Proc. of the International Conference on Circuits and Systems (ISCAS), Taipeh, Taiwan, accepted (2009)

A Pencil Balancing Robot using a Pair of AER Dynamic Vision Sensors

J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, RJ. Douglas, T. Delbruck
Institute of Neuroinformatics
UZH-ETH Zurich, Switzerland
Email: {conradt, cook, raphael, patrick, rjd, tobi}@ini.phys.ethz.ch

***Abstract*—Balancing a normal pencil on its tip requires rapid feedback control with latencies on the order of milliseconds. This demonstration shows how a pair of spike-based silicon retina dynamic vision sensors (DVS) is used to provide fast visual feedback for controlling an actuated table to balance an ordinary pencil. Two DVSs view the pencil from right angles. Movements of the pencil cause spike address-events (AEs) to be emitted from the DVSs. These AEs are transmitted to a PC over USB interfaces and are processed procedurally in real time. The PC updates its estimate of the pencil's location and angle in 3d space upon each incoming AE, applying a novel tracking method based on spike-driven fitting to a model of the vertical shape of the pencil. A PD-controller adjusts X-Y-position and velocity of the table to maintain the pencil balanced upright. The controller also minimizes the deviation of the pencil's base from the center of the table. The actuated table is built using ordinary high-speed hobby servos which have been modified to obtain feedback from linear position encoders via a microcontroller. Our system can balance any small, thin object such as a pencil, pen, chop-stick, or rod for many minutes. Balancing is only possible when incoming AEs are processed as they arrive from the sensors, typically at intervals below millisecond ranges. Controlling at normal image sensor sample rates (e.g. 60 Hz) results in too long latencies for a stable control loop.**

I. Introduction

Balancing an object has been used for many years as a demonstration of controller design. Such demonstrations in teaching robotics and robotics contests often are limited to a single pole rotating about one constrained axis for simplicity, and typically use a position encoder at the bottom of the object providing the current angle relative to desired balanced orientation.

Normal image sensors are hard to use for balancing small objects because the frame rate limits the response latency, necessitating complex nonlinear control methods that can control despite the large signal nonlinearities that develop with controller delay. One example available online [1] shows a Sarcos industrial robot balancing a pole approximately 1 m long with a weighted top and two colored markers which are used for tracking. Since the angular acceleration by gravity is inversely proportional to the distance of the center of mass (COM) from the base, the weighted top shifts the center of mass away from the hand, easing the task significantly. An unmodified rod of length L has its COM $c = \frac{1}{2} \cdot L$ from the base. The angular acceleration ($\ddot{\alpha}$ in rad/s^2) caused by gravity (g) for small angle α (where $sin(\alpha) \approx \alpha$) is given by $\ddot{\alpha} = (\alpha \cdot g)/c$, representing an angle that increases by a factor of e every sqrt(c/g) seconds. For a normal pencil of length L=20cm, the angle increases 10% every 10ms. A simple linear controller must react within a few ms to balance such an object t.

Here we show how the use of data-driven spike based sensors facilitates the application of straightforward linear control policies with low computational requirements, that allows visually guided balancing of normal pencils, which otherwise represents a challenging control problem. The paper describes the vision sensors and their spiking output data, the algorithm to compute the pencil's position in 3d space, the linear control policy and the balancing robot. We present and discuss measurements from the running system, and conclude with a comparison using conventional image sensors.

II. Dynamic Vision Sensor

The dynamic vision sensor (DVS) [2, 5] used as a pair in this demonstration is an address-event silicon retina that responds to temporal contrast (Fig 1). Each output spike 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.

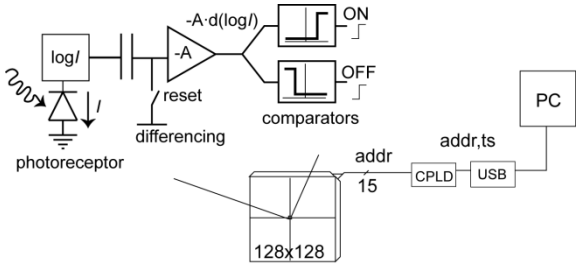

Figure 1. DVS pixel and camera architecture. The DVS pixel outputs events that represent quantized log intensity changes as shown by the simplified pixel schematic (top left). After transmission of the pixel address (addr), the pixel is reset. The addresses are timestamped (ts) by the camera's CPLD on the way to the USB chip's FIFO, and packets of up to 128 TAEs are transferred to the host PC over USB.

The DVS has a USB interface that delivers time-stamped address-events (TAEs) with a resolution of 1us to a host PC, although in the present applications the timestamps are so far unused. The DVS' specifications are summarized in table I.

TABLE I. THE DVS'S ELECTRICAL SPECIFICATIONS

Dynamic range	120dB
Contrast threshold mismatch	2.1%
Pixel array size	128x128 pixels of $(40u)^2$
Photoreceptor bandwidth	>= 3KHz
Event saturation rate	1 M-event per second
Power consumption	23mW

The USB interface delivers packets of up to 128 events to the host PC at a rate of at least 100Hz, but the packets of TAEs can be sent at a minimum interval of about 128us, limited by the USB2.0 high speed polling interval.

III. BALANCER HARDWARE

The balancer hardware (Fig. 2) consists of a custom-built table capable of moving its hand cup in X and Y directions within a range of 100x100mm. Two parallel linear rollers, mounted 120mm apart, support a bar with a third linear roller mounted perpendicular on them. The small cart on the upper linear roller slides with little friction in Y-direction, but can also slide easily in X-direction by moving the support bar on the parallel rollers. The hand mounted on the upper cart is a small cup lined with foam rubber into which a pencil or other sharp pointed object can be placed; the foam rubber provides sufficient friction to be able to rapidly accelerate the hand without slippage, but does not otherwise support the pencil.

The table is actuated in X and Y directions by two independent servo motors. The rotary heads of these servos connect via a 2-segment arm of length 2x100mm to either one of the slides, which allows the servos to move the linear rollers by rotating their servo head. The lengths of the arms and the types of servos (lower bar Futaba BLS451, upper cart Futaba BLS251) have been selected to yield sufficient torque and fast response times, maximizing the speed of motion for the two slides based on their different mechanical characteristics (e.g. weight, friction).

Mounted on the bar and on the underlying table are linear position sensors (SpectraSymbol SoftPot SP-L-0100-101-ST), whose voltage output represents the current position of the cart in X and Y directions. A 32-bit microcontroller (NXP LPC2103) running at 64MHz reads the position information at a rate of 8KHz, low-pass filters the data, and computes servo control signals. This uC implements a high-speed low-level control loop to quickly reach desired hand positions in X/Y space.

Both servo motors have been modified so that their internal angle feedback potentiometer is replaced by a constant voltage divider. Each servo's build-in control electronic assumes the servo to be permanently in center position; therefore the pulse width of a PWM signal applied to the servo motor now directly corresponds to the rotary speed of the servo head. The servos receive such PWM control input generated by the uC at a frequency of 250Hz, and measurements show that changes in PWM width cause movement of the hand within about 25ms.

The uC is interfaced to a host PC via a serial-to-USB converter (FTDI-chip FT232R), allowing the PC to specify desired hand positions. The rate of commands sent is limited to a minimum interval of 2ms in software, to avoid saturating the FTDI's USB bus interface.

The cost of assembling the entire table was about US$1500, strongly dominated by the custom machining cost for the servo arms, which are aluminum that is milled out for low moment of inertia.

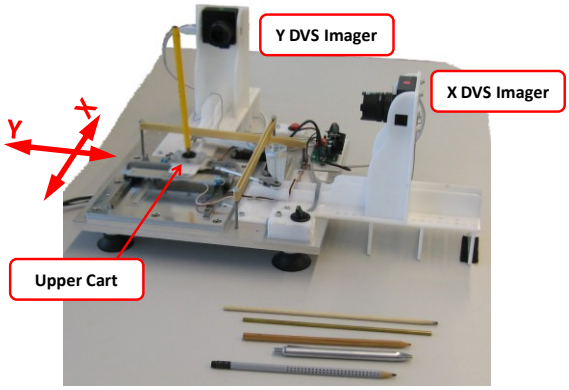

Figure 2. Photo of balancer hardware: 2 DVS (top center and top right), the motion table (center left) on linear rollers actuated by two servos. The system can balance all objects show at the bottom without modification of parameters.

IV. BALANCER SOFTWARE ARCHITECTURE

Processing of TAEs from the 2 DVSs and generation of desired hand positions is computed on a 2.6GHz Core2 Duo PC running Windows XP and jAER [4, 6], a Java software package that facilitates computation in AE-based systems. The TAEs from the 2 DVSs are captured over separate USB2.0 interfaces. Each interface is served by its own thread running at maximum priority, and the data from each DVS is processed at the moment that it is delivered from the USB driver. Optional rendering of DVS output is done in a separate lower-priority thread that combines the 2 streams into a single binocular stream displayed on the PCs monitor along with annotations such as the pencil tracker output. A jAER GUI allows adjustment of tracker and control parameters. All source code for the balancer is open-source and can be examined in the package ch.unizh.ini.jaer.projects. pencilbalancer in the jAER project [6].

V. PENCIL TRACKING ALGORITHM

The algorithm we use to track the pencil proceeds in two stages. The first stage, done independently for each vision sensor, uses each incoming event to efficiently update an estimate of the line where the pencil appears to be from that vision sensor's point of view. The second stage combines the two line estimates from the two vision sensors, taking perspective into account, to generate a 3D estimate of the line containing the pencil.

The goal of the first stage is to identify the line corresponding to where the pencil is in vision sensor coordinates from the sensor's point of view. Every event that arrives is treated individually to update the estimate of this line. The estimate of the line is maintained as a Gaussian in the "Hough

space" of possible lines. Since the log of the Gaussian is a quadratic, our estimate of the line is stored as just the 5 coefficients of the quadratic (not including the constant term). This allows our estimate to be stored very compactly, with no discretization.

The equation for the line, x=m·y+b, describes a line in m and b (the two dimensions of the Hough space) when x and y are given by the event. To update the quadratic for a newly arrived event, we simply decay the old quadratic slightly, replacing the decayed portion with the information from the new event. Since the new event corresponds to a line in the Hough space, it contributes a quadratic which has its minimum on this line, and grows parabolically away from the line. This quadratic has coefficients $A=y^2$, B=2·y, C=1, D=-2·x·y, and E=-2·x, given an event from pixel (x,y). When we need to produce an estimate of the slope and x-intercept of the pencil, we report the lowest point of the quadratic, which is given by (b,m) = (D·B - 2·A·E, B·E - 2·C·D)/q, where q=4·A·C-B·B.

The second stage combines the two lines from the first stage into a single line in 3D space. If the true position of the pencil is given by $(x,y,z)=(X+t\cdot\alpha_X, Y+t\cdot\alpha_Y, t)$, then the sensors at (0,-yr,0) and (xr,0,0) will see the line as:

$$(x,z) = ((X+t\cdot\alpha_X)/(Y+t\cdot\alpha_Y+yr), \quad t/(Y+t\cdot\alpha_Y+yr))$$
$$(y,z) = ((Y+t\cdot\alpha_Y)/(xr-X-t\cdot\alpha_X), \quad t/(xr-X-t\cdot\alpha_X)).$$

We receive these as a base and slope for each sensor:

$$b1 = (X/(Y+yr), 0) \qquad s1 = dx/dz = \alpha_X - X\cdot\alpha_Y/(Y+yr)$$
$$b2 = (Y/(xr-X), 0) \qquad s2 = dy/dz = \alpha_Y + Y\cdot\alpha_X/(xr-X).$$

We can solve these for X,α_X,Y,α_Y in terms of b1,s1,b2,s2:

$$X = (b1\cdot yr + b1\cdot b2\cdot xr) / (b1\cdot b2+1)$$
$$\alpha_X = (s1 + b1\cdot s2) / (b1\cdot b2+1)$$
$$Y = (b2\cdot xr - b1\cdot b2\cdot yr) / (b1\cdot b2+1)$$
$$\alpha_Y = (s2 - b2\cdot s1) / (b1\cdot b2+1)$$

This yields the position (X,Y) of the pencil at the height of the cameras, as well as the slope of the pencil (α_X,α_Y).

VI. Control System

The pencil tracking algorithm reports its estimate of the pencil's position in 3d space, represented as a pair of position coordinates (X,Y), and corresponding slopes in x and y directions (α_X,α_Y). A PD-controller generates desired hand positions (X_{des},Y_{des}) based on these four inputs and the position time derivatives $(\dot{X},\dot{Y})$. In our system all final desired target values (positions, slopes, and velocities) are zero to keep the pencil upright in the center of the table. X_{des},Y_{des} are computed as follows:

$$X_{des} = g_P X + g_\alpha \alpha_x + g_D \dot{X}$$
$$Y_{des} = g_P Y + g_\alpha \alpha_y + g_D \dot{Y}$$

Here g_P, g_α, and g_D denote the gain parameters for base position, slope, and base velocity, respectively. Selecting $g_P = 1$ (with g_α, $g_D = 0$) moves the hand exactly underneath the current center of the pencil; whereas values of g_p slightly larger than 1 move the hand further outside, helping the pencil to tilt backwards towards the center of the table. Intuitively, the middle term has a similar effect based on the pencil's current slope: for larger gains g_α, the current tilt of the pencil more strongly influences the future position of the hand. The last term counteracts recent drift of the pencil.

We found a large range of gain settings for which the system exhibits consistent performance. Typical settings are:

$$g_P \approx 1.3\pm0.2 \quad g_\alpha \approx 250\pm100 \quad g_D \approx 70\pm20$$

VII. Results

Videos of the balancer in operation that demonstrate its performance are available online [7].

Our system normally balances an object for several minutes before losing it. During operation, the PC processor load varies between 10% and 22% when rendering is disabled; rendering at 60 FPS increases processor load to around 33%. Events are processed by the core algorithm at a rate of about 3 million events per second. The total event rate during operation is between 200k and 300k events/second. Fig 3 shows a histogram of measured update intervals, which typically range between 125us and 300us, but can reach up to 1.5ms when the pencil is not moving rapidly, or when a USB1.0 table command is sent. Although USB, Windows XP, and Java are not typically considered real-time control environments, we were pleasantly surprised that the maximum observed update interval was below 10ms. The minimum observed intervals of 125us between received packets of TAE suggest that USB2.0 polling interval limits the rate. The average update rate is about 4 times higher than observed in a previous robotic goalie [3], because here the average event rate is much higher.

We used a variety of illumination sources ranging from uneven (shadowed) sunlight through the windows to fluorescent lighting of 300lux to incandescent lighting from a table lamp. Low illumination degraded performance slightly by increasing DVS noise and decreasing pixel bandwidth, but generally the balancer is tolerant to a wide range of illumination conditions as long as it is relatively steady.

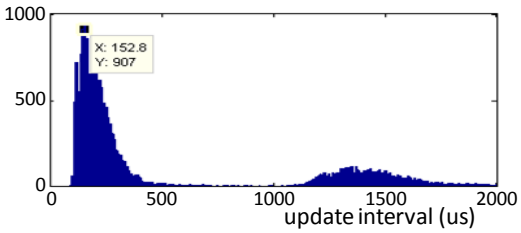

Figure 3. Histogram of update intervals. Median interval was 236us; maximum interval was 9.8ms. The bimodal distribution is due to USB1 and USB2-devices sharing the bus.

Fig. 4 shows TAEs emitted from one of the two DVS due to pencil motion along the blue time axis. The solid black axes represent the field-of-view in sensor space: an event's position on the vertical axis corresponds to its height along the pencil; its position on the horizontal axis shows its displacement with respect to the center of the setup. Each black dot denotes a reported event at a given position in sensor space and time. The red pencil at t = 640ms shows the current estimate of pencil base and slope. This space-time plot shows oscillations of pencil position and tilt within the 640ms time window.

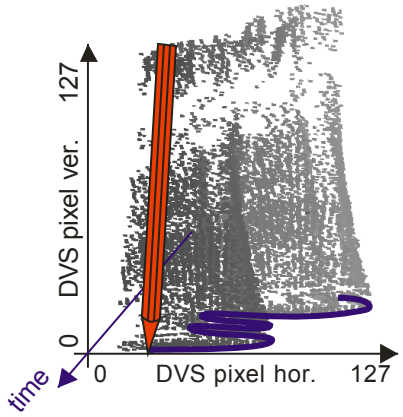

Figure 4. Space-time plot of 54k events (dots) reported from one DVS sensor during balancing in a time window of 640ms. The pencil's base over time and the last tracked position are shown in blue and red. The lower density of events close to the top reflects lower contrast of the pencil's rubber holder in silver-metallic.

Fig. 5 shows recorded control data from our system during 2 seconds of operation. For clarity, we display data of a single dimension only. The top panel shows raw unfiltered data whereas the bottom panel shows the same data processed by a low-pass filter for clarity. The blue trace shows the estimated position of the pencil (X) over time; the red trace a 100-fold amplification of the pencil's slope (α_X). Both these signals are obtained based on spiking visual input only. The green trace shows the desired position of the cart, as computed in section VI. The blue position trace follows the green desired position trace with an average delay of about 50ms. This delay is probably dominant in limiting the possible object length that can be balanced.

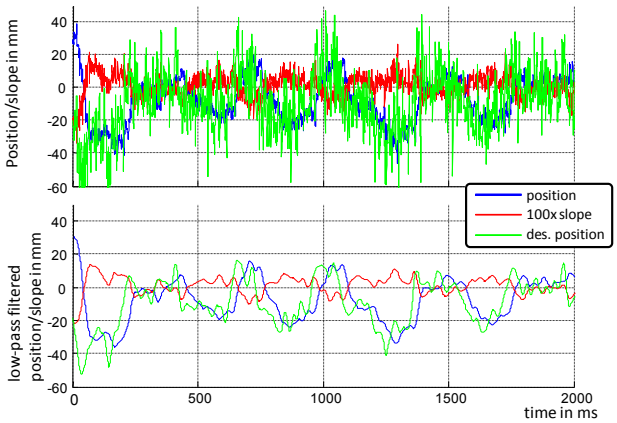

Figure 5. Recorded traces of position, slope and desired position during a 2s time window. Upper graph: raw data; lower graph: same data filtered in 3rd order Butterworth filter (-3dB cutoff frequency set to 30Hz) for clarity.

Fig. 6 is a histogram of X, Y-positions visited by the table during balancing. The plot clearly shows that the cart typically stays close to the center of the table, but occasionally needs much of the available motion space. The center of balancing is shifted relative to the table's origin, indicating an offset between the centers of the two DVS and the center of the table. In fact, we never properly calibrated the visual system with the actuated table.

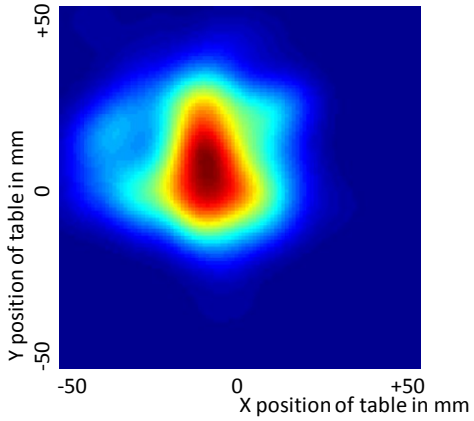

Figure 6. Histogram of relative occurrances of true table positions, red denoting areas often visited.

VIII. Conclusions

This paper describes a balancer demonstration that uses spike-based vision sensors coupled to a standard PC running a standard preemptive OS. The low latency and sparse output of the sensors enable a straightforward solution to this balancing problem, which challenges conventional imaging based systems. A solution to balancing small objects using standard image sensors requires running at >1 KHz frame rate. Even at the relatively low spatial resolution of 128x128 = 16k-pixels of the DVS sensors, image analysis requires processing pixels at a rate of 2·16k·1k=32M pixels/second for data acquisition, which would saturate a USB2.0 hub. Log conversion for temporal contrast extraction and subtraction against stored values to obtain event-like equivalents to the DVS output would require several hundred MIPs of processing before the remaining processing described here. Quantization noise at the low end of the conversion scale would limit low light performance severely, as would the 1ms exposure times, necessitating bright and uniform lighting. The use of AER sensors and event-driven methods for computation has simplified and reduced the cost of implementing this demonstration, and has shown the advantages of the event-driven style of computation used in brains.

Acknowledgment

We thank the Institute of Neuroinformatics and the UZH-ETH for supporting this work.

References

[1] Available: http://www.youtube.com/watch?v=lwvTyC7m4LQ

[2] A 128×128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor, (2007) Lichtsteiner, P., C. Posch and T. Delbruck. IEEE Journal of Solid State Circuits, Feb. 2008, 43(2) 566-576.

[3] Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. (2007) T. Delbruck, T. and P. Lichtsteiner, ISCAS 2007, New Orleans, 27-30 May 2007 Page(s):845 - 848.

[4] Frame-free dynamic digital vision, T. Delbruck, Proceedings 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.

[5] Available: http://siliconretina.ini.uzh.ch

[6] Available: http://jaer.wiki.sourceforge.net

[7] Available: http://www.ini.uzh.ch/~conradt/projects/PencilBalancer.