Towards Developing Micro-Scale Robots for Inaccessible Fluidic Environments[*]

Martin Colley, Gustavo de Souza, Hani Hagras,
Anthony Pounds-Cornish, Graham Clarke, Victor Callaghan
Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK
{martin, gdesou, hani, apound, graham, vic}@essex.ac.uk

Abstract - *In this paper we introduce the development of dedicated hardware capable of controlling autonomous micro-scale robots for fault detection/repair in complex inaccessible fluidic environments. This work is part of a European Union funded project entitled SOCIAL, (Self-Organized societies of Connectionist Intelligent Agents capable of Learning No IST-2001-38911). The project's aim is to produce a swarm of micro-scale ($5cm^3$) autonomous robots that, through indirect communication, are capable of achieving fault detection and reparation in difficult, challenging and inaccessible environments. An application benchmark for this project is the on-line monitoring and maintenance of underwater pipelines like those found in the oil industry or desalination plants. The robots would move through the fluidic environment, continuously sensing for corrosion and scaling faults in the pipeline.*

Keywords: Spiking Neural Networks, Evolutionary Robotics, FPGA, Hardware-in-the-Loop, Simulation.

1 Introduction

The primary aim of the SOCIAL project is to investigate methods for engineering emergent collective behaviour in large societies of micro scale collaborative autonomous robots that can learn and evolve. The robots are viewed as independent individuals – having an identical structure, yet able to exhibit varying behaviours – capable of perceiving / exploring their environment, selectively focusing attention, communicating with peers, initiating and completing corrective tasks, and learning both at the individual and communal levels. The use of spiking neural networks (SNNs) allows the behaviours associated with the robots to be evolved within a simulation environment, allowing them to be studied and verified before the robots are released into the physical environment. This paper considers the benefits of using spiking neural networks and describes the construction of the development environment.

1.1 Spiking Neural Networks

SNNs are deemed computationally more powerful than conventional artificial neural network formalisms on the basis of extensive theoretical work by Maass [1]. "Computationally more powerful" implies that SNNs need fewer nodes to solve the same problem than conventional artificial neural networks [1]. From an implementation viewpoint, this means that SNN circuits of the same complexity can provide "more for less" compared to other neural network implementations (such as the multi layer perceptrons). In addition, SNNs provide a number of other desirable features such as noise-robustness (tolerance to background noise) and simple real-world interfaces [2]. The computational power of SNNs exist because of the intrinsic time-dependent dynamics of spiking neurons that allow the temporal patterns of sensory-motor events to be captured and exploited more efficiently than the other connectionist models (i.e. with fewer neurons and simpler circuits) [1], [3]. Moreover, SNNs can be mapped easily to hardware because the spikes are in essence binary events and the non linear dynamics and the coding of spiking circuits can be provided by spiking times, rather than by non linear, real valued activation functions used in the traditional connectionist neuron models. In other words, a few logic operations and instructions to move around single bits over time would be sufficient to embed large circuits of spiking neurons that display complex abilities and behaviours into tiny and low power chips. Therefore such SNNs will be appropriate control mechanisms for our micro-scale autonomous robots as they can give a very good response dealing with noise using tiny chips that consume little power in inaccessible environments. This is a big advantage especially as both memory and power are extremely limited on the micro and micro-scale platforms currently being developed by the project.

There have been many applications of SNNs to robotics; most of these applications are focused on the first stages of sensory processing and on relatively simple motor control [4], [5]. Despite these interesting implementations, they did not produce methods for

[*] **0-7803-8566-7/04/$20.00 © 2004 IEEE.**

developing complex SNNs that could display minimal cognitive functions or learn their behaviours through autonomous interactions with the environment. Implementations of SNNs are difficult as the hand design of SNNs that display a desired functionality is not a trivial task because of the highly non linear dynamics. Furthermore, the learning algorithms developed for SNNs are often restricted to very simple and application specific architectures. Artificial evolution through Genetic Algorithms (GAs) is therefore an interesting method to discover SNNs. In our previous work [6] we introduced an adaptive GA to evolve SNN controllers for robots.

1.2 Application

One of the important technological problems in secondary oil production is scale formation. The problem varies in severity depending on the composition of the water used during flooding. Scale deposits consisting either of Calcium Carbonate or Calcium and Barium Sulphate cause clogging in the pipes and damage to the pumping systems. The process of calcium carbonate formation is accompanied with proton release in the fluid and subsequent pH drop. Calcium Carbonate deposits tend to form around various nuclei of foreign material. Bulky and tenaciously adhering calcium carbonate deposits are formed in riser pipes used to control water levels in secondary oil recovery processes. Moreover these deposits are encountered in heater-treatment units, i.e. storage tanks in which water is heated to be used to raise the temperature of the produced fluids facilitating the breakdown of water and oil emulsions in order to achieve separation of the two fluids. The process of steam-flooding of high-viscosity oil reservoirs also involves heating water in tanks.

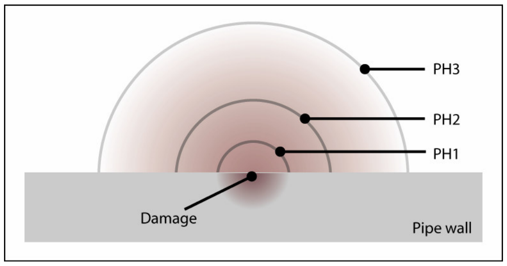

Figure 1 : Representation of the damage

Hardness leakage through the ion exchangers usually employed, in combination with the bicarbonates in water, result in the formation of Calcium Carbonate scale that causes a dramatic reduction of the heat transfer coefficient. As a consequence the rupture of the high pressure tubing has been reported. Investigations on the formation of scale deposits in industrial systems indicate that the slower the fluid velocity the more intense the scaling problems. This is due to the formation of insoluble salts like Calcium Carbonate [7]. As a consequence of the formation of scale, local pH drops as one moves away from the damage (see Figure 1).

1.3 Target Scenario

The target scenario consists of agents circulating in one compartment of the fluidic environment, continuously sensing for “fault signals”. The “fault signal” (i.e. the environmental stimulus that signals the presence of a faulty element) will be an altered chemical/physical property (i.e. pH/conductance). Using ISFET pH sensors, robots are able to perceive any “fault signals”, and start moving towards its source (i.e. the fault area) simultaneously releasing a quorum sense[1] signal, which ultimately “attracts” other robots. Modulation of the emission of the quorum sense signal is contingent upon the local value of the fault and quorum sense fields, but also critically depends on the path the individual robots have traced in the environment as this is reflected in the spiking history of the neurons that sense and actuate the generation of the quorum sense field. As quorum signals are summed more agents are eventually attracted towards the fault area; when a quorum signal threshold is exceeded a spatio-temporally ordered community is formed, which starts repairing the fault. When robots repair the fault the strength of the original environmental stimulus drops, quorum signals diminish and hence the community disperses.

The project’s main objectives are:

- Design and implement a generic modular agent architecture that can be mapped to the physical design of the tangible agents.
- Design and implement a Development Environment for rapid prototyping and evaluation of successive generations of robots.
- Design a life cycle model for robot development and a methodology to realize it.
- Develop hybrid robots which will consist of SNNs ported into FPGA hardware and the necessary software to interface with the simulated environment.
- Develop a robotic platform and a roadmap to its miniaturisation.

[1] Taken from biology, quorum sense is the phenomena that allow a bacterium "to know" the number of bacteria of certain species that are in its proximity based in the accumulation of signalling molecules.

- Implement a proof-of-concept test bench using bypass test tubing, to replicate aspects of fluidic systems in the priority application, in order to investigate in-situ behaviour of robots.

2 Development Methodology

The robot development life-cycle model consists of four stages: Specification, Implementation-Integration, Simulation, and Evaluation-Evolution. In Specification, a formal model capable of describing the topology and execution of SNN (computational component) as well as the sensor/actuator components has been defined. Based on this formal model, a designer is able to define robot controllers. This specification together with robot morphology is compiled and VHDL or simulator language instructions are produced. The generated code is then used by the Simulator embedding the software version of the controller into the virtual robot to test its performance in the virtual environment. The generated VHDL can also be compiled into FPGA boards that can then be tested in simulation in order to verify the execution of SNNs in hardware. The robot's performance is then evaluated according to measurement results and success criteria. Controllers are then evolved and a candidate specification is fed back into the cycle.

2.1 Development Environment

An Integrated Development Environment platform has been designed with the aim of producing a complete solution for SNN robot controllers' development, from the specification to hardware implementation, covering all intermediate stages such as simulation, evaluation and real-time monitoring of prototyped hardware. An overview of this environment, its modules as well as the inter-module communication is depicted in Figure 2.

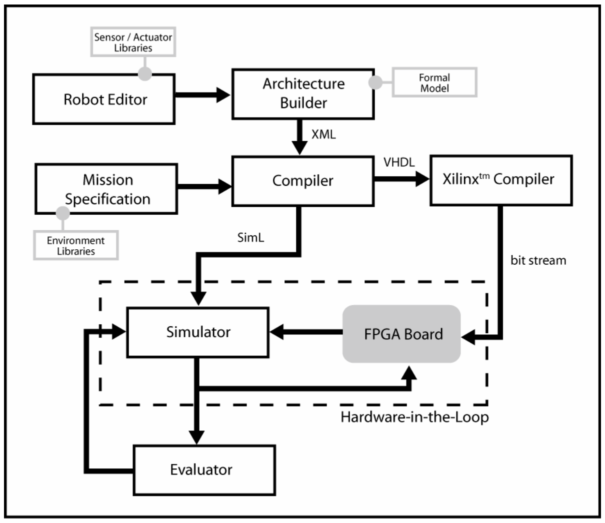

Figure 2 - Development Environment Architecture

The robot's morphology can be defined using the Robot Editor which utilises a library of predefined sensors and actuators. The Architecture Builder module enables users to specify robot controllers based on a formal specification according to a predefined morphology. The Compiler then takes the specification from the Architecture Builder and produces controllers in either VHDL which is used to program the FPGA boards or simulator code (SimL) integrating characteristics of the targeted environment defined in the Mission Specification module and gluing together the controller with simulated robots.

Hybrid societies (i.e. virtual robots and virtual robots with hardware controllers) are then tested in benchmark multi-agent task scenarios. During simulation, robots develop certain skills according to their context and experience and are thus no longer identical. In this heterogeneous society, the Simulator measures a number of success criteria and feeds them to the Evaluator module for further processing. The Evaluator assesses each robot's performance in the targeted benchmark scenarios which will entail the localization and repair of a failure in the tube walls. Their collective performance is evaluated and a new generation is specified with the use of evolutionary algorithms. Evaluation criteria include the time to detect a fault, time to amass the minimum number of robots required to act collectively towards a solution, number of sensors / actuators required per robot, etc.

2.2 Simulation and evaluation modules

The Simulator and the Evaluator allow the user to simulate, analyze, and evaluate prototyped robots. It plays a key role in the development process. It allows designers to learn more about the robot's behaviour or to investigate the potential of alternative architectures and researchers to probe the relationships between robot-controller architectures, behaviours and the environment.

Simulation supports robot development in different ways:

- Facilitating the development and evaluation of different configurations. Robot morphology can be tested under differing conditions before committing to a particular design.

- Visualizing the whole model allows researchers to concentrate on the general behaviour of the system and contribute to the discussion and conception of new ideas.

- Robot's defective behaviour can be recognized during simulation and it can be altered before the hardware realization is implemented.

- Simulation helps to identify variables that are essential for the development of accurate behaviours and that have not been taken into account.

What follows is a description of a preliminary implementation of the simulator. This model allows developers to easily test different robot architectures (i.e. defining different types of sensors, actuators, controllers and their relationships) without affecting the core components of the Simulator. The level of abstraction in the interfaces allows a different robot configuration to be benchmarked which is an essential feature for the experimentation on new prototypes. Interfaces to *Actuator* and *Sensor* classes have been kept as simple as possible. This allows the developer to define different types of sensors/actuators and their physical characteristics without the necessity of re-implementing other components or to be concerned about a particular configuration of the environment. (i.e. robot controllers' implementation is usually tightly coupled to sensors and actuators). Entities with a physical representation inherit most of the functionality from the SolidObject class whose position and orientation is updated every time step of simulation after applying environmental forces (i.e. gravity, drag and buoyancy, etc) and internal forces (i.e. accumulated actuators-force vectors). SolidObject objects can be grouped together in tree-form structures (i.e. parent-child relationships). This enables developers to define components and glue them together in new robot designs ready to be tested against environmental forces.

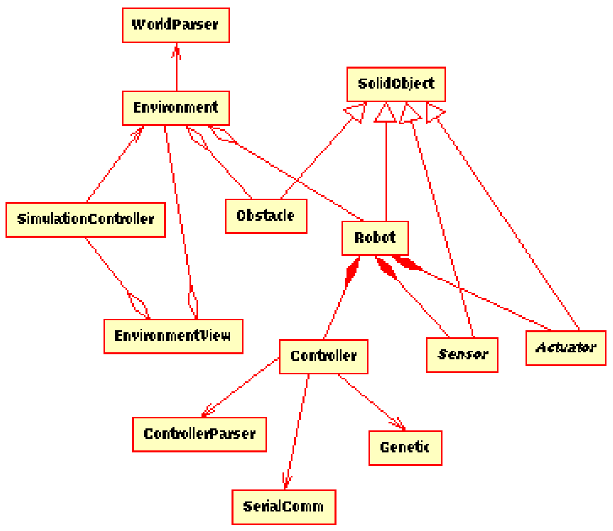

Figure 3 - Simulation and Evaluation modules

The SimulatorController is in charge of integrating forces defined in the environment and how robots are affected by them including algorithms for collision detection/response. It also manages the global timer and provides functions to control its performance throughout the simulation as well as generating log files to be used as a measuring and debugging resource. The Environment class holds information about all the entities participating in the simulation and acts as a unique entry point from which users can specify new configurations to be simulated. Environment is integrated with the Open Dynamics Engine (ODE)[8]. Based on Newtonian mechanics, the ODE handles masses of arbitrary geometry updating the environment in discrete time steps. The ODE provides support for collisions, joints (restrictions of relative motion), second order constraints, and friction.

The EnvironmentView class is in charge of rendering the graphics and the Graphical User Interface (GUI). Robot is a generic class whose realisations represent different types of robotic platforms. Implemented robots are controlled using custom implementations of the Controller class. In order to simplify experimentation while testing new types of sensors and actuators, users are required to implement abstract classes *Sensor* and *Actuator*. Developers need to write the code that emulates a specific sensor's functionality, obtaining information by querying the Environment class. In the same way, users can implement a particular type of actuator which will be able to alter the internal state of the Environment class or the Robot to which it belongs. The Controller class provides hooks to the algorithms that supply the functionality required to control robot's performance according to the SNN's formal specification. The SerialComm helper class provides serial communication functionality allowing the interfacing of simulated robots with hardware versions of SNN controllers. The Genetic class is in charge of assigning fitness to specific robot controller parameters, using these parameters to evolve the controllers using Genetic Algorithms and produce a number of possible parameter sets that allow the robots to reach certain goals.

ControllerParser is in charge of reading Simulation Language (SimL) configuration files. These are text files with an XML description of the SNN controller that is automatically generated by the Compiler (see Figure 2). Features described by these files are mapped into Controller SNN modules integrating them with the simulator. In the same way WorldParser takes files representing environment features (i.e. features dimension and location, fluid characteristics, etc) and these are mapped to Environment attributes.

3 Hardware implementation

3.1 Preliminary tests: Integrating FPGA based SNN controllers with prototyped 2D simulator

As a first attempt to integrate hardware controllers we evolved software SNNs within the real world, translated them into VHDL and then downloaded them into the prototyped hardware. Although the final platform for the robot is within 3D space and these experiments were carried out in a 2D space, this work represents an important investigation into how SNN solutions, regardless of the task they face, can be ported from software into FPGA boards that will be used in the final robotic platform. In order to compare SNN transferred into hardware with their software counterpart, the results from a series of wall following experiments were compared with three different scenarios:

- Scenario 1: Software implementations of the SNN running and evolved in a robot.
- Scenario 2: A hardware implementation of the same controller interfaced with the simulator.
- Scenario 3: The same hardware implementation running in the real robot.

The C code along with the evolved parameters of the wall following network trained in a real robot was converted into VHDL and downloaded onto a FPGA board. This process was made markedly easier by producing lookup tables for the FPGA which saved a great deal of computational work at run time. Much of the physical structure of the spikes in the Spike Response Model (SRM) was constant in terms of decay rates and delays. Because FPGA devices are more suitable for fixed point computation rather than floating point, look-up-tables were used to store the height of the spike at a particular time step, rather than dynamically computing the spike response with the necessary floating point arithmetic. Whilst this method involved a reduction in the accuracy of plotting the spike curve, the complexity of the lookup tables could be adjusted to meet the need of the specific navigational problem being solved. The FPGA was connected to the simulator and the robot using serial communications. Once the FPGA was connected to the simulator or secured to the robot, the experiments in Scenario 1 were repeated with the same robot starting point and layout of obstacles but using the FPGA as the controller rather than the software SNN. By comparing the log readings of scenarios 1 and 3 we intended to see what quality, if any, was lost by transferring the SNN software into hardware while comparing scenarios 1 and 2 allowed us to verify the accuracy of hardware in the loop simulation.

3.2 Component scaling

The size of the core components in the development system are shown in Figure 4 and by the end of the project miniaturization within the range of 10-18mm is planned focussing mainly on packaging and interconnections of various hardware modules as well as power management issues. The FPGA layer contains a Xilinx Spartan IIE Field Programmable Gate Array (FPGA). The Spartan IIE 1.8V FPGA family gives high performance, abundant logic resources, and a rich feature set. The device integrated into the 25mm^3 Printed Circuit Board (PCB) series is the XC2S300E-7FG256. This is a mid-range device with density of up to 300,000 system gates. Features include dedicated block RAM, distributed RAM, programmable I/O and DLLs (Delay-Locked Loops) for minimization of clock skew.

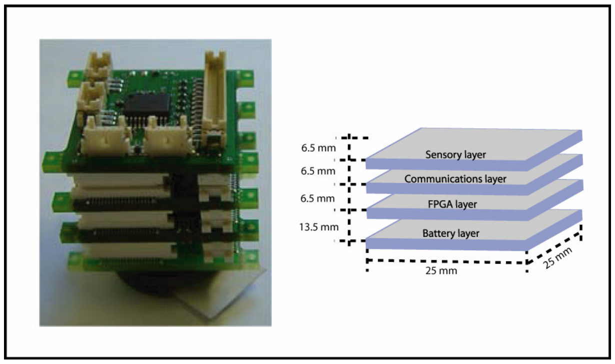

Figure 4 - 25mm^3 Core

The module features an on board 4 MHz crystal oscillator chosen to give a moderate processing rate while conserving power. 1.8V and 3.3V low drop out power supply regulators to provide the maximum module lifetime from a coin cell battery attachment for the core and LVTTL IO voltage requirements respectively. The module also features an on-board Flash serial EPROM such that the FPGA configuration memory is automatically downloaded on power-up. The stackable connector system used allows simple connectivity to other modules such as the RF module, coin-cell power supply and sensor modules. The current version of the 25mm Communication layer consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator and a modulator (Nrf2401 Single chip 2.4GHz Transceiver). Output power and frequency channels are easily programmable. Current consumption is very low, and a built-in Power Down mode makes power saving easily realizable. The module also features an on board antenna. The embedded microcontroller is based on the ATmega128L, an 8-bit microcontroller with 128K bytes in system programmable flash. This programmable transceiver has been designed to connect with a separate battery module and FPGA layer depending on the configuration required.

The Sensory layer is a wired communication and sensor interface to the FPGA module. The current version contains a dual channel RS232 transceiver, the Maxim MAX3224ECAB in a small SSOP20 package enabling wired serial communication with a PC for test purposes. The module also contains two TLC549CD Analogue to Digital (A to D) converters from Texas Instruments for interfacing analogue sensors to the FPGA. The module also allows interfacing to seven external sensors of the type which change their resistance according to the parameter being measured (e.g. light dependent resistors). The necessary conditioning circuitry for the sensors defined will be incorporated into a new version of this PCB. In the Battery layer, coin cells may be used to provide power to the 25mm^3 system. A PCB has been designed to directly interface coin cells using the stackable connector system. A range of options exists for choice of coin cell including support for 20mm and 24.5mm cells. Single and double coin cell holders are also available. A future revision of this module will provide power management circuitry to prolong the lifetime of the controller. The modules use a stackable connector system to make the electrical and mechanical interconnections between themselves. These high density interconnect have 0.5mm pitch and are available in range of interlayer spacing from 5mm to 8mm to allow for different component heights on the PCBs. The connectors facilitate an 80 pin general purpose bus and a 40 pin bus for configuration and data transfer between the modules. The RF transceiver also has a 20 pin connector for 4 low noise analogue input channels.

4 Conclusions

In this paper, we have introduced SOCIAL's development architecture for SNN based controller definition, simulation, evolution and testing. Inspired by biological neural networks, SNNs provide a fast processing system that provides tolerance to background noise and can be mapped in small programs and thus requires few logic operations and instructions to move around single bits. Thus large SNNs can be embedded in tiny and low power chips that can achieve complex tasks and behaviours. This make them appropriate control mechanisms for the micro-scale autonomous robots developed by our project as they can give a very good response when dealing with noisy, inaccessible environments whilst consuming little power.

A methodology for the development of controllers with these characteristics was outlined together with a description of the architecture of a simulation and evaluation environment which allows us to train controllers off line and download them into FPGA boards. Finally, we include a description of the modules implemented in a hardware prototype and consideration towards miniaturization.

4.1 Acknowledgement

We are pleased to acknowledge the funding support from the EU Future and Emerging Technology programme for the project entitled "Self-Organised Societies of Connectionist Intelligent Agents capable of Learning ", No IST-2001-38911. We are also pleased to acknowledge our project partners: the University of Patras, Greece; CTI, Greece and NMRC, Ireland.

References

[1] W. Maass, "Networks of spiking neurons: the third generation of neural network models", Proc. Australian Conference on Neural Networks, pp. 1-10, Apr. 1996.

[2] T. Lehmann and R. Woodburn, "Biologically-inspired on-chip learning in pulsed neural networks", *Analog Integrated Circuits and Signal Processing*, 2 ed., vol. 18. pp. 117-131, Feb. 1998

[3] D. Floreano and C. Mattiussi, "Evolution of spiking neural controllers for autonomous vision-based robots", Proc. International Symposium on Evolutionary Robotics (ER-2001), Berlin, pp. 38-61, Oct. 2001.

[4] M. A. Lewis, R. Etienne-Cummings, A. H. Cohen and M. Hartmann, "Toward biomorphic control using custom VLSI CPG chips", Proc. IEEE International Conference on Robotics and Automation, San Francisco, pp. 494-500, Apr. 2000.

[5] G. Indiveri, "Neuromorphic analog VLSI sensor for visual tracking: circuits and application examples", *IEEE Trans.on Circuits and Systems II*, 11 ed., vol. 46. pp. 1337-1347, Nov. 1999

[6] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan and G. Clarke, "Evolving spiking neural network controllers for autonomous robots", Proc. IEEE International Conference on Robots and Automation, New Orleans, pp. 4620-4626, Apr. 2004.

[7] J. C. Cowan and D. J. Weintritt, *Water-formed scale deposits*, Gulf Pub. Co, Huston Texas, 1976

[8] R. Smith, "Open Dynamics Engine", http://www.ode.org/, May 2004