2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Event-based neural computing on an autonomous mobile platform

Francesco Galluppi[1], Christian Denk[2], Matthias C. Meiner[2], Terrence C. Stewart[3], Luis A. Plana[1], Chris Eliasmith[3], Steve Furber[1] and Jörg Conradt[2]

***Abstract*— Living organisms are capable of autonomously adapting to dynamically changing environments by receiving inputs from highly specialized sensory organs and elaborating them on the same parallel, power-efficient neural substrate. In this paper we present a prototype for a comprehensive integrated platform that allows replicating principles of neural information processing in real-time. Our system consists of (a) an autonomous mobile robotic platform, (b) on-board actuators and multiple (neuromorphic) sensors, and (c) the SpiNNaker computing system, a configurable neural architecture for exploration of parallel, brain-inspired models. The simulation of neurally inspired perception and reasoning algorithms is performed in real-time by distributed, low-power, low-latency event-driven computing nodes, which can be flexibly configured using C or specialized neural languages such as PyNN and Nengo. We conclude by demonstrating the platform in two experimental scenarios, exhibiting real-world closed loop behavior consisting of environmental perception, reasoning and execution of adequate motor actions.**

I. INTRODUCTION

The nervous system shows a rich repertoire of responses to dynamic contexts using many individual, yet interacting, computing units, which together exhibit extraordinarily coordinated adaptive behaviors. Neuro-physiological models of increasing complexity have been proposed to explain perception and cognition in behaving systems; most of such models, however, are typically simulated only in restrained or virtual environments. Emerging cognitive architectures, capable of solving some classical cognitive tasks by flexibly coordinating perceptual, cognitive and motor areas, hint to the ability of organisms to solve a variety of tasks using the same universal, parallel neural substrate in different contexts [7]. Simulating biological systems is a resource-intense task, and power consumption is a relevant issue when considering autonomous agents embodying such models. Inspired by the parallelism and power frugality characteristics of the nervous system, a new class of *neuromorphic* devices and sensors have been developed in the last years [11].

In this paper we present an integrated robotic platform (shown in Figure 1) to explore the behavior of complex, biologically-inspired models; the platform is equipped with neuromorphic sensors and with SpiNNaker [8], a digital, configurable event-driven system that can interpret incoming events and translate them into neural spikes, using them to control the robotic platform. SpiNNaker offers a combination of off-the-shelf, low-power RISC processors, and a custom packet-switched network infrastructure, aimed at the simulation of large neural network models in real time. The platform is configurable at various levels of abstraction, from C programming to the use of specialized neural description languages, making it user-friendly. The result is a system for implementing and testing event-based neuronal models on mobile robots in an accessible way. The use of cognitive architectures can enhance the capabilities of autonomous agents to rapidly adapt to dynamic environments and situations, while the low-power consumption of neuromorphic sensors and devices can increase their autonomy.

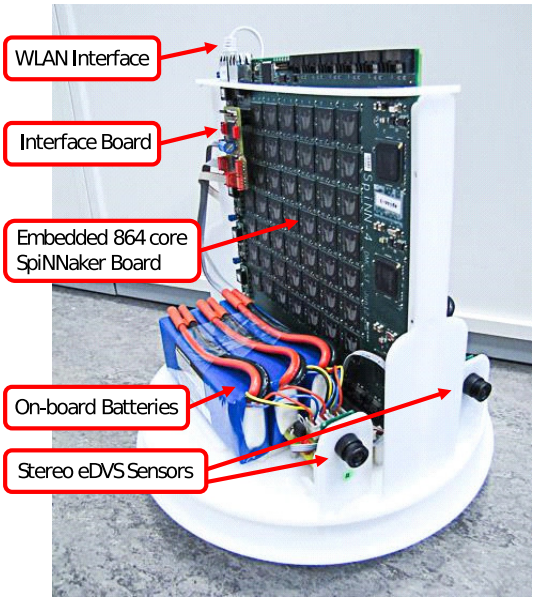

Fig. 1. The SpiNNaker robotic platform, comprising a mobile robot, two silicon retinas and a 48 node SpiNNaker board to perform neural control.

The rest of the paper is structured as follows: the next Section introduces related work exploring the brain inspiration. Section III presents SpiNNaker, the robotic platform and the sensors. Section IV shows two experiments performed with the platform, demonstrating its flexibility; finally Section V concludes with a brief discussion summarizing the results and the novel characteristics of the platform.

II. RELATED WORK

Simulating large portions of neural tissue is a resource-intensive task, mainly due to the cost of message (spike)

[1]Advanced Processors Technologies Group, School of Computer Science, University of Manchester, Oxford Road, M13 9PL, United Kingdom steve.furber@manchester.ac.uk
[2]Neuroscientific System Theory, Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. conradt@tum.de
[3]Centre for Theoretical Neuroscience, University of Waterloo, Canada celiasmith@uwaterloo.ca

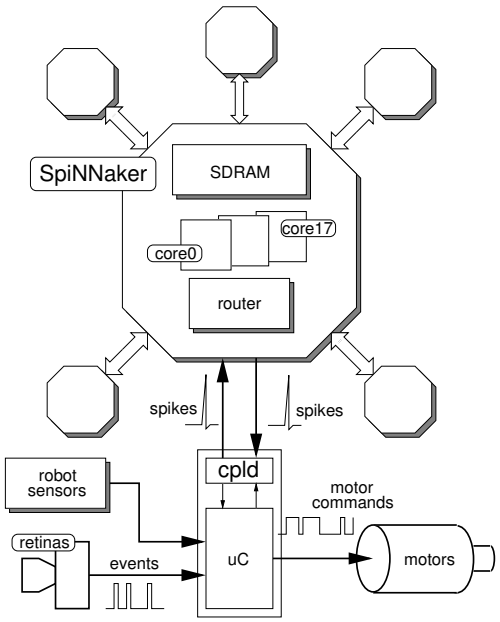

Fig. 2. System overview: a SpiNNaker chip (octagon) is connected to the robot and sensors using an interface board, responsible of translating events into spikes and motor commands. The interface board is connected on one of the 6 asynchronous links connecting SpiNNaker chips together; each chip sees the robot and sensors as other SpiNNaker chips.

delivery, particularly on supercomputers using MPI [17]. Recently, in a report called "10^{14}", IBM showed the results of a structural simulation of the brain comprising 53×10^{10} neurons and 1.37×10^{14} synapses, running $1542\times$ slower than real time on their Sequoia BlueGene system [27] (1,572,864 cores, 17,173 TFlops/s and a power consumption of 7,8 MW). Interestingly, they report that the time spent communicating spikes via MPI messaging is the biggest cost. As simulation of large parallel systems on standard (or clusters of) computers is such a power- and communication-intense task, the cognitive architectures running on them cannot easily be embodied in a real-time robotic platform.

In order to circumvent performance and power limitations, biologically inspired - or *neuromorphic* [13] - parallel VLSI circuits have been produced. Such platforms are based on the organization principles of the nervous system: they take inspiration from the inherent parallelism and the communication mechanism of the brain to execute computation more efficiently in terms of power and performances. They do so by instantiating arrays of simple, neuron-like units, communicating asynchronously in the network through means of all-or-non signals called *spikes*, as their biological counterparts. Silicon retinas, for instance, only produce events if a local change of light is detected, hence having a very fast response (low latency) and reducing visual information redundancy [12][21].

Brain-style parallel processing architectures (as Stanford's Neurogrid [15], or the EU funded BrainScaleS hardware [20]), and neuromorphic sensory systems such as artificial retinas [14] and cochleas [26], have been successfully developed and used to run event-driven simulations. These sensors and systems use an Address-Event Representation (AER) scheme, where each unit produces a stereotypical message identifying its source coordinates, mimicking the biological *spike* mechanism. Messages are delivered to neurons through *synapses* (the point of contact of two neurons), which are usually located close to the destination (*post-synaptic*) neuron.

The use of such systems and methods has generated a research environment where large-scale, truly parallel platforms where real-world, real-time sensory processing, learning, and generation of complex motor output can be explored. Silicon neurons are low power, efficient systems for neural-inspired computation, but their availability is scarce and they are often tailored for a particular neural model [20] or network topology [15], as specific circuit solutions depend on requirements of an application [11]. Projects such as CAVIAR [21] have demonstrated the advantages of the integration of neuromorphic sensory perception compared to standard, frame-based mechanisms.

FPGAs and GPUs are more readily available, configurable hardware platforms which have been used for neural simulations [1][16]. Neural simulation is however often bounded by communication, while GPUs and FPGAs are typically better suited for tasks where the ratio communication/computation favours the latter; additionally their power requirements hamper the possibility to use them in autonomous agents.

Current systems using dedicated hardware are specialized to particular tasks or models, and are accessible only through a steep learning curve, making them scarce resources in the scientific community. This leads to a gap between neural modellers and neuromorphic, parallel platforms. In an effort to narrow this gap we present an integrated neuro-style computing system directly interfaced to multiple neuromorphic sensors mounted on-board of a mobile robot (Figure 1), accessible at different abstraction levels.

III. THE ROBOTIC PLATFORM

This section describes the SpiNNaker system (A), the developed autonomous mobile robot platform (B) the on-board available event-based vision sensors (C) and the interface between robot, sensors and the SpiNNaker computing system (D). All required devices operate on board of the mobile robot for several hours with the provided battery pack and a WLAN connection for system boot-up and monitoring.

A. SpiNNaker

1) Hardware: SpiNNaker [8] is a digital multi-core multi-chip architecture oriented to the simulation of large neural networks in real-time. Each SpiNNaker chip (octagons in Figure 2) is equipped with 1Gbit SDRAM, storing synaptic information and accessible by parallel DMA requests (for an aggregate bandwidth of 900 MBytes/s [18]), by 18 programmable ARM968 cores embedded in a configurable packet-switched asynchronous fabric, based on an on-chip Multicast (MC) Router capable of handling one-to-many communication of spikes (packets) very efficiently, and linked to 6 neighbour chips through asynchronous links [19].

In this work we use a 48-chip SpiNNaker board, for a total of 864 ARM968 Cores and 48 Gbit of memory, where every chip has a top power consumption of 1W. A system, like the one presented in Figure 1, is capable of simulating up to a quarter million neurons and more than 80 million synapses in real time, within a power budget of less than 40W [25]. Typically the most expensive process in neural network simulations is the one concerning propagation and integration of spikes [17][27]; with the current software infrastructure, the system used in this paper is capable of delivering 1.8 billion synaptic events per second, using a few nJ per event and per neuron.

2) Software: Specialized neural languages provide a development environment where neurons and synapses are programming entities: through the use of standard neural models, their mathematical formulation is abstracted to the final user, and this guarantees repeatability across different simulators, promoting model sharing in the community. Using a specialized neural language to configure the SpiNNaker platform is a natural choice. PyNN [4] is a platform-independent description language, capable of defining the structure of a neural network by instantiating *populations* of different models of neurons and connecting them through *projections*. Alternatively, neural computation can be performed in a more *functional* way, by using the Neural Engineering Framework (NEF) [6], a method to encode functions and dynamical systems in networks of spiking neurons. Using the NEF it is possible to build complex cognitive architectures such as SPAUN [7], currently the largest functional model of the brain. Known neurophysiological data acts as constraints on neural parameters, making comparisons with human neural and behavioral data possible. Regardless of the choice of the language, the model is mapped and distributed to the machine using the Partition and Configuration Manager (PACMAN) [10], which hides the complexity of configuring a massively-parallel machine to the final user, by exposing the system through interfaces with PyNN and Nengo [9][24], the software that implements the NEF principles, automatically translating the system in biologically plausible neural circuitry, and performing all computation in neurons, as presented in Section IV-B.

The code running on the platform is written in a C-based API [22], offering an accessible interface to the hardware substrate and to real-time event scheduling facilities. SpiNNaker applications are loaded into the ARM cores and executed in parallel; they are used to configure *callbacks* that respond to *events*, such as the arrival of a spike, or a timer tick initiating the neural update process. Neural kernels are fully parametrizable and can support different classes of neural models and connectivity patterns.

B. Mobile Robot

The mobile robot used in this project is a custom developed omni-directional mobile platform of 26cm diameter, with embedded low-level motor control and elementary sensory systems. An on-board ARM7 micro-controller receives desired motion commands in x and y directions and rotation through a UART communication interface, and continuously adapts three motor control signals (PID-control) to achieve desired velocities. The robot's integrated sensors include wheel encoders for position estimates, a 9 degrees of freedom inertial measurement unit (3 accelerometers, 3 gyroscopes, and 3 compasses) and a bump-sensor ring which triggers binary contact switches, located at 60° intervals on the robot circumference, upon contact with objects in the environment.

C. Silicon Retinas

The embedded DVS system (eDVS) [2] used as spiking sensory input in based on an address-event silicon retina that responds to temporal contrast [12], connected to an ARM7 micro-controller that initializes the DVS and captures events. All 128×128 pixels operate asynchronously and illumination changes are signalled within a few microseconds after occurrence (without having to wait for a frame to send information) through a spike, representing a quantized change of log intensity at a particular location. In this project the micro-controller streams all obtained events over a UART port through the SpiNNaker interface board.

D. Robot-SpiNNaker Interface Board

Computing nodes in SpiNNaker exchange data on a proprietary energy and speed efficient asynchronous interface [19]. We designed a generic CPLD/micro-controller interface board [5] (Figure 1, top left) to receive and to transmit SpiNNaker data packets, while communicating with multiple external devices. The on-board micro-controller translates the protocols between SpiNNaker and peripherals, integrated into an existing SpiNNaker system as further computing node (refer to Figure 2).

IV. EXPERIMENTS

This section describes two independent experiments. First we present a trajectory stabilization task, using perceived optic flow and a distributed neural information processing model written in C. In the second task, written in Nengo, the robot maintains a constant distance and orientation to the most salient target through stereopsis.

A. Trajectory stabilization using optic flow

In this first experiment the autonomous mobile robot traverses a man-made corridor composed of vertically-striped misaligned card-board boxes (shown in Figure 3a; overhead sketch of spatial layout in Figure 3b). The robot stays centered exclusively based on real-time locally computed optic flow (OF), with raw visual information solely provided by two laterally mounted event-based vision sensors. The complete setup resembles similar flight experiments performed with bees [23].

While forward-traversing the corridor marked with black and white stripes on both walls (Figure 3a), two laterally mounted eDVS report perceived illumination changes as spike trains: the upper left panel in Figure 3c (corresponding to the robots left-facing eDVS) clearly shows a lower spatial frequency compared to the lower left panel (the robots right-facing eDVS), because of the robots position close to the

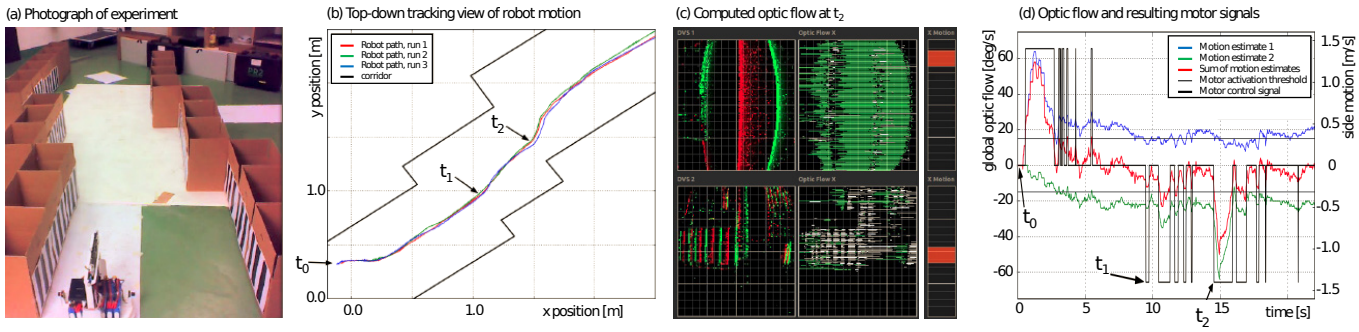

Fig. 3. Trajectory stabilization using optical flow: (a) photograph of experiment arena, front end the mobile robot; (b) top-down tracking view of arena and robot path in three consecutive experiments; (c) left: display of observed events from eDVS, middle: derived horizontal local optic flow (green and white indicate different polarities), right: combined flow estimate equal to lateral motion motor command; (d) time series of one experiment showing global optic flow (blue, green and combined in red) and resulting motor commands.

left wall (Figure 3a). While driving, all spike trains are processed retinotopically in multiple distributed computing nodes on SpiNNaker. Each node estimates local horizontal OF, within a limited region of the total field-of-view. The resulting horizontal flow vectors are shown in Figure 3c, center, for the left and right-facing eDVSs respectively, with green indicating negative flow and white indicating positive (opposite) horizontal flow.

Local OF is computed by estimating time differences between nearby local events, which is inversely-proportional to underlying motion velocity, and hence to local OF. All local OF computations contribute to a common global OF estimate, as we assume a stationary world in which only the robot moves. In a generic mobile robot task such a transformation might be an under-constrained operation; here, however, we assume that neither vertical OF nor rotational OF exist, because of the construction and motion of the wheeled robot. We compute an individual horizontal global OF estimate for each retina by summing all local OF into a leaky integrator. The position/height of the red bar in Figure 3c, right box, represents the current overall OF for each of the two eDVS.

The robotic control loop is closed by computing and executing desired motor commands based on those horizontal OF estimates to keep both OF estimates balanced and thus the robot centered. Assuming identical spatial frequencies of the patterns on both walls of the corridor, we also expect to perceive equal amounts of OF from the left and the right eDVS. We add the two sign-corrected global flow estimates (Figure 3d, blue and green trace) into a combined flow-mismatch signal (red trace), which upon passing a threshold triggers a sideways motor response (black trace) to correct the robots motion.

The high frequency oscillations in all estimates can be traced back to the heterogeneous structure of the observed environment, and could be compensated for by a light low-pass filtering. In multiple independent experiments the robot stays centered within the hallway, even in scenarios with turns and detours, as confirmed by overhead tracking (Figure 3b).

B. Stimulus tracking in Nengo

While it is possible to write custom applications, or define a particular network structure to perform a task, using the NEF it is possible to rapidly build networks representing functions in populations of neurons and their interconnection parameters. The interface with Nengo exposes the robot and sensors directly in the neural language.

The network embodied in the agent is presented in Figure 4, and the task here is to keep the most salient stimulus at a fixed distance and orientation using stereopsis [3]. There are five groups of neurons, each representing different values needed in this algorithm, with the NEF giving a biologically plausible method for representing scalar values and functions in a distributed manner across populations of spiking neurons. The retina input is a combination of information from the $128 \times 128 \times 2$ silicon neurons representing the retinas of the robot. The input population estimates the angle of the most salient object by averaging the number of events in its field of view; an LED, blinking at 100 Hz and thus producing a large number of events, is used as the stimulus to track. This input is connected to the 200 neurons in the *angles* population to form a distributed representation of the relative angles α and β between the retinas and the stimulation, trying to keep $\alpha = \beta$ and the stimulation in the center of the field of view, as illustrated in Figure 4.

If the stimulus is converging into the field of view then the angle value is negative; conversely, if the stimulus is diverging from the field of view (or on the same direction of the sensor) the angle is positive. The concept is illustrated in Figure 5: if the stimulus is present within the central part of the field of view (as in 1 and 2 in Fig. 4), both angles are negative and no rotational movement is required. If the angles are different (as in case 3 and 4) then the *angles* population drives the *turn* population, encoding rotational movements, trying to keep $\alpha = \beta$ and the stimulation in the center of the field of view.

The algorithm was provided to the Nengo software, which automatically performs the neural optimization and loads it onto SpiNNaker. The NEF is also used to compute the connection weights from the 200 *angles* neurons to the 150

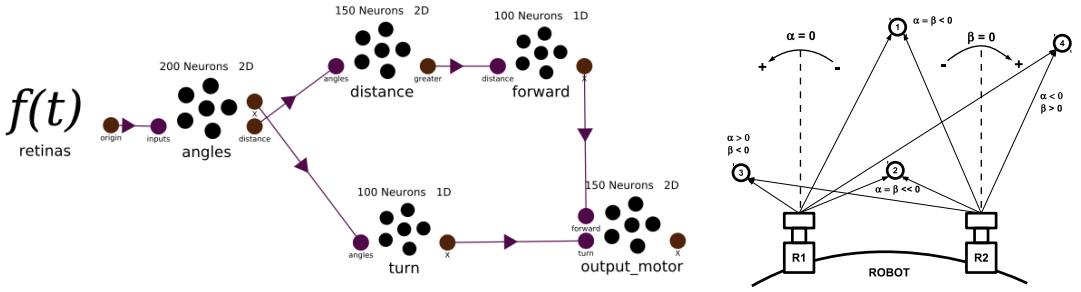

Fig. 4. (Left) Network structure: The *angles* population receives input from the silicon retinas, and uses it to compute the motor command to turn the robot (based on the angle difference) and to move forwards and backwards using the *distance* population, which estimates distance through stereopsis to keep the stimulus at a constant distance and orientation. (Right) The *angles* population represents the relative angles α and β between the R1 and R2 retinas and the stimulation. (1-2) if the stimulus is converging into the field of view then the angle value is negative; conversely, if the stimulus is diverging from the field of view the angle is positive (3-4) when the stimulus is located laterally, $\alpha \neq \beta$, and the robot turns, trying to keep $\alpha = \beta$.

distance neurons. In this case, the weights are optimized to take these α and β values and to estimate the distance to the object via triangulation. The *distance* neurons are connected to the forward population with weights encoding the function "*if d > .5: 1; else: -1*". This value is sent to *output_neurons* to control the movement of the robot keeping the distance constant. The system for turning the robot to face the stimulus involves turning left if the mean of α and β is negative, and otherwise turning right.

All the neurons and connections are simulated directly on SpiNNaker. The model is effectively a neural computational module, where values (the spiking retina input) are fed into the system, manipulated by a sequence of spiking neurons and their connections, and finally decoded and sent to the motors (or recorded for display purposes as in Fig. 5). The operation of the system is presented in Figure 5, and in the video included in the Proceedings. In total the model uses 10 out of 16 ARM cores in a single SpiNNaker chip (out of 48 chips present on the board, for a total of 764 neural cores) for modelling 1,450 neurons firing at 100-150 Hz.

V. CONCLUSIONS

This paper has presented a neuromorphic robotic platform aimed at the exploration of parallel, event-driven neural models behaving in real-world scenarios. As the use of cognitive architectures can improve the adaptability of robotic agents in dynamic contexts, the neuromorphic approach offers specialized systems for perception and computation, that improves their autonomy within a low-power budget.

Silicon retinas are used as a sensory front-end, asynchronously delivering visual events to SpiNNaker, a configurable parallel platform that controls the behavior of the robot. The use of C and alternatively of specialized neural languages makes the platform accessible at different levels, hiding the difficulties of configuring a parallel platform from final users. The optical flow experiment shows how C-based kernels can be used to solve problems in real-time in a parallel fashion, while using the Neural Engineering Framework it is possible to encode and process information in spiking activity, automatically translating a functional description of the system into neural circuitry, as shown in Section IV-B. The experiments presented run in real time on the SpiNNaker robot. Interaction with the environment is possible through a custom interface which connects SpiNNaker, AER-based-sensors and the robotic platform. While integration with AER sensors makes use of the event-driven nature of the platform, interfacing the system with robots performing in a real environment makes use of its real-time capabilities. The integration of such systems provides non-experts with a standard interface to bring their models to the platform, ready for exploration of networked computation principles and applications.

The overall system is a stand-alone, autonomous configurable platform with no PC in the loop, provided with a set of tools and interfaces that make it a usable exploratory platform for embodied brain-inspired models.

ACKNOWLEDGMENTS

The system used in the second experiment was developed during the Capo Caccia Cognitive Neuromorphic Workshop 2013[1]. We would like to thank the workshop sponsors and organizers. SpiNNaker is partially supported by the Engineering and Physical Sciences Research Council of the U.K. through Grant EP/G015740/1.

REFERENCES

[1] R Brette and DFM Goodman. Simulating spiking neural networks on GPU. *Network: Computation in Neural Systems*, 23(4):167–182, 2012.

[2] J Conradt, R Berner, M Cook, and T Delbruck. An embedded AER dynamic vision sensor for low-latency pole balancing. pages 1–6. IEEE, 2009.

[3] J Conradt, P Simon, M Pescatore, and PFMJ Verschure. In *Int. Conference on Artificial Neural Networks (ICANN2002), Madrid, Spain (2002)*.

[4] AP Davison, D Brüderle, JM Eppler, J Kremkow, E Muller, D Pecevski, L Perrinet, and P Yger. PyNN: A Common Interface for Neuronal Network Simulators. *Frontiers in Neuroinformatics*, 2(0):11, 2008.

[5] C Denk, F Llobet-Blandino, F Galluppi, LA Plana, SB Furber, and J Conradt. Real-Time Interface Board for Closed-Loop Robotic Tasks on the SpiNNaker Neural Computing System. In *Artificial Neural Networks and Machine Learning ICANN 2013*, volume 8131, pages 467–474, 2013.

[6] C Eliasmith and CH Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press, Cambridge, MA, 2003.

[1] capocaccia.ethz.ch/capo/wiki/2013/spinnaker13

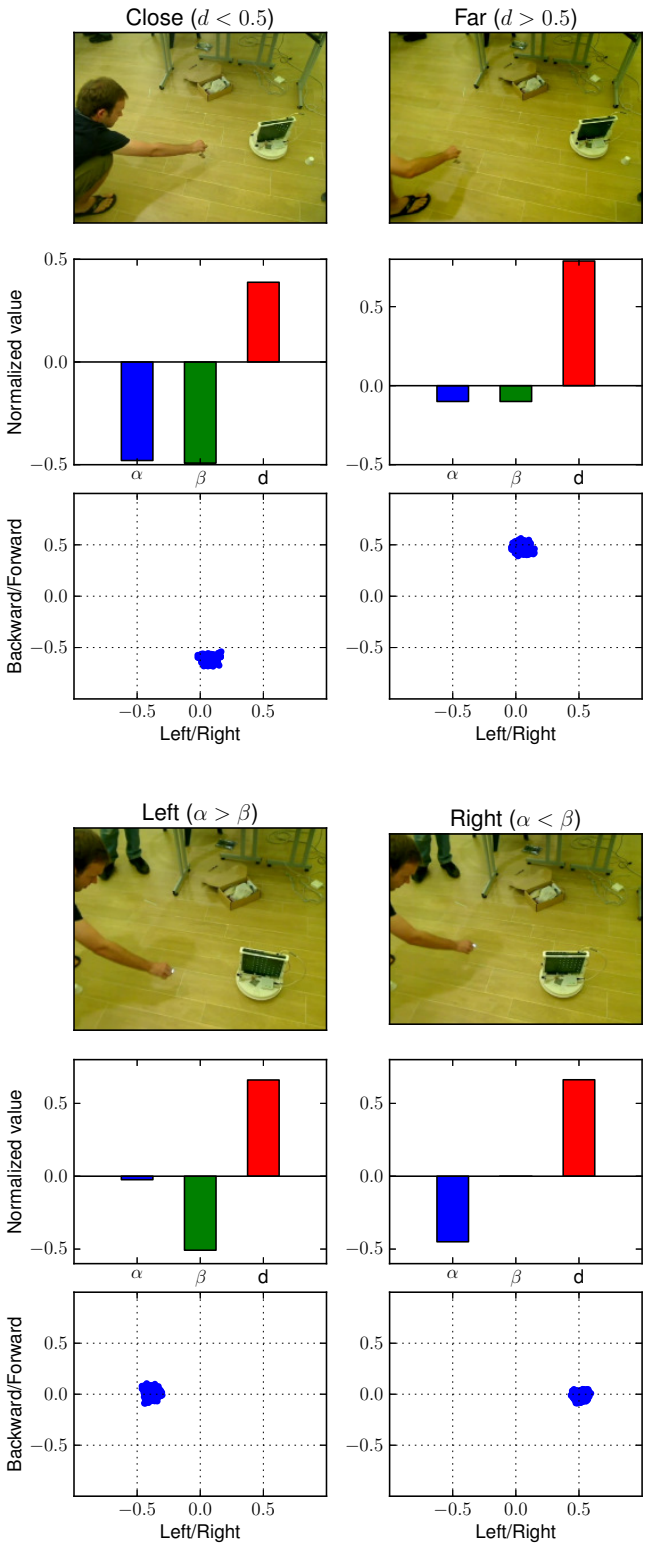

Fig. 5. Nengo simulation running on the SpiNNaker robot (Top): when the stimulus is located centrally to the field of view, but far from the agent, the estimated distance $d > 0.5$, $\alpha \approx \beta$, and the agent moves forward; when the stimulus converges in the field of view, d decreases and both angles become negative; the agent moves backward. (Bottom) When the stimulus is located to the left α is greater than β, and the robot turns left. Conversely, when $\alpha < \beta$ the robot turns right. Angles, distances and movement command values are normalized.

[7] C Eliasmith, TC Stewart, X Choo, T Bekolay, T DeWolf, Yichuan Tang, and Daniel Rasmussen. A Large-Scale Model of the Functioning Brain. *Science*, 338(6111):1202–1205, 2012.

[8] SB Furber and S Temple. Neural Systems Engineering. *Computational Intelligence: A Compendium*, 4(13):763–796, April 2008.

[9] F Galluppi, S Davies, S Furber, T Stewart, and C Eliasmith. Real time on-chip implementation of dynamical systems with spiking neurons. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, June 2012.

[10] F Galluppi, S Davies, AD Rast, T Sharp, LA Plana, and S.B. Furber. A Hierarchical Configuration System for a Massively Parallel Neural Hardware Platform. In ACM, editor, *CF '12 Proceedings of the 9th conference on Computing Frontiers*, pages 183–192, New York, 2012.

[11] G Indiveri, B Linares-Barranco, T Hamilton, A Van Schaik, R Etienne-Cummings, T Delbruck, SC Liu, P Dudek, P Häfliger, S Renaud, J Schemmel, G Cauwenberghs, J Arthur, K Hynna, F Folowosele, S Saighi, T Serrano-Gotarredona, J Wijekoon, Y Wang, and Boahen. Neuromorphic Silicon Neuron Circuits. *Frontiers in neuroscience*, 5(May):23, 2011.

[12] P Lichtsteiner, C Posch, and T Delbruck. A 128x128 120dB 15us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid State Circuits*, 43:566–576, 2008.

[13] C Mead. Neuromorphic electronic systems, 1990.

[14] C Mead and M Mahowald. A silicon model of early visual processing. *Neural Networks*, 1(1):91–97, 1988.

[15] PA Merolla, JV Arthur, B Shi, and K Boahen. Expandable Networks for Neuromorphic Chips. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(2):301–311, February 2007.

[16] SW Moore, PJ Fox, SJT Marsh, AT Markettos, and A Mujumdar. Bluehive - A Field-Programable Custom Computing Machine for Extreme-Scale Real-Time Neural Network Simulation. *2012 IEEE 20th International Symposium on Field Programmable Custom Computing Machines*, pages 133–140, 2012.

[17] A Morrison, C Mehring, T Geisel, AD Aertsen, and M Diesmann. Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural computation*, 17(8):1776–801, August 2005.

[18] E Painkras, LA Plana, J Garside, S Temple, F Galluppi, C Patterson, DR Lester, AD Brown, and SB Furber. SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, August 2013.

[19] L Plana, S Furber, S Temple, M Khan, Y Shi, J Wu, and S Yang. A GALS Infrastructure for a Massively Parallel Multiprocessor. *IEEE Design & Test of Computers*, 24(5):454–463, 2007.

[20] J Schemmel, D Brüderle, A Grübl, M Hock, K Meier, and S Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proc. 2010 Int'l Symp. Circuits and Systems (ISCAS 2010)*, pages 1947–1950, 2010.

[21] R Serrano-Gotarredona, M Oster, P Lichtsteiner, A Linares-Barranco, R Paz-Vicente, F Gomez-Rodriguez, L Camunas-Mesa, R Berner, M Rivas-Perez, T Delbruck, SC Liu, RJ Douglas, P Hafliger, G Jimenez-Moreno, A Civit Ballcels, T Serrano-Gotarredona, AJ Acosta-Jimenez, and B Linares-Barranco. CAVIAR: a 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking. *IEEE Trans. on Neural Networks*, 20(9):1417–1438, 2009.

[22] T Sharp, LA Plana, F Galluppi, and SB Furber. Event-Driven Simulation of Arbitrary Spiking Neural Networks on SpiNNaker. In *The International Conference on Neural Information Processing (ICONIP)*, volume 2011, pages 424–430. Springer, 2011.

[23] Mandyam V Srinivasan. How Bees Exploit Optic Flow: Behavioural Experiments and Neural Models. *Philosophical Transactions of the Royal Society B*, 337(1281):253–259, 1992.

[24] TC Stewart, B Tripp, and C Eliasmith. Python scripting in the Nengo simulator. *Frontiers in Neuroinformatics*, 3(0), 2009.

[25] E Stromatias, F Galluppi, C Patterson, and SB Furber. Power analysis of large-scale, real-time neural networks on SpiNNaker. In *The International Joint Conference on Neural Networks - IJCNN 213*, pages 1570–1577, 2013.

[26] A van Schaik and SC Liu. AER EAR: A matched silicon cochlea pair with address event representation interface. In *IEEE International Symposium on Circuits and Systems ISCAS*, pages 4213–4216, 2005.

[27] TM Wong, R Preissl, P Datta, M Flickner, R Singh, SK Esser, E McQuinn, R Appuswamy, WP Risk, HD Simon, and DS Modha. 10^14. Technical report, IBM, 2013.