Diverse, Noisy and Parallel: a New Spiking Neural Network Approach for Humanoid Robot Control

R. de Azambuja*†, A. Cangelosi *, S. V. Adams*

*School of Computing, Electronics and Mathematics
University of Plymouth
Plymouth, Devon, United Kingdom

†CAPES Foundation
Ministry of Education of Brazil
Brasilia, DF 70040-020, Brazil

Email: ricardo.azambuja@ieee.org, {a.cangelosi, samantha.adams}@plymouth.ac.uk

***Abstract*—How exactly our brain works is still an open question, but one thing seems to be clear: biological neural systems are computationally powerful, robust and noisy. Using the Reservoir Computing paradigm based on Spiking Neural Networks, also known as Liquid State Machines, we present results from a novel approach where diverse and noisy parallel reservoirs, totalling 3,000 modelled neurons, work together receiving the same averaged feedback. Inspired by the ideas of action learning and embodiment we use the safe and flexible industrial robot BAXTER in our experiments. The robot was taught to draw three different 2D shapes on top of a desk using a total of four joints. Together with the parallel approach, the same basic system was implemented in a serial way to compare it with our new method. The results show our parallel approach enables BAXTER to produce the trajectories to draw the learned shapes more accurately than the traditional serial one.**

***Index Terms*—spiking neural networks, liquid state machines, reservoir computing, humanoid robots, BAXTER, V-REP, parallel processing.**

I. Introduction

The use of a robot that resembles the human body and behaves alike - a humanoid robot - is often proposed as one possible solution to facilitate the deployment of such machines among us. However, humanoid systems suffer from the Bernstein's degree of freedom problem as their redundancy creates multiples ways to solve a task and it is still not clearly known how a complex redundant biological body is controlled by the nervous system [1].

Following the neurorobotic outlook [2], a good amount of inspiration for solving those problems comes from biology. As natural stimuli are made of spatio-temporal patterns and cortical neurons are naturally sensitive to those, any model of cortical processing needs to be able to deal with this information [3] and Spiking Neural Networks (SNN) have this ability [4]. Despite several works published in the domain of Artificial Neural Networks (ANN), comparatively few studies can be found in the literature addressing humanoid robot control based on SNN [5]–[8].

Liquid State Machines (LSM), introduced in [9], are recurrent SNN where an external output layer called *readout* is the only part of the network that is subject to a learning process. Therefore the great attractiveness is the fact that *readout* learning can be as simple as a linear regression. Some principles of LSM can be found in *in vitro* neural networks according to [10] and parameters applied to the neuron model, connection probability and short-term plasticity are derived from studies with the rat cortex [11]. Inspired by the results from [12], the LSM could be seen as a possible model to implement a neural arm controller. Any dynamical system, in addition to Turing machines, could be simulated by an LSM with the use of appropriate feedback [13]. According to [14], it is also possible to emulate complex, non-linear computations with the use of a simple linear regression when Reservoir Computing methods like an LSM are used. The use of an LSM to control a very simple planar 2 degrees of freedom simulated arm was already implemented in [15]. Apart from these works, a search in the literature could not provide any further studies with implementations of robot arm controllers using the LSM framework.

This paper presents the experimental results of a novel humanoid robot control framework based on parallel, diverse (each liquid was randomly generated) and noisy (using random injected currents and initial membrane values), sets of biologically inspired LSM. Some of the motivations to apply the LSM model are related to the idea that the neuron model and the inherent network connectivity could have an influence in the results of the computations as suggested in [16]. Also the concepts of movement decomposition from [12], [17], [18] were used as inspiration for the parallel system as their results are averaged and composed together to generate the final movement.

The chosen task was based on the principles of action learning and focuses on the ability to learn from a teacher how to draw simple shapes on top of a table. This is part of a wider approach to robot learning and development with embodied and situated interaction [14], [19] . The trajectories start and end with zero velocity and acceleration following a smooth human inspired profile [20]. A total of four joints were necessary to draw the shapes (square, triangle and circle) while the distance and angle between the pen and the table had to be kept constant. The performance and analysis of the learned movements using the parallel and the serial approaches were done using a model of the Rethink Robotics Inc. industrial humanoid robot BAXTER inside the Versatile and Scalable

Robot Simulation Framework (V-REP) [21]. At the end, as a proof of concept, one of the shapes was also tested using the real BAXTER robot.

II. Methods

The main steps in the procedure for the neurorobotics experiment presented in this work (subsections II-A, II-B, II-C, II-D, II-E, II-F and II-G) are summarised below[1]:

1) Shapes were defined in the $2D$ space (Z was kept constant as it represented the table top height). Human inspired endpoint velocity profiles were applied during the discretization of the shapes (see II-C).
2) The trajectories were translated to joint angles using a BAXTER robot model in V-REP (see II-F) controlled directly from an IPython notebook [22].
3) Joint angles were converted according to a special input code convention developed to increase the system's resolution and simplify the input/outputs to the liquid (see II-D).
4) Training data was generated by the LSM (see II-B).
5) Readout weights for each joint (see Figure 2) were trained to predict the next value using the data generated by the LSM using a linear regression algorithm (see II-E).
6) Testing data was generated using the proposed framework (Figure 1a) and the analysis of the results was done by comparing against a serial (Figure 1b) approach (see II-G).

A. Parallel and Serial Approaches

When working with a stochastic system, as in the case of the LSM approach, each time an experiment is done a new unique value will be generated. Considering the random process as stationary, traditionally an averaging of the results from multiple simulations, i.e. the mean value or first moment, is used.

In [15] and [13] LSM systems with an added feedback connecting the output to the input were employed in order to increase their computational power. Each LSM received individual feedback and its outputs were averaged between trials to improve the results. This method is called here the *serial* approach (Figure 1b).

The alternative framework proposed here follows a slightly different approach. Inspired by the idea of a parallel noisy brain model, multiple LSM, randomly created and initialized, are used in parallel and the feedback each individual LSM receives (Figure 1a) is the average of all the readout outputs. This *parallel* approach (Figure 1a) is compared to the *serial* one (Figure 1b). Both systems used during the experiments had the same number of LSM (with 600 artificial neurons each, totalling $3,000$ per trial) in order to allow a comparison between them.

B. Liquid State Machine

An LSM usually is composed of Leaky Integrate-and-Fire (LIF) spiking neurons [4] connected in a recurrent pattern

[1]Source code available at github.com/ricardodeazambuja/IJCNN2016

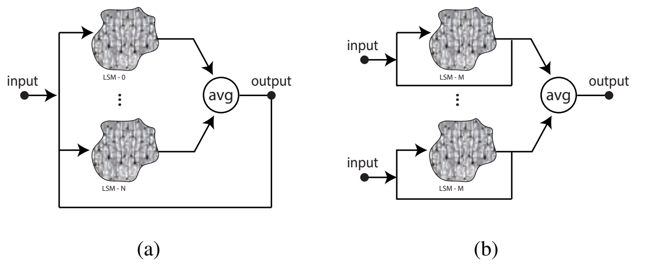


Figure 1. Simplified diagram with the parallel (a) and serial (b) approaches. All LSM were randomly generated and initialised for (a) and only randomly initialized for (b).

as suggested in [9] forming what is known as Small-World Network (SWN). The authors of [23] suggest SWN presents an appealing way to model the brain connections based on empirical and theoretical motivations.

Non-spiking neuron model based applications traditionally do not alter the network after the learning process is finished. Noise is only applied during the initialization as stated by [14]. However, stochastic processes seem to be an important part of brain computational strategy [24] and the LSM technique implements noise levels compatible with what was found in *in vivo* recordings [3].

In order to make it possible to benchmark the performance of the framework proposed here, the same neuron model and LSM parameters from [15] were applied with a few modifications. The system implemented in this work does not make use of Short-Term Plasticity (STP) or forced transmission delays. According to [3], STP is only one of the properties generating hidden network states. Therefore, as they slow down simulations by demanding extra variables and calculations, STP and delays were not used. The diagram of the implementation of one individual LSM can be seen in Figure 2.

A variant of the Leaky Integrate and Fire (LIF) neuron model with exponential currents is used in this work (Equation 1). The basic LIF model behaves as a capacitor-resistor circuit with an added circuitry in order to generate the spike (action potential) and also to keep it discharged during the refractory period [4]. It can be partially represented by the set of differential equations as seen in the Equation 1 where c_m is the membrane capacitance (in F), τ_m the membrane time constant (in s), τ_{syn_e} and τ_{syn_i} decay time of the excitatory and inhibitory synaptic current respectively (in s), v_{rest} the membrane resting potential (in V), i_{offset} a fixed noisy current and i_{noise} a variable noisy current (in A).

$$\frac{dv}{dt} = \frac{i_e(t) + i_i(t) + i_{offset} + i_{noise}}{c_m} + \frac{v_{rest} - v}{\tau_m} \tag{1a}$$

$$\frac{di_e}{dt} = -\frac{i_e}{\tau_{syn_e}} \tag{1b}$$

$$\frac{di_i}{dt} = -\frac{i_i}{\tau_{syn_i}} \tag{1c}$$

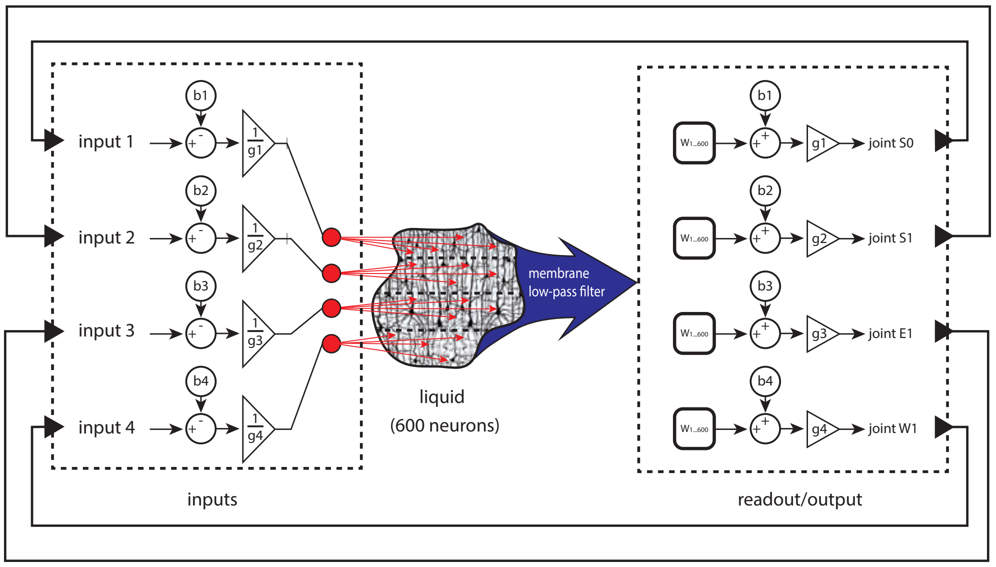


Figure 2. Illustrative representation of one individual LSM using only its own feedback. Each joint input/output has its unique bias (b1 to b4) and gain (g1 to g4). Readout weights sets (W1 to W600) are trained individually for each joint. As the weights are directly connected to the output of the low-pass membrane filter, they are trained using the normalised values (see II-D and Figure 5).

All simulations employed IPython and a custom software[2] entirely written in C. The parameters used for the neuron model were: $\tau_{syn_i} = 6ms$, $\tau_{syn_e} = 3ms$. $c_m = 30nF$, $\tau_m = 30ms$. Each LSM had the i_{offset} randomly drawn (see [15] for details about the distributions) during its creation, but the values were kept constant (by the use of the same random seed) after that. The liquid internal structure (connections) was also kept after the initialisation. It was necessary to keep those values otherwise each trial would have a different liquid instead of one with added noise. The initial membrane voltage and the current i_{noise} were randomly drawn during learning and testing phases where new i_{noise} values were drawn every time step. For all simulations, a time step of $2ms$ was used.

C. Definition of the 2D shapes

Three shapes were used in this work as a teaching task for the robot: square, triangle and circle (Figure 3). The joint angles generated by the inverse kinematics to draw a triangle on top of the table are presented in Figure 4. The system needs to obey not only the individual joint curves but also the synchrony between them to accomplish its task.

Based on the human inspired model reported by [20] the velocity profile was not constant all over the trajectories and the effect can be seen by the concentration of points in Figure 3. A time step of $2ms$ was adopted with the whole trajectory taking $2s$. Shapes containing sharp bends (square and triangle) were designed using several straight trajectories where the velocity reached zero at the corners. The idea here was to simulate a human teacher guiding the robot's arm.

[2]Available at github.com/ricardodeazambuja/LiquidStateMachine-Python

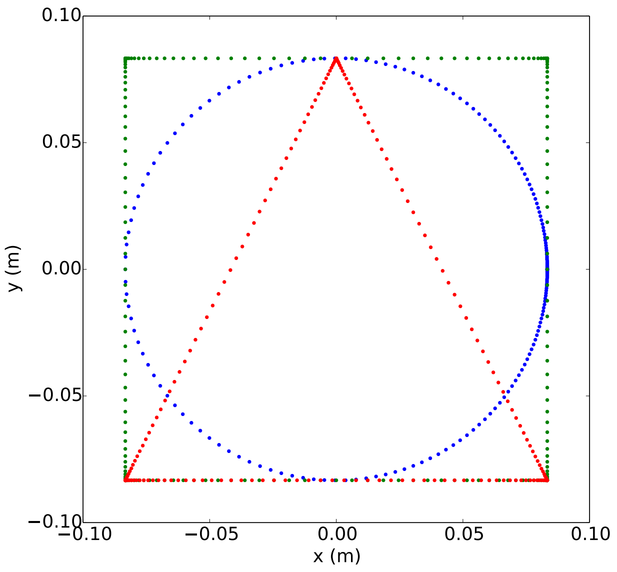


Figure 3. Shapes used to teach the robot (green:square, blue:circle and red:triangle). The Z axis is not presented here as it was kept constant (non zero) for all the shapes. The increase in the concentration of points is a consequence of the human inspired velocity profile [20]. Only one fifth of the points are being presented to help the visualization.

D. Input and Output Code

The input code uses a simplified population code inspired by what was presented in [15] to discretise analogue values. In [25] a VLSI friendly implementation of an LSM based system is presented where 77 discrete inputs are used. Therefore this setup is appealing for future conversion of LSM to a VLSI digital binary based system. Despite its simplicity it needs a

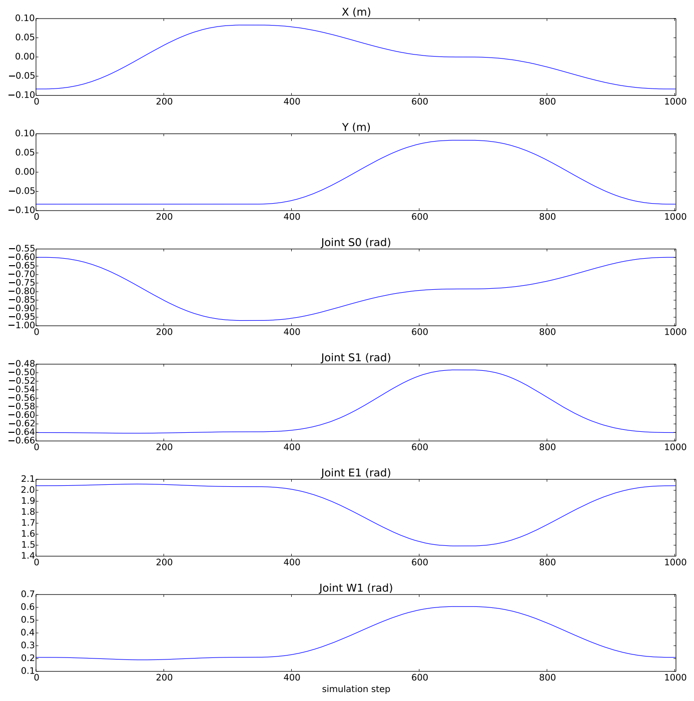


Figure 4. Resultant individual Cartesian X and Y movements as well as joints S0, S1, E1 and W1 (see Figure 7) necessary to draw the triangle (Figure 3) on top of a table.

large number of neurons if a fine scale is used.

Inspired by electronics instrumentation signal conditioning, analogue values suffer a translation (*bias*) making them start at zero and compression/elongation through a *gain* that fits its total range to one (normalisation) (see Figure 2). That way, using only two extra variables (*bias* and *gain*) the system normalises the inputs and uses as much of the population code range as possible in this situation producing a higher resolution (Figure 5).

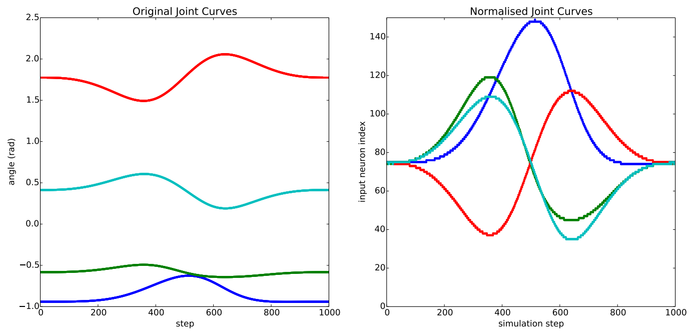


Figure 5. Input normalisation. The figure on the right presents the normalised version of the joint angle values necessary to draw the triangle on top of the table (figure on the left). The normalised values go from 0 to 149 as they are related to the input neuron index offset.

The pseudo-code used to convert from analogue values to neuron indices is presented as Algorithm 1. Both *bias* and *gain* are unique for each shape and could be seem as a form of mapping from the desk space (where the shapes were drawn) to the LSM space. For the injection of the input spikes, the liquid was divided into four slices. Each slice receives spikes from only one input (joint angles). The weights connecting the liquid's neurons and the input spikes are formatted to create a redundant code with a Gaussian shape where its mean value lays on the input neuron index (Figure 6).

Algorithm 1 Input code normalisation

```
procedure INPUTNORM($J\theta$)
    Receives joint angle $J\theta$          ▷ $J\theta$ in rad
    Extracts the bias          ▷ the curve starts from zero now.
    Divides by the gain          ▷ gain makes the range unitary
    Discretizes          ▷ closest value on the input neuron array
    return index          ▷ value passed to the Gaussian input
end procedure
```

For the output joint values the same *gain* and *bias* calculated for the input are used. It works as the opposite calculation of the Algorithm 1 having as inputs the analog values from the membrane low-pass filter (time constant $30ms$) output.

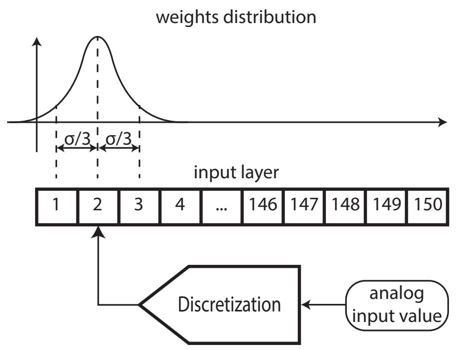


Figure 6. Gaussian weighted input distribution. The maximum weight value used was $100nA$.

E. Linear regression

When the LSM framework is employed, normally a linear regression is implemented to train the weights (W1 to W600 from Figure 2) connecting the *readout* to the liquid. The Ordinary Least Squares (OLS) method solves the minimization problem expressed by the Equation 2. The Ridge Regression [26] from Scikit-learn (0.16.1) [27] was used here with its default settings. It solves a slightly different version of the OLS problem as can be seen on Equation 3 (λ is the regularization parameter). Differently from the OLS, the Ridge Regression is still solvable even if $\mathbf{X}$ is not full rank.

$$\min_{w} \|y - \mathbf{X}w\|_2^2 \tag{2}$$

$$\min_{w} \left(\frac{1}{2}\|y - \mathbf{X}w\|_2^2 + \frac{\lambda}{2}\|w\|_2^2 \right) \tag{3}$$

In this work the matrix $\mathbf{X}$ was composed by the membrane low-pass filtered values of the liquid spikes ($\tau_m = 30ms$), y the normalised joint angle values (before the *bias* and *gain* being added back) necessary to generate the shape and w represents the readout weights.

F. BAXTER Robot

BAXTER, from Rethink Robotics Inc., is a humanoid robot designed to be safe and operate among humans. However, in order to keep all its safety mechanisms activated, the researcher must use the Joint Position Control mode[3]. Therefore the experiments using the framework proposed here always command the robot using that control mode. In total this work used four joints: S0, S1, E1 and W1 (see Figure 7). It is important to emphasize that although it is drawing 2D shapes, the robot moves in the 3D space, keeping an angle of 90 degrees between the felt pen and the table, and it needs to keep the Z axis constant (table top height). This made necessary the use of four joints (S0, S1, E1 and W1) as can be seen in the Figure 7. The translation from Cartesian space to joint space was made using the available Damped Resolution Method (damping:0.10 and max.iterations:3) for inverse kinematics in V-REP (version 3.2.2. rev.1) after the arms were positioned in their default untucked pose. In addition the V-REP remote API was employed to control the simulator from IPython making it easier to run several trials using the V-REP headless mode. Figure 8 shows the simulation result of one trial using the framework presented here.

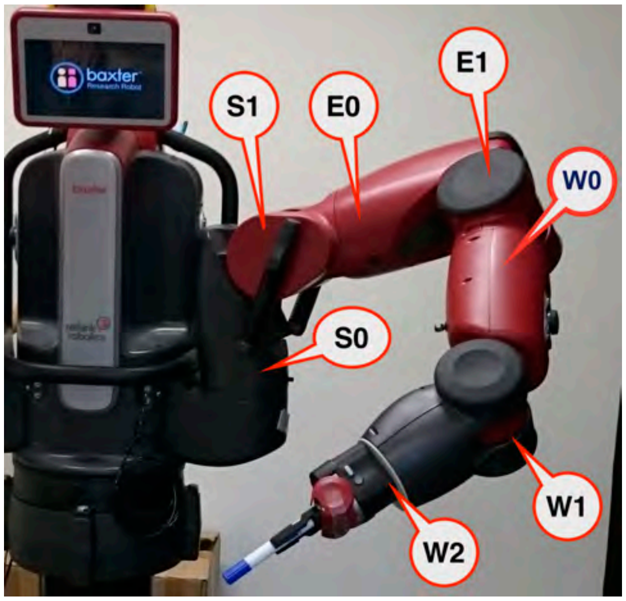
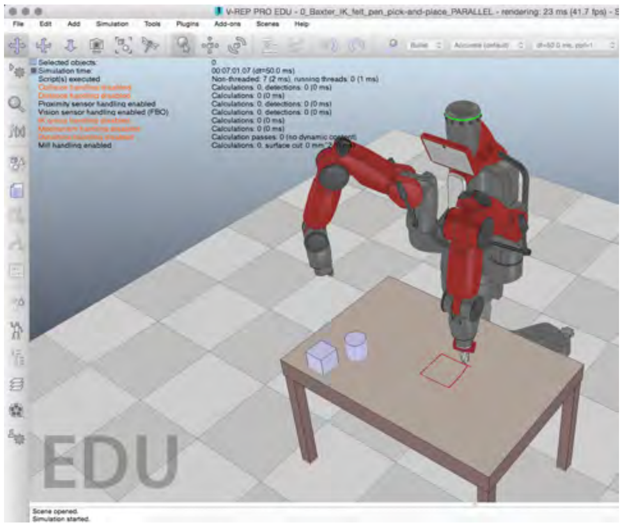


Figure 7. BAXTER robot left arm with the joint names highlighted. Only the joints S0, S1, E1 and W1 are used during the experiments presented here.

As a proof-of-concept only, the square shape using the novel parallel framework proposed here (Figure 1a) was tested using the real BAXTER robot (Figures 7 and 20).

G. Testing and Analysis tools

Analysis was done to verify if the novel framework proposed here (Figure 1a) would perform better or worse than


V-REP PRO EDU - 0_Baxter_IK_felt_pen_pick-and-place_PARALLEL - rendering: 23 ms (41.7 fps) - S
File
Edit
Add
Simulation
Tools
Plugins
Add-ons
Scenes
Help
EDU

```
Selected objects:                      0
Simulation time:                       00:07:01.07 (dt=50.0 ms)
Script(s) executed                     Non-threaded: 7 (2 ms), running threads: 0 (1 ms)
Collision handling disabled            Calculations: 0, detections: 0 (0 ms)
Distance handling disabled             Calculations: 0 (0 ms)
Proximity sensor handling enabled      Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)   Calculations: 0, detections: 0 (0 ms)
IK group handling disabled             Calculations: 0 (0 ms)
Mechanism handling disabled            Calculations: 0 (0 ms)
Dynamics handling disabled             Calculation passes: 0 (no dynamic content)
Mill handling enabled                  Calculations: 0, surface cut: 0 mm^2
```

```
Scene opened.
Simulation started.
```

Figure 8. V-REP setup used in this work. A simulated felt pen was employed to draw the shapes. The shape drawn on top of the table corresponds to the square generated by the novel system presented in this work (one trial).

a single LSM after several trials where the final results were averaged (Figure 1b). Pilot experiments, not presented in this paper, showed that the positive effects of multiple LSM in parallel could be more clearly seem with at least five of them. Therefore, five different LSM (600 neurons each) were randomly generated to test each individual shape. Here the term *randomly* is in respect to the initialization of the random seeds used during the definition of the liquid structure's main parameters.

Readout weights were trained for each one of the five LSM for a total of five hundred trials (one hundred trials each LSM). The testing phase employed all five LSM created for each shape in a batch of ten trials for the proposed parallel framework (with five LSM running in parallel - Figure 1a) and ten trials for the traditional serial averaged one (where each trial was composed of five simulations of identical LSM with the results averaged at the end. In order to have the same number of final shapes, each LSM was employed twice - Figure 1b).

Having the same total number of trials for both systems and using the same five different LSM generated makes it possible to compare the two approaches. Instead of simulating only $2s$ (1000 simulation steps), during the testing phase the system was subjected to twice the number of steps. The use of more simulation steps helped to verify if the systems had the ability to keep the end position.

Testing phase results could be analysed through several ways as the system generates the final resultant X, Y and Z movements following a specific velocity profile and also the individual joint curves. The experiments showed some results where the movement had a constant value zone creating a delay in time, and this makes it harder to apply traditional metrics such as correlation. Instead Dynamic Time Warping (DTW) [28] was used.

[3]See sdk.rethinkrobotics.com/wiki/Arm_Control_Modes

The DTW method generates a new set of pairs that applies a correction (time warping) making it easier to compare two signals in a similar way to the method human beings use [29]. One example of mapping can be seen in the Figure 9. A simple version of the DTW algorithm was implemented in C with a Python interface[4] supporting multidimensional arrays and Euclidean distance. The final cost is calculated by summing all the distances (represented by the red lines connecting the circle and the cardioid - left hand side on Figure 9) mapped by the generated path (blue curve, right hand side on Figure 9) and normalising using the maximum cost value generated for each shape using the parallel and serial approaches. Using this metric, the smaller the cost the better the match.

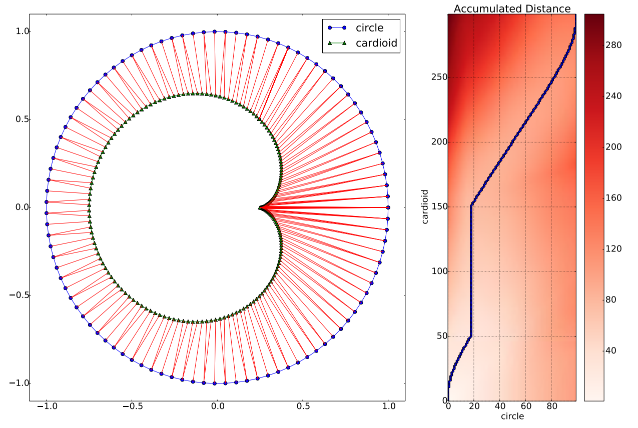


Figure 9. Example of 2D trajectory-matching generated by the DTW method. Although looking perfect in the figure on the left, the cardioid was modified to have a constant value zone from time step 50 to 150. The DTW correctly matches the values as can be seen as a straight blue line in the Accumulated Distance plot (right).

III. Results and Discussion

The main task proposed in this study as a benchmark was the ability to teach a robot, using principles of action learning [30] and embodiment [14], [19] and controlled by a SNN to draw three simple shapes (Figure 3). The analyses were made by comparing the resultant curves from the parallel and serial methods by visual inspection as well as using the DTW method. As the task involved the control of a pen in the 3D Cartesian space with the added time dimension, in order to simplify the figures, the comparisons were divided into the analysis of: 2D Shape (III-A), Time (III-B), Space-Time (III-C) and Z Axis (III-D).

A. Final 2D Shape Analysis

For the analysis of the resultant 2D shape, all the ten trials were plotted together with the parallel and serial approaches side-by-side. Figures 10, 11 and 12 show the square, circle and triangle shapes respectively.

Starting with the square (Figure 10), the comparison between the two methods (parallel and serial - Figure 1) showed the second one was not able to complete the task even after all the trials. As the square has four sides, it is easy to see the serial approach could not reach much more than one half of the total trajectory. In the case of the triangle (Figure 12), the serial method was able to perform better on only one out of ten trials, but failing in the final $1/6$ of the trajectory. The circle (Figure 11) was the only shape with results where the traditional approach was able to follow the trajectory in more than one trial. Although not a perfect fit to the dashed line (the original shape), the resultant curves from the parallel method presented here were able to match the original ones with just some small errors when compared to the serial one. The only shape with slightly worse results for the parallel case was the circle.

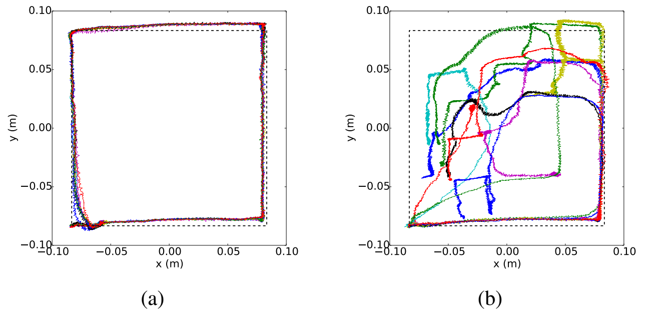


Figure 10. Results of the ten trials (square). The dashed line represents the original shape (Figure 3). Using the framework presented here (a) and the serial averaged method (b).

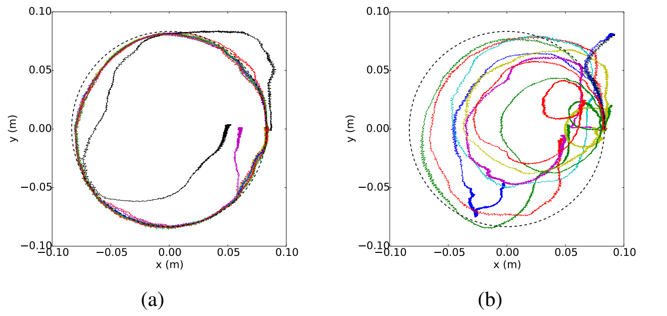


Figure 11. Results of the ten trials (circle). The dashed line represents the original shape (Figure 3). Using the framework presented here (a) and the serial averaged method (b).

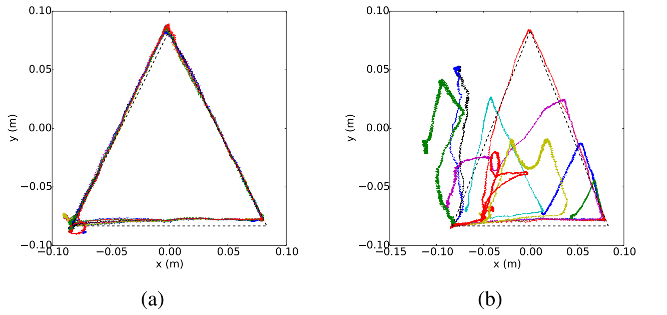


Figure 12. Results of the ten trials (triangle). The dashed line represents the original shape (Figure 3). Using the framework presented here (a) and the serial averaged method (b).

[4]Available at github.com/ricardodeazambuja/DTW

B. Time Series Analysis

The teacher signal used in this work contained more than spatial information, as it also had time information in the form of a velocity profile (II-C, Figure 4). Consequently using only the analysis of the final 2D shape it is not possible to verify the behaviour in relation to time. Through a visual inspection of the Figures 13, 14 and 15 it is possible to realize that the solution proposed in this paper was able to follow more closely the original X and Y time series. However, in some of the trials phase delays occurred mostly within the parallel approach (Figures 13a, 14a and 15a). The serial approach (Figures 13b, 14b and 15b) clearly could not take advantage of the diversity of the LSM as most of the curves had a very erratic shape.

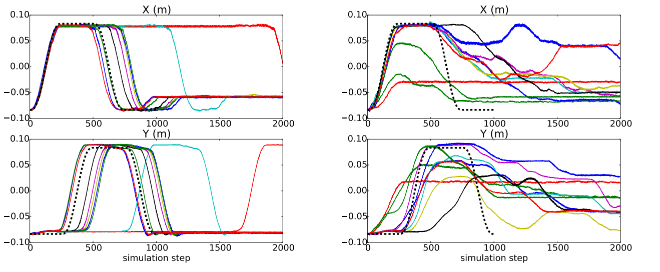


Figure 13. Visualization of the X and Y resultant curves (square) in time (all ten trials). The dashed line represents the original shape (Figure 3). Using the framework presented here (Figure 13a) and the serial averaged method (Figure 13b).

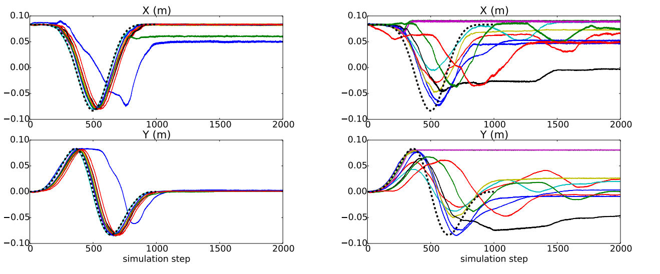


Figure 14. Visualization of the X and Y resultant curves (circle) in time (all ten trials). The dashed line represents the original shape (Figure 3). Using the framework presented here (a) and the serial averaged method (b).

C. Space-Time Analysis

During the time series analysis (III-B) phase delays, disturbances and constant value zones were observed in some of the signals generated. The DTW algorithm was applied generating a final metric closer to what a visual inspection can detect. The results from the comparisons using the final cost generated by the DTW can be seen in the Figures 16, 17 and 18 for the square, circle and triangle respectively. In only one situation (Figure 16, trial number 2) the serial method had approximately the same cost as the parallel one. This happened because the parallel method can get stuck in a constant value (see Figure 13a) and the simulation finished before it could get unstuck. Besides this special case, the parallel approach cost was always smaller than the serial one and the DTW method confirms what the visual inspection tells us.

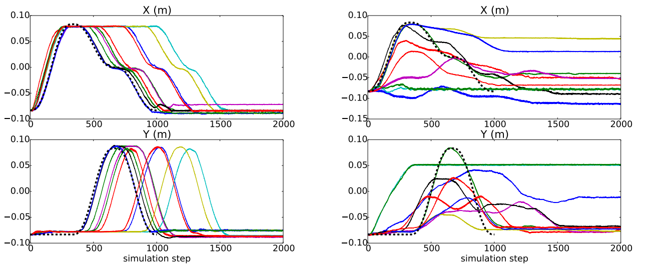


Figure 15. Visualization of the X and Y resultant curves (triangle) in time (all ten trials). The dashed line represents the original shape (Figure 3). Using the framework presented here (a) and the serial averaged method (b).

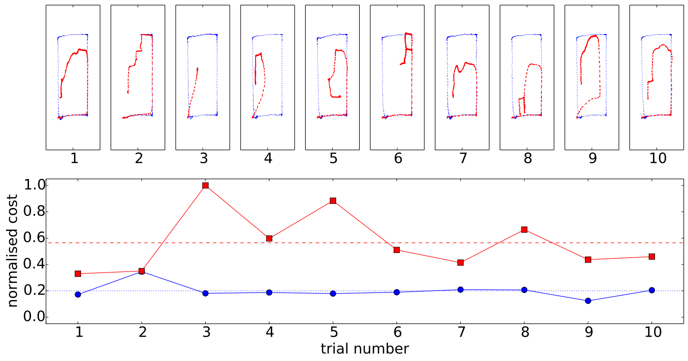


Figure 16. Resultant curves (square) for all trials - top figure. The DTW generated cost is presented at the bottom figure. The serial (dashed line, small squares - red) and parallel (dotted line, small circles - blue) approaches had a normalised mean cost value of 0.56 and 0.20 respectively.

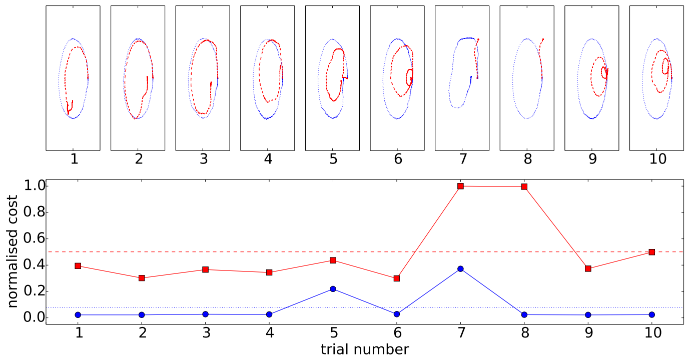


Figure 17. Resultant curves (circle) for all trials - top figure. The DTW generated cost is presented at the bottom figure. The serial (dashed line, small squares - red) and parallel (dotted line, small circles - blue) approaches had a normalised mean cost value of 0.50 and 0.08 respectively.

D. Final Z Axis Analysis

During the simulations done in V-REP the table did not exert any kind of reaction against the pen. Therefore values where the Z height is below the table surface were generated.

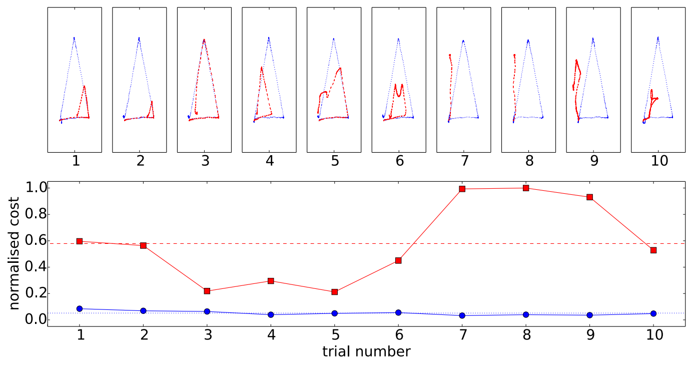
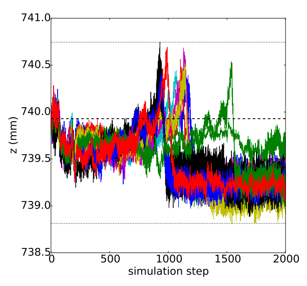
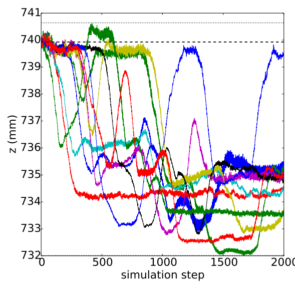


Figure 18. Resultant curves (triangle) for all trials - top figure. The DTW generated cost is presented at the bottom figure. The serial (dashed line, small squares - red) and parallel (dotted line, small circles - blue) approaches had a normalised mean cost value of 0.58 and 0.05 respectively.

In order to draw the shapes the BAXTER robot should keep the Z height approximately constant. The Figures 19a and 19b present the results obtained during the testing phase of the square shape for the Z axis. The maximum *delta* (absolute distance from the original Z value) for the parallel method was $1.12mm$ for the square, $2.10mm$ for the circle and $0.71mm$ for the triangle while, the serial averaged one had respectively $7.92mm$, $5.09mm$ and $6.35mm$. These numbers show the novel approach was on average more than six times better controlling the Z axis when compared with the serial.



(a)



(b)

Figure 19. Visualization of the Z curves (square) in time (all ten trials). The dashed line represents the original curve ($Z_{original} = 739.93mm$) and the dotted ones the maximum/minimum values. Using the framework presented here (a) - $Z_{max} = 740.74mm$ and $Z_{min} = 738.81mm$ and the serial averaged method (b) - $Z_{max} = 740.64mm$ and $Z_{min} = 732.01mm$.

E. Real BAXTER robot experiment

As stated at the beginning of this paper, a proof-of-concept experiment was done using the real BAXTER robot. It consisted of drawing a square (Figure 3) making use of its left arm and the framework presented here, but in an open loop configuration as the joint values were recorded from the robot simulations using V-REP. In Figure 20 it is possible to see the sequences of steps until the complete square is drawn. Not surprisingly the final drawing did not have as much noise as the one from Figure 10a because the robot arm together with its actuators functioned as a low-pass filter. Although simple, this experiment was very useful to test the tools developed to communicate from the SNN to BAXTER using UDP packets as well as the arm's calibration and table levelling. The Z axis spike (pen lifts up off the table) seen on Figure 19a could also be seen in the trial presented here, but not in all trials with the real robot as a result of the morphological filter formed by the whole physical set-up.

IV. Conclusions and Future Work

A new simple, yet powerful, idea of using parallel Liquid State Machines sharing the averaged outputs as their feedback was presented in this work. The task used as benchmark was based on the ideas of action learning where an external teacher showed the humanoid BAXTER robot how to draw simple shapes on top of a table. Results presented in this paper show a clear improvement in the task when the parallel method (Figure 1a) was used.

A similar idea of multiple liquids (or columns) to improve performance was already presented in [9], but only one *readout* was used for all columns, there were no feedback connections from the output of the system to the input and the results came from serial averaging many trials instead of the parallel system proposed here.

Several directions for future work are being considered. Firstly, the use of the real Baxter robot to generate the trajectories under human instruction could be investigated, as the robot can be safely guided in a gravity compensated mode. Additionally the possibility of closing the feedback loop using the real robot will be verified since in this situation it will be possible to experience delays and other external disturbances that were not present during the simulations. As Plymouth University is part of the BABEL project (a project on robot control and learning via the SpiNNaker neuromorphic system - see babel-project.org), the neuromorphic system SpiNNaker [31] will be employed for the Spiking Neural Networks in order to be able to increase considerably the number of neurons simulated as well as the speed and robustness.

Acknowledgment

This work was in part supported by the CAPES Foundation, Ministry of Education of Brazil (scholarship BEX 1084/13-5) and UK EPSRC project BABEL (EP/J004561/1 and EP/J00457X/1).

References

[1] E. Guigon, P. Baraduc, and M. Desmurget, "Computational Motor Control: Redundancy and Invariance," *Journal of Neurophysiology*, vol. 97, no. 1, pp. 331–347, Jan. 2007.

[2] T. Mergner and K. Tahboub, "Neurorobotics approaches to human and humanoid sensorimotor control," *Journal of Physiology-Paris*, vol. 103, no. 3-5, pp. 115–118, May 2009.

[3] D. V. Buonomano and W. Maass, "State-dependent computations: spatiotemporal processing in cortical networks," *Nature Reviews Neuroscience*, vol. 10, no. 2, pp. 113–125, Feb. 2009.

[4] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[5] M. A. Nugent and T. W. Molter, "AHaH Computing–From Metastable Switches to Attractors to Machine Learning," *PLoS ONE*, vol. 9, no. 2, p. e85175, Feb. 2014.

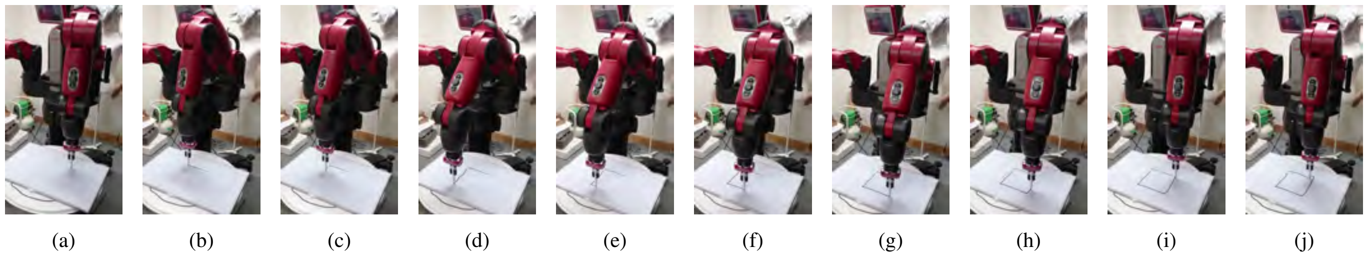


Figure 20. Proof-of-concept experiment where the resultant joints generated by the system presented in this work were directly injected into the real BAXTER robot in order to generate the square drawing (Figure 10a).

[6] S. Dura-Bernal, G. L. Chadderdon, S. A. Neymotin, X. Zhou, A. Przekwas, J. T. Francis, and W. W. Lytton, "Virtual musculoskeletal arm and robotic arm driven by a biomimetic model of sensorimotor cortex with reinforcement learning," Dec. 2013.

[7] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-DoF robotic arm based on spike timing-dependent plasticity," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, 2010, pp. 1–8.

[8] D. Gamez, R. Newcombe, O. Holland, and R. Knight, "Two simulation tools for biologically inspired virtual robotics," in *Proceedings of the IEEE 5th chapter conference on advances in cybernetic systems, Sheffield*, 2006, pp. 85–90.

[9] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.

[10] K. P. Dockendorf, I. Park, P. He, J. C. Príncipe, and T. B. DeMarse, "Liquid state machines and cultured cortical networks: The separation property," *Biosystems*, vol. 95, no. 2, pp. 90–97, Feb. 2009.

[11] W. Maass, T. Natschläger, and H. Markram, "Fading memory and kernel properties of generic cortical microcircuit models," *Journal of Physiology-Paris*, vol. 98, no. 4–6, pp. 315–330, Jul. 2004.

[12] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. L. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, no. 6810, pp. 361–365, Nov. 2000.

[13] W. Maass, P. Joshi, and E. D. Sontag, "Computational Aspects of Feedback in Neural Circuits," *PLoS Comput Biol*, vol. 3, no. 1, p. e165, Jan. 2007.

[14] H. Hauser, R. M. Füchslin, and R. Pfeifer, *Opinions and Outlooks on Morphological Computation*, H. Hauser, R. M. Füchslin, and R. Pfeifer, Eds., Zürich, 2014.

[15] P. Joshi and W. Maass, "Movement generation with circuits of spiking neurons," *Neural Computation*, vol. 17, no. 8, pp. 1715–1738, 2005.

[16] S. V. Adams and C. M. Harris, "A Computational Model of Innate Directional Selectivity Refined by Visual Experience," *Scientific Reports*, vol. 5, p. 12553, Jul. 2015.

[17] A. P. Georgopoulos, R. E. Kettner, and A. B. Schwartz, "Neuronal Population Coding of Movement Direction," *Science*, vol. 233, pp. 1416 –1419, 1986.

[18] T. E. Milner, "A model for the generation of movements requiring endpoint precision," *Neuroscience*, vol. 49, no. 2, pp. 487–496, 1992.

[19] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. MIT Press, Jan. 2015.

[20] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[21] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1321–1326.

[22] F. Pérez and B. E. Granger, "IPython: a System for Interactive Scientific Computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, May 2007.

[23] D. S. Bassett and E. Bullmore, "Small-World Brain Networks," *The Neuroscientist*, vol. 12, no. 6, pp. 512–523, Dec. 2006.

[24] E. T. Rolls and G. Deco, *The noisy brain: stochastic dynamics as a principle of brain function*. Oxford university press New York, 2010, vol. 28.

[25] Y. Zhang, P. Li, Y. Jin, and Y. Choe, "A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–1, 2015.

[26] R. M. Rifkin and R. A. Lippert, "Notes on regularized least squares," 2007.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb. 1978.

[29] R. Ratcliff, "Continuous versus discrete information processing: Modeling accumulation of partial information." Master Thesis, Radboud University Nijmegen, Nijmegen, The Netherlands, 2004.

[30] V. J. Marsick and J. O'Neil, "The Many Faces of Action Learning," *Management Learning*, vol. 30, no. 2, pp. 159–176, Jun. 1999.

[31] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker System Architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.