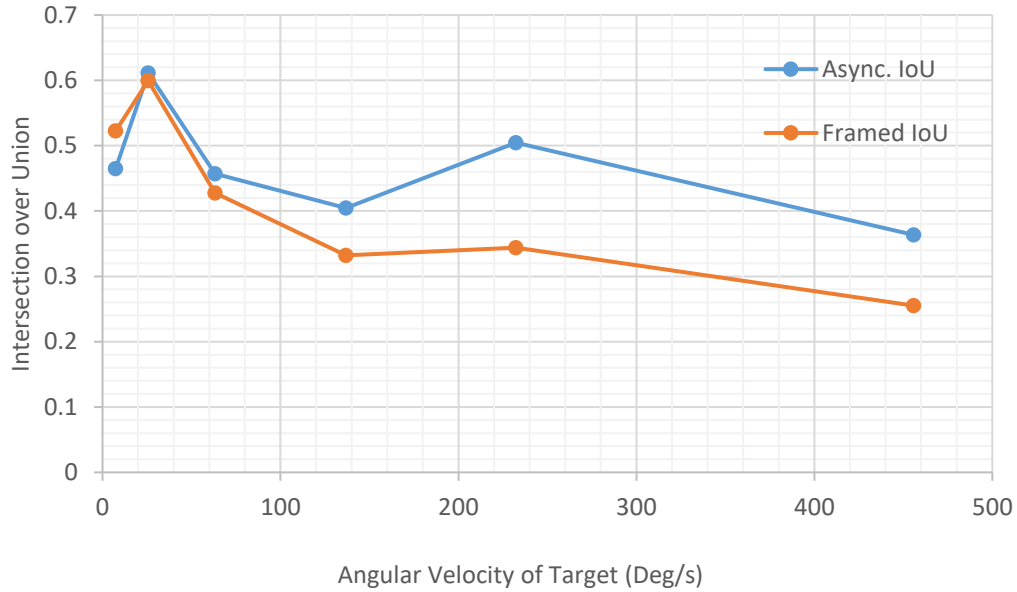
Figure 7: Intersection over Union vs. Target Speed

Note that active tracking time specifically means that the tracker was active and not suppressed, so frames that were not actively tracked may have still contained an inactive or suppressed tracker on target. This impact of tracker suppression on active tracking time can be observed in the relatively lower active tracking time of the asynchronous method for lower velocity targets. Although the target tracker was suppressed in some frames, the asynchronous method maintained relatively the same or better IoU than the frame-based method with the same velocity targets.
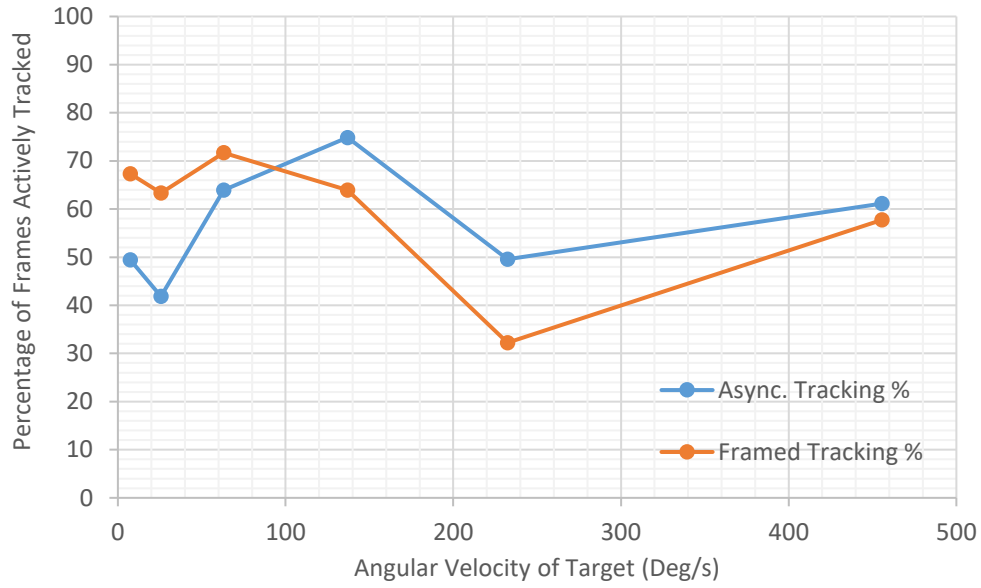
Figure 8: Percentage of Actively Tracked Frames vs. Target Speed

Figures 9 and 10 show the same accuracy metrics measured for the 11 additional trials with multiple targets and changes of target direction. Since these trials did not have precise measurements of target velocity, they are presented per experiment. The additional trials showed similar trends as the linear tests, with the asynchronous approach exhibiting better IoU measures across almost all of the experiments, but with comparable or relatively smaller percentage of actively tracked frames. While the frame-based method exhibited some instances of slightly better IoU, the asynchronous approach showed greatly improved IoU for trials with multiple targets.
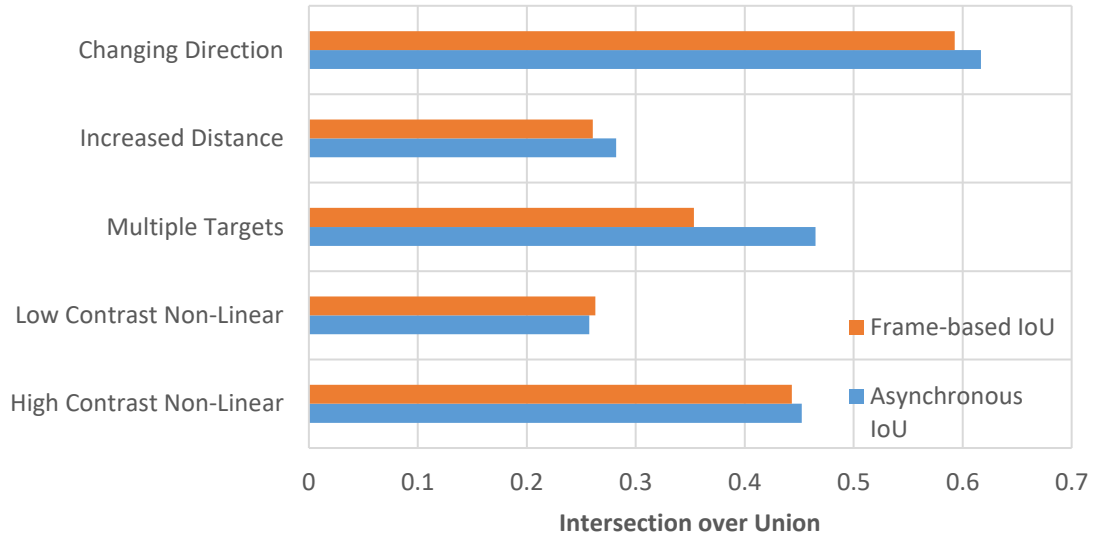
Figure 9: Intersection over Union per Experiment

Although many of the trials displayed only a slight difference between the percentage of frames tracked for both methods, the non-linear, increased distance trials showed a significant improvement for the asynchronous approach. Since the increased distance leads to a relatively smaller target size, this result supports the asynchronous method's proposed superior, single-point tracking ability.
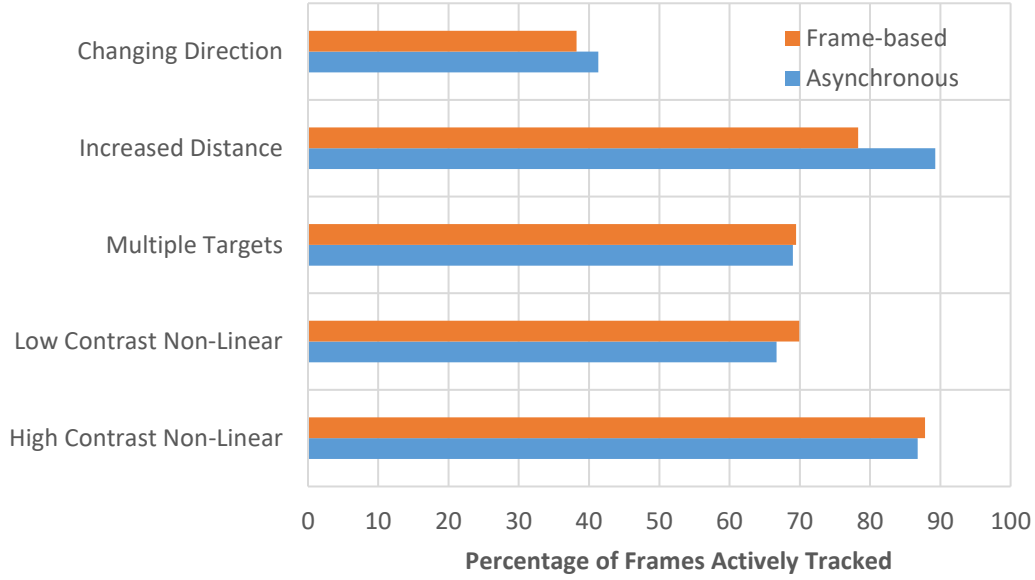
Figure 10: Percentage of Actively Tracked Frames per Experiment

Although the algorithm parameters were tuned to maximize tracking accuracy in the previous experiment analysis, additional metrics were recorded to assess tracking effectiveness within a range of tracking parameters. The graphs displayed in Figures 11 and 12 show averaged accuracy results with varied position update parameters for the trackers. The position update parameter refers to α_1 from Equation 3-4, which controls the magnitude of tracker displacement when a new event is associated with it. In general, larger α_1 values cause trackers to require greater number of events to result in significant movement of the tracker. This additional analysis was conducted with trial recordings at the highest level of contrast, i.e. no background lighting, but with varied speed and distance, in order to determine the effect of the position update factor on tracker accuracy. Disregarding the base differences between the accuracies of the two approaches, both methods showed the same general trends as a result of increasing the value of the position update parameter. In general, the range of 0.4 to 0.7 showed minor differences in both IoU and actively tracked frames, but both accuracy measures showed significant improvement beginning

42

at a value of 0.8, with some exceptions. Interestingly, the increased distance trials showed comparable or even better accuracy with much smaller update values and diminishing accuracy with increased values. This trend can be explained by the much smaller number of events seen in these trials, and thus the corresponding targets were composed of far fewer events as well. As a result, the position update parameter must be smaller such that fewer events are required for the trackers to accurately track the targets. Since these reported values are averaged across all trials, the average accuracy metrics are somewhat skewed due to the relatively low accuracies measured in the slowest target trials. Full results can be found in Figures 19, 20, 21, and 22 in Appendix.
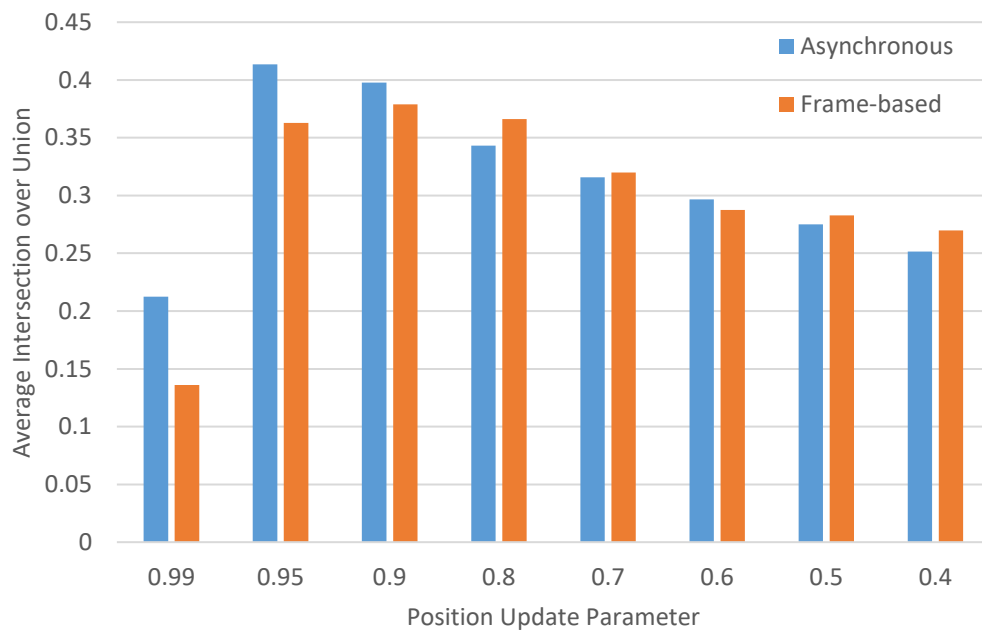

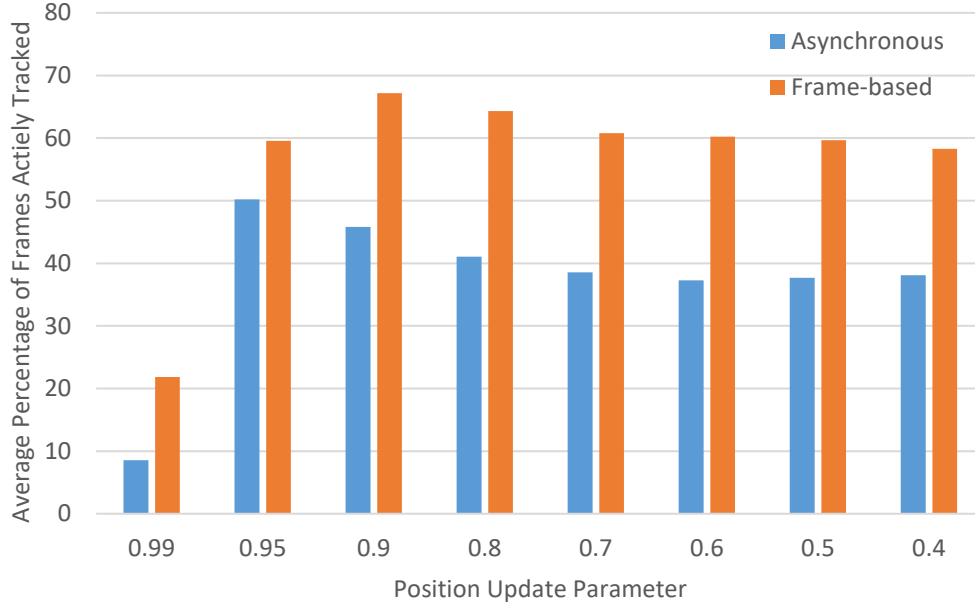**Figure 11: Intersection over Union vs. Position Update Parameter**

Figure 12: Percentage of Actively Tracked Frames vs. Position Update Parameter

4.5 Visualized Tracking Results

For demonstration purposes, Figure 14 (a-c) depicts the reconstructed frame output with ground-truth, frame-based, and asynchronous tracking results superimposed for several trials, while Figure 13 (a-c) shows the corresponding unfiltered frames. Positive- and negative- polarity events are indicated by black and grey pixels respectively. The ground-truth minimum enclosed circle is denoted by a red circle, while the frame-based and asynchronous trackers are overlaid with brown ellipses. Each of the trackers that is deemed to be tracking the target also displays a blue Bezier curve fit to its predicted trajectory obtained via DESP. As referenced earlier, both asynchronous and frame-based have many inactive trackers that are hidden in the reconstructed frames, though only active trackers are displayed. The events displayed in the frames are the result of the asynchronous filtering technique and several erroneous active trackers can be seen in each.

These erroneous trackers belong only to the asynchronous approach; however, these trackers have no estimated trajectory and promptly become inactive between frames. Trajectories are drawn for both the asynchronous and frame-based approaches, which can be seen to diverge most significantly in the trials containing multiple targets.
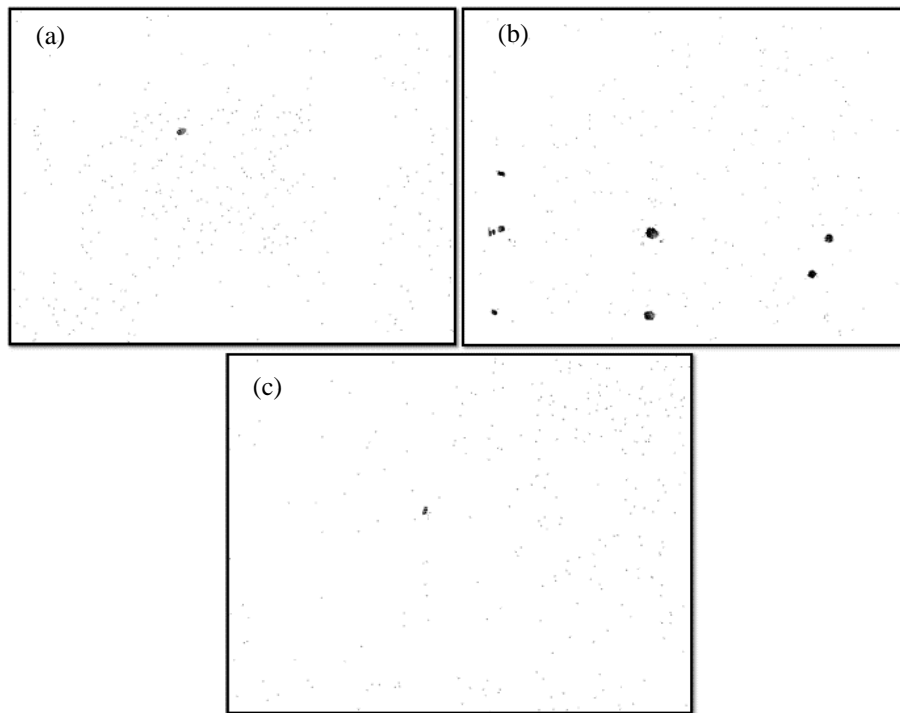

Figure 13: Original Unfiltered Frames (a) High Contrast 3 Experiment (b) Multiple Targets 1 Experiment (c) Changing Direction Experiment
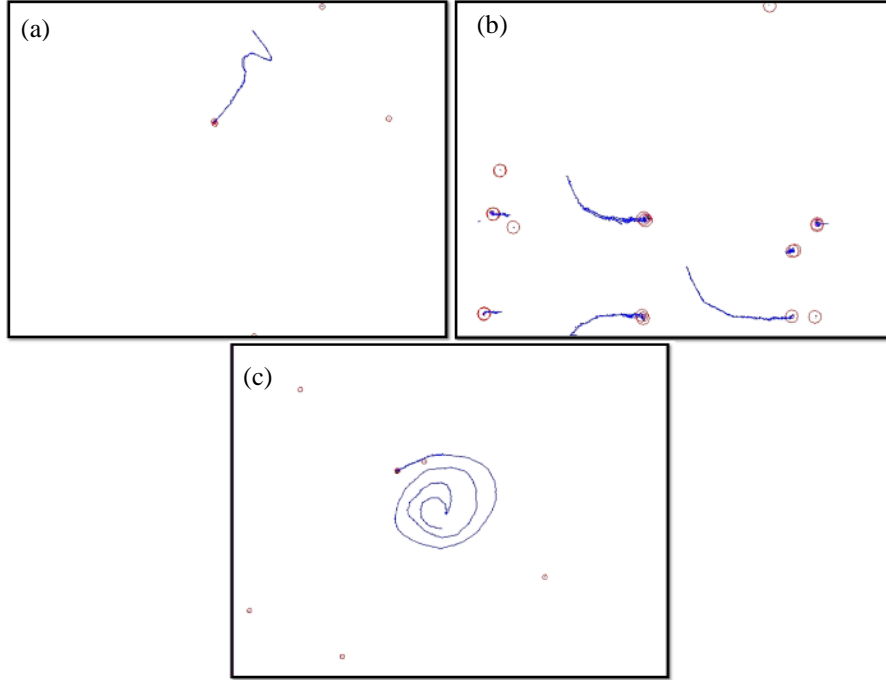
Figure 14: Filtered and Tracked Frames (a) High Contrast 3 Experiment (b) Multiple Targets 1 Experiment

(c) Changing Direction Experiment

4.6 Event-Simulation Tracking

As a supplement to the experimental trials performed, the tracking and noise filtration algorithms were also applied to an event stream simulated from the footage in [31] in order to assess their performance in a more realistic scenario. The event stream was simulated using a luminance threshold of zero, allowing the adaptive threshold functionality to prevent an excessive number of events being generated, but emulating the maximum sensitivity expected to be used for space applications. However, lateral inhibition was not used in this case due to its incompatibility with the frame-based noise filtration method. In other words, performing non-maximum suppression would have resulted in many more isolated events that would then be removed through

noise filtration. In addition, the OpenCV object localization scheme used for ground-truth comparison created many enclosing contours around large sections of luminance change on the Earth's surface, radically decreasing the IoU measured when compared to both the frame-based and asynchronous trackers. As a result, the frame-based and asynchronous tracking schemes were compared directly without determining accuracy with a ground-truth tracker position. Furthermore, unlike in the experimental trials, the number of trackers instantiated must be limited due to some instances in the video where the entire visual field changes in luminance, and thus generates an exceedingly large number of events. Although this event stream is merely simulated from a frame-based video, it is likely that in some cases the DVS' entire pixel array will be excited concurrently, which would result in the large number of events as seen in simulation. Figures 15 and 16 show a reconstructed frame of the simulated event stream with trackers drawn from the frame-based and asynchronous approaches respectively.
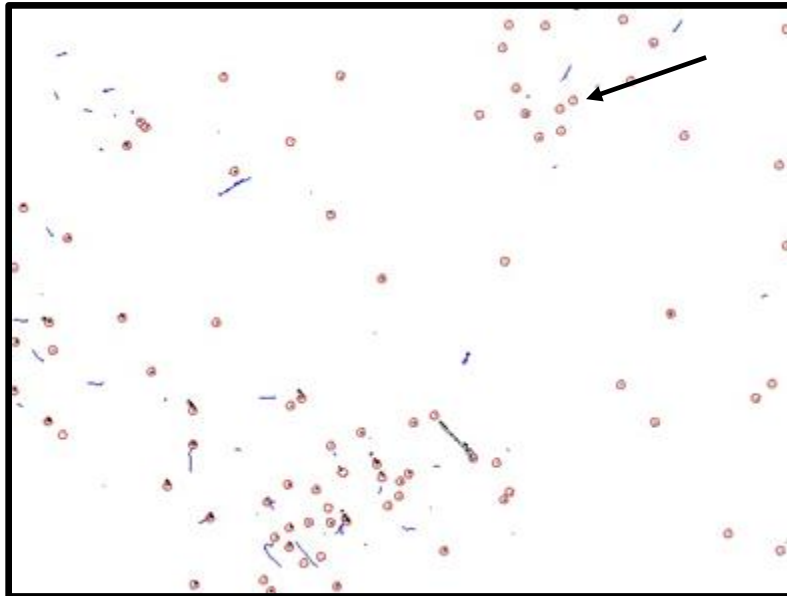

Figure 15: Reconstructed Frame of Frame-Based Tracking Applied to Event Stream Simulation
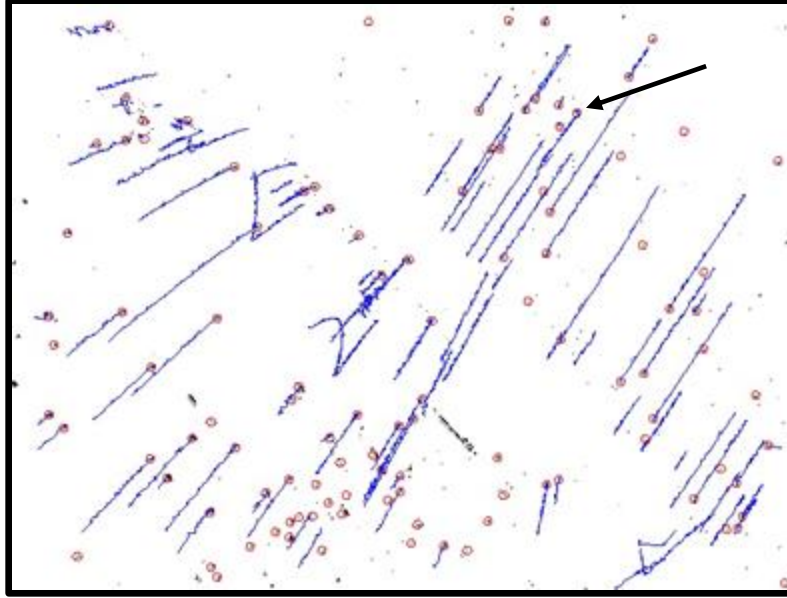
Figure 16: Reconstructed Frame of Asynchronous Tracking Applied to Event Stream Simulation

Unlike in the experimental trials, the asynchronous approach showed a slight, approximately 7% slowdown in execution time as compared to the frame-based approach, despite filtering almost 3.8 times the number of simulated events. However, as can be seen in the captured frames shown in Figures 15 and 16, the trackers instantiated by the frame-based approach frequently lose the trajectory trails as a result of the trackers being deactivated between frames. Furthermore, the target of interest, i.e. the rocket, is frequently filtered out as noise by the frame-based noise filtration method, resulting in repeated loss of tracking. Conversely, the asynchronous approach effectively tracks the target of interest throughout a majority of the frames and the negative polarity and forward-backward error suppression reduce the number of active trackers instantiated for uninteresting events. Since the frame-based and asynchronous approaches were only compared directly, Table 4 includes the relevant metrics collected.

Table 4: Event Stream Simulation Tracking Results

Tracking/Filtration Approach	Execution Time	Filtered Events	Neg. Polarity Suppressions	Forward-Backward Error Suppressions
Frame-based	64511 ms	1466379	N/A	N/A
Asynchronous	69408 ms	5546043	49344	149267

5.0 Discussion

In most instances, the experimental results showed anticipated trends when comparing the frame-based and asynchronous approaches. Generally, the asynchronous approach showed exceptionally lower execution times, while maintaining or even improving across accuracy metrics. However, some interesting exceptions did arise in several experiments as well as in the results of the event-simulation testing.

5.1 Performance and Noise Filtration Outcomes

In terms of average performance, the asynchronous approach had an average of $7\times$ better performance but had a much larger variation in relation to the total number of events. The reason for this disparity is primarily the differing forms of noise suppression used by both algorithms. The frame-based approach will always perform approximately the same number of comparisons whereas the asynchronous approach leverages the sparsity of the event stream to dramatically reduce computation time as evidenced by the differing trends in Figure 4. The two forms of noise suppression also resulted in very different ratios of unfiltered to filtered events, with the asynchronous approach filtering out many more events across all experiments. The multiple targets trials are an outstanding example of this with the asynchronous approach exhibiting an average of about 60% of filtered events compared to the frame-based approach's mere 6%. This large discrepancy is most likely due to the asynchronous approach also filtering events that may not necessarily be noise, but constituent events of potential targets. However, for the purposes of this

research, filtering redundant events can reduce computation time without negatively impacting the tracking performance itself.

In addition to large differences in the number of filtered events, the filtering techniques also differed greatly in terms of the number of trackers instantiated. The asynchronous algorithm exhibited many more trackers being instantiated across all trials as compared to the frame-based approach. According to both algorithms, new trackers will be created if no pre-existing tracker has a p score above the predefined threshold. The asynchronous approach having more instantiated trackers would indicate more solitary events passing through the filter, though these trackers are largely hidden as they are correctly deemed to be associated with noise. In the asynchronous approach, the negative-polarity threshold served to suppress the majority of erroneous trackers. However, there were several trials in which the target had few or no negative-polarity events associated with it, and which relied entirely on the forward-backward error threshold to filter noise-attracted trackers. These occurrences highlighted the need for both forms of tracker suppression as well as the necessity of tuning the suppression thresholds to the behavior of the intended target.

5.2 Tracker Accuracy Comparisons and Outcomes

In regard to tracker accuracy, the asynchronous approach displayed either comparable or, in some cases, superior results in the IoU measurements. While the IoU scores for both approaches did fall below what might be considered an acceptable level of 50% in several trials, these results can be explained by a number of factors. As mentioned earlier, the IoU metric was calculated using the circle circumscribed around the larger axis of each elliptical Gaussian tracker. This approximation was made to ease the calculation involved in the metric as well as due to the

assumption that the target is circular and would not cause significant change in tracker shape. However, this assumption was not always necessarily the case, especially with the additional non-linear and multiple target trials. As a result, the area of the tracker would be overestimated and cause the IoU metric to be much smaller than expected.

Another issue with the IoU metric is the use of reconstructed frames to form the ground-truth comparison. Since the ground-truth position of the target is evaluated per frame, events must be accumulated over the chosen integration time in order to reconstruct a conventional image frame. However, sufficiently fast-moving targets, such as those in the highest speed trials, would exhibit motion blur due to events from multiple positions of the target falling into the same reconstructed frame. While the event-based tracking would track these targets accurately due to the serial nature of the tracker update, the ground-truth comparison would fit a larger than necessary circle around the target and thus artificially decrease the IoU metric. Nonetheless, the IoU metric still shows that the asynchronous noise suppression did not result in a loss of tracker accuracy compared to the frame-based approach. Conversely, the percentage of actively tracked frames does seem to favor the frame-based approach across many of the experiments. This general trend is largely due to the tracker suppression employed by the asynchronous approach, where the target tracker may be made hidden if the negative-polarity activity falls below threshold or the forward-backward error rises above threshold. However, the tracker is not destroyed and positions that may not be actively tracked are in most cases correctly interpolated by the smoothing and curve-fitting operations. Even so, the asynchronous approach showed comparable active tracking time across most experiments.

With respect to varying the Gaussian tracker position update parameter, both approaches showed similar trends with respect to the resulting tracking accuracy. As previously mentioned,

the position update weight tunes the number of events that are required to shift the center of the tracker. As a result, the noise suppression employed and size and speed of the target influence the optimal parameter value required to maximize tracking accuracy. During testing, the frame-based approach showed peak accuracy with position update parameters, α_1, between 0.9 and 0.95, while the asynchronous approach exhibited better accuracy with value 0.8 for several trials. This disparity is most likely due to the asynchronous noise suppression, which as previously mentioned likely removed target events in addition to noise. As a result, the update factor would need to be lower in order to weight each event of the target greater when updating position and maintain accuracy. This behavior indicates that the asynchronous approach is more sensitive to changes in algorithm parameters and would likely need to be tuned according to the expected targets of interest.

For the event-simulation experiment, the frame-based approach surprisingly showed a slightly faster execution time than the asynchronous method, albeit with almost a quarter of the number of events filtered. However, this performance disparity with the experimental trials can be explained by the differences between the simulated and recorded event streams themselves. Since repeated spikes are generated at a greater rate to indicate larger changes in luminance in simulation, these redundant spikes are filtered out in the asynchronous approach, but have no effect on the frame-based method. The frame-based noise filtration is unaffected by redundant spiking due to the fact that it only checks for the previous support of events in adjacent pixel locations. This differing filtration behavior can explain not only the execution times, but also the large discrepancy in numbers of events filtered. The removal of isolated events led to the frame-based trackers being deactivated frequently between frames. This consistent deactivation not only resulted in losing track of the object of interest, i.e. the rocket, but also resulted in the trajectories being refreshed

between most frames. Since drawing the trajectory curves for the large number of trackers instantiated can be a costly process, it is likely that maintaining object tracking is responsible for a significant increase in the asynchronous approach's execution time. While the asynchronous method actively tracks the trajectories of many more trackers within a given timeframe compared to the frame-based method, additional steps could be taken to isolate specific targets of interest, such as selecting trackers with an expected direction and speed of motion.

6.0 Conclusions

This research has introduced an asynchronous noise and tracker suppression scheme for Gaussian-blob tracking with dynamic vision data. The intent of this work was to establish a method of effectively tracking point-source targets that would likely be of interest in space-based applications. The need for point-source tracking was motivated by the incredibly large distances between objects in space coupled with the low resolution of most currently available neuromorphic sensors. Furthermore, the radiation effects expected in the space environment necessitated a new noise filtration scheme tailored to the one-dimensional, asynchronous data received from the DVS. The inclusion of trajectory interpolation and forecasting served to increase the robustness of the target tracking algorithm, while the tracker suppression methods helped mitigate erroneous target tracking. Experimental testing was conducted to assess both the speed and accuracy of this asynchronous approach as compared to a more conventional, frame-based tracking approach. To evaluate the method's effectiveness in a realistic scenario, an event-stream simulation method was applied to space-based footage in order to generate a relevant dataset for further testing.

Experimental results show comparable or superior tracking accuracy with the asynchronous approach, while also exhibiting dramatically better performance as compared to the frame-based approach across all trials. Although the asynchronous method exhibited slightly larger execution times in the event-simulation experiment, this effect was largely due to the frame-based method's frequent loss of target tracking and subsequent reduced processing time devoted to trajectory interpolation. Furthermore, unlike the frame-based and other previous event-based noise-suppression techniques, the asynchronous approach is shown to be effective in tracking objects that may only appear as a single pixel within the DVS's view. While the frame-based

approach was shown to be at least as effective across many of the experimental trials, the method failed to track the target of interest in several trials with low contrast and/or slower targets as well as the more realistic scenario presented in the event-simulation experiment. These results suggest that asynchronous method established in this research achieves superior point-source tracking compared to the conventional approach as was the primary intent of this work.

In regards to the differing forms of noise filtration, the asynchronous method exhibited much higher ratios of events filtered across all trials, which contributed not only to increased tracking accuracy, but also reducing the overall execution time. The Multiple Targets trials in particular greatly highlighted the advantages of the asynchronous method by displaying much larger ratios of filtered events that consequently led to a significant increase in tracking accuracy. However, although the blob tracking associated with both methods requires the tuning of multiple parameters, the asynchronous approach introduces additional parameters, i.e. noise filtration thresholds, that must be tuned for detecting and tracking specific types of objects in terms of speed, size and relative luminance. While the frame-based approach may be more easily applied to multiple scenarios, some knowledge of a given scene and target of interest is required to maximize the effectiveness of the asynchronous method.

The possibility for several extensions exists for both the core noise filtration and tracking algorithms as well as the event simulation method used for experimental testing. Modifications to the asynchronous method could be made that suppress trackers based on a variety of characteristics in order to fine tune the type of objects actively tracked. The number of constituent events and duration of active tracking time are both readily available metrics that could be used to distinguish objects of interest based on size and/or relative importance. The velocity, in terms of both speed

and direction, of targets could be calculated via interpolated tracker positions over time and thus provide another means of selecting particular objects of interest for active tracking.

Although it would be ideal to capture event recordings in space, improvements could be made to the event stream simulator that would allow simulated event streams to more closely mimic the properties of true event streams, such as randomizing the temporal distance between pixel spikes and/or implementing a refractory period in the pixel spiking similar to actual spiking neurons. The asynchronous nature of DVS imaging also makes the event simulation process inherently parallelizable given that the output generated at each pixel is unaffected by other pixels. As a result, event simulation could be accelerated using multiprocessing techniques on both CPU and GPU architectures to achieve real-time simulation. This accelerated event simulation could be used to assess the effectiveness of neuromorphic vision techniques in the absence of neuromorphic hardware, using either a conventional camera or in simulations with 3D computer graphics software.
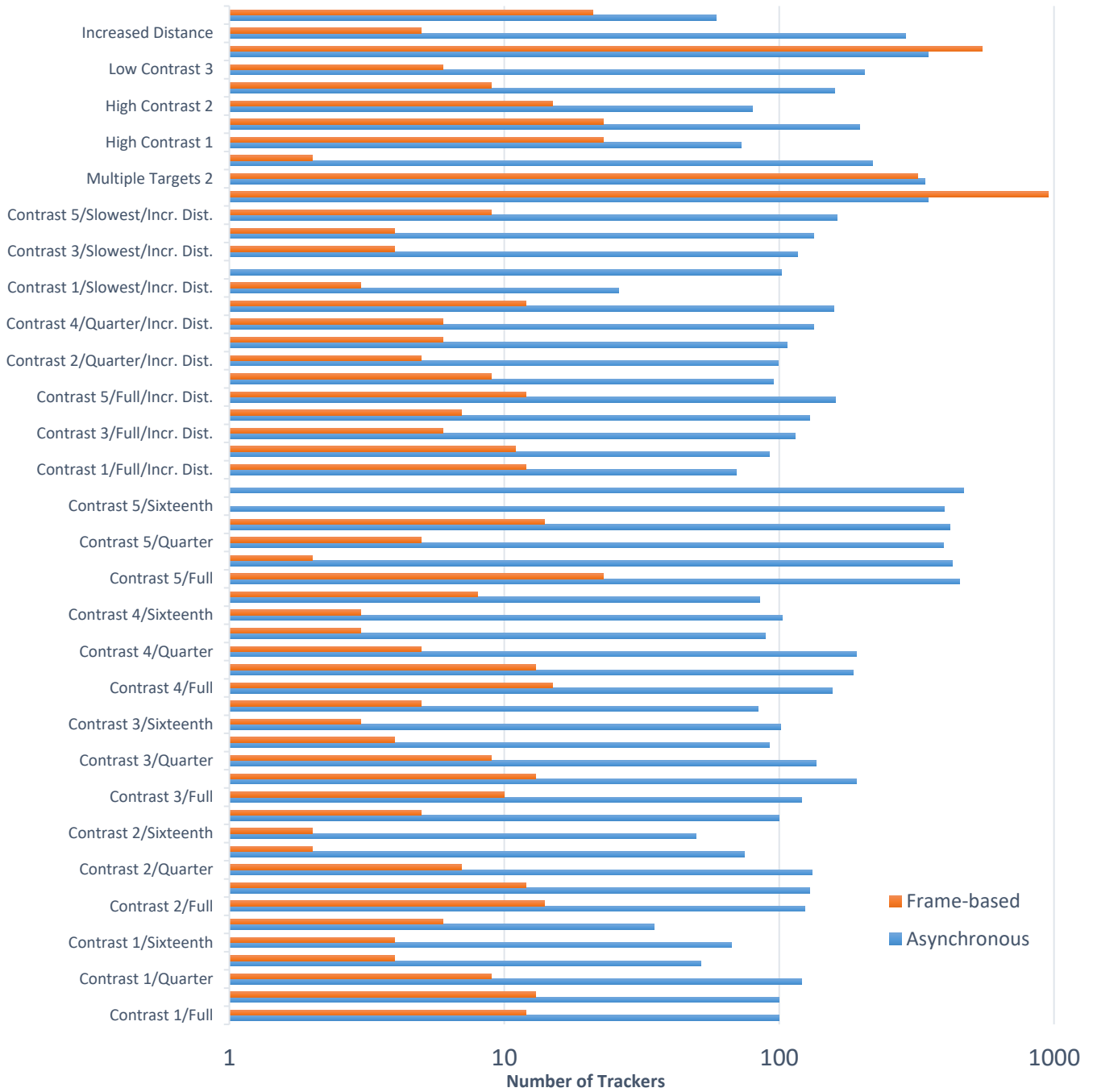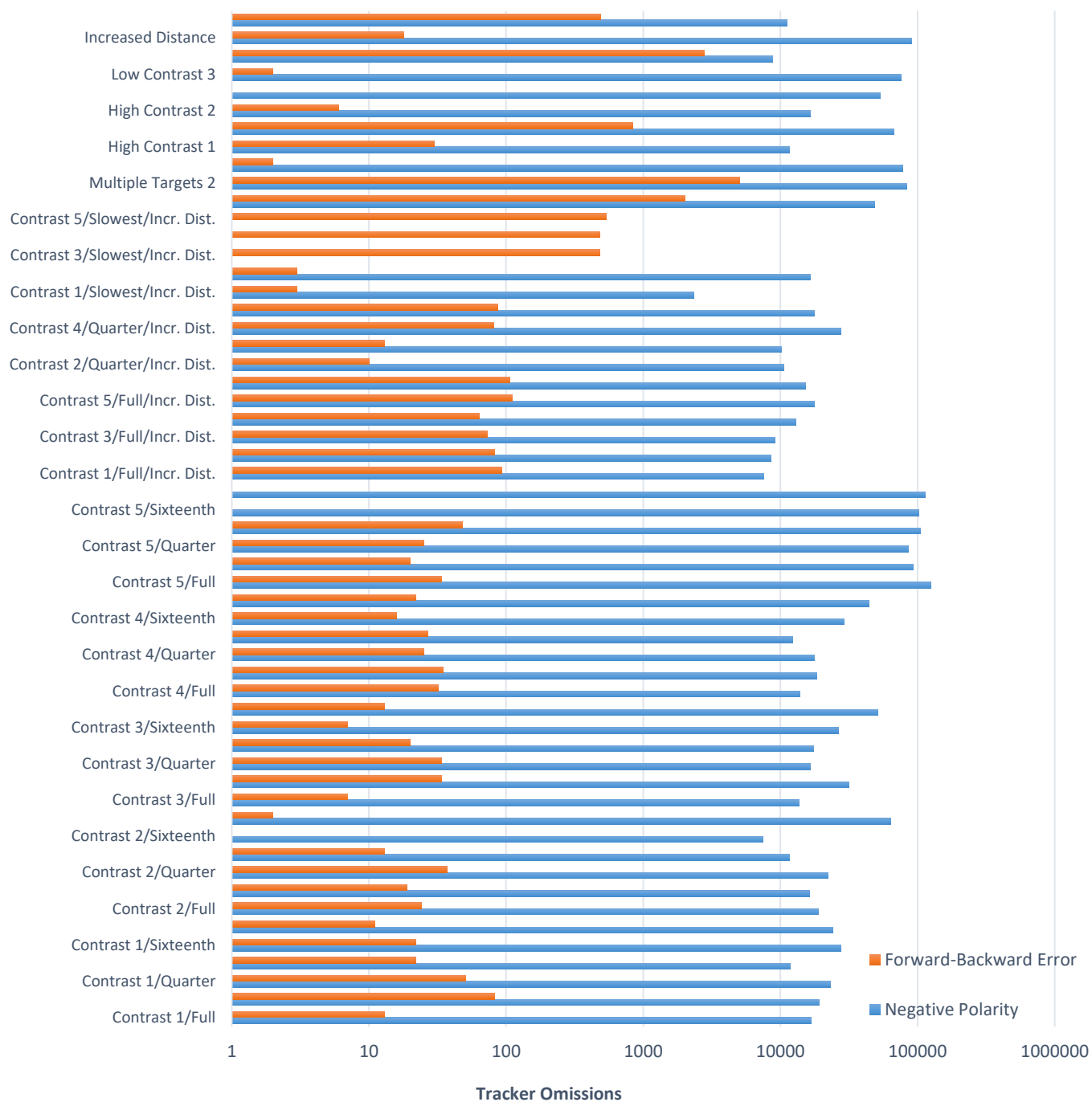
Figure 17: Trackers Instantiated per Experiment

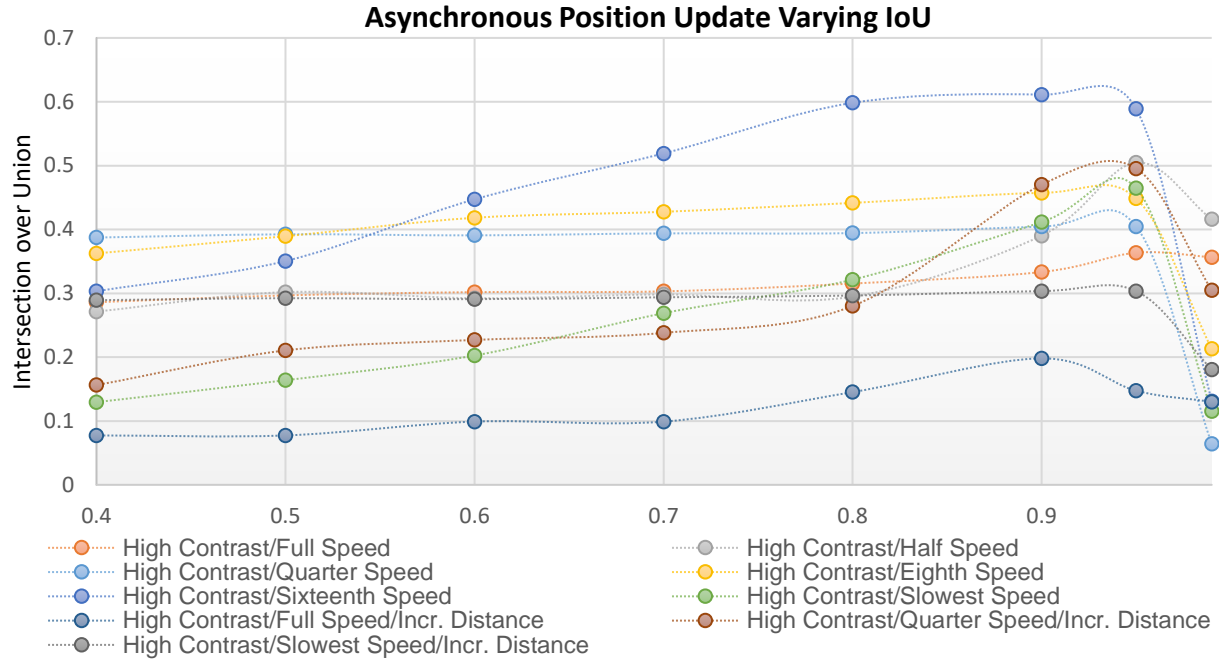**Figure 18: Tracker Suppression Across All Experiments**

Figure 19: IoU Tracking Results for Asynchronous Position Update Parameter Variation
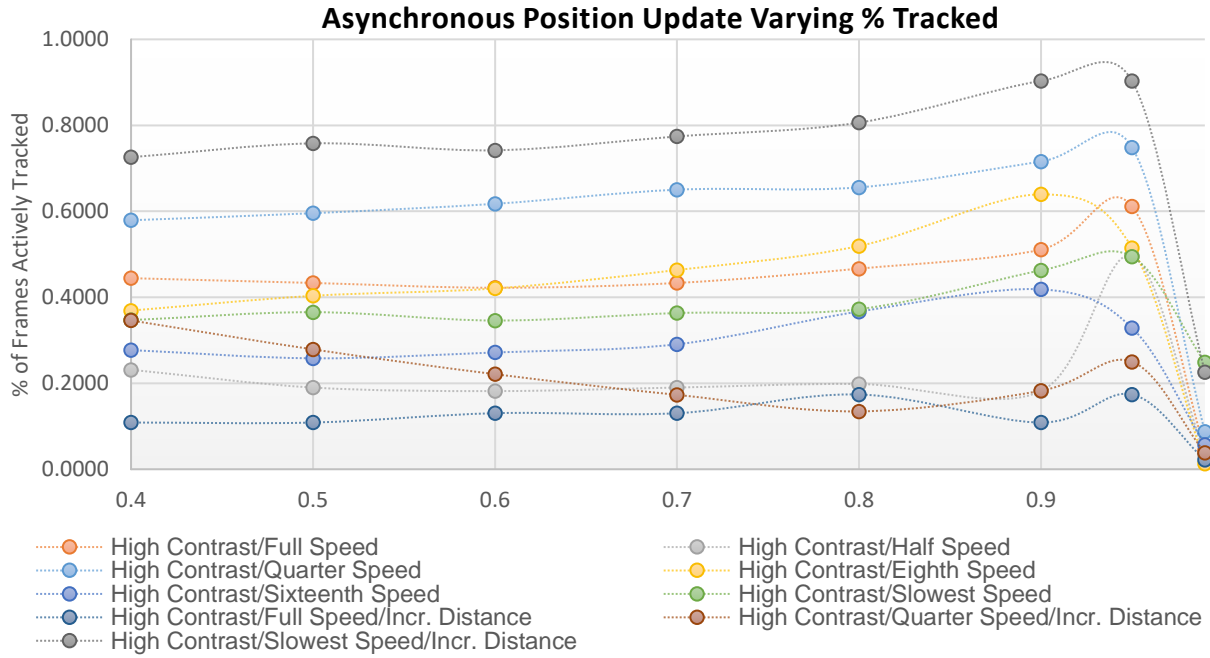

Figure 20: % Tracked Frames Results for Asynchronous Position Update Parameter Variation
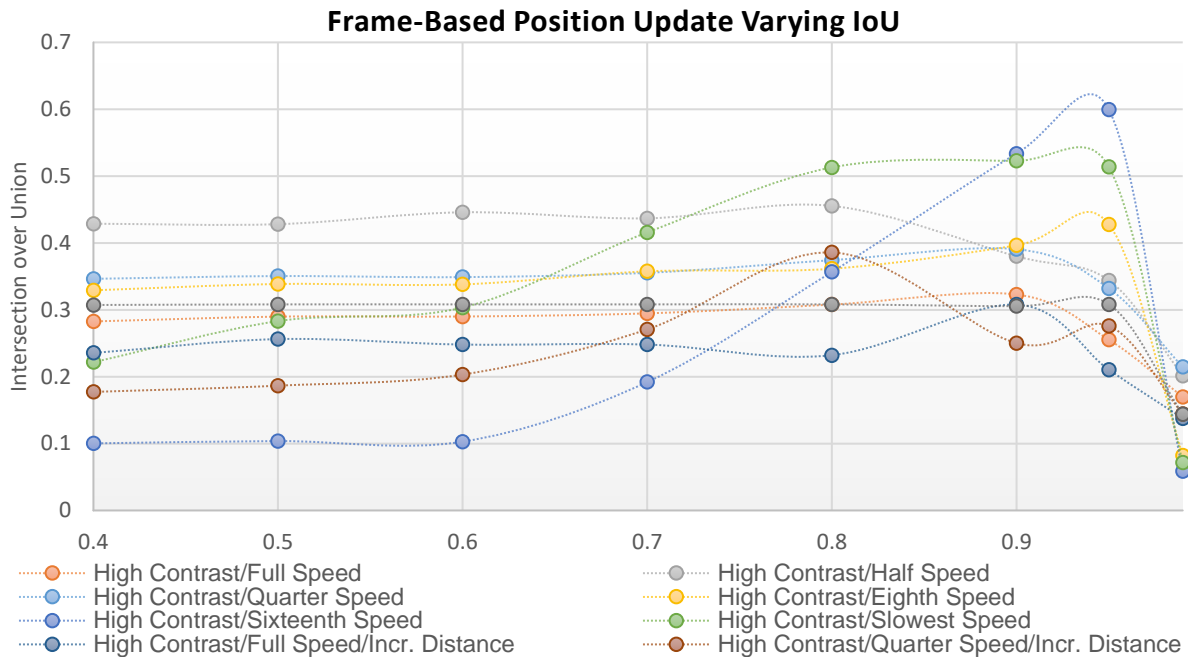
Figure 21: IoU Tracking Results for Frame-based Position Update Parameter Variation
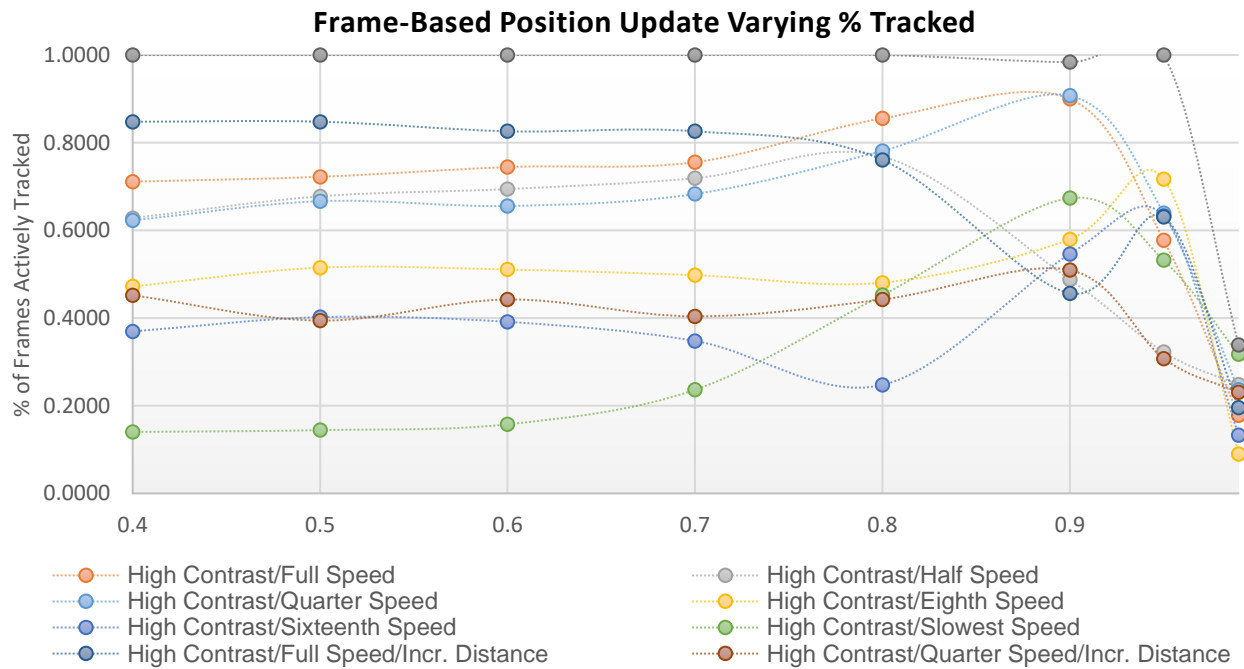

Figure 22: % Tracked Frames Results for Frame-based Position Update Parameter Variation

Bibliography

[1]   M. Mahowald, "VLSI analogs of neuronal visual processing: a synthesis of form and function," 1992.

[2]   T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010.

[3]   C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," *2008 IEEE International Symposium on Circuits and Systems*, 2008.

[4]   F. Bennet, C. D'Orgeville, Y. Gao, W. Gardhouse, N. Paulin, I. Price, F. Rigaut, I. T. Ritchie, C. H. Smith, K. Uhlendorf, and Y. Wang, "Adaptive optics for space debris tracking," *Adaptive Optics Systems IV*, 2014.

[5]   J. Sharma, "Space-Based Visible Space Surveillance Performance," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 1, pp. 153–158, 2000.

[6]   G. Cohen, S. Afshar, B. Morreale, T. Bessell, A. Wabnitz, M. Rutten, and A. V. Schaik, "Event-based Sensing for Space Situational Awareness," *The Journal of the Astronautical Sciences*, vol. 66, no. 2, pp. 125–141, Mar. 2019.

[7]   F. Wang and V. D. Agrawal, "Single Event Upset: An Embedded Tutorial," *21st International Conference on VLSI Design (VLSID 2008)*, 2008.

[8]   G. Hopkinson, "Radiation effects in a CMOS active pixel sensor," *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2480–2484, 2000.

[9]   V. Goiffon, M. Estribeau, O. Marcelot, P. Cervantes, P. Magnan, M. Gaillardin, C. Virmontois, P. Martin-Gonthier, R. Molina, F. Corbiere, S. Girard, P. Paillet, and C. Marcandella, "Radiation Effects in Pinned Photodiode CMOS Image Sensors: Pixel Performance Degradation Due to Total Ionizing Dose," *IEEE Transactions on Nuclear Science*, vol. 59, no. 6, pp. 2878–2887, 2012.

[10] S. Buchner, D. McMorrow, N. Roche, L. Dusseau, and R. L. Pease, "The Effects of Low Dose-Rate Ionizing Radiation on the Shapes of Transients in the LM124 Operational Amplifier," *IEEE Transactions on Nuclear Science*, vol. 55, no. 6, pp. 3314–3320, 2008.

[11] G. Reitz, R. Beaujean, E. Benton, S. Burmeister, T. Dachev, S. Deme, M. Luszik-Bhadra, and P. Olko, "Space radiation measurements on-board ISS—the DOSMAP experiment," *Radiation Protection Dosimetry*, vol. 116, no. 1-4, pp. 374–379, 2005.

[12] D. Tedaldi, et. al. "Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)." *Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP).* 2016.

[13] T. Delbruck, "Frame-free dynamic digital vision," in *International Symposium on Secure-Life Electronics Advanced Electronics for Quality Life and Society,* 2008.

[14] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Networks*, vol. 27, pp. 32–37, 2012.

[15] Z. Ni, C. Pacoret, R. Benosman, S. Ieng, and S. Régnier, "Asynchronous event-based high speed vision for microparticle tracking," *Journal of Microscopy*, vol. 245, no. 3, pp. 236–244, 2011.

[16] S. Seifozzakerini, et. al. "Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor." *BMVC*. 2016.

[17] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 8, pp. 1710–1720, 2015.

[18] Mueggler, Elias, et al. "Lifetime estimation of events from dynamic vision sensors." *IEEE international conference on Robotics and Automation (ICRA).* IEEE, 2015.

[19] J.-M. Maro, G. Lenz, C. Reeves, and R. Benosman, "Event-based Visual Gesture Recognition with Background Suppression running on a smart-phone," *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, 2019.

[20] X. Xie, et al. "DVS image noise removal using K-SVD method." *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)*. Vol. 10615. International Society for Optics and Photonics, 2018.

[21] X. Xie, J. Du, G. Shi, H. Hu, and W. Li. "An Improved Approach for Visualizing Dynamic Vision Sensor and its Video Denoising," *Proceedings of the International Conference on Video and Image Processing* (*ICVIP 2017*). Association for Computing Machinery. pp. 176–180, 2017.

[22] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis \& Machine Intelligence*, no. 5, pp. 564–575, 2003.

[23] J. J. Laviola, "Double exponential smoothing: an alternative to Kalman filter-based predictive tracking," *Proceedings of the workshop on Virtual environments 2003 - EGVE* 03, 2003.

[24] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," *2010 20th International Conference on Pattern Recognition*, 2010.

[25] T. Böttger, P. Follmann, and M. Fauser, "Measuring the Accuracy of Object Detectors and Trackers," *Lecture Notes in Computer Science Pattern Recognition*, pp. 415–426, 2017.

[26] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[27] M. Katz, K. Nikolic, and T. Delbruck. "Behavioural emulation of event-based vision sensors." *2012 IEEE International Symposium on Circuits and Systems*. IEEE, 2012.

[28] E. Mueggler, et al. "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM." *The International Journal of Robotics Research* 36.2 (2017): 142-149.

[29] Y. Bi, and Y. Andreopoulos. "PIX2NVS: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams." *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017.

[30] G. Garcia, et al. "pyDVS: An extensible, real-time dynamic vision sensor emulator using off-the-shelf hardware." *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016.

[31] R. Rossi. "Rocket Launch as Seen from the Space Station." YouTube, 23 Nov. 2018, https://www.youtube.com/watch?v=B1R3dTdcpS