Steering with an aVLSI Motion Detection Chip

Rico Moeckel, Roger Jaeggi, and Shih-Chii Liu
Institute of Neuroinformatics
University of Zürich and ETH Zürich
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
Email: moeckel,jaeggir,shih@ini.phys.ethz.ch

Abstract— **We demonstrate the capabilies of a 1D motion detection chip in steering a car in a simulated racing game. The chip implemented in a 1.5 μm CMOS VLSI process, is comprised of 24 motion pixels that extracts optical flow in parallel. The local optical flow values go to a single-layer perceptron whose output is used to steer the car in the racing game. Because of the continuous-time operation of the motion detection chip, the computationally expensive task of generating a control signal for the car based on the visual scene is largely alleviated.**

I. Introduction

Optical flow or motion is a signal widely used in robot steering. The typical approaches used in robotic systems for extracting optical flow include digital cameras with a supporting processor [1], [2]; optical mouse chips [3]; discrete electronic circuits [4]; and single-chip continuous-time analog motion processors [5]. The most common approach described in the robotics literature is to transmit picture frames from digital camera systems to a digital processor that then performs the computationally expensive task of extracting optical flow from these images. Because of the high computational power needed for optical flow estimation, the computation is often not performed on the robot but on an external personal computer (PC) [4], [6].

We believe that single chip analog motion processors can simplify the estimation of optical flow on this robotics systems with high temporal resolution and consuming very little power. This approach has the advantage that the motion detection is done on a single chip that directly estimates the optical flow on the focal plane. Examples of the use of these motion detection chips in controlling robots can be found in [5], [7], [8]. The algorithms implemented in single chip motion processors can be divided into either (a) intensity-based and (b) token-based algorithms. In intensity-based algorithms, the optical flow is computed from the gradient of pixel intensities over space and time [9], [10] or by correlating neighboring pixel intensities [11], [12]. Token-based algorithms first extract low-level features like contrast edges, and then estimate the optical flow from the correlation of these local features or by measuring the time that a feature takes to travel across two pixels as in [13].

Our motion detection chip is based on a token-based time-to-travel algorithm. Various schemes have been adopted on chip to reduce the dependence of the computed motion on background intensity and contrast. The computed motion is invariant to changes in background intensity over at least three decades due to the inclusion of the adaptive photoreceptor circuits in [14]. The computed motion is also invariant to image contrast, C down to values of 2.5% due to the contrast edge detection circuits. (The image contrast C is defined as $C = \frac{I_{max}-I_{min}}{I_{max}+I_{min}} * 100\%$, where I_{max} and I_{min} are the maximum and minimum intensity values of the scene.)

The circuits run in continuous time thus the temporal resolution of the computed motion does not depend on a clock frequency as would happen in the case of motion estimation using image frames from a digital camera.

We first describe the chip in Section II and the demo setup in Section III. In Section IV we give details on the implementation of the controller and finally we show results in Section V.

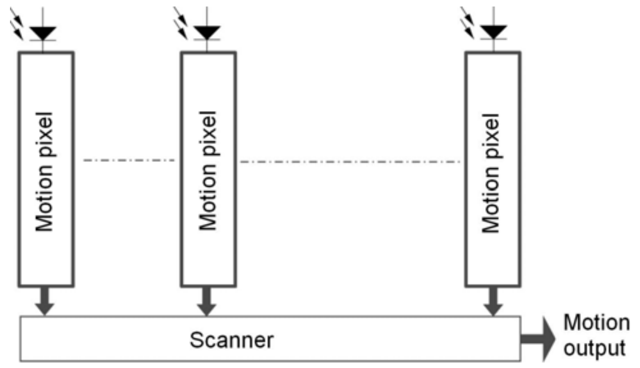

Fig. 1. Architecture of motion detection chip. 24 motion detection pixels are aligned in a linear array and calculate optical flow based on a time-to-travel algorithm. A scanner circuits supports access to the local motion values.

II. Motion Detection Chip

The details of the motion detection chip have been presented in [15]. It was fabricated in a 2-metal 2-poly 1.5 μm CMOS process and consumes 5mW. An overview of the chip architecture is shown in Fig. 1. The chip has a one-dimensional array of 24 motion pixels that implement a time-to-travel algorithm. Each motion pixel contains a photodiode and an analog photoreceptor circuit [14] that convert a temporal change of light intensity into a transient voltage signal. The output signal is further amplified and high-pass filtered by a circuit descibed in [12].

From this amplified high-pass filtered output, we look for a contrast edge. Optical flow is determined by measuring the

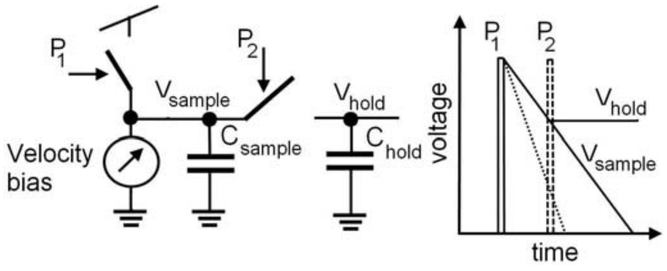

Fig. 2. Left: Sample-and-hold circuit for measuring feature speed. Right: Behavior of voltage V_{sample} that measures and V_{sample} that stores traveling time of contrast edge traveling from a pixel 1 to its neighbor pixel 2. V_{sample} is shown for a given (solid line) value and a higher value of the velocitybias (dotted line) parameter.

time that a contrast edge takes to travel between neighboring pixels. This time delay is measured by a sample and hold circuit shown in Fig. 2: When a motion pixel sees a contrast edge it outputs a pulse P_1 that is used to charge up the capacitor C_{sample}. The charge on the capacitor leaks away once P_1 has disappeared so the V_{sample} voltage decreases linearly thereafter. The neighboring pixel outputs a pulse P_2 once the contrast edge travels across it. This pulse is used to sample V_{sample} which has a high value when the speed of the contrast edge was high and a lower value for a lower speed. This sampled value is stored on a separate capacitor C_{hold}. The discharge rate of C_{sample} can be adjusted by the velocity bias parameter thus allowing us to tradeoff between the range and resolution of speeds that can be measured. A high discharge rate gives us a better resolution for high speeds but a smaller range of detectable speeds (see Fig. 2). A scanner circuit similar to that described in [16] allows us access to the two channels of direction at each of the 24 motion pixels.

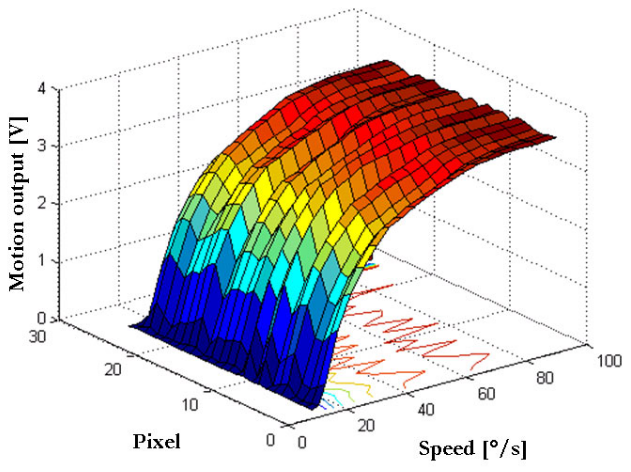

Fig. 3. Speed characterization for the 24 motion pixels for a given value of the velocity bias. The speed of the visual stimuli was varied between 0 and 100 degrees per second. The motion output was measured for one direction of motion of the stimuli.

In Fig. 3, we show a characterization of the output of the 24 motion pixels for visual stimuli moving from right to left. We used sinusoidal gratings of various spatial and temporal frequencies. The measured motion outputs proved to be dependent only on the speed of the grating over 2 orders of magnitude. The speed is computed from V_{hold} in Fig. 2. Fixed pattern noise on the chip leads to the variance in the motion values at different pixels for the same stimulus speed.

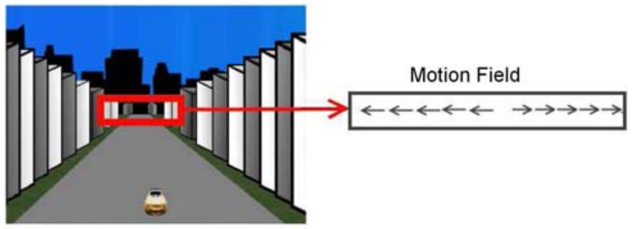

Fig. 4. Screen shot of one frame in the racing car simulation. The visual field of the motion detection chip is indicated by the red box. The box on the right shows the local motion values of the chip that are used to steer the car.

III. Experimental Setup

To demonstrate the capabilies of the motion chip in a real time task we used it to control a simulated car in a racing game. The simulation environment was completely generated in Matlab. The visual stimuli were created with Psychtoolbox [17], [18]. A typical screen shot of the game is shown on the left side of Fig. 4. During the game the car at the bottom of the screen is steered using the local motion outputs of the motion chip to avoid collisions with the walls on both sides of the road. The chip was placed so that it monitors the far end of the street as indicated by the red box. For a straight street, the motion chip outputs a typical 1D optical flow field shown on the right of Fig. 4. If the street curves to the left or right of the screen, the optical flow field will shift and this shift is used to adjust the postion of the car in the next frame.

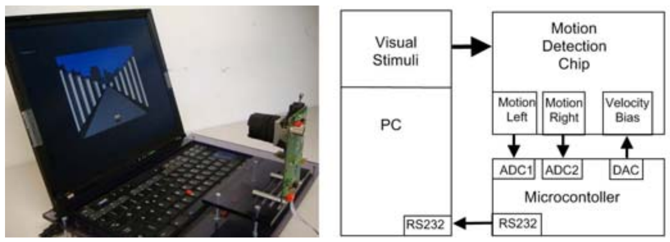

Fig. 5. Demo setup. The motion chip computes optical flow based on the racing game generated by a PC. A microcontroller continuously scans the local motion values from the chip, converts the analog values to digital ones and sends them via a serial interface (RS232) to the PC where a simple controller updates the position of the simulated car based on the motion values. The microcontroller is further responsible for adjusting the velocity bias that sets the current detectable speed range.

A picture and a schema of the whole setup are shown in Fig. 4. The analog local motion values from the motion chip are continuously scanned by a microcontroller in a frame-based fashion and converted into digital values with a resolution of 7 bit. Furthermore the microcontroller adjusts the velocity bias of the motion detection chip. Due to the limitation of the analog-to-digital converter, the minimum readout time per pixel is currently set to $100\mu s$ which corresponds to a sampling rate of 10 KSamples/s. To reduce the amount of data that needs to be transmitted to the PC and because the motion values change slowly over time, we only do the motion readout at 100 frames per second. We send the motion data with a baud rate of 38400 baud/s to the PC that runs the controller for adjusting the position of the car.

IV. Controller

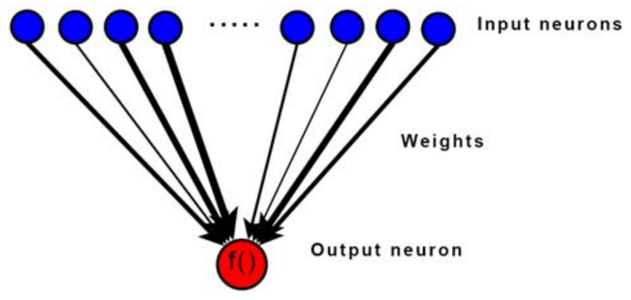

Fig. 6. The perceptron calculates the current position of the street based on the 24 local motion values.

For steering the simulated car based on the local motion values, we decided on a simple controller based on the well-known perceptron shown in Fig. 6. The 24 motion values are inputs to the perceptron that produces an output based on the weighted sum of the inputs following the equation:

$$output = f(\sum_{i=1}^{24} weight_i * input_i) \quad (1)$$

where f is the sigmoid function. The weight vector is updated using the following supervised learning rule

$$\Delta weight = \eta * (supervisor - output) * input \quad (2)$$

where the weight change $\Delta weight$ is determined based on the difference between the $supervisor$ signal (that is the desired output) and the current $output$ of the perceptron. η is a constant that is used to adjust the amount of weight update per learning step.

V. Results

During the training phase, the perceptron was trained with different segments of the racing simulation. A typical evolution of the displacement error during this training phase is shown in Fig. 7. The error value defined in screen pixels is the difference between the distance of the current position of the center of the street and the perceptron output based on the local motion values. The displacement error decreases below 60 pixels after 8000 weight updates.

An example of a final weight distribution at the end of a learning phase is shown in Fig. 8. For this experiment, the motion chip was placed in front of the screen so that the center

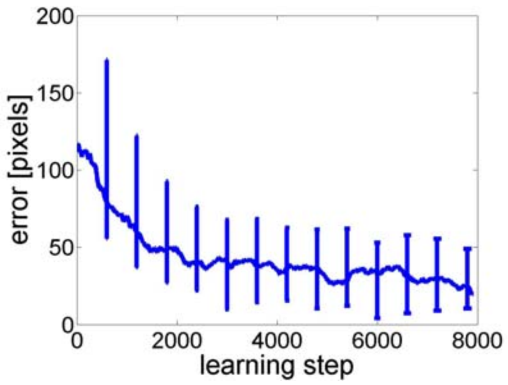

Fig. 7. Average estimation error of the current position of the center of the street measured in pixels. Error bars indicate the standard deviation. After the supervised training phase of the perceptron, the error decreases below 60 pixels.

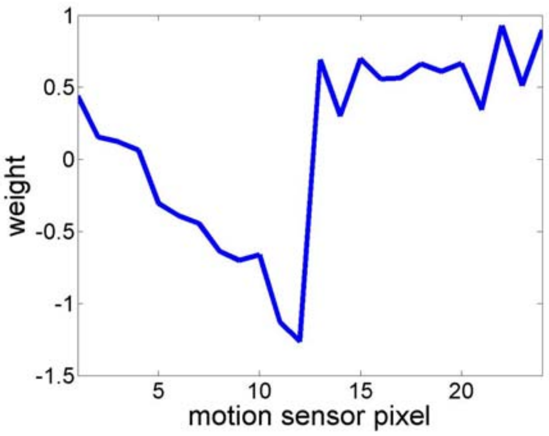

Fig. 8. Final weight values for the inputs of the perceptron after 8000 learning steps. The perceptron learned to steer the simulated car towards the center of the street which was approximately in the field of view between motion pixels 12 and 13.

of the street was approximately in the visual field of pixel numbers 12 and 13. Even though the weight distribution is asymmetric, the net output of the perceptron is close to zero which means that the car will stay in its current horizontal position. The unsymmetric weight distribution is probably due to the fact that the motion chip was not looking at the center of the screen. The perceptron has the potential to correct for displacement errors of the chip with respect to the center of the screen as well as for fixed pattern noise in the motion output and lens aberrations.

The trajectory of the car through the racing game after training is shown in Fig. 9. The car was able to drive close to the center of the street without colliding with the walls. However a more complex controller like a PID controller that allows better estimation of the next street position based on the history would be necessary to reduce the displacement error to zero.

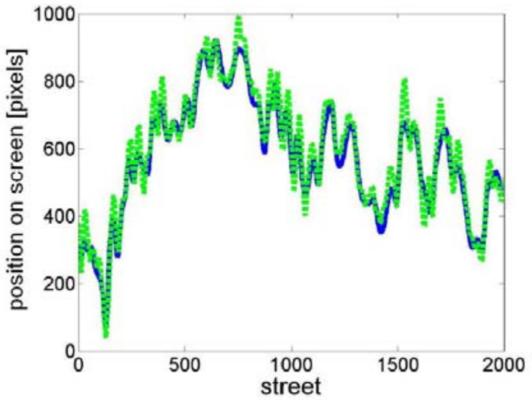

Fig. 9. Horizontal position of the center of the street (solid line) and the location of the car (dashed line) on the screen in units of pixels.

VI. Conclusion

We describe an experimental setup for steering a car in a racing simulation based on the outputs of a novel motion detection chip. The chip computes 24 local motion values in parallel and in continuous time. We fuse the motion values to find the displacement error of the simulated car in relation to the center of the street with the help of a one-layer perceptron that produces a steering output based on the weighted sum of the local motion values. Because of the inclusion of circuits on the motion chip to model the properties of cells in the stages prior to the motion computation in biological systems, and the learning through the perceptron, we get a robust setup that (i) is adaptive to approximately three orders of magnitude of change in background light intensity; (ii) operates over a large range of contrasts down to a contrast of 5%; (iii) supports pixel-level optical flow variations over two orders of magnitude without bias changes; (iv) adapts to displacement errors of the placement of the motion chip in front of the screen, and (v) is robust to variance across the local measured motion values. So we demonstrate how to largely simplify the difficult task of steering a car with the help of the motion chip that performs the computationally expensive processing optical flow estimation at a high temporal resolution. To give the reader a feeling of the capabilities of this chip, we allow them to compete against the chip in the demo setup.

Acknowledgment

The authors would like to thank T. Delbrück for discussions on the circuits. This work is partially supported by SNF Research Grant 200021-105545/1.

References

[1] J.C. Zufferey, A. Klaptocz, A. Beyeler, J.D. Nicoud, and D. Floreano, “A 10-gram microflyer for vision-based indoor navigation,” in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, October 2006.

[2] M. Epstein, S. Waydo, S. Fuller, W. Dickson, A.D. Straw, M.H. Dickinson, and R.M. Murray, “Biologically inspired feedback design for Drosophila flight,” in *American Control Conference*, 2007.

[3] H.J. Dahmen, “Extracting egomotion from optic flow: Limits of accuracy and neural matched filters,” in *Motion Vision*, Springer, 2000.

[4] F. Ruffier, and N. Franceschini, “Optic flow regulation: the key to aircraft automatic guidance,” in *Robotics and Autonomous Systems Journal*, vol. 50, pages 177-194, 2005.

[5] L. Reichel, D. Liechti, K. Presser, and S.C. Liu, “Range estimation on a robot using neuromorphic motion sensors,” in *Robotics and Autonomous Robots*, vol. 51, pages 167-174, 2005.

[6] S. Bermudez i Badia, P.F.M.J. Verschure, “A collision avoidance model based on the Lobula giant movement detector (LGMD) neuron of the locust,” in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 3, pages 1757-1761, July 2004.

[7] R. Etienne-Cummings, V. Gruev, and M. Abdel-Ghani, “VLSI implementation of motion centroid localization for autonomous navigation,” in *Advances in Neural Information Processing Systems 11*, MIT Press, pages 685-691, 1999.

[8] M. Massie, C. Baxter, J.P. Curzan, M.C. McCarley, and R. Etienne-Cummings, “Vision chip for navigating and controlling micro unmanned aerial vehicle,” in *Proceeedings of IEEE International Symposium on Circuits and Systems*, pages 786-789, 2003.

[9] R. Etienne-Cummings, J. Van der Spiegel, and P. Mueller, “Hardware implementation of a visual motion pixel using oriented spatiotemporal neutral filter,” in *IEEE Transaction on Circuits and Systems II*, vol. 46(9), pages 1121-1136, 1999.

[10] A. Stocker, “An improved 2D optical flow sensor for motion segmentation,” in *Proceeedings of IEEE International Symposium on Circuits and Systems*, pages 332-335, 2002.

[11] R.R. Harrison, “A biologically inspired analog IC for visual collision detection,” in *IEEE Transaction on Circuits and Systems I*, vol. 52, no. 11, pages 2308-2318, November 2005.

[12] S-C. Liu, “A neuromorphic aVLSI model of global motion processing in the fly,” in *IEEE Transactions on Circuits and Systems II*, pages 1458-1467, 2000.

[13] J. Kramer, R. Sarpeshkar, and C. Koch, “Pulse-based analog VLSI velocity sensors,” in *IEEE Transaction on Circuits and Systems II*, vol. 44, pages 86-101, 1997.

[14] P. Lichtsteiner and T. Delbrück, “64x64 event-driven logarithmic temporal derivative silicon retina,” in *Proceedings of the 2005 IEEE Workshop on Charge-Coupled Devices and Advanced Imager Sensors*, June 2005, Nagao Prefecture, Japan, 9–11 June.

[15] R. Moeckel and S-C. Liu, “Motion detection circuits for a time-to-travel algorithm,” in *IEEE International Symposium on Circuits and Systems (ISCAS 2007)*, pages 3079-3082, May 2007.

[16] C.A. Mead and T. Delbrück, “Scanners for visualizing activity of analog VLSI circuity,” in *Analog Integrated Circuits and Signal Processing*, vol. 1, pages 93-106, 1991.

[17] D.H. Brainard, “The psychophysics toolbox,” in *Spatial Vision*, vol. 10, pages 433-436, 1997.

[18] D.G. Pelli, “The VideoToolbox software for visual psychophysics: Transforming numbers into movies,” in *Spatial Vision*, vol. 10, pages 437-442, 1997.