STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXII**, Number 2, 2017
DOI: 10.24193/subbi.2017.2.07

ROBBY: A NEUROROBOTICS CONTROL FRAMEWORK USING SPIKING NEURAL NETWORKS

CĂTĂLIN V. RUSU, TIBERIU BAN, AND HOREA ADRIAN GREBLĂ

Abstract. The variety of neural models and robotic hardware has made simulation writing time-consuming and error prone, forcing thus scientists to spend a substantial amount of time on the implementation of their models. We developed a framework called "Robby" that allows the quick simulation of large-scale neural networks designed for robotic control by spiking neural networks. It provides both mechanism for robotic communication and tools for building and simulating neural controllers. We present the basic building blocks of "Robby" and a simple experiment to show its practical value.

1. Introduction

As hardware becomes more diverse and affordable the need for controlling different hardware platforms within similar contexts becomes more prominent. The difficulty lies in the fact that each robot has a different underlying physical layout with different programming interfaces. Thus, similar control programs would have different implementations depending on the robotic platform. In this context neural simulators, that are able to simulate large-scale neural networks efficiently, and robotic frameworks, that allow them to interact with robotic devices, are highly desirable. Such frameworks: (i) allow the facile control of physical cognitive agents; (ii) enable scientist to spend less time on programming details and more on detailing experiments; (iii) provide a basis to easily explore theoretical principles in the context of real computational tasks involving physical autonomous agents; (iv) help increase our understanding of how large neural networks mediate cognitive functions. Popular frameworks either provide a collection of software and algorithms focused on robot communication, sensing and navigation while leaving the development of control

Received by the editors: October 20, 2017.

2010 *Mathematics Subject Classification.* 68T40.

1998 *CR Categories and Descriptors.* I.2.9 [**Artificial Intelligence**]: Robotics – *Operator interfaces*.

Key words and phrases. neural simulators, robotic frameworks, cognitive robotics, spiking neural networks.

programs (neural networks) to the user, like "Player/Stage" [1] (biased towards wheeled robots) or "YARP" [2] (biased towards humanoid robotics), or either provide limited support like "Pyro" [3] or "Orocos" [4]. Thus, it would be of interest to have a system that provides both the abstraction layer for robot communication and the logic to support the development of neural controllers. We introduce "Robby", a flexible and distributed framework for robotic control with spiking neural networks, ideal for large-scale simulations. It enables the control of robotic platforms occupying different physical locations by multiple types of neural networks. In the framework, controllers are primarily neural networks, but in principle they can be any user-defined controller. Additional support for joystick controllers is provided to allow direct manipulation of devices. While this setup might seem restrictive it is sufficient for common simulations in neurobotics while keeping the architecture of the system simple. Since "Robby" makes easy to simulate and explore spiking neural networks with different architectures and properties with the aim of training autonomous robots it could be of interest to the scientific community interested in cognitive robotics. In the following we present the basic principles behind "Robby" and provide several future development directions together with a simple evaluation to show its use.

2. Architecture and implementation details

Low level programming is tedious because it requires a deep understanding of the underlying hardware platform and knowledge of complex languages and programming interfaces. As complex behavior generally requires complex hardware, a large amount of time is spent on writing even the most basic simulations. Essentially, "Robby" is a fast and lightweight platform aimed at simulating spiking neural networks and facilitating the control of various types of robotic devices. From a software development point of view it is written to promote reusability, extensibility and flexibility. The time spent writing code is thus minimized and scientists are able to spent more time modeling rather than setting up complex environments and debugging. The framework is written in C++ and adheres to the POSIX standards. Even if C++ is considered to be a high level language, it offers the means to interact at low level with the hardware in an efficient and portable way.

The architecture of "Robby" is modular. It consists of a control structure (the server), a behavioral component (the client) and a commons component (Fig. 1). The server is in strict relation with devices through an instantiation of corresponding drivers. It forwards commands received from clients, and awaits and reads replies from devices. Besides providing the communication functionality it also provides an interface to plot the raw sensory data received

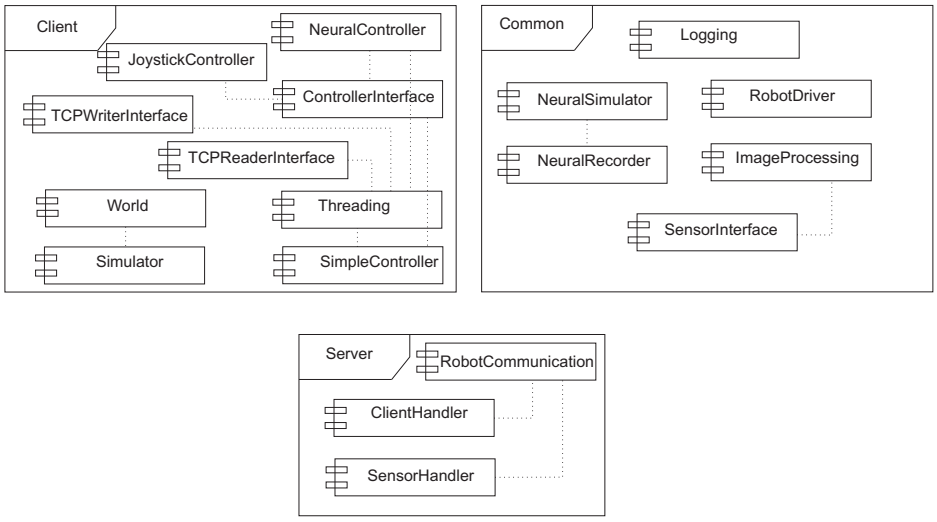
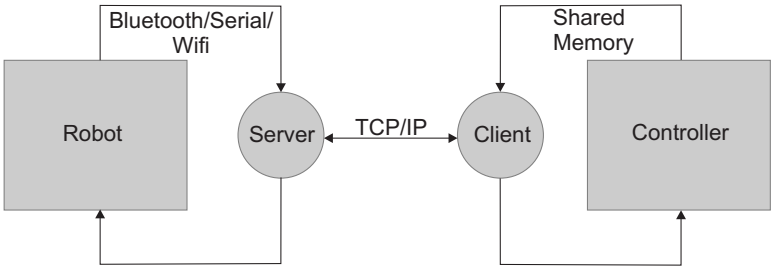


FIGURE 1. "Robby" architecture.

from the devices together with basic communication cycle parameters. The client reads data from the controller and maps it into robot commands which are later sent to the server where they are processed and forwarded. After it sends them to the server it awaits a reply before notifying the controller that the communication cycle is over. A controller implements the *ControllerInterface* and runs in a separate thread. Currently the available controller types are: a *NeuralController* which is a spiking neural network and a *JoystickController* which is useful for direct manipulation of devices. To allow flexibility, other user-defined controller types can be added with the restriction that they must implement the *ControllerInterface*. The common component contains the neural simulator which facilitates the creation and simulation of spiking neural networks [5], device drivers and various image processing algorithms like Laplacian of Gaussian and log-polar filters [6, 7] used to process device video data. Other sensorial controllers can be added provided that they implement the *ControllerInterface*.

This server/client strategy acts as proxy between the client controller and server driver entities to increase flexibility in control and allow the controller and robot to be in different locations with server and client communication mediated through Ethernet. Besides this obvious geographical benefit, such a separation allows the decoupling between the computationally inexpensive communication process and the highly time consuming simulations performed by the controller. What is actually transmitted through the TCP channel are

robotic commands embedded into packets. Their aim is to keep communication uniform and enable a seamless control of many types of robots. Each package contains the command together with parameters and optional sensor values. Thus, as long as a physical connection between the server and the robotic device can be initiated multiple controllers and devices can co-exist. "Robby" makes no assumption about the connection medium between the server and the robotic devices, but, as stated previously, a driver needs to be supplied. Efforts are made to increase the number of supported connection types, but currently only drivers for devices with serial connections are provided. Fig. 2 depicts the communication processes inside "Robby". The client and server communicate by using TCP sockets. At the server side, communication between devices and the server could be achieved via bluetooth, wireless or serial depending on the device capabilities. At the client side, communication between the client and the controller is achieved by using a common memory buffer guarded by a critical section. This setup allows the existence of multiple controllers at the same time. Any of the controllers can be replaced by other controllers if they comply with the *ControllerInterface*.



1

FIGURE 2. "Robby" Communication. Communication between the server and client is implemented using the TCP/IP protocol.

While the speed of the simulation itself arguably is not important for detailed modeling of complex biophysical entities and small simulations, in the case of large-scale simulations for the purpose of robotic control it still remains an important constraint. Such simulations of large neural systems consisting of thousands of neurons are heavily time consuming because of the amount of interaction in the network which needs to be evaluated. As memory becomes an inexpensive commodity the trade-off between memory usage and simulation speed needs to be carefully investigated. Recent computing optimization techniques [8, 9] propose the usage of lookup tables to avoid the repeated computation of a value. Thus the runtime computation of what might be expensive is replaced with a simple indexing operation with constant complexity. These approaches increase code size and memory consumption, but the speed gain outweighs the cost. "Robby" implements lookup tables to improve the performance when simulating large neural networks. They are used when computing postsynaptic responses, a process that involves, for some neural models, repetitive evaluations of exponential functions. In addition, when simulating neural networks some operations are independent and can be executed in parallel (for example the update of a neuron membrane potential). These operations are implemented using OpenMP directives [10] to allow multi-threading.

3. Robby as a framework for robot learning

As outlined in previous sections, "Robby" is designed for the simulation of large-scale neural networks for robotic control in a computationally efficient way with as little code as possible. It is able to simulate different types of spiking neurons at different levels of detail. In the current implementation, the available models are the integrate-and-fire [5] and Izhikevich [11]. Because of its simplicity the integrate-and-fire neuron is commonly used in large-scale simulations [12, 13, 14] while the Izhikevich neuron can reproduce the complex behavior observed only at more detailed models while at the same time allowing an efficient implementation [11]. Thus, this selection of neuron models albeit small is sufficient, since complex models are computationally expensive and networks composed out of them would not be feasible as robotic controllers. Different types of static and dynamic synapses together with various plasticity rules (short-term plasticity [15], spike-timing-dependent plasticity (STDP) [16], synaptic scaling [17] or intrinsic neuronal plasticity [18]) are available in order to facilitate learning. In the case of static synapses a fixed current is injected into the postsynaptic neuron at the time of the presynaptic activation while dynamic synapses feature facilitation or depression mechanisms. In addition, different supervised learning rules for spiking neural networks [19, 20]

together with reward modulated spike-timing-dependent plasticity [21] are implemented in order to create a framework for reinforcement learning [22].

In the following we present a simple experiment to demonstrate some of the features of "Robby" and their application. Consider the setup presented

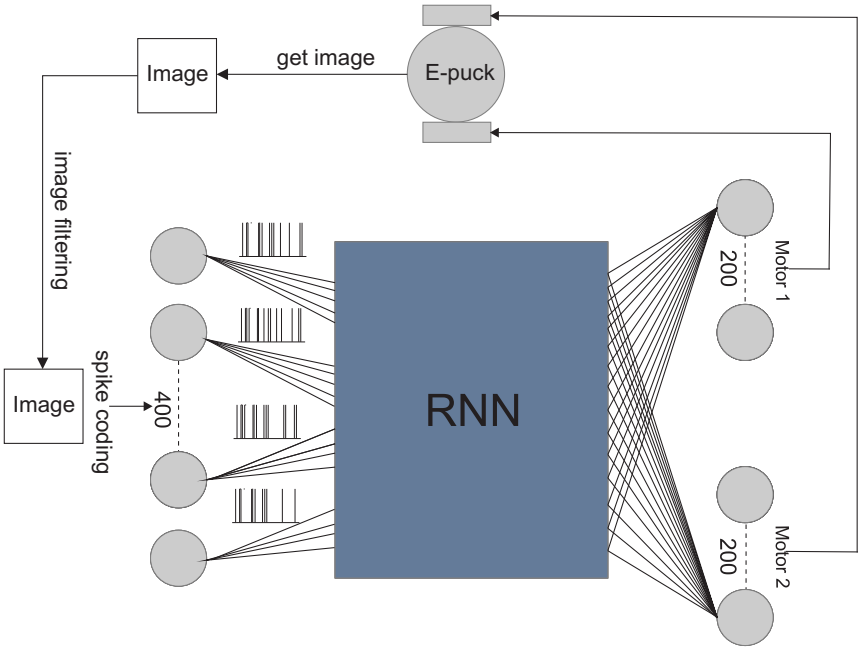


FIGURE 3. Experiment setup. A spiking neural network controlled an E-puck robot located in a rectangular arena. The network received as input the video information from a camera mounted on top of the robot and controlled the motor activation.

in Fig. 3. An E-puck robot with an externally mounted camera navigated through a rectangular arena which contained no obstacles. The controller of the robot was a spiking neural network with three layers of neurons. The first and third served as sensor and motor neurons respectively.

The input neurons conveyed video information about the environment. More precisely, the pixels of the image received from the camera were averaged to provide input for the 400 input neurons. These activation values were normalized between 0 and 1. The input neurons fired Poisson spike trains with rate proportional to the activation, between 10 and 50 Hz. The 400 output neurons served as motor neurons, 200 for each motor (left and

right). Each of these neuron populations is further divided into two 100 neuron pools. These two pools of neurons are assigned to one motor with its speed proportional to their corresponding average firing rate. The spikes were converted to effector activation by integrating them with a leaky accumulator of time constant $\tau = 500$ ms. The activation of two 100 motor neurons populations were averaged to yield the activation of one effector at a timestep. In this antagonistic setup given the activation a_+ and a_- of the two 100 neuron populations, the motor was given a relative command $(a_+ + a_-)$. The network was thus composed out of 400 input neurons and 400 motor neurons. The network also had 1500 hidden neurons. All the neurons were modeled as integrate-and-fire. Each non-input neuron in the network sent connections to 30% of hidden and motor neurons. Input neurons projected onto 45% of the hidden neurons. All the connections were chosen randomly from the uniform distribution. The connections between the first and second layer together with the recurrences within the hidden layer were static spiking synapses while to ones that projected onto the motor neurons were static STDP synapses. In this simple experiment the goal of the robot was to freely explore the environment for 15 seconds. The distance traveled by the robot from the initial point is depicted in Fig. 4. It was computed from the information received from a camera located on top of the environment which recorded every position of the robot. The distribution of synaptic weights of a randomly selected motor neuron at the beginning and at the end of the simulation together with the activation values of the motors received at each timestep are also depicted in Fig. 4. The resulted bimodal weight distribution is consistent with experimental results [16]. The duration of a simulation step which includes the timestep of the neural controller together with the time required to send data to and from the robot has on average the value of 70 ms (see Fig. 4). Due to the low data transfer rates of the robot connection the duration of the communication cycle increases to an average of 400 ms if the video from the E-puck on-board camera is transmitted instead from the external camera on top of the robot. This value is dependent upon the settings of the camera like for example number of pixels or color depth. Although this experiment is simplistic it presents some of the basic features of "Robby" and how they can easily be used in the context of robot learning and control with spiking neural networks.

The experiment was aimed to present the flexibility of the framework that had been developed around *Robby*. As previously stated, this framework has a major advantage of being able to include elements of supervised learning rules with respect to the goal of achieving results in reinforced learning.

The next step that is under development is to apply the neural network framework of Robby to a new business domain in gathering information from

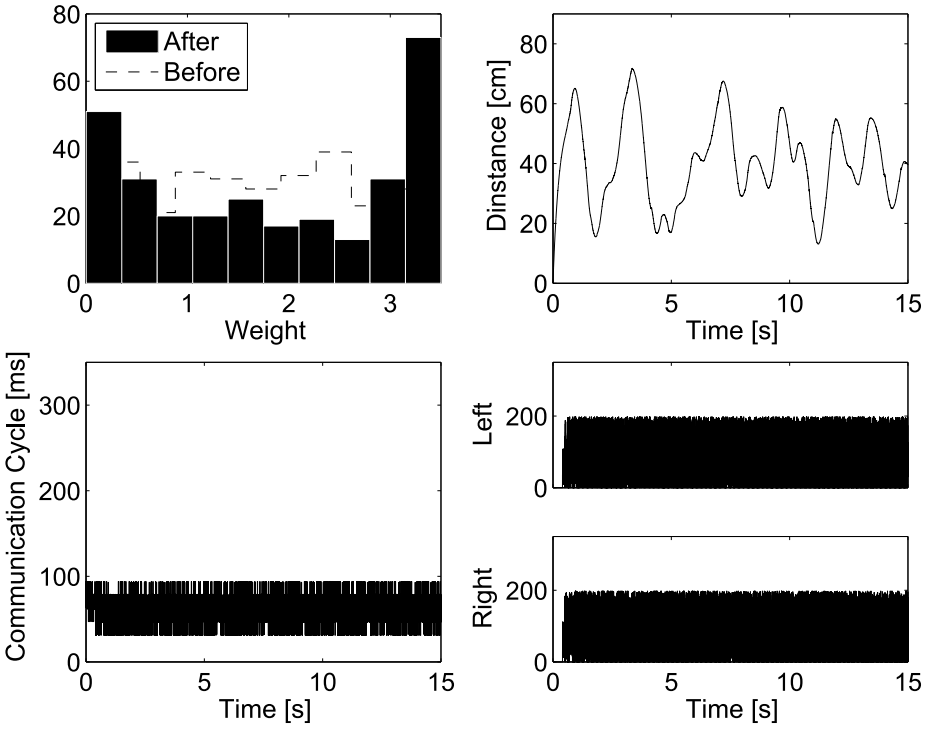


FIGURE 4. Experiment results. (a) The weight distribution of a randomly selected motor (output) neuron at $t = 0$ s (dashed line) and $t = 15$ s. (b) The distance traveled by the robot from the initial point. (c) The duration of a simulation step in ms. (d) The activation value of each motor.

patterns of mistakes found by analysing test papers, as presented as a theoretical approach in [23]. Results already obtained in developing a mathematical model [24] proved that in order to maximize the robustness of the framework, elements of unsupervised learning (i.e. generation of frequent item sets and determining association rules) need to be linked with elements of supervised learning, in order to prune out coincidental occurrences that are not relevant in terms of knowledge gathered as patterns of mistakes.

So far results from [24] as well as [23] were heavily based on rationales that followed strongly the mathematical model of association rules. This business domain can be extended with a new approach, incorporating elements of neural networks in addition to the existing mathematical model. Using the framework of Robby the goal is to include results gathered from association rules and

frequent item set discovery in order to modify existing algorithms based on neural networks that are currently used in credit card fraud detection, in order to study the possibility of predicting frauds in evaluation tests, based on patterns of mistakes.

4. Conclusion

We have introduced a flexible distributed control framework for robotic interaction with spiking neural networks ideal for large-scale simulations. Our aim was to create a multi-threaded, flexible, lightweight framework which promotes code reuse. "Robby" is not intended to be a multi-purpose tool, but it proved to be a convenient tool for quickly exploring new ideas and write experiments with a small amount of code. At the current stage in the development the number of supported robotic platforms and neuron models is still limited. Future plans include support for further commonly used robotic devices and neuron models.

References

[1] B.P. Gerkey, R.T. Vaughan, K. Stoy, A. Howard, G.S. Sukhatme, and M.J. Mataric. *Most Valuable Player: A Robot Device Server for Distributed Control.* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Wailea, Hawaii, 2001.

[2] P. Fitzpatrick, G. Metta, and L. Natale. Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29–45, 2008.

[3] D.S. Blank, D. Kumar, L. Meeden, and H. Yanco. Pyro: A python-based versatile programming environment for teaching robotics. *Journal of Educational Resources in Computing*, 2004.

[4] Bruyninckx H. Open robot control software: the orocos project. 2001.

[5] W. Gerstner and W. Kistler. *Spiking neuron models: Single neurons, populations, plasticity.* Cambridge University Press, 2002.

[6] R. Haralick and L. Shapiro. *Computer and Robot Vision.* Addison-Wesley Publishing, 1992.

[7] G. Wolberg and S. Zokai. *Robust image registration using log-polar transform.* IEEE International Conference on Image Processing, 2000.

[8] M. Hall and J. Mayfield. *Improving the Performance of AI Software: Payoffs and Pitfalls in Using Automatic Memoization.* Proceedings of the Sixth International Symposium on Artificial Intelligence, Monterrey, Mexico, 1993.

[9] M. Hall and J.P. McNamee. Improving software performance with automatic memoization. *Johns Hopkins APL Technical Digest*, 18(2), 1997.

[10] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. *Parallel Programming in OpenMP.* Morgan Kaufmann, 2000.

[11] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14:1569–1572, 2003.

[12] Brunel N. The evidence for neural information processing with precise spike-times: A survey. *Natural Computing*, 3:195–206, 2000.

[13] Cessac B and Vieville T. On dynamics of integrate-and-fire neural networks with adaptive conductances. *Frontiers in Computational Neuroscience*, 2, 2008.
[14] Soula H. Alwan A. and Belson G. Learning at the edge of chaos: Temporal coupling of spiking neuron controller for autonomous robotics. 2005.
[15] L.F. Abbott and W.G. Regehr. Synaptic computation. *Nature*, 431:796–803, 2004.
[16] S. Song, K. D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926, 2000.
[17] G.G Turrigiano and S. B. Nelson. Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, 5:97–107, 2004.
[18] W. Zhang and D. J. Linden. The other side of the engram: Experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, 4:885–900, 2003.
[19] F. Ponulak. *ReSuMe-new supervised learning method for Spiking Neural Networks*. International Conference on Machine Learning, ICML, 2005.
[20] R. Florian. The chronotron: a neuron that learns to fire temporally-precise spike patterns. *PLoS ONE*, 7(8), 2012.
[21] R. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.
[22] Sutton RS and Barto AG. Reinforcement learning. 1998.
[23] Ban T. Fuzzy computing for complexity level of evaluation tests. *Studia Universitatis Babes-Bolyai, Seria Informatica*, LVIII:81–93, 2013.
[24] Ban T. Generating and assessing test papers complexity using predictions in evolutionary algorithms. 2009.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: rusu@cs.ubbcluj.ro
E-mail address: tiberiu@cs.ubbcluj.ro
E-mail address: horea@cs.ubbcluj.ro