Aquila: An Open-Source GPU-Accelerated Toolkit for Cognitive and Neuro-Robotics Research

Martin Peniak, Anthony Morse, Christopher Larcombe, Salomon Ramirez-Contla and Angelo Cangelosi

***Abstract*—This paper presents a novel open-source software application, Aquila, developed as a part of the ITALK and RobotDoC projects. The software provides many different tools and biologically-inspired models, useful for cognitive and developmental robotics research. Aquila addresses the need for high-performance robot control by adopting the latest parallel processing paradigm, based on the NVidia CUDA technology. The software philosophy, implementation, functionalities and performance are described together with three practical examples of selected modules.**

I. Introduction

RECENT approaches that attempt to understand the nature of cognition have shifted their focus from emphasising formal operations on abstracts symbols to a rather different approach where cognition is seen as an embodied or situated activity and therefore largely determined by the physical form of an embodied system [1][2]. Artificial intelligence, developmental psychology, neuroscience, and dynamical systems theory have directly inspired a completely novel approach called developmental robotics, which is a highly interdisciplinary subfield of robotics also known as epigenetic or epigenetic robotics [3][4].

Artificial cognitive systems based on the developmental robotics approach need to undergo an autonomous and gradual mental development from "infancy" to "adulthood". Interaction with their environments in an autonomous manner with little or no human intervention is necessary where no tasks or goals are pre-defined. This aids the process of achieving a good level of environmental openness where an artificial cognitive system is able to cope with different and previously unexpected environments. Another important aspect of the systems based on developmental robotics is that they are able to learn from their previous experience and use it to assist the acquisition of new skills. Developmental robotics attempts to understand how the control system's organisation of a single robot develops through various experiences over time.

Embodiment plays a significant part for achieving the goal to develop autonomous artificial embodied agents capable of acquiring complex behavioural, cognitive and linguistic skills through individual and social learning. For example, it has been demonstrated that research in action and language learning in natural and artificial cognitive systems can directly benefit from this approach, inspired by the developmental models and phenomena studied in children, allowing re-enactment of gradual process that have the potential to bootstrap various cognitive capabilities and integrate them into a unified interactive cognitive system [4][5]. The main theoretical hypothesis is based on the assumption that the parallel development of action, conceptualisation and social interactions permits the bootstrapping of language capabilities, which leads to the enhancement of cognitive development [6].

The developmental robotics approach to action and language learning is consistent with recent brain-inspired approaches to mental development since computational neuroscience considers the neural development constrains on embodiment, as well as on cognition [7][8][9]. It is not surprising that an overwhelming number of studies from various fields suggest that actions and language are closely integrated together. This highlights the importance of embodiment as well as supports the hypothesis of usage based language as opposed to classical explanations of language development that assumes an extensive language-specific cognitive hardwiring.

The modelling of the integration of various cognitive skills and modalities requires complex and computationally intensive algorithms running in parallel while controlling high-performance systems. The processing requirements are increasing with every added feature and it is not uncommon that at the end of the software development stage a particular system is unable to cope with fast-response robot-control tasks. This is very likely when a system requires applying filters to millions of pixels from a robot's cameras, running large-size neural networks with millions of synaptic connections and using multiple self-organising maps while controlling the robot in the real-time.

Around the year 2003, to overcome the energy consumption and heat-dissipation problems of standard PC processors, manufacturers started to produce computers with multiple cores. This had a strong impact on the software developer community [10]. In the meanwhile, manufacturers have been looking into new technologies that would increase the number of transistors per wafer. However, reducing these dimensions comes at a price since the current leakage becomes a problem.

Since 2003, the production of semiconductors has been

Martin Peniak, Anthony Morse, Christopher Larcombe, Salomon Ramirez-Contla and Angelo Cangelosi are with the Centre for Robotics and Neural System of the University of Plymouth, Drake Circus, Plymouth, PL4 8AA, UK (Tel.: +44-1752-586217; fax: +44-1752-586300; (email: {martin.peniak, anthony.morse, christopher.larcombe, salomon.ramirez-contla, a.cangelosi}@plymouth.ac.uk).

The authors would like to thank Mirokhin Ivan, a Russian software developer who dedicate his spare time to improve multiplatform portability of Aquila.

This work was supported by the EU Integrating Project - ITALK (214886) within the FP7 ICT programme - Cognitive Systems and Robotics.

divided into multicore and manycore design trajectories [11]. Manycore design aims to increase the processing power by increasing the number of cores in a processor. This number was doubling with each semiconductor process generation starting with dual-core chips and reaching hyper-threaded hexa-core systems. A manycore system is fundamentally different with regards to its design philosophy. While CPUs (Computer Processing Units) are optimised for the processing of sequential code and feature sophisticated control logic and large cache memories, the GPU (Graphic Processing Units) design philosophy emerged from the fast growing video industry where massive numbers of floating point operations are required to render every single frame. As a result, a GPU chip has most of its area dedicated to processing of the floating point operations and features only tiny cache memories.

In 2006, NVidia released GeForce 8800 GPU, which was capable of mapping separate programmable graphics processes to an array of GPUs, which paved the way to first general purpose computing using parallel GPU processors. GPGPU was an intermediate step where graphics card programmers had to use the OpenGL or DirectX API to implement their programs. Using the GPGPU technique many different applications have achieved dramatic speed improvements. For example, Kruger and Westermann developed a framework for solving linear algebra [12], Harris and colleagues designed a cloud dynamics simulation based on partial differential equations [13], Rodrigues and colleagues implemented molecular dynamics simulation [14] and Nyland and colleagues N-body simulation [15].

More recent GPU developments, based on the NVidia Tesla GPU architecture, provide clusters of GPUs used as individual programmable processors and allow more efficient parallel processing tools. The CUDA (Compute Unified Device Architecture) programming tool has been designed on purpose to support mutual CPU/GPU application execution.

Parallel computing using CUDA and GPU cards is being increasingly taken up by industry and academies. Many commercial and research applications have migrated from using solely standard CPU processors to a collaborative CPU/GPU use where each architecture does what is best at. In general, most of these applications can achieve tremendous speed-ups in performance, which is anything between 1.3x to 2,600x [16]. Since quantum computing is still in its infancy and CPUs are approaching the processing limits constrained by the physical laws, it seems that parallel computing using GPU devices is the next paradigm that is yet to become fully recognised and widely used.

CUDA has been employed in a wide variety of applications, however, only a handful of these have any relevance for the cognitive and neuro-robotics domains. Only few studies have to date applied CUDA to neural networks and visual processing (e.g. [17][18]), as this field requires further investigation in applying the CUDA technology to neural computation and robotics research.

This paper presents a novel software tool, named Aquila, suitable for GPU-based cognitive and neuro-robotics applications and that makes use of the latest parallel processing paradigm based on the CUDA technology. This software has been developed as a part of the ITALK and RobotDoC projects as well as the open-source project on the iCub humanoid robot[1]. In the next sections we first provide an overview of Aquila's philosophy, its software implementation and its functionalities and performance. We then discuss three different examples on the use of Aquila for various cognitive robotics studies.

II. Overview

Aquila is an open-source project that was inspired by the recent advancements in supercomputing making use of GPU enabled devices for humanoid robotics research with the iCub robot platform. The development of Aquila was driven by the increasing need for an open-source high-performance modular software that would provide not only a convenient interaction with the iCub humanoid robot and its simulator (see section II-A) but also for the design of multiple tools and bio-inspired models (see section II-B) that are useful for cognitive robotics research.

The source code that implements functions running on CPUs is mostly written in C++ programming language. Parallel functions running on GPU cards are written in CUDA-C, which is an extension to C programming language providing access to the virtual instruction set and memory in CUDA-capable GPU cards. Aquila is based on freely available multi-platform libraries and currently compiles on Linux and Mac OS X operating systems. The graphical user interface (GUI) is clearly structured and divided into several modules that can be easily changed using tabs at the top of the application (see Figure 1). Fast and efficient interaction with the iCub is facilitated through the integration of many useful modules in a simple and intuitive GUI based on Qt libraries. The communication with the iCub humanoid robot or its simulator is provided by the YARP protocol [19][20]. Image processing is implemented through OpenCV libraries[2] as well as native functions. The subsequent rendering to the GUI is realised through the native use of OpenGL where the incoming images are mapped into textures and sent to Qt's QGLWidget. 2D and 3D visualisations are based on QwtPlot and QwtPlot3D libraries respectively[3]. Some of the modules use speech recognition, which is provided by the Julius engine.

A. The iCub Humanoid Robot and Simulator

The iCub is a small humanoid robot that is approximately 105cm high, weights around 20.3kg and its design was inspired by the embodied cognition hypothesis. This unique robotic platform with 53 degrees of freedom (12 for the legs, 3 for the torso, 32 for the arms and six for the head) was designed by the RobotCub Consortium [21], which

[1] www.italkproject.org, www.robotdoc.org, www.icub.org

[2] www.opencv.willowgarage.com - OpenCV supports CUDA architecture

[3] www.qwt.sourceforge.net, www.qwtplot3d.sourceforge.net

involves several European universities. iCub is now widely used by other cognitive robotics projects such ITALK and RobotDoc. The iCub project strictly follows the open-source philosophy, and therefore its hardware design, software as well as documentation are released under general public license (GPL).

Tikhanoff et al. have developed an open-source simulated model of the iCub platform [22]. This simulator has been widely adopted as a functional tool within the developmental robotics community[4], as it allows researchers to develop, test and evaluate their models and theories without requiring access to a physical robot.

B. Modules

Aquila currently provides a number of functional modules for robotics research most of which support CPU and GPU execution. Below is a brief description of each module and its functionalities. For a more detailed information about their implementation and use see Aquila manual (section V).

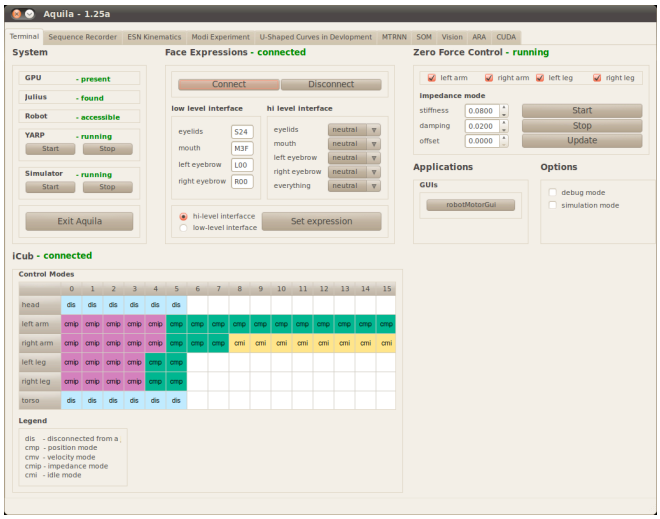

Fig. 1. Top part of the image shows multiple tabs that are used for switching between different modules. All the other controls that are below the tabs are part of the *Terminal* module, which is the initial default module. The top part of the image shows system information, initialised zero-force control and face expressions interfaces. The bottom part provides information about iCub's joint control modes that are currently used

Terminal - provides relevant information about the external systems that Aquila interacts with, displays status messages about the presence of GPU devices, Julius speech recognition system[5], YARP server, the iCub robot and its simulator. Terminal provides way to set face expressions of the iCub either by using low or hi-level module and regardless whether a user is connected to the real robot or the simulator. This module integrates with the force control system developed by Italian Institute of Technology [23] and is able to start/stop force control mode on iCub as well as change several parameters such as joints stiffness, damping and offsets. In addition, Terminal displays colour visualisation of the iCub's joint control modes (e.g. position, velocity, impedance), provides different options such as lunching external applications or modifying Aquila's global settings.

Sequence Recorder - provides a simple and convenient way of recording, saving and replaying motor sequences. This module uses the *Zero Force Control* interface to enable running iCub robot in a compliant mode while recording sequences.

ESN Kinematics - implements echo state networks.

Modi Experiment - runs attention system, speech module and provides a simple way to demonstrate the modi experiment (see section III-B).

U-Shaped Curves in Development - runs and visualises multiple self-organising maps that are connected by Hebbian weights. This biologically inspired model provides one explanation of why children are better at recognising phonemes when they are 8 months old. Their performance gets worse before it improves again later [24].

Multiple Timescales Recurrent Neural Network - trains complex continuous multiple time recurrent neural networks (MTRNN) using the backpropagation through time algorithm. The module executes neural network control systems on the iCub robot or the simulator as well as visualises many parameters (see section III-A).

Self-Organising Maps - trains, saves, loads and visualises self-organising maps, which are very useful and powerful computational tools capable of preserving the topological relations in multi-dimensional data.

Abstraction-Reaction Accumulator - explores a novel approach to synthesising complex adaptive behaviour and non-task-specific control systems.

Vision - renders video streams from the iCub robot, its simulator as well as cameras connected to a host machine. It allows users to take screenshots, record individual frames and videos, maximise and minimise individual viewports and apply various image processing filters.

Simulator - provides various interfaces for the iCub simulator such as video projection from remote and local video files, cameras connected to a host machine or from a YARP port. The module also implements an object management system, which allows users to create and delete various objects (e.g. box, cylinder, sphere, 3D model), modify their properties. save objects into XML files and load them back to the simulator.

CUDA - displays relevant information about CUDA devices found on the system during Aquila initialisation. This information can be very useful in aiding the process of setting GPU execution parameters (e.g. number of threads) that have dramatic influence on the performance.

C. Performance Benchmarking

A performance benchmarking study was carried out in order to demonstrate the potential of using GPU devices for iCub research. It is important to note that these are only preliminary comparisons of unoptimised GPU and CPU code. The performance can be further increased for both architectures, however, the present demonstration of the GPU-CPU differences in performance is to give a general

[4]For example in the following projects: RobotCub, ITALK, Poeticon, Chris, RobotDoc, Roboskin, Amarsi, IM-CLeVeR, emorph, ROSSI

[5]www.julius.sourceforge.jp

indication of the extent to which GPUs outperforms CPUs when code is not extensively optimised.

The test was based on simulation of the multiple timescales recurrent neural network (MTRNN) system benchmark using the backpropagation through time (BPTT) algorithm (Table I), on single forward pass through the network (Table II) as well as for self-organising maps (SOM) training (Table III).

These tests were performed on MTRNN and SOM modules using different parameters to show how both architectures scale when the amount of processing increases.

Fig. 2. system setup used for benchmarking consisting of 8 x 2.67GHz hyperthreaded processors, 16GB RAM, 1 x GeForce GTX470 and 3 x Tesla c1060 GPU cards

number of neurons	CPU	GPU	speedup
336	11.14min	0.90min	12.31x
1104	171.41min	3.83min	44.64x

TABLE I
MTRNN BACKPROPAGATION THROUGH-TIME TIMES

number of neurons	CPU	GPU	speedup
336	30ms	0.4ms	75x
1104	370ms	1ms	370x
4176	5022ms	5ms	1004x

TABLE II
MTRNN FORWARD PASS TIMES

number of neurons	CPU	GPU	speedup
64	0.22sec	0.20sec	1.14x
256	1.75sec	0.45sec	3.88x
1024	14.2sec	1.15sec	12.36x
4096	126.17sec	3.20sec	39.37x

TABLE III
SOM TRAINING TIMES

The benchmarking results of unoptimised code show that achieved speedups vary from 1.14x to 1004x. It is clear that the performance of GPU devices stands out soon after a large amount of data needs to be processed. A good example of such scaling can be seen in the MTRNN forward pass times that vary from 75x to 1004x speedups.

III. Robotics Experiments

This section provides brief descriptions of selected modules, their underlying systems as well as some preliminary results of robotics experiments. Detailed descriptions of these experiments is beyond the focus of this paper. However, the full details can be found in the referenced papers.

A. Multiple Timescales Recurrent Neural Network

Humans are able to acquire many skilled behaviours during their life-times. Learning complex behaviours is achieved through a constant repetition of the same movements over and over while certain components are segmented into reusable elements known as motor primitives. These motor primitives are then flexibly reused and dynamically integrated into novel sequences of actions.

For example, the action of lifting an object can be broken down into a combination of multiple motor primitives. Some motor primitives would be responsible for reaching the object, some for grasping it and some for lifting it. These primitives are represented in a general manner and should therefore be applicable to objects with different properties. This capacity is known as generalisation, which also refers to the ability to acquire motor tasks by different ways. In addition, one might want to reach for the object and throw it away instead of lifting it up. Therefore, these motor primitives need to be flexible in terms of their order within a particular action sequence. The amount of combinations of motor primitives grows exponentially with their number and the ability to exploit this repertoire of possible combinations of multiple motor primitives is known as compositionality.

The hierarchically organised human motor control system is known to have the motor primitives implemented as low as at the spinal cord level whereas high-level planning and execution of motor actions takes place in the primary motor cortex (area M1). The human brain implements this hierarchy by exploitation of muscle synergies and parallel controllers. These have various degrees of complexity and sophistication that are able to address both the global aspects of the motor tasks as well as fine-tune control necessary for the tool use [25].

Models such as MOSAIC [26] or mixture of multiple recurrent neural network systems [27] have implemented functional hierarchies via explicit hierarchical structures. The motor primitives are represented through local low-level modules, whereas the higher-level modules allow the recombinination of these primitives using extra mechanisms such as gate selection systems. These systems, based on predefined hierarchical structures, are appealing because of their potential benefits. For example, the learning of one module does not interfere with the learning of other modules

and it would also seem that by adding extra low-level modules the number of acquirable motor primitives would increase as well. However, it has been demonstrated that the similarities between various sensorimotor sequences result in competition between the modules that represent them. This leads to a conflict between generalisation and segmentation, since generalisation requires the representation of motor primitives through many similar patterns present in the same module whereas different primitives need to be represented in different modules to achieve a good segmentation of sensorimotor patterns. Because of the conflict that arises when there is overlap between different sensorimotor sequences, it is not possible to increase the number of motor primitives by simply adding extra low-level modules [28]. The learning of motor primitives (low-level modules) and sequences of these primitives (hi-level modules) need to be explicitly separated through subgoals [29][27].

Yamashita's multiple timescales recurrent neural network model [30] attempts to overcome the generalisation-segmentation problem through the realisation of functional hierarchy that is neither based on the separate modules nor on structural hierarchy, but rather on multiple timescales of neural activities that seem to be responsible for the process of motor skills acquisition and adaptation as well as perceptual auditory differences between formant transition and syllable level (e.g. [31][32][33]).

Aquila implements the MTRNN model together with the backpropagation through time (BPTT) algorithm for both CPU and GPU architectures. As it is demonstrated in section II-C, GPU significantly improves the speed of the MTRNN activation as well as of the training algorithm.

Multiple timescales recurrent neural network model can be seen as an extension the continuous time recurrent neural network (CTRNN), which was successfully used for producing sensorimotor sequences [34][35][36]. The MTRNN used in the preliminary experiments [37] consists of input-output neurons and context neurons that have different decay rates. The input-ouput neurons receive sensory inputs from a self-organising map that transform multidimensional vectors from the iCub robot into topological maps that directly set neural activations on this layer. Context neurons are divided into two categories (fast and slow neurons) where each category represents different timescales characterised by decay rate of neurons. In this study the MTRNN was fully connected except for no presence of direct connection between input-output neurons and 'slow' neurons.

In the initial preliminary study the MTRNN system was required to learn 8 different sequences of actions (slide left and right, lift up, left and right, swing, push and pull). At the end of the training, the learned neural network was tested on the iCub in the same setup as during the tutoring part. The MTRNN system was found to be able to replicate all the eight sequences while successfully manipulating with the object.

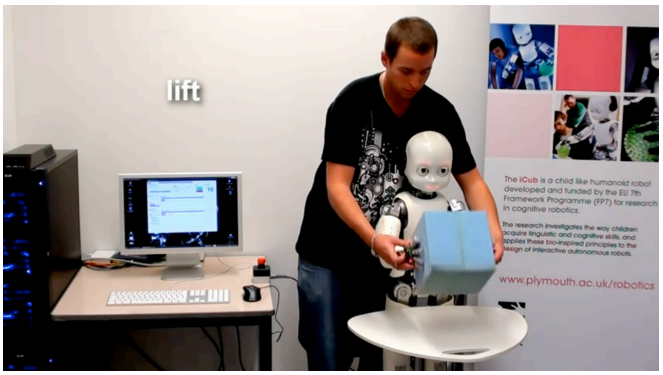

Fig. 3. tutoring the iCub robot while recording the sensorimotor sequences using the *Sequence Recorder* module of Aqula

B. 'Modi' Experiment

Our perception of continuous contact with a rich visual world laid out in front of us is somewhat misleading. In fact our actual sensory input is highly impoverished. Visual acuity for example is focused on an area the size of a thumb nail at arms length. Sensorimotor theories suggest that this rich perception is constructed from knowledge of the sensory consequences of performing various actions, thus you can perceive a chair in the periphery of your vision because you can predict that if you look over there you will see the chair. Similarly identifying objects is not so much about processing static images, but rather comes from identifying a profile of manipulations and their consequences in the dynamics of interaction. Such embodiment centric accounts of perception are supported by a large number of psychology experiments exposing various bodily biases in categorisation.

Using developmental robotics experiment it is possible to model these experiments both to develop the perceptual skills of the iCub robot, and to further understand our own categorisation abilities and our own bodily biases in perception [38][39].

In a series of experiments conducted by Linda Smith and Larissa Samuelson [40] children between 18 and 24 months of age are repeatedly shown two different objects in turn, one consistently presented on the left, and the other consistently presented on the right. After several presentations of the objects, the childs attention is drawn to one side or the other and the linguistic label 'modi' is presented in the absence of either object. Finally the children are presented with both objects in a new location and asked to find the 'modi'. Not surprisingly 71% of the children select the spatially correlated object.

In a follow up experiment following the same basic procedure one group of children is presented with only a single object which is labeled while in sight, the other group are repeatedly presented with a consistent spatial relationship until finally an object is labeled while in sight but in the wrong spatial location. In the control group 80% correctly pick the labeled object while in the spatial competition group the majority of 60% select the spatially linked object rather

than the object that was actually labeled.

In both experiments changes in posture from sitting to standing eradicate the effect, while other visual or auditory distracters do not. This is strong evidence challenging the hypothesis that names are associated to the thing being attended at the time they are heard.

Our model consists of a number of self-organising maps capturing variations in visual, auditory, and body posture input. These maps are then linked together via the body posture map (acting as a hub) in real time based on the experiences of the robot. With the addition of motion detection in the periphery of the robots vision, causing the robot to look at moving objects or changes in the scene, we are able to replicate the psychology experiments using the robot.

The *'modi'* module in Aquila gives access to the programs used to replicate these psychology experiments with the iCub robot and provides algorithms for motion detection, eye saccades and object tracking. Speech recognition is provided as is a key-press terminal connection should you wish to use your own speech recognisers. The specific experiment can be performed by following the 6 steps outlined below but the resulting behaviour is not limited to this sequence alone.

1) Object A is presented to the right
2) Object B is presented to the left
3) Steps 1 and 2 are repeated
4) The robots attention is drawn to the right with no objects present and the word MODI is spoken
5) Steps 1 and 2 are repeated again
6) Both objects are placed in a new location and the robot is asked where is the modi

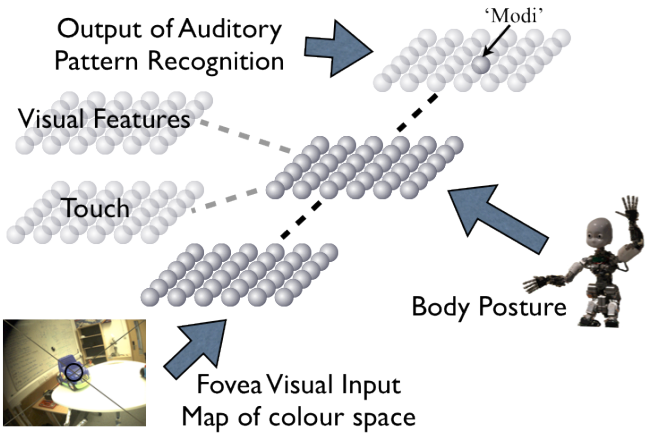

Fig. 4. The general architecture of the model. SOMs are used to map the color space, the body posture, and the word space. These maps are then linked using Hebbian learning with the body posture map acting as a central hub. The model can easily be extended to include other features such as visual and touch information in additional SOMs

C. Abstraction-Reaction Accumulator

The Abstraction-Reaction Accumulator (ARA) is an experimental adaptive control system, inspired by early cybernetic work [41]. The model explores a novel approach to synthesising complex adaptive behaviour and non-task-specific control systems. The system is structured such that a growing repertoire of adaptive behaviours is observed to emerge when coupled to an appropriate environment. Through a gradual process of cumulative learning, behavioural reactions are associated with perceptual abstractions. The current Aquila implementation is specific to the embodiment of the iCub humanoid robot: abstractions are defined as low-dimensional states dependent on high-dimensional sensory data obtained from the physical or simulated iCub, such as joint encoder values or eye camera images; reactions are defined as states in the iCub robot that determine or describe observable behaviours, such as postures, joint velocities or joint positions. A shared vision module is used to obtain the raw camera images, while another shared module (iCubControl) is used to obtain proprioceptive joint-position data, and to move individual joints on the iCub robot.

Intermittent feedback from the environment, provided through the Aquila GUI, modulates the parameters in the system such that certain reactions (components of overt behaviour) are more likely to occur in the presence of certain abstractions (sensory states). This feedback simulates perturbation to essential variables, which quantify the appropriateness of the state of specific groups of joints on the robot, referred to as synergies. Each multidimensional synergy has its own essential variable, and can take one of several different states at any one time. The number of syneriges and a mapping between each joint and synergy is also specified by the user in the GUI. Any joint on the robot can be mapped and controlled by the ARA. When a perturbation to an essential variable occurs, a parameter (dimension of variation) corresponding to the present reaction and present state of abstraction will be modulated, such that future disturbance to that essential variable is minimised. For example, in a particular sensorimotor state, where a red ball is held above the 'closed' right hand of the robot, a tutor may perturb an essential variable corresponding to the synergy 'right hand' (a set of joints), positively or negatively, causing the stability of a relevant set of parameters in the system to change. As a result, the state 'closed' of the synergy 'right hand' will be more or less likely to be observed in future, depending on the polarity of the perturbation, if and when the sensory situation recurs.

The ARA module provides many configuration settings (see Figure 5) as well as visualisations of the continuously changing variables, allowing users to observe the state of the system in real-time (see Figure 6).

IV. Conclusion and Future Work

We have briefly mentioned the motivation for the development of Aquila, described its modules and showed few example applications of their practical use. The work described here has the potential of being used not only for the ongoing research projects on the iCub robot, but in general as a general, open-source tool for cognitive and neuro-robotics, and embodied neural computation research in general. The availability of various modules implementing tools for neural network simulations (e.g. Kohonen maps, Hebbian learning, feedforward ad recurrent neural networks,

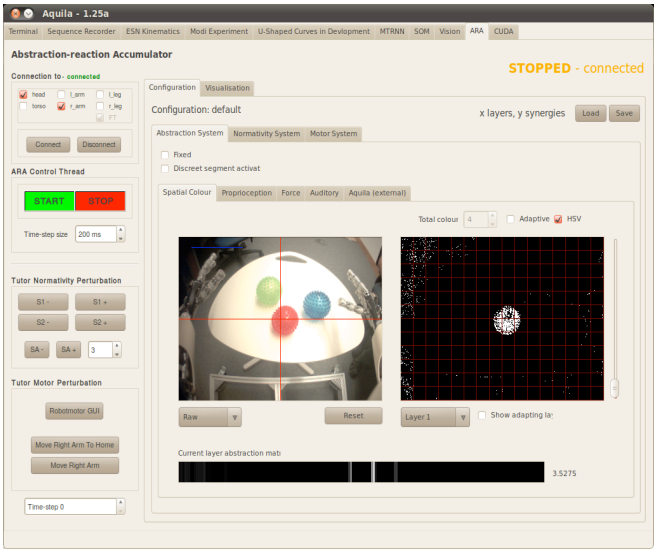

Fig. 5. settings that determine the behaviour of coupled sub-systems

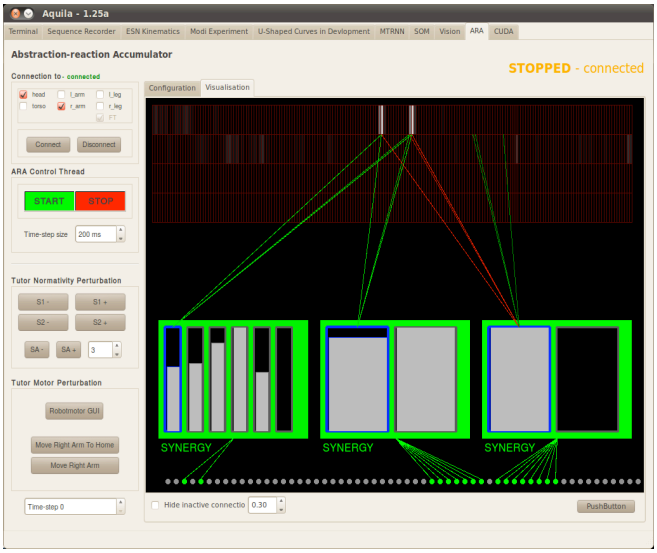

Fig. 6. real-time visualisation of the system state

MTRNNs), and the availability of linking of these software tools with the iCub simulator can also allow researchers with no direct access to the iCub physical robot platform to design novel experiments on neural and cognitive modelling using embodied sensorimotor agents.

Despite the various modules and features already present, Aquila is still in early development stages. The reusability of components allows for rapid development of new modules. However a lot of work still needs to be done with regards to programming of individual classes that need improvements and more testing. Many new features will be made available as we proceed with our research and optimisation of the existing systems, which will improve the stability of the application.

Current work focuses on porting Aquila to Windows, fixing known bugs, improving interface, extending the MTRNN module with vision and language systems to allow experiments of action and language acquisition. New module, available in the next version, will facilitate investigation of spatial representation with the focus on peripersonal space.

V. Software Repository and User Manual

Aquila can be downloaded directly from SourceForge or iTalk Project software repositories. The project page is on SourceForge (*http://sourceforge.net/projects/aquila/*) where new developers can joint our team.

The user manual that provides detailed module descriptions and installation instructions is available from this link: *http://dl.dropbox.com/u/81820/Software/Aquila/Aquila.pdf*

References

[1] A. Clark and D. Chalmers, "The extended mind," *Analysis*, vol. 58, no. 1, pp. 7–19, 1998.

[2] V. Gallese and G. Lakoff, "The brain's concepts: The role of the sensory-motor system in reason and language," *Cognitive Neuropsychology*, vol. 22, pp. 455–479, 2005.

[3] M. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi, "Cognitive developmental robotics as a new paradigm for the design of humanoid robots," *Robotics and Autonomous Systems*, vol. 37, pp. 185–193, 2001.

[4] M. Lungarela, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: A survey," *Connection Science*, vol. 15, no. 4, pp. 151–190, 2003.

[5] A. Cangelosi and T. Riga, "An embodied model for sensorimotor grounding and grounding transfer. experiments with epigenetic robots," *Cognitive Science*, vol. 4, pp. 637–689, 2006.

[6] A. Cangelosi, G. Metta, G. Sagerer, S. Nolfi, C. Nehaniv, K. Fischer, J. Tani, T. Belpaeme, G. Sandini, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders, and A. Zeschel, "Integration of action and language knowledge: A roadmap for developmental robotics," *IEEE Transactions on Autonomous Mental Development*, in press.

[7] J. Weng, "On developmental mental architectures," *Neurocomputing*, vol. 70, no. 13-15, pp. 2303–2323, 2007.

[8] D. Mareschal, M. Johnson, S. Sirios, M. Spratling, M. S. C. Thomas, and G. Westermann, *Neuroconstructivism Volume 1: How the brain constructs cognition*. Oxford University Press, 2007.

[9] G. Westermann, S. Sirois, T. R. Shultz, and D. Mareschal, "Modeling developmental cognitive neuroscience," *Trends in Cognitive Sciences*, vol. 10, no. 5, pp. 227–233, 2006.

[10] H. Sutter and J. Larus, "Software and th concurrency revolution," *ACM Queue*, vol. 3, no. 7, pp. 54–62, 2005.

[11] W. W. Hwu, K. Keutzer, and T. Mattson, "The concurrency challenge," *IEEE Design and Test of Computers*, pp. 312–320, 2008.

[12] J. Kruger and . R. Westermann, "Linear algebra operators for gpu implementation of numerical algorithms," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 908–916, 2003.

[13] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra, "Simulation of cloud dynamics on graph-

ics hardware," in *ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 2003.

[14] C. I. Rodrigues, D. J. Hardy, J. E. Stone, K. Schulten, and W. M. W. Hwu, "Gpu acceleration of cutoff pair potentials for molecular modeling applications," in *Conference on Computing Frontiers*, 2008.

[15] L. Nyland, M. Harris, and J. Prins, *GPU Gems 3*. Addison Wesley, 2007, ch. Fast N-Body simulation with CUDA, pp. 677–795.

[16] NVidia, "Cuda community showcase," December 2010. [Online]. Available: www.nvidia.com/object/cuda_apps_flash_new.html

[17] H. Jang, A. Park, and K. Jung, "Neural network implementation using cuda and openmp," in *2008 Digital Image Computing: Techniques and Applications*, 2008.

[18] S. Oh and K. Jung, "View-point insensitive human pose recognition using neural network and cuda," *World Academy of Science, Engineering and Technology*, vol. 60, 2009.

[19] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 29–45, 2008.

[20] G. Metta, P. Fitzpatrick, and L. Natale, "Yarp: Yet another robot platform," *International Journal on Advanced Robotics Systems*, vol. 3, no. 1, pp. 43–48, 2006.

[21] G. Metta, D. Vernon, L. Natale, F. Nori, and G. Sandini, "The icub humanoid robot: an open platform for research in embodied cognition," in *IEEE Workshop on Performance Metrics for Intelligent Systems*, 2008.

[22] V. Tikhanoff, P. Fitzpatrick, F. Nori, L. Natale, G. Metta, and A. Cangelosi, "The icub humanoid robot simulator," in *International Conference on Intel ligent RObots and Systems IROS*, Nice, France, 2008.

[23] M. Fumagalli, M. Randazzo, F. Nori, L. Natale, G. Metta, and G. Sandini, "Exploiting proximal f/t measurements for the icub active compliance," in *EEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1870–1876.

[24] A. F. Morse, T. Belpaeme, A. Cangeloci, and C. AmpatzisFloccia, "Modelling u shaped performance curves in ongoing development," submitted to the Cognitive Science Conference 2011.

[25] G. Rizzolatti and G. Luppino, "The cortical motor system," *Neuron*, vol. 31, pp. 889–901, 2001.

[26] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, pp. 1317–1329, 1998.

[27] J. Tani and S. Nolfi, "Learning to perceive the world as articulated: an approach for hierarchical learning in sensory–motor systems," *Neural Networks*, pp. 1131–1141, 1999.

[28] J. Tani, R. Nishimoto, J. Namikawa, and M. Ito, "Codevelopmental learning between human and humanoid robot using a dynamic neural-network model," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 38, pp. 43–59, 2008.

[29] J. Tani, R. Nishimoto, and R. Paine, "Achieving "organic compositionality" through self-organization: reviews on brain-inspired robotics experiments," *Neural Networks*, vol. 21, pp. 584–603, 2008.

[30] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment," *PLoS Computational Biology*, vol. 4, no. 11, 2008.

[31] R. Huys, A. Daffertshofer, and P. J. Beek, "Multiple time scales and multiform dynamics in learning to juggle," *Motor Control*, vol. 8, pp. 188–212, 2004.

[32] F. Varela, J. P. Lachaux, E. Rodriguez, and J. Martinerie, "The brainweb: phase synchronization and large-scale integration," *Nature Reviews Neuroscience*, vol. 2, pp. 229–239, 2001.

[33] D. Poeppel, W. J. Idsardi, and V. van Wassenhove, "Speech perception at the interface of neurobiology and linguistics," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 368, pp. 1071–1086, 2008.

[34] Y. Doya and S. Yoshizawa, "Adaptive neural oscillator using continuous-time back-propagation learning," *Neural Networks*, vol. 2, pp. 375–386, 1989.

[35] R. Nishimoto, J. Namikawa, and J. Tani, "Learning multiple goal-directed actions through self-organization of a dynamic neural network model: A humanoid robot experiment," *Adaptive Behavior*, vol. 16, no. 2-3, pp. 166–181, 2008.

[36] E. Tuci, "An investigation of the evolutionary origin of reciprocal communication using simulated autonomous agents," *Biological Cybernetics*, vol. 3, pp. 183–199, 2009.

[37] M. Peniak, D.Marocco, J. Tani, Y. Yamashita, K. Fisher, and A. Cangelosi, "Multiple time scales recurrent neural network for complex action acquisition," submitted to ICDL2011.

[38] A. F. Morse, T. Belpaeme, A. Cangelosi, and L. B. Smith, "Thinking with your body: Modelling spatial biases in categorization using a real humanoid robot." in *Proceedings of the Cognitive Science Conference 2010*, 2010.

[39] A. F. Morse, J. DeGreeff, T. Belpeame, and A. Cangelosi, "Epigenetic robotics architecture (era)," . *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 325–339, 2010.

[40] L. B. Smith and L. Samuelson, *Thinking Through Space: Spatial Foundations of Language and Cognition*. Oxford, United Kingdom: Oxford University Press, 2010, ch. Objects in Space and Mind: From Reaching to Words.

[41] W. R. Ashby, *Design for a Brain*. London: Chapman Hall, 1952.