Integration of Dynamic Vision Sensor with Inertial Measurement Unit for Electronically Stabilized Event-Based Vision

T. Delbruck, V. Villanueva, L. Longinotti
Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland

Abstract – **Neuromorphic spike event-based dynamic vision sensors (DVS) offer the possibility of fast, computationally efficient visual processing for navigation in mobile robotics. To extract motion parallax cues relating to 3D scene structure, the uninformative camera rotation must be removed from the visual input to allow the un-blurred features and informative relative optical flow to be analyzed. Here we describe the integration of an inertial measurement unit (IMU) with a 240x180 pixel DVS. The algorithm for electronic stabilization of the visual input against camera rotation is described. Examples are presented showing the stabilization performance of the system.**

I. Introduction

Animal vision is made stable against motion blur caused by the animal's head movement by a vestibulo-ocular reflex system. The vestibular system measures the head rotation and rotates the eyes in the opposite direction. People with damaged vestibular systems cannot see clearly during walking or running [1]. Many video cameras also include some form of image stabilization. However, stabilization of conventional video requires expensive rotation and translation of entire image frames and cannot compensate for motion blur during pixel exposure. Some high-end photographic SLR lenses include optical stabilization via a mechanical glass plate that can translate the image slightly, but these mechanical components greatly complicate the lens design and increase its cost, and they cannot rotate the image.

The DVS is a camera that produces address-events as output [2][3]. Each address-event (**AE**) asynchronously signals a log change of intensity at the pixel address since the last event from that address. The AE timestamp (in microseconds) codes the time of the events. DVS sensors are neuromorphic abstractions of the responses of retinal ganglion cells in biological retinas. Their sparse sub-ms latency output and kHz pixel bandwidth has led to applications requiring high speed object tracking with short-latency feedback [8][9]. So far, their application in visual obstacle avoidance, terrain classification, and navigation in mobile robotics has not been widely-explored, although they could enable high speed capabilities at lower computational cost and higher dynamic range than conventional image sensors. Their high pixel bandwidth and event-based output opens the possibility of purely electronic stabilization of the camera output.

A prerequisite of mobile robotic vision systems is that they should be able to maintain a clear view of the scene despite rapid camera movements, particularly camera rotations. The most interesting information for navigation is the relative feature movement in the scene, which is informative about object distance when the robot translates through the environment. In order to measure this image movement, the effect of camera rotation around its own axes must be suppressed so that the relatively small motion signals resulting from ego motion (motion parallax) or target motion can be reliably extracted. The live demonstration paper [6] demonstrated that roll stabilization was possible with a DVS and could be applied by an FPGA, but no details of the implementation were published. The aim of this paper is to describe initial steps in the direction of a full 3-axis rotational stabilization. Specifically, this paper reports the details of hardware integration of an angular rate gyro into a DVS camera and the algorithms used to compute the mathematical transforms that are applied to the sensor output events. This paper also reports experimental results obtained from the electronically stabilized neuromorphic camera.

II. Stabilizing Visual Input With Rate Gyro

Camera rotation is defined as the camera's rotation around its own axes. The effect of camera rotation on the image is to cause transformation of the original image, consisting of both image translation and image rotation, along with other distortions not discussed here.

The IMU is mounted on the back of the DVS camera, directly behind the vision sensor chip so that it shares nearly the same center of rotation The IMU

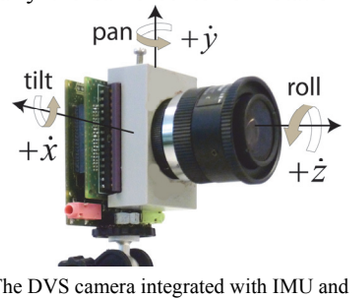

Fig. 1. The DVS camera integrated with IMU and showing the rate gyro axes.

978-1-4799-3432-4/14/$31.00 ©2014 IEEE

supplies rotational angular rates around the 3 camera axes as indicated in Fig. 1: The quantities $\dot{x}, \dot{y}, \dot{z}$ denote the tilt (up/down), pan (right/left) and roll (camera axis rotation) angular rates. The DVS output is stabilized against camera rotation by first integrating these angular rates over time, starting at some selected moment, to produce total rotational angles x, y, z . Each DVS event address $\vec{\mathbf{e}} = \{e_x, e_y\}$ consisting of the e_x and e_y pixel address is then transformed according to the integrated camera rotation to bring the event back to the arbitrary starting point, by applying the transform (1):

$$\vec{\mathbf{e}}_t = \left[\mathbf{R}^{\mathrm{T}}\left(\vec{\mathbf{e}} - \vec{\mathbf{e}}_0\right) - \vec{\mathbf{T}}\right] + \vec{\mathbf{e}}_0 \qquad (1)$$

where the 2-vector $\vec{\mathbf{e}}_t$ is the transformed event address. $\vec{\mathbf{e}}_0$ is the pixel address nearest the center of the IMU, $\mathbf{R}$ is the 2x2 roll-angle image rotation matrix, and $\vec{\mathbf{T}}$ is the pan and tilt translation 2-vector in the image. (The reader is reminded that $\vec{\mathbf{T}}$ is the translation of the image and has nothing to do with unknown camera translation through space.) Eq. (1) rotates and then translates the event address, as illustrated in Fig. 2. These computations assume no lens distortion as indicated in Fig. 2 by the unchanged shape of the rectangle. This assumption becomes increasingly imprecise for wide-angle lenses.

$\mathbf{R}$ and $\vec{\mathbf{T}}$ are computed by integrating the IMU angular rates over time and then high-pass filtering the results by a first-order IIR filter. The high-pass filter is used so that after camera motion stops, the transform eventually decays back to zero transform, so that the transformed output is centered. The high-pass filter also deals to some extent with non-zero gyro offsets, which we observed can be as large as 2 °/s. The *k*'th IMU samples $\dot{x}_k, \dot{y}_k, \dot{z}_k$ are captured at times t_k. The integrated angles are denoted *x,y,z* and the high-pass filtered integrated angles are x_f, y_f, z_f . The time constant of the high-pass filter is τ. *x(k)* and x_f(k) are computed by (2):

$$\begin{aligned} \Delta t_k &= t_k - t_{k-1}, \qquad \Delta x_k = \dot{x}_k \Delta t_k \\ x(k) &= x\left(k-1\right) + \Delta x_k \\ x_f(k) &= (1 - \frac{\Delta t_k}{\tau}) x_f(k-1) + \Delta x_k \end{aligned} \qquad (2)$$

Analogous computations are performed for *y* and *z*. When any of the x_f, y_f, z_f values exceed a programmed limit, then all are set back to zero, to deal with continued camera rotation. These resets in effect implement saccades to re-center the transformed output. The translation vector **T** and the rotation matrix **R** are computed by (3):

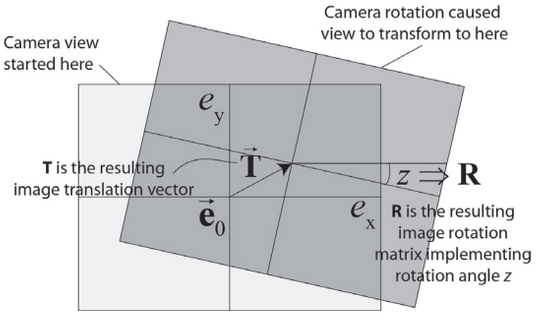

Fig. 2. Transform coordinates.

$$\vec{\mathbf{T}} = k\begin{bmatrix} y_f \\ x_f \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \cos z_f & -\sin z_f \\ \sin z_f & \cos z_f \end{bmatrix} \qquad (3)$$

The factor *k* scales from camera pan and tilt rotation angle radians to DVS pixels, and is computed from the square pixel size of $w = 18.5$um and the lens focal length $l = 6$mm . If one pixel subtends an angle θ_{pix}, then $\tan\theta_{\mathrm{pix}} = w/l$ and *k* is given by (4):

$$k = 1/\tan^{-1}\left(w/l\right) = 5.66 \text{ [pixels/rad]} \qquad (4)$$

III. CAMERA IMPLEMENTATION

For the work presented here, a new generation DVS sensor with 240x180 pixels and concurrent image sensor output was used [4][5] (the image sensor output is not considered in this paper). The IMU was integrated into the prototype camera. Fig. 1 shows the camera with IMU and Fig. 3 shows the block diagram of the integration.

This MPU-6150 IMU was selected because of its high level of integration and I²C output. It integrates a 3 axis user-programmable gyroscope with selectable full-scale ranges of ±250, ±500, ±1000, ±2000 °/s and a user-programmable accelerometer with selectable full-scale ranges of ±2g, ±4g, ±8g and ±16g, which was not used here. It has a small 4x4x0.9mm QFN package. It integrates six 16-bit ADCs for digitizing the gyroscope and accelerometer outputs at 1kHz sample rate. These outputs can be processed by the an integrated Digital Motion Processor™ although only the basic synchronized data capture feature has been used in this work.

The IMU was positioned as closely as possible to the center of the sensor image plane, with only a few millimeters displacement due to thickness of the PCB

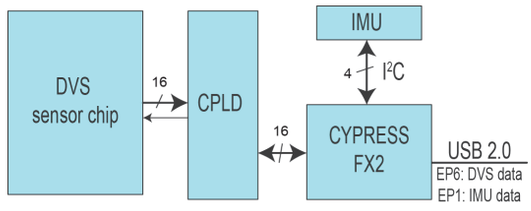

Fig. 3. Camera implementation block diagram.

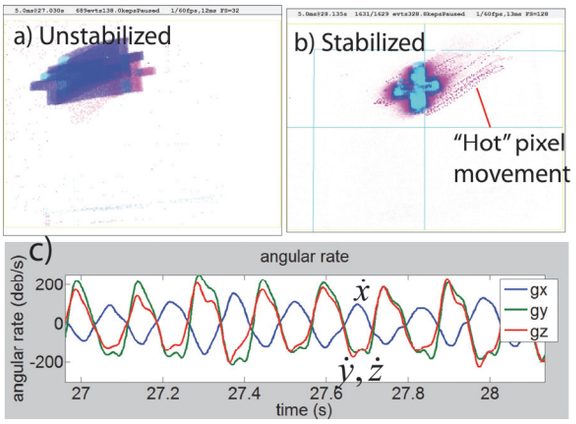

Fig. 5. Stabilization example. Scene was 1s of time viewing black cross on white paper with edge of paper at bottom left at distance of 50cm with 6mm lens. DVS events are rendered as ON=red, OFF=green. a) Unstabilized output. b) Stabilized output. A "hot" pixel shows the image movement. c) The rate gyro outputs during time slice showing 7Hz oscillation frequency.

and the sensor socket. The IMU axes are aligned with the camera axes. The IMU sensors are built with a small piece of silicon floating in 3 axes. Because the total mass of this floating piece is so small it is easily attracted by electromagnetic fields, disturbing the measurements. To avoid these electromagnetic fields and obtain better position data, the IMU is protected by a guard ring connected to ground in the same plane as the IMU and two ground planes between the DVS sensor and the IMU. The IMU interfaces to the camera's Cypress FX2 USB microcontroller via the FX2 I²C interface. IMU samples are captured by the FX2 and transmitted to the PC host over different USB endpoint than the DVS event stream endpoint.

Communication between the MPU-6150 and Cypress FX2 is performed by I²C at 400kHz. The FX2 captures the 6 gyro/accelerator samples by polling the data-ready I²C register and then reading the samples. Only about 50 lines of C code are needed on the FX2 to set up the IMU and to handle data capture and USB transfer.

On the USB host, the IMU samples are merged into hybrid event packets that contain both DVS events and IMU samples. The sensor GUI enables control of the transform computation and viewing of the IMU samples together with the DVS events. The GUI controls the highpass filter time constant and provides a button to capture zero angular rate calibration values. (The gyro on the camera had a persistent 2 °/sec offset for one axis, so this calibration function was useful.)

The stabilization algorithm is implemented in the open-source jAER project [7], as the Java class named *Steadicam,* which is in the project package *org.capocaccia.cne.jaer.cne2012.vor.* jAER supplies infrastructure and algorithms for event-based sensor

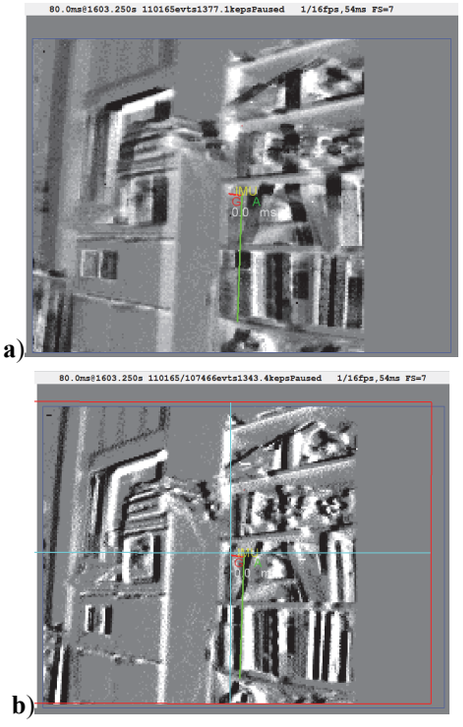

Fig. 4. Stabilization example showing 80ms of DVS events from an office scene drawn as 2d histograms of events. a) Unstabilized output. b) Stabilized output. The red square and blue cross hairs show the current transform.

processing. *Steadicam* is 950 lines of Java code. The stabilization transform is updated after each IMU sample is received. Therefore the computational cost to apply the electronic stabilization is a bit more than 6 multiply-adds per event.

IV. RESULTS

YouTube videos demonstrating stabilization are available [10]. To characterize the electronic stabilization performance the camera was mounted on a pan-tilt unit using two HiTec digital servos and controlled by jAER [7] using the *PanTiltAimer* class.

Fig. 4 shows DVS data captured from an office scene with a bookshelf and file cabinet and rendered as a 2d histogram of ON and OFF events over a time slice of 80ms. Fig. 4a shows that without electronic stabilization the features are blurred out over about 20 pixels. Fig. 4b shows that when *Steadicam* stabilization is enabled, the data from the same time slice shows edges that remain sharper and blur reduced to about 5 pixels. The red square indicates the current transform. At that moment, the camera was panning to the right and the transform indicates this by showing a shift to the left.

Fig. 5 shows a 1s time slice example of stabilization where the camera was viewing a black cross on white paper. In the unstabilized output (Fig. 5a) the cross is completely blurred out. In the stabilized output

(Fig. 5b) the cross is clearly visible despite camera rotation causing image movement larger than the cross at a frequency of ~7Hz and a peak amplitude of 200 °/sec (Fig. 5c). The camera movement is visible in the indicated track of the always active "hot" pixel. The measured real-time cost of processing in *Steadicam* [7] was 58ns/event on a Core i7 975 @ 3.33GHz PC running Windows 7x64 and Java 1.7. The overhead on this machine simply to process event packets is 38ns/event, so each the additional cost to stabilize the retina output is 30ns/event.

V. CONCLUSION

This paper reports the integration of a commercial IMU with an event-based neuromorphic camera to result in electronically stabilized output. The main accomplishment is the concrete realization of the system and its open-source software implementation. The addition of the IMU to the camera increases camera power consumption by about 10mW when the IMU samples at 1kHz. Compared to the 10mW for the vision sensor [3][5] and the approximately 100mW for the rest of the camera electronics (USB and CPLD) the 10% increase in power consumption is small and provides a rich source of additional information for visual navigation.

There are two drawbacks to the current implementation. The first is that although the I^2C interface to the FX2 is simple to implement, it complicates time-stamping the IMU data because the DVS event timestamps are not available to the microcontroller. (They are part of the FX2 FIFO interface to USB, which the microcontroller has no access to during high speed USB transfers). Currently the IMU timestamps are generated by a timestamp counter on the FX2 that runs simultaneously with the CPLD timestamp counter; however because of hardware limitations the timestamps clocks cannot be kept perfectly in synchronization for long periods of time. The second drawback is that sending the data over a separate endpoint to the host greatly complicates the later merging of the data into a single processing thread on the host processor. Both difficulties will be addressed in the next generation implementation by integrating the I^2C interface to the IMU onto the CPLD. The CPLD logic will be modified so that the IMU data is sent along with the DVS events on the same endpoint using the same timestamp source.

When the angular velocity can change suddenly, it is important to minimize the delay between measurement and computed transform. The measured IMU sample rate on the host PC is 1kHz (1ms sample interval achieved with 400kHz I^2C bus speed and minimum endpoint polling interval). The specified MPU-6150 gyro bandwidth is 188Hz with a delay of 1.9ms at the sample rate of 1kHz used in this work [11]. These specifications suggest that further improvement in stabilization performance should be possible with tighter synchronization of the timing of the IMU samples and DVS events.

In its current form, the stabilized camera can reduce feature blur significantly. Future work will improve the hardware implementation, include lens distortion effects in the mathematics, and include non-linear predictive filtering of the rate gyro outputs. After this, we intend to explore the extraction of relative feature motion that cues object distance and can act as a powerful source of information for obstacle avoidance.

Acknowledgements

This work was supported by the Swiss NCCR Robotics, EU projects SEEBETTER and VISUALIZE, and the DARPA SyNAPSE project. We gratefully acknowledge the opportunity to initiate this work at the Telluride and Capo Caccia Neuromorphic Workshops, the gift of MPU-6150 samples from the Invensense University Program, and the helpful comments of the reviewers.

References

[1] E.R. Kandel, J.H. Schwartz, T.M. Jessell, eds. *Principles of neural science.* Vol. 4. New York: McGraw-Hill, 2000.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor," IEEE J. Solid-State Circuits, vol. 43, no. 2, pp. 566–576, 2008.

[3] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-Driven, Event-Based Vision Sensors," IEEE Intl. Symp. Circuits and Systems (ISCAS 2010), Paris, 2010, pp. 2426-2429.

[4] R. Berner, C. Brandli, M. Yang, S. -C. Liu, and T. Delbruck, "A 240×180 120dB 10mW 12μs-latency Sparse Output Vision Sensor for Mobile Applications," Proc. of the 2013 Intl. Image Sensor Workshop (IISW 2013), pp. 41-44, 2013.

[5] R. Berner, C. Brandli, M. Yang, S. -C. Liu, and T. Delbruck, "A 240×180 120dB 10mW 12μs-latency Sparse Output Vision Sensor for Mobile Applications," IEEE VLSI Symposium, pp. 186-187, 2013.

[6] A. Jimenez-Fernandez, et al.. "Live demonstration: Neuro-inspired system for realtime vision tilt correction." Proc. of IEEE Intl. Symp. Circuits and Systems (ISCAS 2010), pp. 1393-1393.

[7] *jAER Open Source Project.* Available: http://www.jaerproject.org . Accessed 3 Feb 2014.

[8] M. Lang and T. Delbruck, "Robotic Goalie with 3ms Reaction Time at 4% CPU Load Using Event-Based Dynamic Sensor," *Frontiers in Neuromorphic Engineering*, 2013, pp 223.

[9] J. Conradt, C. Berner, M., and T. Delbruck, "An Embedded AER Dynamic Vision Sensor for Low-Latency Pole Balancing," 5th IEEE Workshop on Embedded Computer Vision (in conjunction with ICCV 2009), Kyoto, Japan, 2009, pp. 1-6.

[10] Stabilizing DVS output with IMU rate gyros. Available http://www.youtube.com/watch?v=_twn_x94ins&feature=share&list=PLVtZ8f-q0U5iROmshCSqzzBNlqpUGbSkS. [Retrieved 12.10.2013].

[11] "MPU-6100 and MPU-6150 Register Map and Descriptions, Revision 2.1," Invensense Inc. 2013.