Live Demonstration: Behavioural Emulation of Event-Based Vision Sensors

M. L. Katz[1], K. Nikolic[1], T. Delbruck[2]
[1] Centre for Bio-Inspired Technology, Dept. of Electrical and Electronic Engineering, Imperial College London, UK
[2] Institute of Neuroinformatics, UZH-ETH Zurich, Switzerland

Abstract— **This demonstration shows how an inexpensive high frame-rate USB camera is used to emulate existing and proposed activity-driven event-based vision sensors. A PS3-Eye camera which runs at a maximum of 125 frames/second with colour QVGA (320x240) resolution is used to emulate several event-based vision sensors, including a Dynamic Vision Sensor (DVS), a colour-change sensitive DVS (cDVS), and a hybrid vision sensor with DVS+cDVS pixels. The emulator is integrated into the jAER software project for event-based real-time vision and is used to study use cases for future vision sensor designs.**

Associated Track 11.1: Sensory Systems: Image and Vision Sensors

I. Demonstration

Recent developments in activity-driven, event-based vision sensors have opened up a promising alternative to conventional frame-based vision. By outputting a sparse and variable-rate stream of data originating asynchronously from pixels with local gain control, these sensors reduce processing cost and latency, while increasing dynamic range. Except for [1], these vision sensor developments have been preceded by silicon chip developments. We thought that it would be useful to develop a behavioural emulator of such chip architectures that would allow us or others to study how to use a proposed vision sensor in application scenarios, and to allow us to model, in software, non-idealities such as fixed pattern noise and temporal noise. In particular, we needed an emulator that is fully integrated into our host-side software infrastructure [3]. In this demonstration, we show how this emulation allows us to model several existing and proposed vision sensor architectures.

II. Demonstration setup

The demonstration setup consists of a laptop, a webcam (Playstation PS3-Eye), and a DVS128 Dynamic Vision Sensor [4]. A large monitor or beamer shows the output.

III. Visitor Experience

Visitors see how the emulator models an existing DVS128, how the improved resolution of the QVGA emulation will benefit applications in wide-area surveillance, and how the emulation models the cDVS sensor we are developing to simultaneously detect colour change and brightness change [5] . By using a high speed rotating stimulus, visitors also see the limitations of the conventional camera in terms of time resolution (8ms for the PS3-Eye compared with 1us for the DVS128), and in terms of dynamic range (<50dB for the PS3-Eye, compared with 120dB for the DVS128).

IV. Demonstration Readiness

The demonstration is fully functional (see Fig. 1). We can simultaneously run the emulation and a real event-based sensor, record data, and play it back. We can display data in a variety of formats such as the spatial histogram shown in Fig. 1 or in 3D space-time, and we can control parameters of the model interactively through the model GUI.

References

[1] Address-Event Imagers for Sensor Networks: Evaluation and Modeling, T. Teixeira, E. Culurciello, E. Park, D. Lymberopoulos, A. B. Sweeney and A. Savvides, Proceedings of Information Processing in Sensor Networks (IPSN), April 2006, pp. 458 - 466.
[2] Activity-Driven, Event-Based Vision Sensors, T. Delbruck, B. Linares-Barranco, E. Culurciello, C. Posch, IEEE International Symposium on Circuits and Systems, 2010. ISCAS 2010, Paris, France, pp. 2426 - 2429.
[3] Welcome to the jAER Open Source Project. Retrieved 10.10.2011. Available: jaer.wiki.sourceforge.net
[4] Dynamic Vision Sensor (DVS) - asynchronous temporal contrast silicon retina. Retrieved 10.10.2011. Available: siliconretina.ini.uzh.ch
[5] Event-Based Pixel Sensitive to Changes of Color and Brightness, (2011) R. Berner, T. Delbruck, IEEE Transactions on Circuits and Systems I (TCAS I), 58:7, pp 1581-1590.

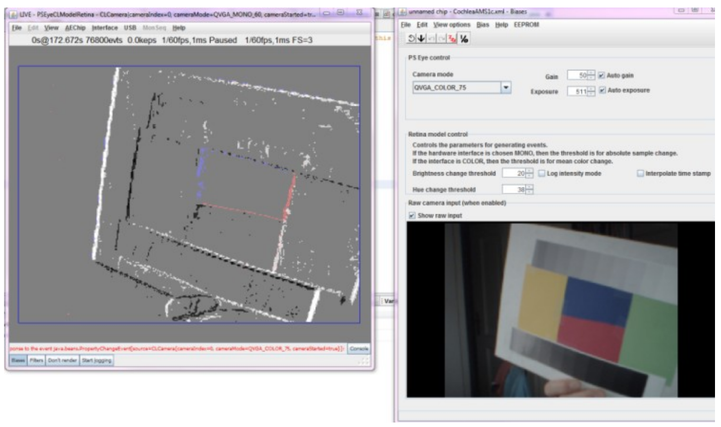

LIVE - PSEyeCLModelRetina - CLCamera[cameraIndex=0, cameraMode=QVGA_MONO_60, cameraStarted=tr...
0s@172.672s 76800evts 0.0keps 1/60fps,1ms Paused 1/60fps,1ms FS=3
ponse to the event java.beans.PropertyChangeEvent[source=CLCamera[cameraIndex=0, cameraMode=QVGA_COLOR_75, cameraStarted=true]]:
unnamed chip - CochleaAMS1c.xml - Biases
PS Eye control
Camera mode: QVGA_COLOR_75
Gain: 50 [✓ Auto gain]
Exposure: 511 [✓ Auto exposure]
Retina model control
Controls the parameters for generating events.
If the hardware interface is chosen MONO, then the threshold is for absolute sample change.
If the interface is COLOR, then the threshold is for mean color change.
Brightness change threshold: 20 [ ] Log intensity mode [ ] Interpolate time stamp
Hue change threshold: 38
Raw camera input (when enabled)
[✓ Show raw input]

Fig. 1. Vision sensor emulation demonstration. The output of jAER simultaneously shows the emulated DVS+cDVS and the raw PS3-Eye camera. Left window shows output of jAER [3]. The emulated pixels respond both to brightness change (black and white pixels) and mean wavelength change (red and blue pixels). The right window shows the control panel which allows camera and emulation parameter control, and also shows the raw PS3-Eye camera output from the colour patch stimulus, which is moved to generated the output events.

Behavioural Emulation of Event-Based Vision Sensors

M. L. Katz[1], K. Nikolic[1], T. Delbruck[2]

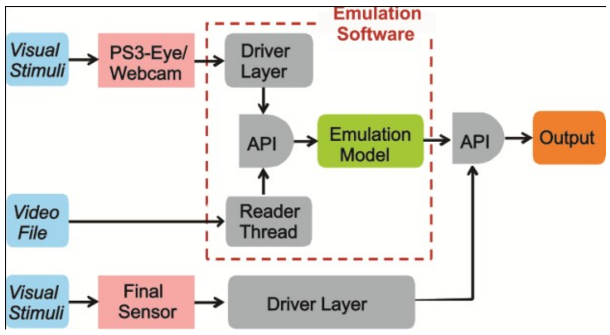
[1] Centre for Bio-Inspired Technology, Dept. of Electrical and Electronic Engineering, Imperial College London, UK
[2] Institute of Neuroinformatics, UZH-ETH Zurich, Switzerland

***Abstract*— Recent advances in activity-driven, event-based vision sensors have demonstrated a need for behavioural emulation tools to explore sensor architectures and applications scenarios in advance of silicon design. This paper describes the technical details behind a live demonstration of the Vision Sensor Behavioural Emulator (VSBE). The VSBE can be used for behavioural software emulation of any event-based vision sensor architecture and functionality, such as for example bio-mimetic silicon retina designs, before the fabrication of sensors. The VSBE uses an inexpensive PS3-Eye camera, which runs at up to 125 frames/second with QVGA (320x240) resolution and mono or colour output. It is used to emulate several event-based vision sensors, including a Dynamic Vision Sensor (DVS) and a colour-change sensitive DVS. The emulator is integrated into the open-source jAER software project which does real-time event-based sensor processing, where we use it to evaluate future vision sensor designs.**

I. Introduction

Recent developments in event-based vision sensors have opened up a promising alternative to conventional frame-based vision [1]. By outputting a sparse and variable-rate stream of data originating asynchronously from the pixels, these sensors reduce processing cost and latency, while increasing dynamic range. These vision sensor developments have generally been led by new silicon chip developments. In connection with the ongoing SEEBETTER project [2], we realized that it would be useful to develop a vision sensor behavioural emulator (**VSBE**) of such chip architectures that would allow us or others to study how to use a proposed vision sensor in application scenarios, and to allow software modelling of non-idealities such as fixed pattern noise and temporal noise. Previous work has already shown the utility of emulation [3], and here we developed a real-time emulation that is fully integrated into our existing host-side processing infrastructure [4-5] to allow side-by-side comparison of new sensors with their emulation in actual application scenarios such as object tracking and robotics, of which there are already numerous existing examples.

This paper describes some details of this emulator. It starts with a description of the hardware camera, driver, and software framework. Next it describes two vision sensor pixels circuits which we emulate. Finally the paper concludes by showing several emulation examples that demonstrate the utility and limitations of emulation.

a)


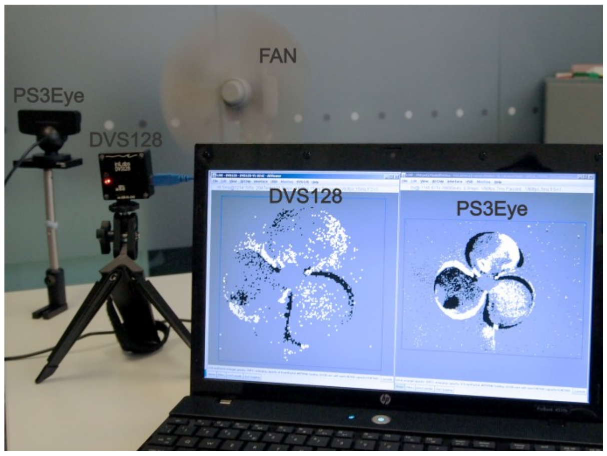
b)


Fig. 1. Vision sensor behaviour emulator. (a) Software architecture. (b) Photo of the emulator setup.

II. Implementation

The VSBE uses a fast conventional imager as its input. Successive image frames are processed in software to generate synthetic time-stamped address-events, as though they had arisen from an event-based sensor. Fig. 1a) shows the software architecture and Fig. 1b) shows a photo of the running implementation, where we compare an existing DVS128 event-based vision sensor with its emulation, using a spinning fan as input.

For the image sensor we chose the Playstation PS3-Eye camera [6]. It is widely available for online purchase and costs about $40. It uses a large-pixel VGA (640x480) Omnivision CMOS image sensor which is optimized for low-light capability. It has a high speed USB interface. A closed-source driver for Windows is available, including a basic Java Native Interface wrapper [7]. The driver starts a native thread that acquires frames from the camera and delivers them on demand to user processes. The driver also supports setting camera modes such and mono or colour, as well as gain and exposure time. In our emulator we presently use only the QVGA (320x240) modes because they provide higher frame rate.

The emulation is integrated into **jAER**, which is a large software project (currently with >700 classes) that has been under development since 2007 and which is our main workhorse for sensor development and application [4-5]. Several existing event-based vision sensors [8-9] are integrated into jAER. The PS3-Eye is integrated into jAER so that events from the VSBE appear to be events from a standard event-based sensor. In the parlance of jAER, the PS3-Eye appears to be just another instance of the class *AEChip* event-based sensor, with the interface *HardwareInterface* that implements the same software interface as other event-based sensors. The resulting setup allows direct comparison of event-based sensors with their frame-based emulation (Fig. 1). To run this emulator, users obtain a working copy of jAER, start the *AEViewer*, and select the chip *PSEyeCLModelRetina*. After installing the PS3-Eye USB driver [7], the PS3-Eye camera appears in the *Interface* menu and is automatically selected as the sensor input.

III. Emulation of the Brightness-Change Dynamic Vision Sensor

The dynamic vision sensor (**DVS**) [1, 9] is an address-event silicon retina that responds to temporal intensity contrast (Fig 2). The output of this vision sensor (like all such event-based sensors) is variable-rate asynchronous stream of pixel addresses. Each output event address represents a quantized change of log intensity at a particular pixel since the last event from that pixel. The address includes a sign bit that distinguishes positive from negative changes (so-called ON and OFF events). Events occur asynchronously, are arbitered by on-chip circuits for access to a shared digital output bus, and are time-stamped to a resolution of 1μs on the camera board. The input to the host PC is a stream of these time-stamped address events.

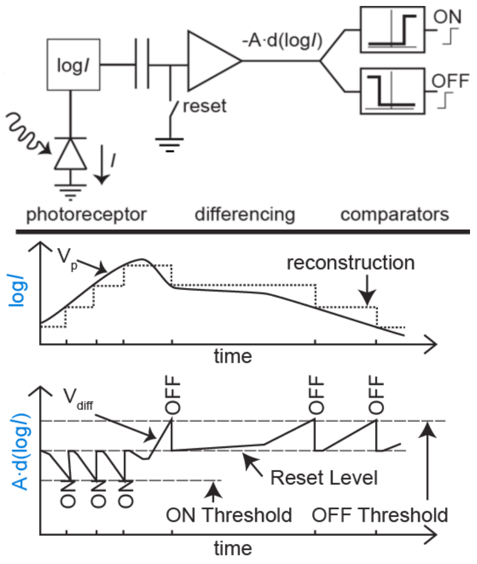


Fig. 2. DVS pixel architecture and operation. The DVS pixel outputs events that represent quantized log intensity changes as shown by the simplified pixel schematic (top). After transmission of the pixel address (not shown) the differencing circuit is reset to memorize the new log intensity value. The time course of internal signals (bottom) shows the internal signals.

To model the DVS, we operate on the sequential image frames from the camera to compute an approximation to the events that would have been generated by the DVS. A 2D memory array holds the sampled brightness values for which each pixel last generated an event. (Here we use the term "brightness" to refer to log intensity.) Pseudo code for DVS event generation for each pixel sample is as follows:

1. Map from the sampled intensity value to its logarithmic brightness (see later).
2. Check to see if the new brightness differs from the stored last-event brightness for that pixel by at least the event threshold. If not, do nothing and go on to step 1 for the next pixel.
3. Output ON or OFF events (according to the sign of the difference) according to the size of the difference divided by the threshold.
4. Store the new sampled pixel value.

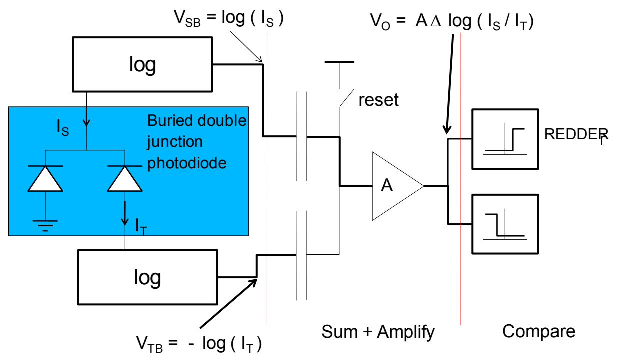
The mapping from linear intensity reading (0-255) to log brightness measure is not useful for discrete sample values near zero because log is a very expansive function for positive values near zero, resulting in a high number of events caused by sensor noise. To handle these small values, we use a "lin-log" mapping which maps linearly up to some value (typically 20) and then maps logarithmically for the rest of the sample values, up to 255.

The timestamp assigned to each synthetic VSBE event is chosen optionally either to be the frame time or a time interpolated between the last frame and this one. If the time is set to the frame time, all events generated by that frame are synchronous, which causes many of the existing event-based processing algorithms which depend on precise event timing to fail. Therefore the optional time interpolation assigns a timestamp to each separate event within a pixel by linearly interpolating between the last and current frame timestamps with the total number of events in that pixel. 5 events from one pixel, for instance, will have timestamps separated by 1/5 of the frame interval. These timestamps serve to desynchronize otherwise simultaneous events but do not increase timing precision, since they are synthetic and the event times are not related to actual visual input.

The pixel-to-pixel variation of the threshold in a real sensor is included in the emulator by randomly assigning a fixed threshold value for each pixel following the normal distribution with the mean and standard deviation which correspond to the emulated sensor.

IV. Emulation of the Colour-Change cDVS Vision Sensor

The colour-change DVS (**cDVS**) vision sensor [10] is aimed at dynamic colour vision applications. The cDVS pixels detect changes of mean colour wavelength by measuring



V_{SB} = log (I_S)
V_O = AΔ log (I_S / I_T)
log
reset
I_S
Buried double junction photodiode
REDDER
A
I_T
log
V_{TB} = - log (I_T)
Sum + Amplify
Compare

Fig. 6. cDVS pixel schematic. The BDJ photodiode (**PD**) currents are continuously converted to log intensity voltages V_{SB} and V_{TB}. I_T is from the top PD and is more sensitive to blue light. I_S is the summed photocurrent from top and bottom PDs. The Sum+Amplify circuit amplifies changes in the difference V_{SS}-V_{TB} since the last event. The Compare circuits asynchronously generate REDDER or BLUER events on changes in the log of I_S/I_T.

changes in the ratio of currents in a buried double junction (**BDJ**) photodiode. The possible applications of this sensor include shadow detection (hue does not change across shadow borders) and tracking of artificial tags labelled by colour patches.

Fig. 6 shows a simplified schematic of the pixel architecture. The switched-capacitor differencing amplifier outputs changes in the log ratio of I_S/I_T, which is the ratio of summed to top photodiode current. This ratio monotonically increases with mean wavelength, as less photons create photoelectrons collected by the top diode PN junction compared with the bottom one.

To emulate the cDVS, we implemented a variety of different algorithms to extract the mean wavelength of the light from the RGB pixel samples. We do not know the spectral sensitivities of the image sensor used in the camera, nor do we know the post processing that is applied to do white-balancing, so any such model will be very approximate. The steps of this algorithm are exactly as for the DVS (Section III), except that rather than computing a brightness value, we compute a synthetic "hue" value from

$$h = b/(r + g),$$

where h is the value for which we detect changes to emit events, and *r, g, b* are the sampled pixel colours.

V. Results

Several recordings were made to compare the emulator DVS with a real DVS128, and to see how the emulated cDVS would respond to colour stimuli.

A qualitative comparison between the real and emulated DVS is shown in Fig. 4a), where the stimulus was a moving cloverleaf pattern from a fan. To perform a quantitative comparison, a moving white bar with linear gradient edges was generated using a 120 Hz LED monitor. The number of events generated by the DVS128 and emulation in response to a single stimulus cycle (0.83s per cycle) is shown in Fig. 5.

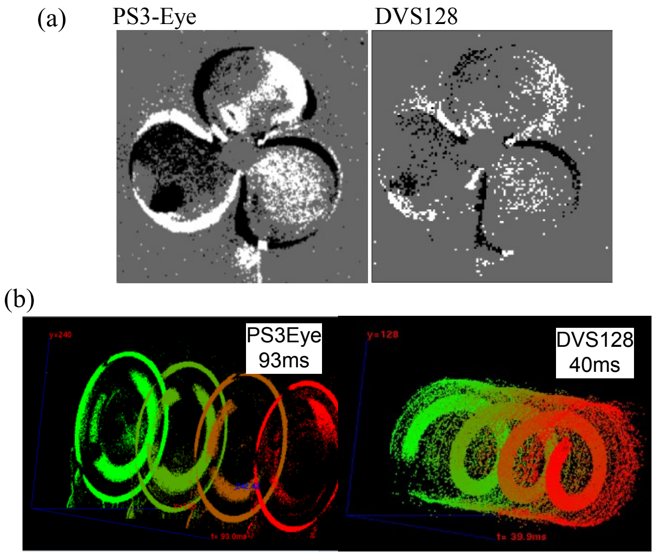


Fig. 4. Comparison of VSBE with DVS128 vision sensor for: (a) a slowly roatating fan, (b) a fast 80Hz rotating black dot.

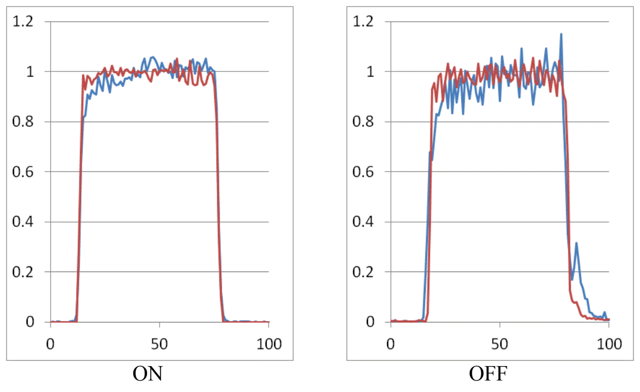


Fig. 5. Comparison of normalised total ON and OFF events generated by the DVS128 (blue) and PS3-Eye emulator (red) in response to a moving bar stimulus binned by the PS3-Eye capture rate (125 fps)

As can be seen, the VSBE is able to correctly emulate the salient features of the DVS behaviour. Both have a characteristic "top-hat" response with the same delay between the onset of ON and OFF peaks (corresponding to the leading and trailing edge of the bar respectively). The VSBE, however, does not capture the "smoothing" of the initial response present in the DVS data. This effect arises from optical distortions due to the lens fitted to the DVS128. We are currently quantifying these distortions and will be adding an optical correction function to future versions of the VSBE.

By comparing the total counts for individual pixels across a single stimulus run the standard deviation of response for the sensor array was calculated. This again showed a very good agreement between the real and emulated DVS with inter-pixel deviations of 13/14% (ON/OFF) and 14/14% (ON/ OFF) respectively.

The limitations of the VSBE vs. DVS can be illustrated by using a fast stimulus. We used the input consisting of a black dot printed on a white disk which spun at 80Hz. This fast stimulus demonstrates the ability of the DVS to capture fast

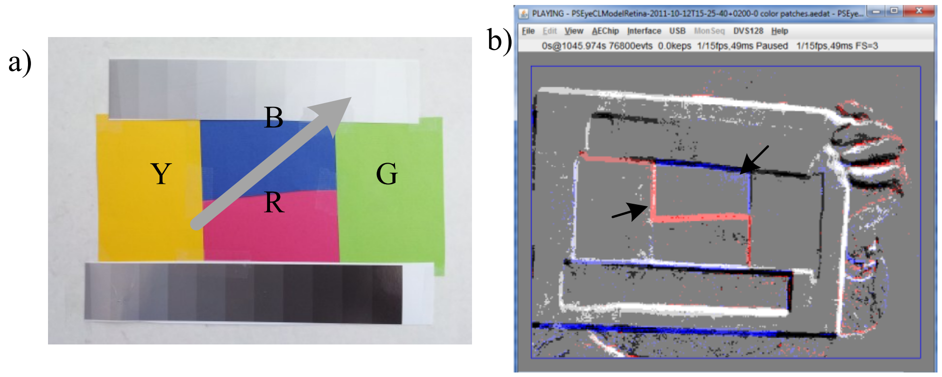

a)
B
Y
R
G
b)
PLAYING - PSEyeCLModelRetina-2011-10-12T15-25-40+0200-0 color patches.aedat - PSEye...
File Edit View AEChip Interface USB MonSeq DVS128 Help
0s@1045.974s 76800evts 0.0keps 1/15fps,49ms Paused 1/15fps,49ms FS=3

Fig. 6. cDVS emulation. a) is the visual stimulus, which moves in the direction of the arrow; patch colours are labelled with letters. The arrow is not part of the pattern. b) shows the DVS+cDVS output for 16ms (1 frame). The REDDER and BLUER events are labelled with arrows. The person holding the pattern is wearing a blue shirt so there are BLUER events at the bottom edge of the pattern.

yet sparse motion, and also clearly demonstrates the limitations of the frame-based emulator (Fig. 4b). The dot causes a continuous helix of events in space-time from the DVS (which are time-stamped to 1us precision), while the VSBE, captured at a frame rate of 60Hz, blurs out the dot over most of each frame, so that the generated events are quantized to the frame times at intervals of 16ms. (The apparent "noise" outside the helix is actually caused by movement of the edge of the white disk, which is not perfectly regular.) Because the frame-based emulation can only generate a single sign of event from each frame – either ON or OFF but not both – it generates events only when the dot moves less than cycle per frame. Otherwise, the dot is blurred into a uniform grey circle. The DVS generates OFF and ON events from leading and trailing edges of the black dot and could respond to multiple dots, for instance.

The operation of the DVS+cDVS emulation is illustrated in Fig. 6. Here the stimulus consisted of a moving pattern of colour and grey scale patches in Fig. 6a, and the model output consists of events of 4 types: BRIGHTER, DARKER, REDDER and BLUER, shown in Fig. 6b as 2D spatial histograms over a chosen time period, in this case one frame of 16ms. The BRIGHTER and DARKER events correspond to the ON and OFF events from the DVS model in Section III and are displayed as gray scale output, while the REDDER and BLUER events are shown in their respective colours. Because the visual pattern moves diagonally upwards to the right, the edges of the blue colour patch are outlined with BLUER and REDDER events at the leading and trailing edges of the patch. Other parts of the pattern are not labelled with colour change events because they produce hue changes (Section IV) that do not exceed the chosen threshold.

VI. CONCLUSIONS

This paper describes a behavioural emulator of activity-driven event-based vision sensors based on a PS3-Eye camera coupled to a standard PC. All code is open-sourced in the jAER project [4]. We believe this emulation platform will be useful for studying the characteristics of proposed event-based sensors. For instance, we are currently interested in studying whether the cDVS colour change output is useful for practical tasks such as shadow vs. non-shadow edge detection and low-cost tracking of artificial colour tags in robotics. The limitations of emulation are also clear: a desktop Core i7 860@3GHz PC runs at a significant load of 40% to emulate the QVGA sensor at 125 frames/sec compared with 0.5% for running the DVS128, and if we want to explore the use of precision event timing, we must drastically slow down the visual input so that the quantized event times from the frame-based emulation are meaningful.

ACKNOWLEDGMENTS

Supported by EU FP6 BioICT project SEEBETTER (270324), Imperial College, the University of Zürich, and ETH Zürich.

REFERENCES

[1] T. Delbruck, et al., "Activity-Driven, Event-Based Vision Sensors," in Proc. of IEEE International Symposium on Circuits and Systems, ISCAS, Paris, France, 2010, pp. 2426 - 2429.
[2] SeeBetter. EU STREP project (number 270324) ICT Call 6 (FP7-ICT-2009-6) in the work programme Brain-inspired ICT. Available: www.seebetter.eu
[3] T. Teixeira, et al., "Address-Event Imagers for Sensor Networks: Evaluation and Modeling " in Proc. of Information Processing in Sensor Networks (IPSN), 2006, pp. 458 - 466.
[4] Welcome to the jAER Open Source Project. Available: jaer.wiki.sourceforge.net
[5] T. Delbruck, "Frame-free dynamic digital vision " in Proc. of Intl. Symposium on Secure-Life Electronics, Advanced Electronics for Quality Life and Society, University of Tokyo, Tokyo, Japan, Mar. 6-7, 2008, pp. 21-26.
[6] PlaystationEye. Available: http://en.wikipedia.org/wiki/PlayStation_Eye
[7] CodeLaboratories. CL Eye Platform Driver. Available: http://codelaboratories.com/get/cl-eye-driver/
[8] Dynamic Vision Sensor (DVS) - asynchronous temporal contrast silicon retina. Available: siliconretina.ini.uzh.ch
[9] P. Lichtsteiner, et al., "A 128×128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor," IEEE Journal of Solid State Circuits, vol. 43, pp. 566-576, 2008.
[10] R. Berner and T. Delbruck, "Event-Based Pixel Sensitive to Changes of Color and Brightness," IEEE Trans. on Circuits and Systems I (TCAS I), vol. 58, pp. 1581-1590, 2011.