$O(N)$-Space Spatiotemporal Filter for Reducing Noise in Neuromorphic Vision Sensors

Alireza Khodamoradi, *Member, IEEE,* and Ryan Kastner, *Member, IEEE*

Abstract—Neuromorphic vision sensors are an emerging technology inspired by how retina processing images. A neuromorphic vision sensor only reports when a pixel value changes rather than continuously outputting the value every frame as is done in an "ordinary" Active Pixel Sensor (ASP). This move from a continuously sampled system to an asynchronous event driven one effectively allows for much faster sampling rates; it also fundamentally changes the sensor interface. In particular, these sensors are highly sensitive to noise, as any additional event reduces the bandwidth, and thus effectively lowers the sampling rate. In this work we introduce a novel spatiotemporal filter with $O(N)$ memory complexity for reducing background activity noise in neuromorphic vision sensors. Our design consumes 10$\times$ less memory and has 100$\times$ reduction in error compared to previous designs. Our filter is also capable of recovering real events and can pass up to 180% more real events.

Index Terms—Neuromorphic Senors, Event Based, Noise, Spatiotemporal Filter, Background Activity.

1 Introduction

Neuromorphic vision sensors are biologically inspired event-based image sensors. Unlike ordinary image sensors they only produce events if they detect changes in light intensity. It enables them to have an efficient output stream by excluding redundant data and only including changes. In addition, their architecture allow each pixel to be sampled at very high frequencies, for example, DAVIS sensor is capable of sampling at $333.3\,\mathrm{kHz}$ per pixel [1, 2, 3].

Neuromorphic sensors have seen growing importance in industry and research [4, 5]. For example, Samsung recently announced that an event-based image sensor, the Dynamic Vision Sensor (DVS) [2] will be used in their products for gesture recognition [6] alongside IBM's TrueNorth processor [7].

Event-based image sensors are extremely sensitive to Background Activity (BA) noise produced by temporal noise and junction leakage currents [2, 8, 9]. BA noise happens when output of a pixel changes under constant illumination. This noise can be removed by spatiotemporal correlation filters [10].

The programmable logic (PL) at the sensor head can be used for implementing the filter. By having the spatiotemporal correlation filter at the sensor side, the BA events will not be sent to a host PC. It can improve both the sensor's bandwidth utilization and processing. Implementing a spatiotemporal filter at the sensor head becomes a must if the sensor's PL hosts an application [4].

However these filters have two main problems: I) $O(N^2)$ memory complexity that makes their hardware implementation challenging and II) Inability of passing all of the real events. To elaborate on the second issue, it happens when an earlier filtered event finds spatiotemporal correlation with a current event. This earlier event, now has support from a current event to pass the filter, but it requires the filter to have additional memory for keeping all the information for the earlier event. This additional memory will increase the memory complexity of the filter even more and requires bigger PLs.

In this work we address these two issues by introducing a novel hardware friendly spatiotemporal correlation filter with $O(N)$ memory complexity for reducing noise in neuromorphic vision sensors.

ARK
November 15, 2017

1.1 Dynamic Vision Sensor (DVS)

In this work we use the Dynamic Vision Sensor, DVS128 from INILabs [11] similar to what is used by IBM and Samsung. The DVS128 sensor is an event-based image sensor that generates asynchronous address events as soon as the changes in log intensity since the last event exceed an upper or lower threshold.

Each pixel independently and in continuous time quantizes local relative intensity changes to generate spike events. If changes in light intensity detected by a pixel since the last event exceed the upper threshold, pixel will generate an ON event and if these changes pass the lower threshold pixel will generate an OFF event. A Pixel will not generate an event otherwise.

By this mechanism, DVS128 only generates events if there is a change in light intensity, therefore, sensor's output stream only includes the detected changes in sensed signal and does not carry any redundant data.

DVS sensor produces two types of events, ON and OFF. These events are in the form of an address-event that are generated locally by the sensor, each ON or OFF event includes polarity, x-position, and y-position of a pixel's event. The timing information of these events is coded in a 32 bits time-stamp.

- *A. Khodamoradi is with the Department of Computer Science and Engineering, University of California, San Diego, CA, 92093. E-mail: akhodamo@ucsd.edu*
- *R. Kastner is with the Department of Computer Science and Engineering, University of California, San Diego, CA, 92093.*

Manuscript received MMDD, YYYY; revised MMDD YYYY.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2788865, IEEE Transactions on Emerging Topics in Computing

To encode all the event information for output stream, DVS sensor uses Address Event Representation (AER) protocol [3] to create a quadruplets for each event as following:

$$e(p, x, y, t) \tag{1}$$

- p: Polarity, direction of change in light intensity
- x: Column number.
- y: Row number.
- t: Time-Stamp.

1.2 Background Activity (BA)

Background Activity noise is produced by thermal noise and junction leakage currents acting on switches connected to floating nodes [2, 9, 12]. These events decay the quality of the data and utilize unnecessary communication bandwidth and processing.

The difference between BA events and the real activity events of a pixel is that the BA events lack temporal correlation with events in their spatial neighborhood unlike the real events that have a temporal correlation with events from their spatial neighbors. Using this difference, the BA noise can be filtered out by detecting events generated by a pixel without the spatiotemporal correlation with the events generated by neighboring pixels and the pixel itself.

Such a filter is a spatiotemporal correlation filter. To process an event, a spatiotemporal filter searches the event's spatial neighborhood for events with time-stamps closer than a dT to the processing event's time-stamp (Fig. 1). If there exists an event with a time-stamp closer than the dT to the processing event's time-stamp, the processing event has *support* and can pass the filter. The processing event will be filtered out otherwise. This principal can be formulated as following:

$$\begin{aligned} e(p, x, y, t)\ is\ not\ BA \iff \exists\ |t - t_{ij}| < dT \\ s.t.\ |i - x| \leq 1\ \wedge\ |j - y| \leq 1 \end{aligned} \tag{2}$$

In the above equation, e is the processing event and t_{ij} is the time-stamp of the most recent event at $col = i$ and $row = j$ excluding the processing event.

It should be clear that for implementing such a filter one memory cell per pixel is required to store the most recent time-stamp.

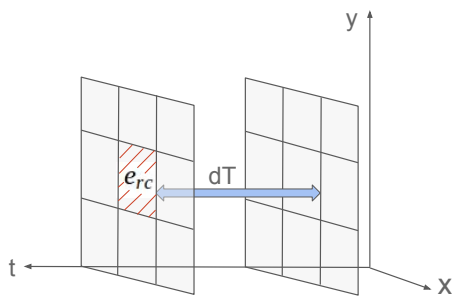

Fig. 1. Principal of spatiotemporal correlation filter. An event can pass the filter if it has correlation with its spatial neighbors within a temporal window dT.

1.3 Contribution

We introduce a novel filter with $O(N)$ memory complexity for reducing BA noise in neuromorphic sensors. Our filter's memory requirement is significantly lower than other related work; this low memory requirement makes our filter desirable for near sensor implementation. By design, our filter stores all the necessary data for recovering recent events. By recovering past real events, we improved the filter's output up to 180% compared to other designs. We also improved the error rates of hardware friendly spatiotemporal filters. Error rates for our filter are 100× smaller than other hardware friendly designs for *false negative* error and zero for *false positive* error.

The rest of the paper is organized as follows. Section 2 reviews the related work in spatiotemporal correlation filter design. Section 3 describes the design of our proposed spatiotemporal filter. Section 4 studies BA noise and provides a mathematical model for it in neuromorphic sensors. In section 5 we define three types of error for spatiotemporal filters and compare them for different filter designs. Section 6 compares and describes the results of hardware implementation for different filter designs.

2 Related Work

Filtering BA noise at the sensor head improves the quality of the data at sensor's output stream, the bandwidth utilization, and saves on processing at a consumer of the data. Filtering BA noise at the sensor head can be inevitable if the existing PL at the sensor be used for implementing a custom application [4].

However implementing a spatiotemporal filter at the sensor head can be problematic. These filters require N^2 memory cells for a sensor with $N \times N$ pixels. Even for small filters, this memory requirement can exceed the available hardware resources at the sensor head. Even if a spatiotemporal filter fit into the sensor's PL, there will not be enough space left for implementing other applications and near sensor processing.

Liu, et. al. [8] designed a filter to address this issue by sub-sampling pixels into groups and projecting each group into one memory cell. An $N \times N$ sensor then will be divided into N^2/S^2 groups with S being the sub-sampling factor. Although this filter does not have significant loss in accuracy for $S = 2$, its error rate increases significantly as the sub-sampling factor grows.

The other problem with *Liu's* filter is the fact that pixels are only compared with other pixels in their sub-sampling group; if a real signal maps on different neighboring sub-sampling groups, *Liu's* filter will not search neighboring pixels in other sub-sampling groups for temporal correlation and it can increase the error rate (Fig. 2). This error can be significant when a DVS is used for observing small objects with limited movement [13].

As mentioned before, the high memory requirement of spatiotemporal filters drives a secondary issue in their design and prevents them to pass both supporting events, meaning if an earlier event that did not pass the filter, provides support for a current event, now the current event is also supporting the earlier event and both events should pass the filter. But this requires extra memory to store all the

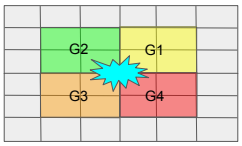

Fig. 2. Sub-sampling groups G1, G2, G3, and G4 each includes 4 pixels. A real world signal that is mapped to different sub-sampling groups may not completely pass *Liu's* filter.

information for the earlier event. *Liu's* filter also lacks this feature and is not designed to store previous events' (x, y) positions and polarity.

3 Proposed Spatiotemporal Filter

In order to search for correlation, spatiotemporal filters need to store the time-stamp of earlier events. But compared to real signals, BA noise is a sparse and random signal. Our observations and calculations show that it is possible to take advantage of this property and store less time-stamps in a specific way to create an accurate and compact filter. In our filter, instead of just saving the time-stamp, we store all the information for an event. By storing all the event information, we are able to search for spatiotemporal correlation in future time of an event and recover all of the real events.

In our approach, Instead of using one memory cell per pixel to store the most recent time-stamp or in *Liu's* case, using one memory cell per sub-sampling group, we assign two memory cells to each row and each column to store the most recent event in the entire row or column (Fig. 3).

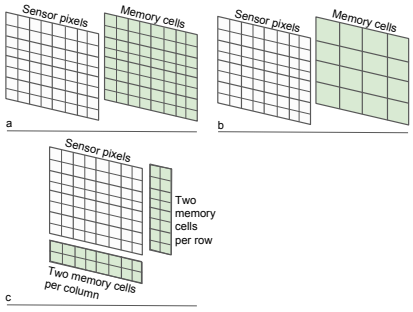

Fig. 3. Memory utilization for different spatiotemporal filter designs: a) Baseline design: using one memory cell per pixel, b) *Liu's* design: using one memory cell per sub-sampling group ($S = 2$), c) Proposed design: two memory cells assigned to each row and column.

Each memory cell is 32 bits, to store the data for the most recent event, we use two memory cells: one for storing the time-stamp and one memory cell for storing a bookkeeping bit, the other axis position, and polarity (Fig. 4.a).

The bookkeeping bit is used for keeping a record of the stored event's status to prevent sending duplicate events. To store the event's polarity one bit is required, it leaves 30 bits from the memory cell for storing the other axis position, it allows the filter to support sensors as big as $2^{30} \times 2^{30}$ pixels.

For example after the filter finishes processing an incoming event $e = (p, x, y, t)$, it updates the cells corresponding to $row = y$ and $col = x$, both time-stamps will be updated to t and both polarity bits will be updated to p the value of *other axis position* for $row = y$ will be updated to x and the value of *other axis position* for $col = x$ will be updated to y. And if the result of processing is that the event is passing the filter, the bookkeeping bit will be set to one and zero otherwise. (Fig. 4.b).

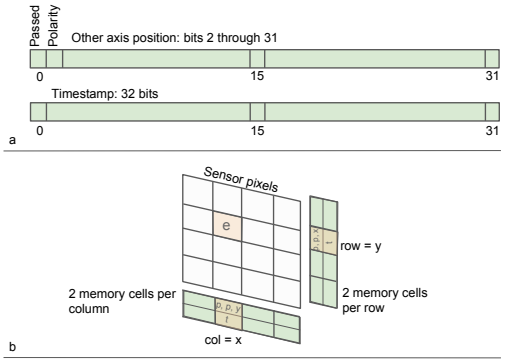

Fig. 4. Memory utilization: a) Two memory cells per row and column, 1 bit to keep track if the event is passed, 1 bit to store polarity, 30 bits to store other axis position, and 32 bits to store time-stamp b) An arriving event $e(p, x, y, t)$ is stored in corresponding memory cells.

Using only two memory cells per row or column, significantly reduces the memory requirements. Compared to other designs, the memory complexity is reduced from $O(N^2)$ to $O(N)$. It makes our filter a much more affordable design for hardware implementation.

Because this filter is able to store all the information for an event, it can recover more real events, later in section 5 we show that this technique improves the data density of real events by about 180%.

4 Noise Model

In a CMOS image sensor, temporal noise is primarily due to the photodetector shot noise, the output amplifier's thermal and $1/f$ noise, and pixel reset, follower, and access transistor thermal, shot, and $1/f$ noise [9, 12]. Hand analysis of the CMOS image sensors published by several authors [1, 2, 14, 15, 16] show that at low illumination the dominant source of noise is reset and readout transistors, while at high illumination the dominant source of noise is the photodiode shot noise.

The DVS sensor is an unconventional CMOS imager. In this sensor, a pixel generates an output if there is enough change in the light intensity since the last event. A pixel then uses two comparators to generate a single bit for reporting an increase or a decrease in the light intensity and sensor generates an ON or OFF event accordingly.

In neuromorphic vision sensors BA events are produced under constant illumination. These events are caused by thermal noise and junction leakage currents acting on switches connected to floating nodes[1, 2, 9]. The hand analysis of DVS128 show that this sensor produces BA events with an average rate of $0.05\,\mathrm{Hz}$ at room temperature and it increases to $1.5\,\mathrm{Hz}$ at $60\,^{\circ}\mathrm{C}$ [2].

These events randomly appear in time independent of each other with an average rate. Although their source is a combination of different noises (Shot, Gaussian, and Pink

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2788865, IEEE Transactions on Emerging Topics in Computing

noises), we assume that their appearance follow a Poisson distribution. Our motivation for making this assumption is based on our observations and previous related studies [12, 17].

To evaluate our assumption, we collected the output stream for a DVS128 sensor in a controlled environment. In our experiments by isolating the setup, we ensured that there are no real events captured by the sensor. We collected sensor's output stream for intervals of $7200\,\mathrm{sec}$ in three different constant illumination settings: dark, normal, and bright. We repeated each test for seven days (Fig. 5). We then used the collected data to measure the goodness of fit between their underlying distribution and Poisson distribution using Kolmogorov-Smirnov test [18].

Fig. 5. DVS128 sensor used in our experiment for data collection. In each data collection experiment, the sensor was kept under constant illumination in an empty room isolated from any activity.

Poisson Distribution

Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time if these events occur with a known average rate and independent of each other:

$$P\{n\ events\} = \frac{(\lambda)^n}{n!}e^{-\lambda} \tag{3}$$

with λ being the average rate of occurrence for the events.

Two-Sample Kolmogorov-Smirnov Test

This test is one of the most popular and important tests for comparing samples with a reference probability distribution [19] and can serve as a goodness of fit test.

If *null hypothesis*, be the position that *"there is no relationship between two measured phenomena"*, Kolmogorov-Smirnov test can check the goodness of fit between samples drawn from an unknown distribution and samples drawn from a known distribution by rejecting or accepting the null hypothesis.

$$D_{m,n} = \sup_x |F(x)_m - F_n(x)| \tag{4}$$

Where $D_{m,n}$ is the Kolmogorov-Smirnov statistic, $\sup_x$ is the supremum of the set of distances, $F_m(x)$ is the cumulative distribution function of the known distribution, and $F_n(x)$ is the empirical distribution function (EDF) for n samples and is defined as:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} 1_{x_i \leq x} \tag{5}$$

The null hypothesis is rejected at level α if:

$$D_{m,n} > \sqrt{-\frac{m+n}{2mn} ln(\frac{\alpha}{2})} \tag{6}$$

Applying the Kolmogorov-Smirnov test on the collected noise from our DVS sensor results an average pValue$= 0.97$ and KS statistic$= 0.02$ that confirms that the null hypothesis can be rejected at level $\alpha = 0.05$ between the BA events collected from the DVS sensor and Poisson process with average BA rate equal to $0.05\,\mathrm{Hz}$ (Fig. 6).

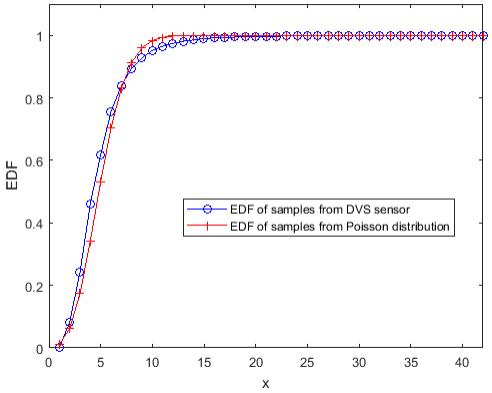

Fig. 6. Kolmogorov-Smirnov test results for DVS128 BA noise and Poisson distribution: pValue = 0.97, KSstatistic = 0.02

Our tests confirm that the BA events from the DVS can be assumed to be drawn from a Poisson distribution, to calculate the number of arrivals for any finite time interval, Poisson process can be used:

$$P\{N(t) = n\} = \frac{(\lambda t)^n}{n!}e^{-\lambda t} \tag{7}$$

In (7), $P\{N(t) = n\}$ is the probability of receiving n BA events during time t from one pixel and λ is the average rate of BA noise-events per pixel.

5 Filters' Error Analysis

In this section, we use the noise model introduced in last section in our analysis to calculate the probability of error for different filter designs. We define three cases of *error* for spatiotemporal filters in our analysis:

- *False positive*: passing an event with no correlation with neighboring events.
- *False negative*: filtering an event with correlation with neighboring events.
- *Past event false negative*: filtering an event with correlation with neighboring events in future time.

Reader should note that if a BA event has correlation with real events it will pass any filter working on spatiotemporal correlation principals.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2788865, IEEE Transactions on Emerging Topics in Computing

5.1 Baseline BA Filter

In this filter one memory cell is assigned to each pixel for storing the last event's time-stamp. This design, does not have *false positive* and *false negative* errors. However, this filter does not have enough memory to store other parameters of previous events, such as polarity therefore it is prone to *past event false negative* error. We use this design as our baseline.

5.2 Liu's BA Filter

This filter uses sub-sampling groups to reduce the memory size. Each sub-sampling group of pixels with sampling factor S, includes S^2 pixels and uses one memory cell for storing the time-stamp of the most recent event of the group (Fig. 3.b).

Grouping pixels in sub-sampling groups bigger than 2×2 causes false spatiotemporal correlation between non-neighboring pixels and will produce *false positive* error.

Although this filter is more efficient than the *baseline* filter for utilizing memory cells, it still does not store any data related to events beside their time-stamp. As a result, this filter is incapable of recovering past events with spatiotemporal correlation with current events and is prone to *past event false negative* error.

False negative error in this filter is caused by its specific design. This filter does not check neighboring groups for supporting an arrival event. In the case of having a real world signal mapped to neighboring pixels in different sub-sampling groups (Fig. 2), this filter may not pass all of the bordering events.

To calculate the probability of error for an incoming event in this filter, we define three pixel groups for a sub-sampling group and calculate their *false positive* and *false negative* probabilities.

A S^2 sub-sampling group of pixels includes: 4 corner pixels, $4(S-2)$ side pixels, and $(S-2)^2$ inner pixels (Fig. 7).

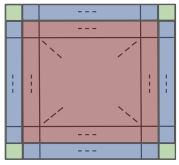

Fig. 7. Three pixel groups for a $S \times S$ sub-sampling group: *green*: 4 corner pixels, *blue*: $4(S-2)$ side pixels, and *red*: $(S-2)^2$ inner pixels.

$$S^2 = 4 + 4(S-2) + (S-2)^2 \tag{8}$$

An arriving event can be from corner, side or inner groups with the probabilities $4/S^2$, $4(S-2)/S^2$, and $(S-2)^2/S^2$ accordingly.

Each corner pixel has five neighboring pixels outside of its sub-sampling group, that can cause *false negative* errors. And has S^2-4 non-neighboring pixels in its sub-sampling group that can cause *false positive* errors.

Each side pixel has three neighboring pixels outside of its sub-sampling group that can potentially cause *false negative* errors. And has S^2-6 non-neighboring pixels in its group with potentials of causing *false positive* errors. For the last group, each inner pixel has S^2-9 non-neighboring pixels that can cause *false positive* errors.

In *Liu's* filter *false positive* error is when a non-neighboring pixel's spatiotemporal correlation with an arriving event is used to pass the event and can be calculated for a temporal window t as:

$$\begin{aligned} P\{false\ positive\ error(t)\} = \\ \frac{4}{S^2}(1-P\{N(t)=0\}^{S^2-4}) \\ +\frac{4(S-2)}{S^2}(1-P\{N(t)=0\}^{S^2-6}) \\ +\frac{(S-2)^2}{S^2}(1-P\{N(t)=0\}^{S^2-9}) \end{aligned} \tag{9}$$

$P\{N(t)=0\}$ is calculated using (7) and $1-P\{N(t)=0\}^k$ is the probability of having at least one BA event from k pixels during time t.

To calculate the *false negative* error for *Liu's* filter, we calculate the possibility of loosing support from a neighboring pixel in a different sub-sampling group:

$$P\{false\ negative\ error\} = \frac{4}{S^2}(\frac{5}{9}) + \frac{4(S-2)}{S^2}(\frac{3}{9}) \tag{10}$$

5.3 Normal Sub-Sampling Filter

To resolve the *false negative* error in *Liu's* filter shown in Fig. 2, we consider a filter that neighboring sub-sampling groups can support each other. To the best of our knowledge this filter is not used in practice and is provided only to demonstrate that resolving the *false negative* error will increase the *false positive* error in sub-sampling approach. In the rest of this paper, we refer to this filter by *sub-sampling filter*.

This filter is also prone to *past event false negative* error, but because neighboring sub-sampling groups in this filter can support each other the *false negative* error is zero. To calculate the *false positive* error we need to calculate how many non-neighboring pixels can cause this error for each pixel in a sub-sampling group. For the corner pixels, there are $4S^2-9$ non-neighboring pixels that can potentially cause this error. For side and inner pixels, there are $2S^2-9$ and S^2-9 pixels accordingly that can potentially cause *false positive* error. We can formulate this for a temporal window t as:

$$\begin{aligned} P\{false\ positive\ error(t)\} = \\ \frac{4}{S^2}(1-P\{N(t)=0\}^{4S^2-9}) \\ +\frac{4(S-2)}{S^2}(1-P\{N(t)=0\}^{2S^2-9}) \\ +\frac{(S-2)^2}{S^2}(1-P\{N(t)=0\}^{S^2-9}) \end{aligned} \tag{11}$$

5.4 Our Proposed Filter

This filter stores the quadruplet (1) of the most recent event in a row or a column. Therefore it is capable of recovering previous events and does not suffer from *past event false negative* error.

To store an event, this filter stores both row and column information, therefore non-neighboring events are not included in the search for spatiotemporal correlation. As a result the *false positive* error in this filter is zero.

The *false negative* error happens in a special case when there is only one real event to provide support for another real event but the data of the older real event gets replaced by BA noise. Lets consider two real events e_1 and e_4 in neighboring of each other with the timing order of $t_1 < t_4$:

$$e_1(p_1, x_1, y_1, t_1) \; and \; e_4(p_4, x_4, y_4, t_4) \; are \; neighbors \iff |x_1 - x_4| \leq 1 \wedge |y_1 - y_4| \leq 1 \quad (12)$$

If $t_4 - t_1 < dT$ then both events must pass the filter, but if BA events overwrite e_1's data, the recent event e_4 will loose its support from e_1. If noise does not support e_4 (no BA event in e_4's neighborhood during dT) e_4 will not pass the filter and results into *false negative* error. Lets consider two BA events $BA_2(p_2, x_1, y_2, t_2)$ and $BA_3(p_3, x_3, y_1, t_3)$ (Fig. 8), *false negative* error will happen if:

$$t_1 \leq (t_2, t_3) \leq t4 \; \wedge \; |y_2 - y_4| > 1 \; \wedge \; |x_3 - x_4| > 1 \quad (13)$$

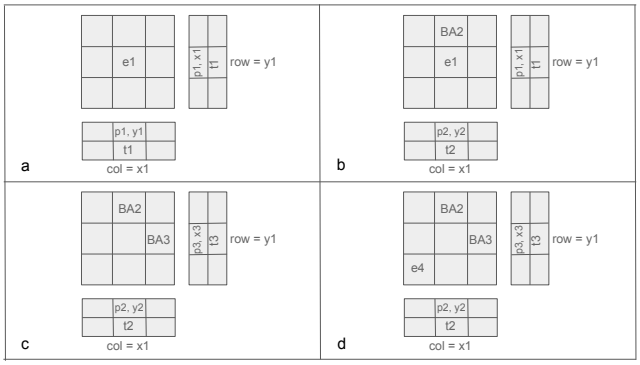

Fig. 8. Example of *false negative* error in our proposed filter: a) at $t = t_1$, real event e_1 arrives, column x_1 stores y_1 and t_1 and row y_1 stores x_1 and t_1. b) at $t = t_2$, noise event BA_2 arrives and changes the values of column x_1 to y_2 and t_2. c) at $t = t_3$, noise event BA_3 arrives and changes the values of row y_1 to x_3 and t_3. At this point information related to e_1 are completely overwritten by noise events. d) at $t = t_4$ real event e_4 arrives and filter can not find a temporal correlation in its neighboring pixels and *false negative* error occurs.

Our filter's *False negative* error for a $M \times M$ sensor and temporal window t can be calculated as:

$$P\{false \; negative \; error(t)\} = (1 - P\{N(t) = 0\}^{M-3})^2 \quad (14)$$

In (14) we are calculating the probability of receiving at least one BA event from the pixels in e_1's row (y_1) excluding e_2's neighbors and at least one BA event from the pixels in e_1's column (x_1) excluding e_2's neighbors during a temporal window t.

5.5 Theoretical Comparison

To compare the filters, we use our developed equations to calculate both *false negative* and *false positive* errors.

In our comparison for *false positive* error, we set the temporal window of the filter to a practical value $dT = 1$mSec and noise frequency to 0.05 Hz for room temperature [2]. This error is zero in our filter and the result for other sensors with different sensor sizes are shown in Fig. 9.

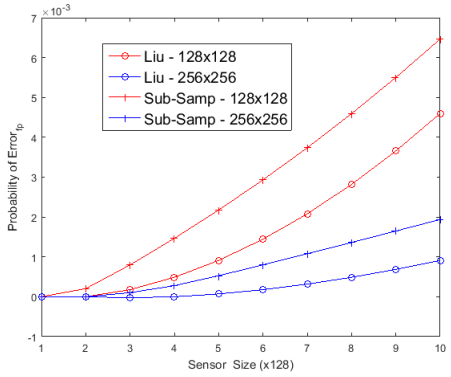

Fig. 9. False positive error calculated for $dT = 1$mSec and $f_{noise} = 0.05$Hz (room temperature). This error is zero for our filter. Increasing the size of the filter for *Liu's* and *sub-sampling* filters results in smaller sub-sampling groups and improves this error.

Increase in temperature increases the average noise rate and results to higher *false positive* error rates for *Liu's* and *sub-sampling* filters.

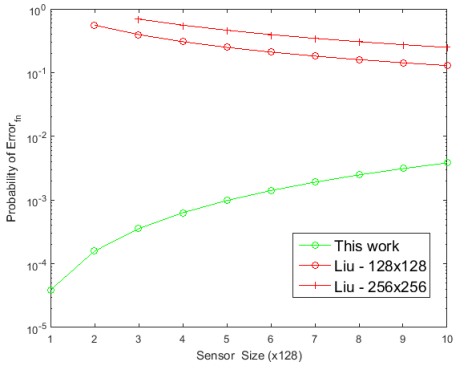

Fig. 10. False negative error calculated for temporal window $dT = 1$mSec and noise rate $f_{noise} = 0.05$Hz. Since this error is significantly lower for our filter, data are presented in logarithmic scale. Increasing *Liu's* filter's size will increase the number of corner and side pixels and results in higher error rates (10).

Comparison between the filters for *false negative* error is done in a different fashion. This error is zero for *sub-sampling* filter, temporal independent for *Liu's* filter (10), and temporal dependent for our proposed filter (14).

Fig. 10 shows this comparison between *Liu's* and our filter. The decay in *Liu's* noise probability is because of fading effect of corner and side pixels for larger sub-sampling groups (10). Increasing the temporal window will increase the probability of *false negative* error for our filter.

5.6 Comparison Between Filters using Real Data

In this subsection, we compare the filters using real data captured from a DVS sensor. To compare the performance of our filter with the baseline filter, we passed the collected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2788865, IEEE Transactions on Emerging Topics in Computing

noise from section 4 to both filters. Our results show that our filter works as expected and we did not observed any error during our observations (Fig. 11).

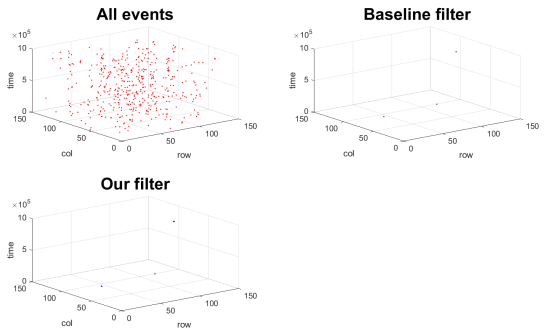

Fig. 11. Comparison between our filter and baseline filter using real data captured by a DVS sensor. Both filters identically remove the BA noise.

To compare the filters for *past event false positive* error, we collected the output of the DVS128 sensor while moving a laser pointer in front of the image sensor. We collected this data for two seconds and used all four filters for denoising. We repeated this test for 20 times and calculated the number of passed events between the filters. We concluded that on average our proposed filter passes 180% more real events compared to other filters. The result of one of our tries is shown in Fig. 12.

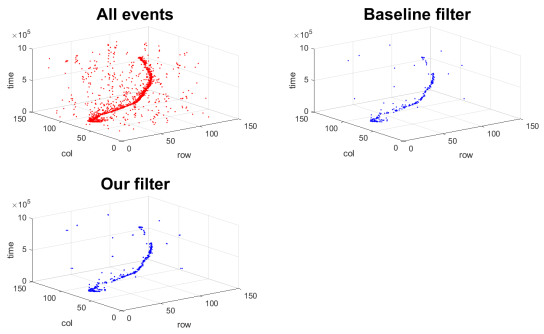

Fig. 12. *Past event false negative* noise. Data captured with a moving laser pointer in front of the camera. Compared to other filters, our proposed filter is able to pass 180% more real events compared to other filters. All other filters are prone to this error and output of the baseline filter can also represent *Liu's* and *sub-sampling* filters for this error.

6 Hardware Implementation

To compare the resource utilization between our proposed filter and other approaches, we implemented all the filters using Vivado ® High Level Synthesis.

Baseline filter's size is equal to the sensor size and *sub-sampling* and *Liu's* filters' sizes can be different depending on their sub-sampling factor (Fig. 3). But for an equal filter size, hardware utilization for these three filters are almost identical. Therefore a reader can assume that the provided result for a *baseline* filter are valid for the same size *Liu's* or *sub-sampling* filters.

In practice, because of limited real estate and to conserve energy and heat, compact FPGAs with high performance-per-watt ratios are used at the sensor head. Therefore we used Artix®-7 from Xilinx® for our synthesis.

The result of synthesis for different sensor sizes range from 128×128 to 1280×1280 is shown in Table 1.

TABLE 1
Comparison between resource utilization

Size	filter	Latency (nSec)	BRAM	FF	LUT	Throughput (MHz)
128×128	baseline	8	8.77%	0.06%	0.32%	14
	this work	35	0.55%	0.31%	0.72%	3
256×256	baseline	8	35.07%	0.06%	0.34%	14
	this work	35	0.55%	0.32%	0.72%	3
512×512	baseline	8	140.27%	0.07%	0.37%	14
	this work	35	0.55%	0.32%	0.72%	3
1024×1024	baseline	8	561.10%	0.10%	0.40%	14
	this work	35	1.10%	0.32%	0.73%	3
1280×1280	baseline	8	1122.19%	0.12%	0.43%	14
	this work	35	2.19%	0.32%	0.73%	3

Table 1 shows that for sensor sizes bigger than 256×256, *baseline* filter can not fit in the fabric. And since *Liu's* and *sub-sampling* filters have similar hardware utilization as *baseline* filter for equal filter size. These two filters also can not fit in the fabric if they are bigger than 256×256 (Fig. 13).

As it is shown in Table 1, our proposed filter consumes less memory and as a trade off, it has lower throughput and higher latency compared to baseline filters. However its latency is three orders of magnitude faster than the sensor, pixels of DVS128 have a latency equal to $15\,\mu\text{sec}$ which is equal to $66.7\,\text{kHz}$ and the sensor itself is capable of producing maximum 1M events per second.

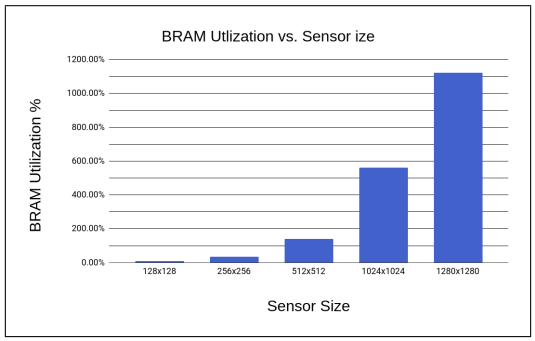

Fig. 13. Memory utilization for baseline filter. Liu's and sub-sampling filters with sizes equal to baseline filter have similar memory utilization.

Compared to other filters, our proposed filter does not have a high demand on resources. Even for the large sizes it utilizes a fraction of memory compared to other filters. Fig. 14 shows a comparison between the smallest *Liu's* filter and our proposed filter for different sensor sizes.

7 Conclusion

This paper presents a novel $O(N)$ spatiotemporal filter for neuromorphic vision sensors. By modeling the noise of neuromorphic vision sensors, we calculated the probability of error for our proposed sensor and other related filter designs. Our error models show that the proposed filter has $100\times$ less *false negative* error compared to other hardware friendly designs and zero *false positive* error. By collecting

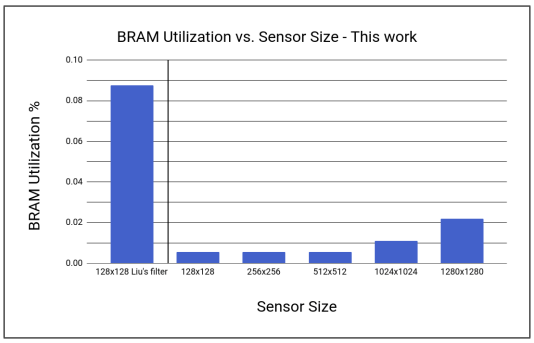

Fig. 14. Memory utilization for proposed filter. Even for large sensor sizes, its memory utilization is significantly lower than a 128×128 Liu's, baseline, or sub-sampling filter.

data from a real sensor, DVS128 we showed that the performance of our proposed filter follows our predictions and developed models. In addition, this filter shows an improved output up to 180% compared to all other designs by passing all of the real events.

In our hardware implementation section we showed that this novel filter reduces the memory utilization by $10\times$ and can fit on fabrics with limited resources and unlike other spatiotemporal filters, it still leaves enough space on the fabric for implementing other possible applications.

REFERENCES

[1] C. Brandli et al. "A 240×180 130 dB $3\,\mu$sec Latency Global Shutter Spatiotemporal Vision Sensor". In: *IEEE Jornal of Solid-State Circuits* 49.10 (2014), pp. 2333–2341. DOI: 10.1109/JSSC.2014.2342715.

[2] P. Lichesteiner, C. Posch, and T. Delbruck. "A 128×128 120 dB $15\,\mu$sec Latency Asynchronous Temporal Contrast Vision Sensro". In: *IEEE Jornal of Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[3] M Mahowald. *An Analog VLSI System for Stereoscopic Vision*. Boston, MA: Kluwer, 1994.

[4] A. Linares-Barranco et al. "A USB3.0 FPGA Event-based Filtering and Tracking Framework for Dynamic Vision Sensors". In: *IEEE International Symposium on Circuits and Systems* (2015), pp. 2417–2420. DOI: 10.1109/ISCAS.2015.7169172.

[5] A. Rios-Navarro et al. "Live demonstration: Real-time motor rotation frequency detection by spike-based visual and auditory AER sensory integration for FPGA". In: *IEEE International Symposium on Circuits and Systems* (2015). DOI: 10.1109/ISCAS.2015.7169040.

[6] CNET. *Samsung turns IBM's brain-like chip into a digital eye*. URL: https://www.cnet.com/news/samsung-turns-ibms-brain-like-chip-into-a-digital-eye/.

[7] Sawada et al. "TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2016), pp. 130–141. DOI: 10.1109/SC.2016.11.

[8] H. Liu et al. "Design of a Spatiotemporal Correlation Filter for Event-based Sensor". In: *IEEE International Symposium on Circuits and Systems* (2015), pp. 722–725. DOI: 10.1109/ISCAS.2015.7168735.

[9] Hui Tian. "Noise Analysis in CMOS Image Sensors". PhD thesis. Stanford University, 2000.

[10] A. Pooresmaelli et al. "Spatiotemporal Filtering and Motion Illusion". In: *Jornal of Vision* 13.21 (2013). DOI: 10.1167/13.10.21.

[11] *iniLabs*. URL: https://inilabs.com.

[12] H. Tian, B. Fowler, and A. Gamal. "Analysis of Temporal Noise in CMOS Photodiode Active Pixel Sensor". In: *IEEE Journal of Solid-State Circuits* 36.1 (2001), pp. 92–101. DOI: 10.1109/4.896233.

[13] T. Delbruck. *Scientific: Particle Image Velocimetry*. URL: https://inilabs.com/videos/dvs-applications/.

[14] S. Decker et al. "A 256×256 CMOS Imaging Array with Wide Dynamic Range Pixels and Column-Parallel Digital Output". In: *IEEE J. Solid-State Circuits* 33 (1998), pp. 2081–2091.

[15] S. Mendis et al. "CMOS Active Pixel Image Sensors for Highly Integrated Imaging Systems". In: *IEEE J. Solid State Circuits* 32 (1997), pp. 187–197.

[16] O. Yadid-Pecht et al. "Optimization of Noise and Responsivity in CMOS active Pixel Sensors for Detection of Ultra Low Light Levels". In: *Proc. SPIE* 3019 (1997), pp. 125–136.

[17] F. Yang et al. "Bits from Photons: Oversampled Image Acquisition Using Binary Poisson Statistics". In: *arXive1106.0954* (2011). DOI: 10.1109/TIP.2011.2179306.

[18] Allen Edward. *Kolmogorov-Smirnov Test for Discrete Distributions*. Monterey, California: Defense Technical Information Center, 1976.

[19] J. Pratt and J. Gibbons. "Concepts of Nonparametric Theory". In: Springer, 1981. Chap. Kolmogorov–Smirnov Two-Sample Tests. DOI: 10.1007/978-1-4612-5931-2_7.