

Interfacing Real-Time Spiking I/O with the SpiNNaker neuromimetic architecture

Sergio Davies*, Cameron Patterson, Francesco Galluppi, Alexander D. Rast, David Lester, and Steve B. Furber

APT group, School of Computer Science
The University of Manchester, Manchester, U.K.
{daviess, patternsc, galluppf, rasta, dlester, sfurber}@cs.man.ac.uk
<http://intranet.cs.man.ac.uk/apt/projects/SpiNNaker/>

Abstract. This paper describes a closed-loop robotic system which calculates its position by means of a silicon retina sensor. The system uses an artificial neural network to determine the direction in which to move the robot in order to maintain a line-following trajectory. We introduce a pure “end to end” neural system in substitution of typical algorithms executed by a standard DSP/CPU. Computation is performed solely using spike events; from the silicon neural input sensors, through to the artificial neural network computation and motor output. At the end of the experiment the robotic system using these methods was able to follow a line consistently on a plain background.

Key words: SpiNNaker, Silicon Retina, Actuators, Spiking neural network, Closed-loop system, spike-based computation

1 Introduction

Artificial spiking neural network (ANN) simulation has been widely investigated in the recent past, with many attempts being made to simulate networks in real-time and with increasing biological realism. ANNs have been widely used to interface with sensors, revealing features and details which are then used for specific purposes e.g. [3] [10]. However these designs typically use spiking ANNs as central processing intelligence, while sensors and actuators remain in the analogue domain. Consequently a conversion is required between the analogue values of sensors and actuators to the spiking inputs required by the ANN. An example of such a system is the Kephra robot [3], where after some training the controlling neural network demonstrated object avoidance. In this Kephra study a NIOS 16-bit processor implemented on an FPGA was used both to mediate between real world sensors and actuators, and to implement the neural network cells. In a similar way, the robot DARWIN V [10] was able to present complex behaviours based on the perception of the world through non-spiking sensors which were pre-processed into spikes for consumption by the neural network. This robot evolved to the point of being able to discriminate between objects captured with its vision system sensors. Few attempts have been made where computation is solely performed by means of spikes in closed-loop systems, where the ANN is a “cortical simulator” between sensors and actuators. The system introduced in this paper is designed as an extendable end-to-end spiking test platform, with the simple initial classical aim of having a robot follow a line. In the future, the same platform can be further developed to explore other (more complex) neural algorithms. The approach of using a full spiking system has many advantages when compared with more familiar robotic implementations. These include more frugal power consumption which is a key factor in low-power embedded applications, better tuned temporal response as real-world stimulus drives the ANN rather than the beat of a synchronous clock, and that spiking ANNs act as a natural noise filter due to the behaviour of the neurons.

2 System description

SpiNNaker is a massively parallel biologically inspired computing platform for modelling ANNs in real-time [9]. In its largest configuration a SpiNNaker system scales to ~ 1 billion real-time neurons. Here we describe a closed-loop system developed as a proof-of-concept for interfacing SpiNNaker hardware with discrete devices in the real world. We use a silicon retina [7] mounted on a self-propelled 6-axis 3-wheeled robot (fig.1).

Architecture of the closed-loop system: The system is outlined by the blocks in fig.1. The retina outputs spike events which are translated to SpiNNaker format spike packets, addressed to the correct input neuron in the ANN. The SpiNNaker board performs the neural network computation and outputs motor spike events on its Ethernet interface that the PC converts into commands to control the wheels of the robot. As the motor controller for the robot accepts only input vectors, the Host PC translates SpiNNaker output spike rate-codes into acceptable values for motor control.

* All the authors are with the APT group at The University of Manchester

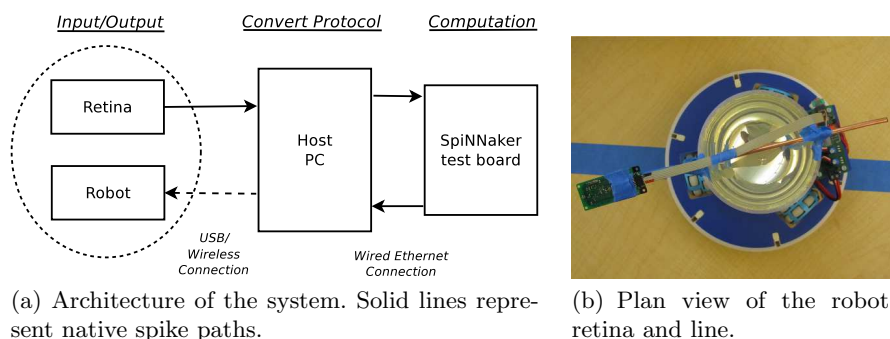


Fig. 1. Schematic and photograph of the robot with mounted retina

The SpiNNaker test board: Four SpiNNaker “test” chips reside on one board [6], each chip (fig.2) having two ARM 968-E processor cores, with tightly coupled instruction and data memories, timer, interrupt and custom DMA controllers and an interface to an external SDRAM chip. One core per chip is used to simulate $\sim 1,700$ LIF [8] or up to 1,000 Izhikevich neurons [5], and the other for system control and communication, giving potentially 4 times this number of neurons per test board.

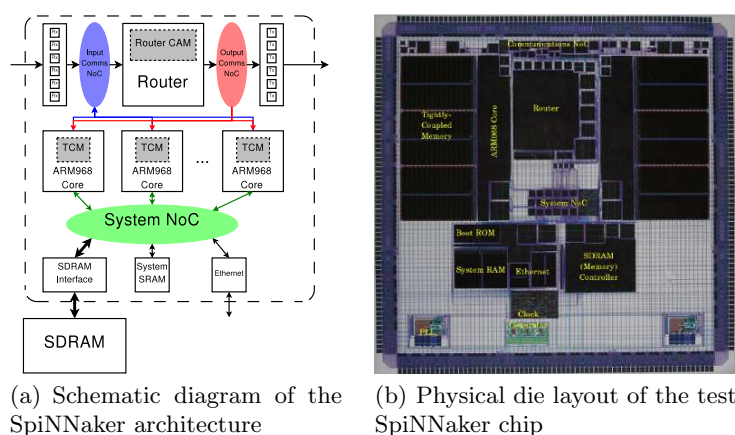


Fig. 2. Schematic and plot of the SpiNNaker two core test chip

The silicon retina: The silicon retina is inspired by the biological processes that take place in the eye [7]. Light intensity changes are captured by each pixel on the sensor and generate discrete spiking events, the sensitivity of which can be adjusted by the user. The sensor is able to generate sub-millisecond responses to changes in individual pixels, and is formed of a matrix of 128×128 elements, totalling 16,384 spike sources. Each spike event generated by an individual pixel is represented by an AER (Address Event Representation) packet including the co-ordinate of the pixel emitting the event and the polarity of the event.

Robot actuators: In this environment the actuators are represented by the wheel motors of a robot (fig.1). The robot has three wheels (placed with radial symmetry) each attached to an independently controlled motor. The commands used in this environment set the robot speed in three directions: forward – backward (y axis), left – right (x axis) and rotate (clockwise – anticlockwise).

3 Model implementation

Sub-sampling input spikes: The silicon retina’s USB interface limited the spike rate sent to the host computer to 120,000 events per second. Further, the DVS128 retina has 16,384 elements, too many for a 1:1 SpiNNaker input map on the test board when using LIF (tests show the SpiNNaker test board with 4 chips can model up to 6,896 LIF neurons [8]). A relevant method of reducing the data flow must therefore be employed, without losing information relevant to the line-following task. We identified and studied the following reduction methods:

- Sub-sampling over space (e.g.: blocking/pixelating). This reduces the number of unique input neurons required, so that all the events received in a region drive a specific neuron. We considered uniform (e.g. 16×16

- fig.3a), and non-uniform methods (fig.3b - with better central image resolution as per a biological eye). This method drops no spikes, but adds categorisation.
- Sub-sampling over space and time (e.g.: only using pixel coordinates which are exact multiples of 4). This drops spiking information that might have been useful in coherent detection of line edges over any noise.
- Windowing (e.g. a thin slot over the full image width - fig.3c). Instead of using the full 'image field' this reduces the number of input spikes by using a high resolution subset conformed for directional decisions.

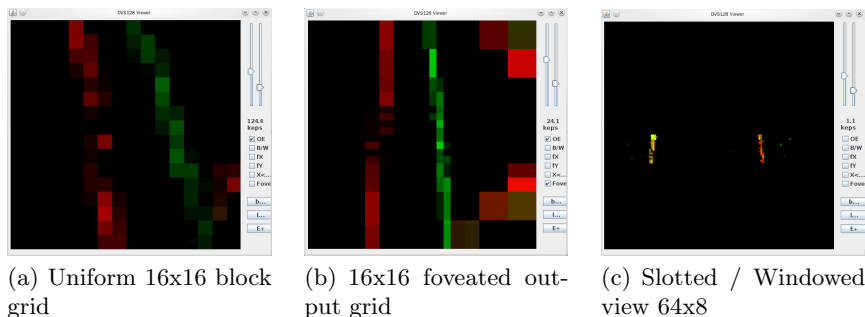


Fig. 3. Sub-sampling the Input Spikes

In our SpiNNaker line-following system we chose space sub-sampling, block pixellating the image down to 8x8 pixels to give a resolution of 16x16 blocks across the full frame (fig.3a). This categorisation into 256 sub-samples is repeated for both spike event polarities, totalling 512 input neurons.

Neural Network Architecture: The neural network couples the sensors with actuators; we used a Leaky Integrate-and-Fire (LIF) [2] feed-forward network organized into 3 layers. The input layer is composed of 2 16x16 matrices combining the input for both polarities, and is connected to a 4x4 sub-sampler layer where units are in competition through lateral inhibition. Each neuron in the sub-sampler maps a 4x4 receptive field in both input matrices. The more frequently cells in the sub-sampling neuron's receptive field fire, the more likely that the sub-sampling neuron fires, inhibiting other sub-sampling neurons - a Winner Take All (WTA) mechanism. Finally, the output layer stimulates the actuators (fig.4), and is designed to maintain the most salient stimuli in the central portion of the visual field. Outputs are also in competition, in order to achieve better stability.

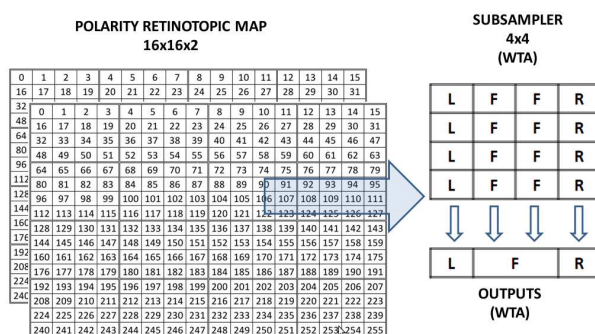


Fig. 4. The Neural Network used. 512 input neurons from the retina feed into a hidden sub-sampler layer, and then to a competitive output layer to give the robot direction.

The neural network was generated with a PyNN script [1], and uploaded to the SpiNNaker chip through an automatic software process [4], taking advantage of PyNN's ability to represent 2D network information.

Spike rate to motor speed conversion: The robot interfaces to the SpiNNaker board through a PC which converts the spiking rate of the output neurons to the speed of the robot in the corresponding direction. To interface the neural network with the robot we used three output neurons: forward, left and right. The conversion uses an internal state variable which holds the current speed for each direction and its last received spike timestamp. The internal state variable decays every millisecond, and when it differs from the actual robot speed beyond a threshold, a new command is sent to the robot to alter its speed. The transfer function is described by the graphs in fig.5. The closer two spikes are, the greater the increase of speed in the corresponding direction.

Both parts of the algorithm (increasing and decreasing) rely on parameterised linear rate-coding functions to permit fine tuning of the behaviour.

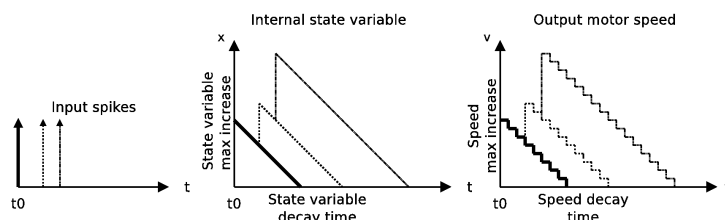


Fig. 5. Impulse response function to spikes generated by a motor neuron. The state variable decays linearly with the time, while the output speed sent to the robot is quantized. The closer together the input spikes, the steeper the increase in robot speed on that axis.

4 Results

In an ideal situation, the robot would sit on the central axis of the line moving forward, however, the silicon retina output relies on changes in intensity of each pixel. In such a case the line would become invisible as no changes in intensity would be detected. To avoid this problem we added an element of jitter in the robot movements, as well as exploiting inherent mechanical vibration, in order to maintain visibility of the edges of the line. The robot has the silicon retina mounted on top towards the front (see fig.1), with some perspective created by slightly angling back the mounting.

Neural Network Activity: Fig.6 analyses the results extracted from 5 seconds of a recorded simulated

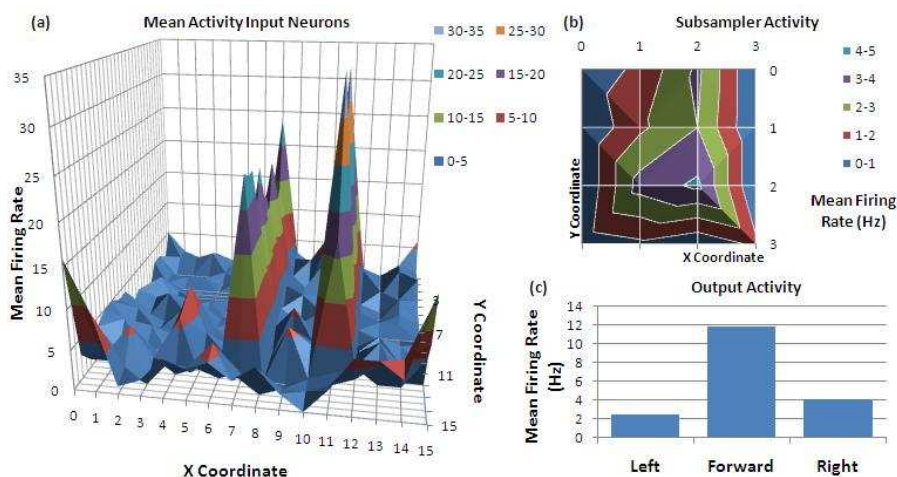


Fig. 6. 5 seconds simulated network activity (a) input neurons' activity: during the trial the most active neurons are the ones corresponding to the edges of the line (b) subsampler activity: the most active neurons are the ones in the center of the visual field (c) output neurons' activity

run, and we used these results to tune network parameters. At the beginning of the trial the robot is placed on top of the line (slightly to the right side). Fig.6a shows the mean activity rate of the input neurons, fed by the silicon retina: it can be seen that the most active neurons are those along the line edges (column 7 and 11) as the robot moves forward. The rest of the input layer is subject to random noise. Fig.6b shows the mean activity in the sub-sampler layer: as the WTA mechanism takes place, we can see that the most active neurons are the ones where the input is more salient (2,2 - 1,2 - 2,1). According to the activity in the subsampler layer and the map described in fig.4, the output forward neuron is the most active. The right neuron is more active than the left because the robot is positioned slightly on the right, and the network is attempting to compensate for this drift in order to keep the line in the central portion of the visual field. The successful line-following outcome at the end of this research is also recorded in an Internet published video.¹

¹ http://www.youtube.com/watch?v=ZQ7FdQ_VJNg

5 Discussion

The approach taken by the system presented in this paper represents a solution complementary to classical approaches to robot control offering alternative benefits rather than attempting to compete feature-for-feature. The closed-loop system presented is formed by interfacing the spiking neural network with sensors and actuators: the whole system runs in real-time and interacts with the outside world exclusively by means of spikes. The simulated ANN receives real-time input from a peripheral sensory input processor, and it outputs spikes to control the movements of the robot, c.f. the motor neurons in the cortex.

Using spiking interfaces between the ANN and the I/O devices places this design in contrast to previous line-following implementations. Two simple passive conversions complete the interface. All spikes from the retina are delivered as spikes to the SpiNNaker board via a lookup table between silicon retina output spikes (from USB) to SpiNNaker board input spikes. A spike-rate to linear vector conversion between SpiNNaker motor neurons and robot wheels handles the output. Such a system demonstrates the feasibility of integrating “cortical” chips such as SpiNNaker with spiking sensors and actuators in time-critical real-world simulations.

6 Conclusions

This paper has covered an alternative approach to the classic robotic line following problem using a fully spike-based technique that delivers potential improvements to power utilisation, input filtering and temporal efficiency. We have described an autonomous closed-loop system using a dedicated programmable neural network simulator chip to emulate the cortical areas of the brain. We used our extendable framework to demonstrate the ability of the SpiNNaker system to interface with the real world through spiking sensors and actuators using only neural algorithms to execute the assigned tasks. In this implementation we used a fixed network topology and fixed synaptic weights, thus the behaviour of the neural network is statically defined. In future work, learning algorithms and synaptic plasticity will enable the robot to understand the pattern of inputs, to follow and search autonomously. In addition a larger scale SpiNNaker chip is currently in preparation which has 18 cores per chip, with enhanced on-board resources. This chip will be used to scale the platform to much larger systems, including the machine expected to scale beyond 1 million processors. This research is a first step towards a long-desired research goal: autonomous robotic systems using bio-inspired spike-based signalling to permit real-time action and adaptation in highly dynamic environments.

Acknowledgements We would like to thank the Engineering and Physical Sciences Research Council (EPSRC), Silistix, and ARM for supporting SpiNNaker research. We would also like to thank Jörg Conradt, Tobi Delbruck and all at the Telluride Neuromorphic Cognition Engineering Workshop 2010.

References

1. Andrew P Davison, Daniel Brderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: A common interface for neuronal network simulators. *Front Neuroinformatics*, 2:11, 2008.
2. Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 1st edition, Dec. 2001.
3. D.Roggen, S.Hofmann, Y. Thoma, and D. Floreano. Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot. In *Proc. 2003 NASA/DoD Conf. Evolvable Hardware*, pages 189–198. IEEE Press, 2003.
4. Francesco Galluppi, Alexander Rast, Sergio Davies, and Steve Furber. A general-purpose model translation system for a universal neural chip. Submitted to ICONIP 2010 - 17th International Conference on Neural Information Processing.
5. X. Jin, S. B. Furber, and J. V. Woods. Efficient modelling of spiking neural networks on a scalable chip multiprocessor. *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2812–2819, September 2008.
6. X. Jin, F. Galluppi, C. Patterson, A. Rast, S. Davies, S. Temple, and S. Furber. Algorithm and software for simulation of spiking neural networks on the multi-chip SpiNNaker system. *Neural Networks, 2010. IJCNN 2010. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2010.
7. Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid State Circuits*, 43:566–576, 2008.
8. A. D. Rast, F. Galluppi, X. Jin, and S. Furber. The Leaky Integrate-and-Fire Neuron: A platform for synaptic model exploration on the SpiNNaker chip. *Neural Networks, 2010. IJCNN 2010. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2010.
9. Alexander D. Rast, Xin Jin, Francesco Galluppi, Luis A. Plana, Cameron Patterson, and Steve Furber. Scalable event-driven native parallel processing: the SpiNNaker neuromimetic system. In *CF '10: Proceedings of the 7th ACM international conference on Computing frontiers*, pages 21–30, New York, NY, USA, 2010. ACM.
10. O. Sporns, N. Almásson, and G. M. Edelman. Plasticity in value systems and its role in adaptive behavior. *Adaptive Behavior*, 8(2):129–148, February 2000.