Real-time event-based sensor simulation for space applications

Iain M. Martin[1], Martin N. Dunstan[1], Manuel Sanchez-Gestido[2] and Joris Belhadj[2], [1]University of Dundee, Scotland, UK, DD1 4HN, [2]ESTEC, ESA, Noordwijk, The Netherlands). *[i.martin@dundee.ac.uk]

Abstract. *Vision-based framing sensors are well-established in the space community as a critical component of guidance, navigation and control systems for autonomous spacecraft control and landing. Event-based sensors detect changes in pixel intensity levels, so output an asynchronous stream of pixel events. Their unique properties have some advantages for space mission scenarios and navigation applications. To support developments in this area, we describe how an established real-time frame-based sensor simulator (PANGU) is being extended to simulate event-based data from logarithmic sensors for a variety of space applications, and how the simulated data can be evaluated with respect to standard event-based calibration tools.*

Introduction. Vision-based sensors have been successfully used as a key component of Guidance, Navigation and Control (GNC) systems onboard spacecraft for a wide variety of autonomous missions and maneuvers such as soft landings on Moons or asteroids, cannister capture and orbital rendezvous [1]. Vision-based sensors have mostly been framing or push-broom style cameras which produce synchronous image data at defined frame rates.

Event-based cameras, inspired by the behavior of biological visual sensors, are asynchronous vision sensors which sample incoming light based on the dynamics of the scene. Instead of producing snap-shot images of fixed size at a specified frame rate, they produce an asynchronous stream of events when a pixel intensity change is above a threshold. Each event is independent and defined by location and polarity of the pixel intensity change [2]. This is a paradigm shift from standard frame-based sensors.

PANGU (Planet and Asteroid Natural scene Generation Utility) is a software suite that combines real and synthetic data to generate simulated frame-based sensor images (visual, thermal and LiDAR) of planetary surfaces, asteroids and spacecraft. It has been designed to support very large models (*e.g.* >50GB) and real-time closed-loop simulations with fast frame rates of 10Hz or better. The tool suite combines a surface modeler tool that can be used to import real data and enhance with synthetic data where required, with an integrated custom, GPU renderer, designed to generate representative sensor images. The renderer has a built-in GPU-based camera model which can apply noise, optical distortion and other camera effects in real-time for image simulation [3].

In this paper we describe a new PANGU feature which provides simulated data to aid the development and testing of event-based sensor applications for guidance and navigation. We have extended our well-established framing sensor simulator to output data that simulates event-based sensors, focusing initially on logarithmic detectors as a popular variant. This adds a new sensor type to the simulation tool suite, producing simulated event-based sensor data in real-time as well as image-based data that can accelerate external event-based sensor tools. The existing PANGU real-time camera model can be used to apply noise to event-based simulations.

Event-based Cameras Overview. An event-based camera consists of three main analogue stages: a photodetector (typically logarithmic), a differencing circuit, and a pair of threshold detectors to generate ON/OFF events. A digital output stage detects, timestamps and emits events to a host system. The photodetector may drive a CMOS pixel for hybrid event-based and frame-based detection. A logarithmic photodetector response means that the system responds to changes of brightness scale rather than level *e.g.* 2× brighter or 2× fainter. A simplified diagram of an event detector pixel is shown in Fig. 1, following Hu [4].

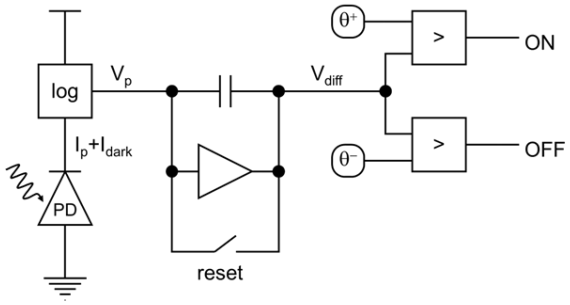

Figure 1. Simplified event-based pixel

The differencing circuit compares the current voltage level to the level at the end of the previous reset. If the level exceeds the ON (θ^+), or OFF (θ^-), threshold levels, the corresponding event is latched. Once the event has been read, the comparison level of the differencing circuit is reset to the *current* voltage level. Since there is a delay between the detection of an event and the corresponding reset, the new comparison voltage is likely to be different to that when the event was raised. This may cause events to be lost. However, by extending the duration of resets (the refractory period), the maximum rate at which a pixel can generate an event is controlled without the loss of events from pixels with low event rates. The choice of thresholds determines how sensitive the detector is to changes in brightness scale.

The individual pixels in an event-based sensor respond independently to dynamic changes of light intensity. For each pixel, the logarithm of the intensity is stored after an event has been emitted. Then whenever the current logarithm of the intensity changes by more than a user-set threshold value, the sensor generates an event.

Events are emitted with the following information:

- *x, y* pixel location
- time *t*,
- polarity (increase or decrease flag).

This produces an asynchronous stream of events with the greater the change in the scene, the greater the number of events per second.

Example sensors. A short summary of the critical parameters from some currently available event-based sensors are summarized in this section to give examples of the resolution, throughput and dynamic. The DAVIS 346 sensor has a resolution of 346×260 pixels, a maximum throughput of 12 MEPS (mega events per second), typically latency of <1ms and a dynamic range of ~120 dB (0.1–100k lux with 50% pixel response to 80% contrast) [5]. The DVXplorer Micro range of sensors have a higher maximum resolution of 640×400, a throughput range of 30–450, typically latency of <1ms and a dynamic range of ~110 dB (0.3–100k lux with 50% pixel response to 80% contrast. The Event Camera Evaluation Kit 4 HD IMX636 Prophesee-Sony sensor has a substantially higher resolution of 1280×720 and a dynamic range of more than 86dB (5 lux–100k lux).

Data formats and sources. DAVIS24 is a publicly available dataset of event camera sample data, mostly taken from a DAVIS346 camera [5]. This contains a set of events generated from different scenarios in the AEDAT 2.0 format. AEDAT format versions 1.0, 2.0, 3.0, 3.1 are 4.0 are defined on the iniVation website [6].

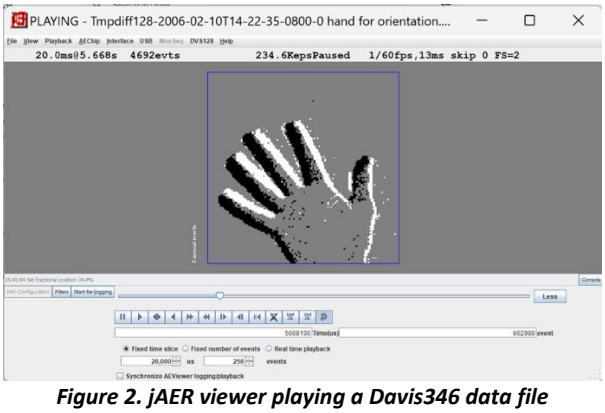

Figure 2. jAER viewer playing a Davis346 data file

AEDAT 2.0 and later formats begin with an XML human-readable header followed by the event data. The event data consists of a series of [address, timestamp] pairs for each event. The address and microsecond timestamp are both 32 bits wide. In AEDAT 2.0 the address format depends on the specific event-detector chip while AEDAT 3.0 and later use a chip-agnostic format. All integer data and fields are signed and big-endian in AEDAT 2.0 and earlier; in AEDAT 3.0 and later they are little-endian [6]. An example snapshot from a DAVIS346 sensor file is shown running in the jAER event-viewer application in Fig. 2 which shows events generated from a moving hand with white pixels denoting positive events and black negative events.

Event-based Sensor characteristics. Event-based sensors have potentially useful characteristics for space applications such as high temporal resolution and low latency (microseconds), a much higher dynamic range than CCD cameras (e.g. ~140dB vs 60dB) and low power usage, especially when the scene is not changing dramatically [2]. Fast motion can be captured without excessive blur more common to frame-based sensors, but noise and blurring are still issues to be managed in event-based sensors [4]. The low latency is inherent because an event can be sent immediately after it is recorded, without a requirement to wait for other events to complete (or a whole frame). The high dynamic range is due to the receptors working on a logarithmic scale. These properties can be useful for real-time space applications, such as surface rovers or planetary landers in challenging lighting conditions [3], e.g., near the lunar South Pole which may have bright peaks and dark shadows in the same image frame, where the high dynamic range may be useful for detecting features for navigation purposes.

Working with event-based sensors can be challenging, the sensor bus can become saturated with events, perturbing the times that events are sent, and processing the events to extract meaningful information can be difficult: algorithms must be developed that are different from frame-based processing systems [2]. Generating sufficient event-based data to support the testing and development of these sensors for use in space-based applications is also a challenge: this can be supported with simulated data if it can be shown to be representative of real scenarios.

Related work. Event-based sensor data has been simulated from image sequences and video. Hu developed a toolbox (***v2e***) which generates events from a sequence of intensity images [4]. In ***v2e***, event-based images may be produced in the form of a 2D histogram: each pixel records the polarity (increasing/decreasing) of the event and the number of events that occurred between two full image frames. The event count is obtained by dividing the difference in voltage by the threshold magnitude; the sign yields the direction [7]. Hu also highlighted issues with noise and hot-pixels, and suggests methods to simulate these artefacts [4].

Mueggler et al. describe an event-based simulator based on the DAVIS sensor [8] using the open-source 3D graphics software tool, Blender [9] to render images from

a 3D scene, along a trajectory, with a small motion between frames (1/3 pixel). A timestamp map is used to store the time of the previous event triggered at each pixel. This is combined with brightness changes between consecutive images frames to provide continuous asynchronous timestamps with events triggered by a contrast threshold [8]. This general approach is extended by Rebecq et al, who created the ESIM event camera simulator which adds an adaptive frame rate determined by the dynamics of the image changes [10].

Sikorski used PANGU to generate a sequence of raw images with floating-point precision and detector noise for realism. The images were converted into voltages by computing the logarithm of each pixel, and then used to determine the events generated by each pixel. Since multiple events might be generated by a pixel between two images, linear interpolation was used to estimate when each event might occur for a given pixel between the two frames taking latency into account [7].

A related resource is EVREAL, which is an analysis suite designed to benchmark event-based video reconstruction techniques [11]. This is a potential resource that could be used to verify real-time event simulations through reconstructing images from the simulated events.

Simulating Events in Real-time in PANGU. The approach taken to generate simulated events from a sequence of images is broadly based on the approach taken by Hu [4] which can be used to generate a realistic stream of events corresponding to that of an event-based detector observing the same scene. They describe how the *v2e* toolbox converts digitized image frames into events, with optional temporal up-scaling via slow-motion interpolation. By including an event-based detector model within PANGU, we benefit from existing camera model features such as optical distortion and multiple sources of noise, the ability to generate floating point radiance images (*e.g.* without 8-bit quantisation), and support for closed-loop simulation with arbitrary inter-frame time intervals. This allows closed-loop simulations to depend on the analysis of events from previous images.

Output data. Two types of data output are provided: the raw event images and the event data. Raw event images encode key information in the floating channels of an RGBA pixel. This includes the signed event count, the time of the first event within the frame, the reference log-intensity level, the temporal low-pass filter log-intensity level and the time of the end of reset/refractory period. This is enough to allow the event data to be extracted and can be delivered to users quickly since it requires no further processing by PANGU. If event data (polarity, coordinates and timestamp) is requested, then PANGU will process the raw event image to extract the events. In doing so it will consider the time taken to scan the detector array extracting and emitting events onto the output bus; this may cause events to be lost in pixels that generate more than one and corresponds to read-out rate limits in real detectors.

GPU-based event generation. The image-to-event conversion on the GPU process begins by obtaining the per-pixel logarithm of a grey-scale luma image which is passed through an intensity-dependent low-pass temporal filter matching the detector bandwidth, to obtain a value proportional to V_p of Fig. 1. This models the motion-blur effect that occurs in event-based detectors where pixels respond faster to large changes in brightness compared to small changes. For each pixel in the image, the filtered log-intensity level from the end of the previous reset period is reduced by leakage and the result, V_{ref}, is subtracted from the current filtered log-intensity level to obtain the difference voltage V_{diff}. If the reset/refractory time is zero, dividing V_{diff} by the two per-pixel thresholds (θ^+,θ^-), yields the number of events that would be generated if the log-intensity varied consistently during the frame interval. The number of ON and OFF events is quantised to an integer using the floor operator to obtain N_{on} and N_{off}; since θ^+ and θ^- have opposite signs, at most one of these can be non-zero. The new reference level for the pixel is:

$$V'_{ref} = V_{ref} + N_{on}\theta_{on} + N_{off}\theta_{off}$$

Since N_{on} and N_{off} are quantised, any extra voltage difference carries over to the next frame.

A non-zero reset/refractory time introduces extra complexity, but it is important for modelling the ability to reduce the maximum event rate of the detector. If a given pixel is in reset for the duration of the frame, its reference level is updated to match the current filtered log-intensity level. Otherwise, V_{diff} is scaled by the remaining reset time relative to the frame duration. This accounts for the time at the start of the frame when the pixel was tracking the filtered log-intensity value in reset. The event count computation is separated into two parts: first whether a single first ON or OFF event is generated; secondly, the number of extra reset-event periods that can occur if a first event is generated and V_{diff} was adjusted after it. The reset period of the last extra event, if any, is not considered because it can extend to the next frame.

Interactive event display. Before extracting events from the raw event images for the user, a GPU shader is used to display the filtered image and mark pixel events. An example of the event display is shown in Fig. 3 taken from a lunar Malapert landing sequence using a PANGU model of the DAVIS346 sensor. The filtered image is displayed with pixels having ON events drawn in green and those with OFF events drawn in red. Users can set the colour of ON and OFF events, and the fraction of the scene colour to use; the scene colour can be also suppressed in pixels that have events to prevent the event colours from becoming washed out. A background colour

can be set if scene pixel colours are not used: this is useful if the ON and OFF event colours are white and black.

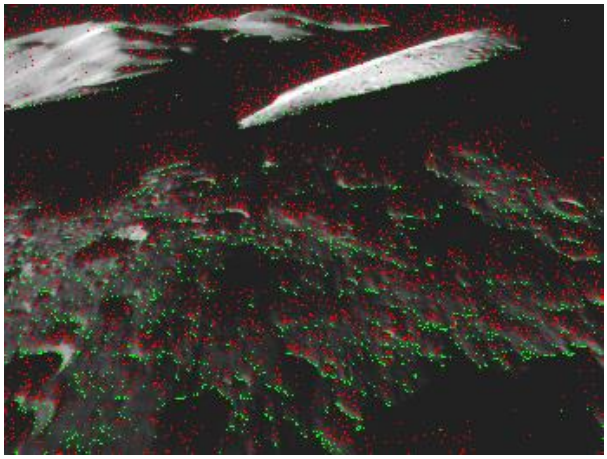

Figure 3. PANGU Davis346 event image of the Moon

Event extraction. Having identified the number of ON or OFF events for each pixel, the time at which each event is raised is determined. It is assumed the events will occur uniformly throughout the frame duration albeit with random jitter. Dividing the frame duration into an equal number of event slots based on the event count determines the ideal time at which each event will occur. A random fraction of the event slot is added allowing the event to be raised at a random point within its slot.

Modelling the Digital Read-out Process. The realism of the event-based detector model may be improved by considering the non-zero time required to detect and emit and event after it has been raised, and the arbitration method by which events are detected. Lichtsteiner et. al created an arbitrated non-greedy system which ensures that a row or column of pixels will not be serviced again until all other rows or columns with events have been serviced [12]. The read-out system can be considered to iterate over each row of the detector in sequence checking to see if any pixels in the row have generated an event. If a row with events is found, each column of the row is checked. If an event is raised for a given pixel, its address and polarity (ON vs OFF) are passed to the output bus, and the pixel is placed in reset. The reset lasts for the duration of the refractory period.

The scanning process may be optimised by skipping rows or columns that have no events. The state of all events in a row could be captured and reset in parallel allowing them to be sent to the bus in a group with their row address. The detector array may be subdivided into a grid of detector tiles to reduce problems associated with scaling up per-pixel addressing to large arrays.

The time needed to read out and reset each pixel or row along with the number of pixels with events will determine the interval between successive checks of a given pixel within a frame. After extracting an event and advancing the time counter, *T*, to account for its read-out, any subsequent events for the pixel that have a time less than *T* are discarded. This is because those events would be lost in the refractory/read-out time while waiting for the extracted event to be read and emitted.

Noise Sources and Distortions. An event-based detector has some noise sources and distortions common with CMOS detectors:

- point-spread function (aperture)
- lens correction (vignetting)
- radial or tangential distortion
- depth of field
- system efficiency, quantum efficiency
- photon shot noise
- per-pixel photo-response non-uniformity
- thermal dark current
- dark current non-uniformity and shot noise

All these can be modelled by the real-time PANGU camera model. In addition, there are noise sources and distortions that are specific to an event-based detector:

- junction leakage
- per-pixel leakage scale variation
- parasitic photocurrent
- temporal bandwidth
- static per-pixel threshold variance
- event jitter/random variation

Junction leakage can be modelled as the average leakage rate scaled by the frame interval, a dynamic random jitter with normal distribution, and a static per-pixel log-normal variance factor.

Parasitic photocurrent results from the leakage of light into the event detection circuitry causing additional photoelectrons to be generated in the differencing circuit. This reduces the reference level by a factor proportional to the light intensity and circuit quantum efficiency.

The temporal bandwidth is proportional to the light intensity. It can be modelled as a leaky integrating low-pass filter: the previous filtered log-intensity level is mixed with the current log-intensity level with the mixing proportion being the clamped product of the filter bandwidth and the light intensity. The clamping range is between a user-specified minimum and 1. The minimum value is needed to prevent low intensities having an infinitely narrow bandwidth and not updating.

Static per-pixel threshold variations model the natural variation in threshold components. They can be modelled as a random bias drawn from a normal distribution with zero mean and user-defined standard deviation.

The time at which events are raised for a given pixel can be modelled by dividing the frame interval into equal sized slots based on the number of events generated. The event is generated at a uniformly random fraction of its slot duration relative to the start of the slot. A slot fraction of 1 is expected which means that events are uniformly randomly distributed across their slots.

Results. A development test image showing a moving checkerboard is given in Fig. 4. The green pixels mark positive ON events and red mark negative OFF events; OFF events are spread out more than ON events due to the bandwidth filter. The visual image of the scene is rendered at 25% brightness.

As a guide to the number of events generated, the measured range of events per frame is shown in Table 1, obtained using some different camera settings. We used the frame dimensions to match two event-based sensors, 346×260 size for the DAVIS346 camera and 1280×720 for the DVS1280x720SD camera. Results from different real-world scenarios at 1280×720 are shown in Table 2.

Table 1: PANGU checkerboard event-based performance

Dimension (pixels)	DN Bias	Rate (Hz)	Events/ frame	Events/ sec
1280×720	0.3	8	30–40k	230–310k
1280×720	0.6	27	20–30k	540–820k
346×260	0.3	86	3–6k	260–520k
346×260	0.6	91	3–5k	270–450k

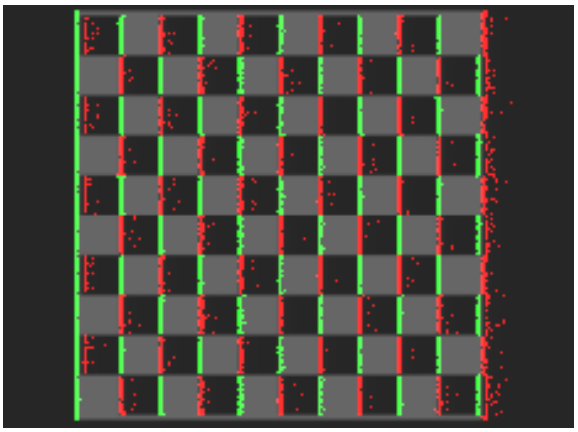

Figure 4. PANGU simulated event-based test image

A more realistic scenario is shown in Fig. 5 which shows half of a PANGU frame from NEAR/MSI Eros flyover sequence 311 in event-based mode on the left (threshold=0.4, bias=0.1), and normal image on the right. Eros is rotating with the bottom half approaching and the top half receding hence the green (ON) and red (OFF) events along the top edges.

The Malapert example is a simulated descent down to Malapert ridge near the Lunar South Pole and the number of events generated per frame varies from lower numbers during the initial phases of the descent and then substantially larger numbers towards the landing site. The Itokawa and Eros asteroid simulation scenarios each have a craft orbiting a small body and show a smaller range of events. In the Itokawa simulation, the body is viewed with the Sun behind the camera which results in a limited brightness variation. This causes events to be mainly around the edge of the body. In the Eros simulation, half of the body is in shadow with the rest showing wide variation in lighting conditions; Eros is also filling the field of view more than Itokawa does.

Table 2: PANGU models event-based performance

Model	Dimension (pixels)	Frame rate (Hz)	Events/frame
Malapert	1280×720	4.7	5–100k
Itokawa	1280×720	8.7	2–3k
Eros	1280×720	11.1	20–30k

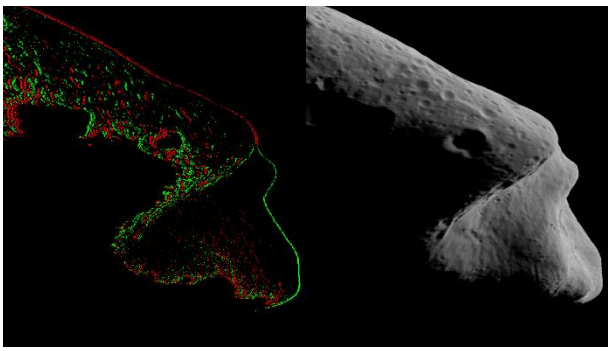

Figure 5. PANGU event (L) and visual (R) images of Eros

Conclusions and future work. We have described the initial implementation of a real-time event-based sensor simulation, implemented within the PANGU tool suite in a real-time rendering system. Initial results are promising and show that up to 100k events/second can be generated on a representative lunar scenario and can be output as event images, event images mixed with visual images, event streams or event files. This allows PANGU to generate simulated events which could support the development and testing of future navigations and guidance systems using event-based sensors through providing simulated data for training and testing.

Future work. Within the scope of the current project, the next plans are to provide TCP/IP access to event images and event data, and to allow event streams to be unicast over UDP (User Datagram Protocol) in a similar way to real event-based cameras. We plan to verify and validate the event-based simulations through comparisons with simulated event-streams from video, by comparison to real-event data and image reconstruction using standardized tools [11]. We will also obtain performance data from representative scenarios using UDP and TCP/IP.

Acknowledgements. PANGU was developed by the University of Dundee for ESA and is being used in many European activities aimed at producing precise, robust planetary lander and rover guidance systems. This work was performed under ESA contract number 4000143650/24/NL/CRS/nh.

References.

[1] I.Martin, S.Parkes, M.Dunstan, M.Sanchez-Gestido, G.Ortega, “Simulating planetary approach and landing to test and verify autonomous navigation and guidance systems”, ESA GNC 2017, Salzburg, May 29th–June 2nd, 2017.

[2] G.Gallego et al., “Event-based Vision: A Survey”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Jan. 2022, pp. 154-180, vol. 44, 2022, DOI: 10.1109/TPAMI.2020.3008413.

[3] I.Martin, S.Parkes, M.Dunstan, M.Sanchez-Gestido, “Simulating Lunar Approach and Pinpoint Landings Through Enhancing Dems and Realistic Image Generation”, 12th International Conference on Guidance, Navigation & Control Systems (GNC), Sopot, Poland, June 2023.

[4] Y.Hu, S-C.Liu, T.Delbruck, “v2e: From Video Frames to Realistic DVS Events”, Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition, 2021, DOI: 10.1109/CVPRW53098.2021.00144.

[5] T.Delbruck, "DAVIS24: DAVIS Event Camera Sample Data", Available from: https://sites.google.com/view/davis24-davis-sample-data/home.

[6] AEDAT Event format, defined in website: https://docs.inivation.com/software/software-advanced-usage/file-formats/index.html. Accessed May 31st 2024.

[7] O.Sikorski, D.Izzo, G.Meoni, “Event-based spacecraft landing using time-to-contact”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2021, pp.1941–1950, DOI: 10.1109/CVPRW53098.2021.00222.

[8] E.Mueggler, H.Rebecq, G.Gallego, T.Delbruck, D.Scaramuzza, “The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM”, International Journal of Robotics Research, Vol. 36, Issue 2, pages 142-149, Feb. 2017.

[9] https://www.blender.org/, accessed 23/09/2024.

[10] H. Rebecq, D. Gehrig, D. Scaramuzza, "ESIM: an Open Event Camera Simulator", 2nd Conference on Robot Learning, Proceedings of Machine Learning Research, No. 87, pp 969-982, 2018.

[11] B. Ercan, O. Eker, A. Erdem and E. Erdem, “EVREAL: Towards a Comprehensive Benchmark and Analysis Suite for Event-based Video Reconstruction”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2023, pp.3943–3952, DOI: 10.1109/CVPRW59228.2023.00410.

[12] P.Lichtsteiner, C.Posch, T.Delbruck, “A 128×128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor”, in IEEE Journal of Solid-State Circuits, 43(2), pp.566–576, 2008, DOI: 10.1109/JSSC.2007.914337.