Programming the MIRTO Robot with Neurons

Christian Huyck[1], Giuseppe Primiero[1], and Franco Raimondi[1]

Middlesex University, London, UK
c.huyck, g.primiero & f.raimondi@mdx.ac.uk

Abstract
MIRTO is a new, inexpensive, open-source robot. The specification, the necessary libraries, and sample code are freely available. It has been used to teach undergraduate students programming and as an extensible base platform for students engineering robots. While it is typically programmed in traditional programming languages, it has also been driven by simulated neurons; point neurons have been used to follow a line. Since neurons are the basis of animal cognition, using them as the basis of a cognitive architecture is a promising idea. The neural MIRTO line following system can be extended into a larger more cognitive neural agent as a way forward in neural cognitive architectures.

Keywords: Neuron, Robot, FLIF and Education

1 Introduction

The recent progress in hardware miniaturization and the increasing processing capabilities of small and embedded devices make it possible to implement software that to date was limited to more powerful (and stationary) servers. Sophisticated software can even run on mobile environments and robots.

This paper describes the MIddlesex Robotic plaTfOrm (MIRTO), an open-source, low-cost robot built using Arduino, Raspberry Pi and various kinds of commonly available sensors and actuators. It then shows how a line following task can be implemented using a biological neural network.

The design specification, including the design of parts that can be cut with a laser printer and the software required to run it, can be found at https://github.com/fraimondi/myrtle. The overall cost of one MIRTO robot is roughly £100, and it has been used for the first year of an undergraduate computer science and engineering degrees, with students implementing line following algorithms in Racket (a Lisp dialect).

This paper presents a Java implementation of the robotic libraries running on the Raspberry Pi to ease interaction with a neural system, also written in Java. The neural model is a fatiguing leaky integrate and fire (FLIF) point model.

The FLIF driven robot has sets of neurons to encode three light sensors that can be used for line following. It associates the input of a light sensor with one or more neurons. A light

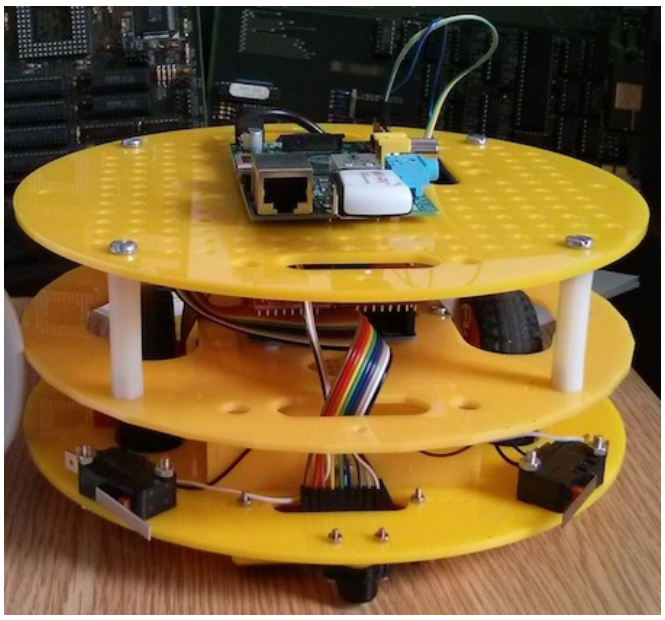

Figure 1: The MIddlesex RoboTic platfOrm MIRTO

sensor mounted on the left-hand side of the robot affects the motor mounted on the right-hand side, with the centre sensor going to both motors. The more neurons that are firing, the faster the motors go. When the robot starts to veer to, for instance, the right, fewer neurons corresponding to the right sensor fire. The right sensor neurons pass less activation to the left motor neurons, the left motor slows, and the robot turns right. If it goes so far to the right that the centre sensor starts to move off the line, the right motor slows and the left motor stops. This simple mechanism is sufficient to follow lines. This is similar to Braitenberg's first vehicle[3].

The rest of the paper is organised as follows: background material is provided in Section 2 and Section 3 describing, respectively, the basic MIRTO architecture and how it is currently used for teaching. Section 4 discusses neural cognitive architectures; the actual implementation is presented in Section 5. Section 6 discusses how the neural robot could be extended into a more cognitive agent, and Section 7 concludes.

2 MIRTO

MIRTO (see Figure 1) has been developed at Middlesex University to support teaching a range of topics in our Computer Science and Engineering programmes. It has been fully designed at Middlesex University and it has been built using widely available hardware. The robotic platform is composed of:

- A base layer incorporating a pair of HUB-ee wheels[1], two bump sensors and six infra-red (IR) sensors. The base layer also incorporates a rechargeable 9000 mAh battery pack, which is enough to cover a full day of teaching.

[1] `http://www.creative-robotics.com/About-HUBee-Wheels`

- A middle layer incorporating an Arduino Uno[2] microcontroller, which is used to collect raw data from the sensors and to drive the motors.
- A top layer incorporating a Raspberry Pi[3], connected to the Arduino in the middle layer by means of a serial connection. The Raspberry Pi runs the standard Debian-derivated Raspbian Linux distribution. This layer can be further extended with a wireless USB card (as depicted in Figure 1), video camera, microphone, etc. The code described in this paper runs at this level and is stored on a standard SD card. Communication between the Raspberry Pi and the Arduino layer employs the standard Firmata protocol available from `http://www.firmata.org/`, extended with some bespoke messages to support wheels quadrature encoders, distance sensors, etc. (these are not part of the standard Firmata protocol).

3 MIRTO for Teaching

In the design of a new Computer Science programme for the academic year 2013/2014 at Middlesex University, London, we have been driven by two main requirements: first, students should be equipped with a strong theoretical basis to be prepared for change; second, the output of software systems increasingly results in tangible actions in the real world. As a result, we expose students to a wide variety of physical devices that are crucial to understanding principles of computer science: from simple logic gates (the building blocks of every computer currently commercially available), to microcontrollers (Arduino) and other specialist devices. Knowledge of these devices is project-driven around basic building blocks. We start with a traffic light system built using Arduino boards [2, 16], LEDs and input switches. Students learn how to control LEDs as lists in a timed loop using *clocks*, how to *read* the values of input switches and to modify the control loop accordingly. Following this Arduino project, students explore a number of applications including, among others, a dungeon game with a GUI to learn data structures and a web server to control an Arduino board. All these elements contribute towards the final project: develop applications for the MIddlesex Robotic PlaTfOrm (MIRTO).

This project starts by reverse-engineering the wheel components of the MIRTO with finite state machines and learning how to control their analogue circuits by the digital inputs available from the Arduino board. We then show the students how to create a (wired in the first place, over a network subsequently) connection to the Arduino layer of MIRTO using a bespoke version of the Firmata protocol [6]. This can then execute simple exercises to drive the wheels at different speeds, knowing how far they have gone by reading the number of their physical rotations, stop the motors and close the connection. The next task is to enable and learn how to read the values of three infra-red sensors mounted under the MIRTO lower layer and its bump-sensors, mounted on the sides. All these task lead towards mastering the definition and use of programming skills that can be seen at work in the interaction with MIRTO. In particular, reading of the sensor is directed towards the construction of procedures to do line-following exercises, by using sensors to measure reflectance: when they are on a white or very reflective surface, they return a low value (below 50). When they are on a black or non-reflecting surface, they return a high value (up to approximately 1900). Line following is a trigger to explore important computer-science and mathematics-related concepts. We present the MIRTO with active sensors as a way to introduce the difference between open-loop and closed-loop systems; we proceed with different versions of the line-following algorithm: from corner-detection, and

[2] `http://www.arduino.cc`
[3] `http://www.raspberrypi.org/`

junction-detection, up to implementing the principle of proportional control, with integral and derivative values. Finally, we ask the students to combine all of these aspects in order to allow MIRTO to solve a maze. Further exercises include the use of additional features of MIRTO: image capture via a USB camera connected to the Raspberry PI and voice recognition, by using `pocketsphinx`.

For a full description of teaching methods, evaluation and assessment, see [20].

4 Neurons as a Cognitive Architecture

Newell [18] states that the ultimate goal is a unified theory of human cognition expressed as a theory of the architecture of human cognition that is

> the fixed (or slowly varying) structure that forms the framework for the immediate processes of cognitive performance and learning.

That is, the person who coined the term, states that a cognitive architecture is the fixed structure that does the cognition and learning. It is pretty clear, and Newell discusses it as one level, that neurons and their synapses are a slowly varying structure that performs cognition.

Of course, popular cognitive architectures are computer programs or systems that can be programmed. For instance, ACT-R [1] and Soar [13] are basic architectures; both have particular programs written to perform particular tasks.

Another type of computer program is simulated biological neural systems. There are a wide range of biological neural models, from simple point models (e.g. [14]) to more complex compartmental models (e.g. [9]). Biological neural models can be combined to create network models. Indeed, there are attempts to model an entire brain [17], though this is a long term project.

Simulated neurons can be, in essence, programmed to perform activities. They are programmed by setting up the connections between them (synapses) so that neural firing propagates to perform the necessary computation. It has been shown that given sufficient neurons, a simulated neural system is Turing complete [5].

Simulated neural systems can be programmed, and it is known that intelligence emerges from the basis of the model. Consequently, biological neural models, and systems developed from them are a great candidate for a cognitive architecture. Moreover, these neural systems are more than just biologically inspired, one goal is to maximize their accuracy as models of the biological system.

One important argument states that for an agent to be intelligent, it needs to be embodied [4]. This fits in well with a system that can learn from its environment; it gets most of its knowledge directly or indirectly from that environment. Consequently, to make progress in developing AI systems, eventually the system must be embodied. While work has been done on virtual neural robots, embodied in virtual environments (e.g. CABot3 see [10]), this paper reports on a physical neural robot.

5 Neural MIRTO

Following a range of simple neural robots, MIRTO was controlled by a simple line following spiking neural mechanism. Like other systems [19, 8], the system was like Braitenberg's first vehicle; it was a simple stimulus response mechanism. Like both, light sensors on one side aligned with a motor to move the robot toward that light. However, the earlier models used

four neurons, two for the sensory input and two for the motor output, and this requires precise timing. The model described below uses many neurons but the timing is quite coarse.

The neural model is a fatiguing leaky integrate and fire model, and all the code for the neural robot is available at https://github.com/fraimondi/myrtle. Equation 1 describes the integrate and fire component. The neuron j integrates activity from other neurons i that fired in the last cycle (V_i) weighted by the synaptic strength w_{ij} from the firing neuron to j. Neuron j fires if activity surpasses a threshold θ and the neuron's fatigue F_j.

$$\theta + F_j < A_j = \sum_{i \in V_i} w_{ij} \tag{1}$$

If a neuron does not fire, some of its activation leaks away. This leak, or decay, is represented by a constant D where $D > 1$. Ignoring external input and assuming i did not fire at $t-1$, activation of neuron i at time t is

$$A_i^t = A_i^{t-1}/D \tag{2}$$

A neuron's fatigue F_j increases by a constant F_c if it fires (line 1 of equation 4). It decreases by a different constant F_r (line 2), but not below 0 (line 3).

$$\begin{aligned} F_j^{t+1} &= F_j^t + F_c \\ F_j^{t+1} &= F_j^t - F_r \\ (F_j^{t+1} < 0) &\rightarrow F_j^{t+1} = 0 \end{aligned} \tag{3}$$

There is a further explanation of this model [11]. The parameters have been tuned to rat somatosensory neurons, and with $\theta = 2.2$, $D = 1.12$, $F_c = 0.045$, and $F_r = 0.01$ the model is reasonably accurate. These are the parameters used for the neurons controlling MIRTO.

The topology is simple. There are 500 neurons, with 100 associated with each of the three input sensors, and 100 associated with each of the wheel motors. Each of the right input sensor neurons has 10 synapses to randomly selected neurons for the left wheel. Each of these has 0.3 weight. Similar connections go from the left sensor to the right wheel motor. Each neuron in the centre sensor sends 10 synapses to neurons in each of the wheel motors, and these weights are 0.15.

The sensor neurons are turned on each cycle depending on the sensor reading. The higher the reading, the more neurons are turned on in the associated sensor. These are randomly selected from the appropriate sensory neurons each cycle. The wheel motors are set based on the number of neurons firing in the set of neurons associated with the left or right motor. The more neurons that are firing, the faster the motor goes.

So, when all sensors are on the line (rarely), more input neurons are turned on, and both motors go faster. If it veers to the left, the left sensor will get less activation, and this will propagate through to the right motor that will slow, causing the robot to turn right. If the centre sensor goes off the line, both motors will slow.

The results are solid but not perfect. The robot typically follows a track. It occasionally loses the line during a tight turn. A demo video is available at
`https://www.youtube.com/watch?v=N8pZZ9sk6N0`.

6 Robotic Cognitive Agents

Clearly, the simulated neural controller for MIRTO is not a cognitive agent. It is a simple stimulus response robot. However, the neural robot can form the basis of a larger neural agent.

By integrating existing simulated neural systems into this agent (e.g. [10]), new versions of the robot should be developed easily. For example, a planning system based on Maes nets [15] has already been implemented in FLIF neurons, and this should support improved behaviour from a new version of the robot.

Of course new neural systems can also be developed, and new sensors can be added to MIRTO, and those can be integrated into the neural system. For example, one obvious extension is to improve reactions. The line following is far from perfect. Here reactions could be closely tuned, by involving extra neural memory; this could be inspired by animal neural behaviour, or even be a model of that neural behaviour.

MIRTO has been extended to use a USB camera. This could be used to improve line following by seeing what is coming, and it could help find the line when it is no longer under the light sensors. While bump sensors are already used on MIRTO, an effector beyond the wheel motors could also be used to extend capabilities.

Clearly one of the weaknesses of the reported neural MIRTO is that it does not use learning. Earlier FLIF systems have used learning (e.g. [10]), and it would be useful to integrate these capabilities into new versions. An obvious extension is to use existing neural reinforcement learning systems to train the systems to follow lines. This would be used initially at the stimulus response level, but could also be used at higher levels, so that some behaviour is more flexible. Furthermore, the system could cache away the track so that it knew what was coming. This could be combined with reinforcement learning, and even planning to optimise performance on the line tracking task. Of course, MIRTO could also be used for other tasks, such as MIRTO in a maze with some reward. This could easily lead to cognitive tasks trying to match rat in a maze data.

It is important to note that this is all done with one architecture: simulated neurons communicating with synapses, and learning is done by changing synaptic weights. Many robots have separate systems that do not communicate. Here communication should be easy to implement. The communication topology is the neural graph, and it is important to note that, like the brain, the topology is not well connected. Different sub-systems can be developed, and those can communicate with other sub-systems. One potential framework for specifying communication is derived from Jackendoff's tripartite theory [12]. A sub-system has its own neurons, but it also has shared neurons with those sub-systems with which it communicates. This mechanism was used for a neural language processing system to combine the lexical, syntactic and semantic sub-systems. This mechanism should be applicable to other sub-systems and communication between them.

It should be noted that the computational power of the Raspberry Pi is quite limited. It should support real time simulation of thousands of FLIF neurons, but the CABot3 agent used over 100,000 and the Pi can not support that many. In particular, existing visual systems start with a 50x50 input net, and then use thousands of centre-surround receptive field neurons, and line, edge, and corner detection neurons.

None the less, new hardware could be added to MIRTO. For instance, MIRTO could easily carry the existing SpiNNaker neuromorphic chip [7]. Moreover, it should be relatively straight forward to integrate communication with off-robot computers via, for example, wi-fi.

One key to improving the overall neural systems, and refining the neural architecture is constraints from the environment. Neurons drive biological agents cognitively, but also sub-cognitively. It is important that humans can learn to improve their motor performance, for example, by practicing a tennis stroke. This is neural change driven by self-controlled performance. There are several neural levels between motor cortex and the neurons that control muscle twitch. Understanding this behaviour is important not only for improved robots but to

understand neuro-cognitive and cognitive function.

7 Conclusion

MIRTO provides a relatively simple and inexpensive platform for developing robotics applications. Control via Java and Racket support teaching of undergraduate computer scientists, and an extensible platform for more complex robots.

Java libraries in MIRTO's Raspberry Pi interact with a Java neural simulator. These simulated neurons act as a stimulus response controller for MIRTO, enabling it to follow lines.

This neural MIRTO provides the basis for a more complex biological neural agent. Earlier virtual neural robots were very simple cognitive agents, using neurons as a low level cognitive architecture. MIRTO provides a platform for simple cognitive agents working in a physical environment. This paper has proposed several extensions, for instance, memorising the immediate environment.

If MIRTO could be given a new task, and could determine how to improve its performance on that task, then actually improve its performance, then it would have made a significant step in biological cognitive architectures. Continued development of these neural robots could lead to better robots, a better understanding of large scale neural dynamics, and a better understanding of neural cognitive architectures.

References

[1] J. Anderson and C. Lebiere. *The Atomic Components of Thought*. Lawrence Erlbaum, 1998.

[2] Massimo Banzi. *Getting Started with Arduino*. Make Books - Imprint of: O'Reilly Media, Sebastopol, CA, ill edition, 2008.

[3] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, Massachusetts, 1984.

[4] R. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:1:139–159, 1991.

[5] E. Byrne and C. Huyck. Processing with cell assemblies. *Neurocomputing*, 74:76–83, 2010.

[6] The Firmata protocol. `http://firmata.org/`. Accessed: 2014-03-20.

[7] S. Furber, D. Lester, L. Plana, J. Garside, E. Painkras, S. Temple, and A. Brown. Overview of the SpiNNaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467, 2013.

[8] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan, and G. Clarke. Evolving spiking neural network controllers for autonomous robots. In *Conference on Robotics and Automation*, pages 4620–4626, 2004.

[9] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[10] C. Huyck, R. Belavkin, F. Jamshed, K. Nadh, P. Passmore, E. Byrne, and D.Diaper. CABot3: A simulated neural games agent. In *7th Intl W/shop on Neural-Symbolic Learning and Reasoning, NeSYS'11*, 2011.

[11] C. Huyck and A. Parvizi. Parameter values and fatigue mechanisms for flif neurons. *Journal of Systemics, Cybernetics and Informatics*, 10:4:80–86, 2012.

[12] R. Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press, 2002.

[13] J. Laird, A. Newell, and P. Rosenbloom. Soar: An architecture for general cognition. *Artificial Intelligence*, 33,1:1–64, 1987.

[14] L. Lapicque. Recherches quantitatives sur l'excitation lectrique des nerfs traite comme une polarisation. *J. Physiol. Pathol. Gen*, 9:620–635, 1907.

[15] P. Maes. How to do the right thing. *Connection Science*, 1:3:291–323, 1989.

[16] M. Margolis. *Arduino Cookbook*. O'Reilly Media, 2011.

[17] H. Markram. The blue brain project. *Nature Reviews Neuroscience*, 7:153–160, 2006.

[18] A. Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[19] J. Nielsen and H. Lund. Spiking neural building block robot with hebbian learning. In *Conference on Intelligent Robots and Systems*, pages 1363–1369., 2003.

[20] F. Raimondi, G. Primiero, K. Androutsopoulos, N. Gorogiannis, M. Loomes, M. Margolis, P. Varsani, N. Weldin, and A. Zivanovic. A racket-based robot to teach first-year computer science. In *Proceedings of the ELS 2014 - 7th European Lisp Symposium*, 2014.