Investigating in-vitro neuron cultures as computational reservoir

Peter Aaser

***Abstract*—The human brain acts as a vastly parallel, self organizing computer far surpassing silicon processors in robustness, plasticity and energy efficiency. To gain an understanding into how the brain works a cyborg (short for cybernetic organism) is currently being made in an interdisciplinary effort known as the *NTNU cyborg project*. In this paper we explore applying a relatively novel machine learning technique, *reservoir computing*, to harness the computational power of living neural networks. These living cultures of neurons have been successfully grown at the department of neuromedicine hospital in *Micro Electrode Arrays* which allow them to be interfaced with using electrical stimuli and readouts. Using the theoretical foundation of reservoir computing these *In Vitro* neural networks are used to control a simple agent in a simulation on a remote computer. Finally we suggest an extension to the idea of cybernetic organisms, extending the idea of a cyborg as a hybrid organism between the biological and mechanic into *RC cyborg*, a robot which utilizes reservoir computing. To this end a software framework is described, facilitating the use of several different reservoirs in addition to neural cultures to be used without drastically changing the code base, enabling different reservoirs to be studied in a similar context.**

I. Introduction

Since the 50s, the Von-Neumann computer architecture has been ubiquitous in the field of computing. This architecture owes its success to its relative simplicity, allowing quick iterations in tact with doubling transistor counts in accord with moores law [1]. However several issues with the Von Neumann architecture have become more and more pressing in recent times. Issues such as the Von-Neumann memory bottleneck are becoming increasingly difficult to hide, and its inherently serial nature of computation makes increasing parallelism difficult and error prone. HiPEAC 2015[2] identifies four key challenges for the future of computer architecture design.

1) dependability by design
2) managing system complexity
3) energy efficiency
4) entanglement between physical and virtual world

Even with the assumption that moores law will hold in the future these issues threaten to drastically reduce the scalability of modern processor designs. These weaknesses necessitate going beyond the Von-Neumann architecture, exploring unconventional computing paradigms. In nature a completely different approach is taken which effectively deal with all these issues: The human brain vastly outperform modern computers on the three first issues, having no single point of failure, being self organized and operating at an estimate of 20 Watts. Not only does the neurons making up the brain easily surpass computers, the same neurons provide a sense of touch, sight and smell handily beating current efforts in the fourth issue. Taking inspiration from nature we look at the paradigm of *cellular computing* [3] where cells which by themselves are unremarkable are combined to perform highly efficient parallel computations. These cellular systems exhibit behavior properties that cannot be traced back to a single cell, they only appear when multiple cells are interacting with each other, known as *emergent behavior*. In nature these systems form without any form of supervision, a property known as *self organization*, with an example being DNA which "describes how to build the system, not what the system will look like" [4]. Understanding how these cellular systems that arise seemingly by themselves can be so successful and how we can harness this power may hold the key to solving problems faced in modern computer architectures, and may lead to advancements in neuroscience as well. In this paper we explore using neuron cultures to create an organic mechanical hybrid organism, a so called *cyborg* as part of the NTNU cyborg project.

II. Background

A. The NTNU Cyborg Project

The NTNU cyborg project is a collaboration between several departments at NTNU including the department of biotechnology, computer and information science, engineering cybernetics, neuroscience and more. [5] The stated goal for the cyborg project is "to enable communication between living nerve tissue and a robot. The social and interactive cyborg will walk around the campus raising awareness for biotechnology and ICT, bringing NTNU in the forefront of research and creating a platform for interdisciplinary collaborations and teaching." Currently the department of neuroscience is growing neuron cultures which are to be used as the biological part of the robot. These neuron cultures are not part of a brain, they are fully dissociated, grown in special chambers in-vitro. The robot part of the cyborg has been developed and implemented by the department of engineering cybernetics, and is currently operational, however it has not yet been integrated with the in-vitro neuron cultures. The challenge faced by the cyborg project is creating the infrastructure for interfacing the neuron cultures and the robot, essentially creating a brain computer interface.

B. Complex Systems

In this paper references are made to computations done by both artificial and real neurons. To make the distinction between these cases clear all computation done by computer simulated approximations of neurons will be prefixed as artificial.

Simply creating a bridge between neural cultures and a machine will not be of any use without a way to make sense of the information coming from the network, or how to encode information sent back to the network. In order to harness the computational power of neural networks a theoretical framework is therefore necessary, starting with the concept of *complex systems*. Complex systems are dynamic nonlinear systems whose behavior is an emergent property that cannot be tracked back to a single component. In these systems positive feedback loops amplify small changes into cascading changes changing the entire system, while negative feedback loops may cause other states to be relatively stable, resulting in multiple meta-stable states, so called *attractors*, that give the system some measure of order. By classifying neural cultures as one of many cellular computing systems that exhibit the properties of a complex system we can model the behavior of neurons with much simpler models. In [3] Sipper explores cellular computations which in contrast with microprocessors consist of relatively simple cells which interact directly only with its neighbors, lacking a global control mechanism. In short, processors are designed, cellular computation *emerges*.

In [6] Langton explores the requirements for cellular systems to support *emergent computation*, where he argues that in order for a system to support emergent computation it must lie in a *critical phase* between order and chaos, drawing parallels to phase transitions in material science. This observation comes from thermodynamics, in which a material may exhibit a *second order phase transition* between a solid and liquid form where the material undergoes a continuous transition in contrast to a first order transition such as melting ice. Using *cellular automata* as an example, he explores the effect of varying the rules for calculating the next state with a parameter, λ, which describes how likely it is for a cell to enter a *quiescent state*, a state where it will not disturb other cells until it leaves the quiescent state itself. At $\lambda = 0$ all cells enter a quiescent state after one step, representing a fully ordered system, thus at $\lambda = 0$ the system is in the ordered phase. At $\lambda = 1$ there are no rules that leads to a quiescent state, leading to very chaotic systems with very high entropy, representing the chaotic phase. As λ is increased from 0 the cellular systems starts to form intricate structures, increasing in complexity and taking longer to reach a steady state. This behavior peaks at $\lambda \approx 0.5$ which is the most critical phase in this particular system, creating complex self-organizing structures. As λ is increased further the self-organizing structures start to give way to more chaotic and seemingly random behavior, gradually transitioning into the chaotic phase.

It turns out then, that with most cellular systems providing the necessary conditions for computation to occur is a tractable problem, however harnessing and shaping this computation is a whole different matter. In their critical phase systems are unstable but not chaotic which causes them to be highly sensitive to small changes in topology and initial conditions. Attempting to maneuver this fitness-landscape by directly changing the properties of a system would in many ways be like calculating the correct way for a butterfly in china to flap its wings in order to cause a hurricane in New York.

C. Reservoir Computing

Faced with the intractability of designing cellular computation systems in a top down manner, and the unpredictable relation between cause and effect in self organizing systems it is necessary to approach complex systems in a different manner. Seemingly, classifying the neural culture as a complex system has not provided any useful tools for understanding how to interact with it, on the contrary it makes such a task look futile. However, in computer science the recent field of *reservoir computing* has emerged, embracing the complexity and unpredictability of certain complex systems. In reservoir computing, a complex systems is used as a *reservoir* [7] which "acts as a complex nonlinear dynamic filter that transforms the input signals using a high-dimensional temporal map, not unlike the operation of an explicit, temporal kernel function." In simpler terms, the properties that make complex systems so hard to work with such as sensitivity to initial conditions also allows them to discern very subtle nuances in input, and their complex behavioral patterns causes the systems to change their behavior to new input based on previous input. Many such reservoirs have been successfully exploited: In [8] an *echo state network* is utilized to solve classification problems. More esoteric reservoirs have been used, for instance in [9] the idea of reservoir computing is taken quite literally using a bucket of water as a reservoir. With reservoir computing there is no need for a designer shaping a system, the systems organize and shape their own behavior, all that needs (and can) be done is to find a good way to present information to the reservoir as either initial conditions or as perturbations, and how to interpret the resulting dynamics. From this perspective using a neural culture is simplified into learning some approximation of the correlation between input and output from the reservoir, treating the internals as a black box. Like the servants of the oracle of Delphi, the reservoir computing system must phrase the question correctly, and interpret the often incomprehensible answer. 1 shows a typical RC (reservoir computing) system with three inputs and two outputs. The inputs are processed in a simple feed-forward neural network before perturbing the reservoir in some way. Similarly the state of the reservoir is being processed by an output layer before leaving the RC system. In the figure the input and output processing is done by feed forward neural networks, but we note that this is only one of many possible filters. Inputs 1, 2 and 3 are snapshots of the current state of the problem we attempt to solve with reservoir computing. Since our filters have no state, at least not beyond some time horizon we see that the history of the system must in some way be encoded in the reservoir in order for the RC system to solve problems in scope wider than the limited amount of state that may be contained in the filters.

D. Neuron Computing

Neurons are cells which communicate with each others using electro-chemical signals in vast interconnected networks.

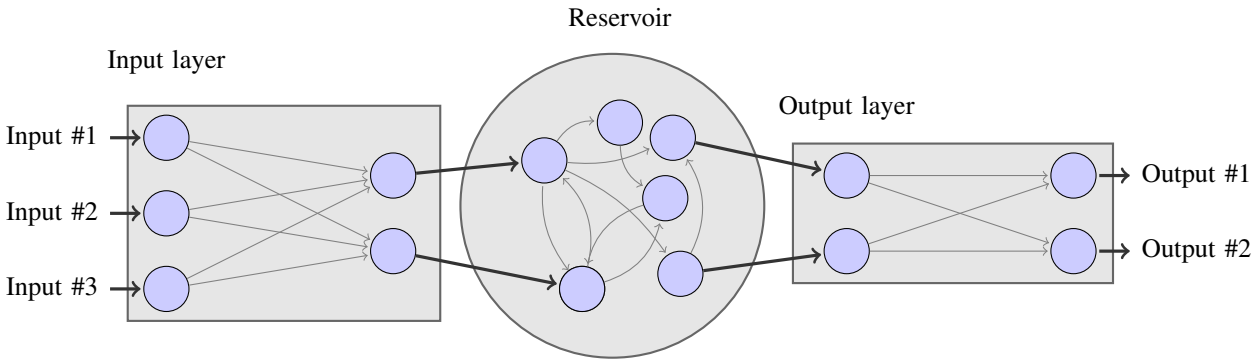

Fig. 1. An idealized reservoir computing system. The square boxes are simple filters, in this particular case represented by feed forward neural networks. The input processing and output layer are both trained with machine learning to process input and to interpret output from a reservoir in order to perform a specific task.

These networks communicate in a complex interplay between neurotransmitters and voltage spikes, and their communication prompt structural changes in the network, deeply intertwining the emergent behavior and self organizing properties on neurons. There is a high degree of correlation between the electrical, chemical signals in these neural network, thus a necessary simplification of considering only the electrical signals when interacting with neural networks can be made at a small cost of information loss. These electrical signals, known as action potentials, are caused by polarity differentials between the ions inside and the ions outside of the cell membrane. A neuron maintains this potential difference by transporting ions of opposite polarity through the cell membrane, essentially preparing a spike of electrical activity. This spike is triggered as the neuron receives electrical stimuli via a network of electrical receptors called dendrites. When excited, a neuron which has built up a sufficient polarity difference will trigger a polarity shift. This polarity shift cascades along the axon, a long tendril that can extend to reach far away neurons, forming branches that may reach multiple neurons.

III. A Hybrid Neuro-Digital Approach

A. Concept

A neural network is grown in a *MEA*, short for *micro electrode array* which allows the neurons to be interfaced with using electrical signals in order to control a robot as shown in 2. Building on the theoretical groundwork described in the background, the neural network is used as a reservoir which is fed sensory input from a robot. This system is a *closed loop*, it does not require any outside intervention, such as a human controller using a joystick to operate. A somewhat subtle but very important consequence of utilizing reservoir computing is that the neural network does not directly control the robot, it simply acts as a classifier which responds to the input, and the resulting dynamics is interpreted by a computer as an action for the robot. This distinction has little practical consequence for the cyborg, but may offer a clue into the inner workings of sentience.

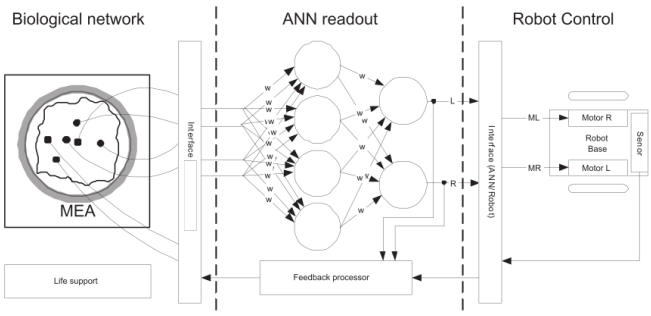

Fig. 2. The proposed architecture for a cyborg using neural cultures as a reservoir

B. Platform

Providing an interface between neurons and a computer allows using neural network for computation, however it is impractical to move the neural cultures outside of the laboratory. The practical, yet thought provoking[1] solution is using a TCP/IP network protocol such that the neuron culture may be interfaced with any robot or simulator without leaving the laboratory. Similar network architectures have been implemented, in [10] a neuron culture is used to control a simple wall-avoiding robot as a proof of concept. In contrast with previous work the robot used in the NTNU cyborg project is a sophisticated robot which is programmed to move by itself, follow a person, take a selfie with someone and upload it to facebook, and even perform a secret handshake. By utilizing an already functioning robot as a host the cyborg project can focus on extending the capabilities of the robot, making it a true hybrid between digital and cellular computing.

C. Growing Neurons In Vitro

The neuron cultures used in the cyborg are being grown in MEAs at the department of neuroscience. The MEAs are

[1]For brevity the philosophical implications is left as an exercise to the reader.

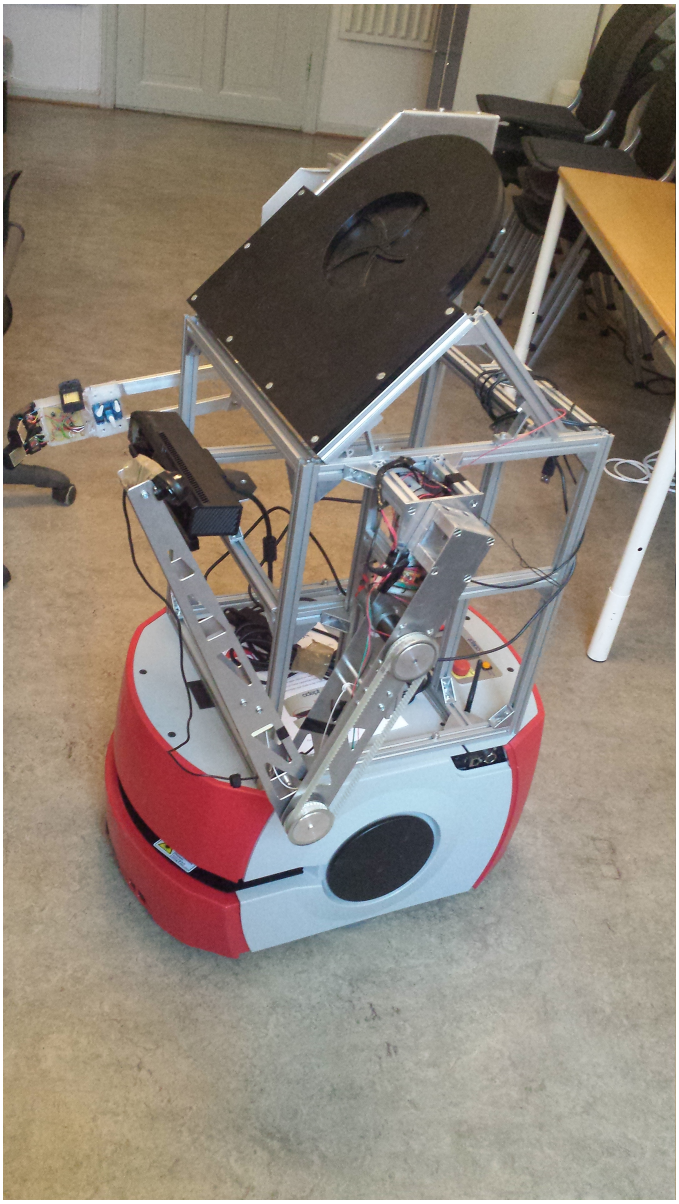

Fig. 3. The NTNU cyborg

seeded with neural stem cells of either human or rat origin which then spontaneously form networks. At seeding there is no network at all, only a "soup" of dissociated neurons which over the course of several weeks start forming networks. As the networks starts "maturing" a common phenomenon is neurons firing monotonic spikes automatically. The activity from these so-called pacemaker neurons can be seen in 7. In the figure each cell corresponds to one of the electrodes as seen in 6, however at this stage the monotonic spiking activity tends to be transient, starting and stopping randomly.

D. Neuron Interfacing Hardware

The hardware used to interface with neuron cultures for the cyborg is an *MEA2100* system purchased from multichannel systems. The MEA2100 system is built to conduct in-vitro experiments electrically active cell cultures such as neurons. The principal components of the MEA2100 systems are:

1) Micro Electrode Array: Introduced in the previous chapter, the *MEA* is equipped with an array of microscopic electrodes capable of sensing and delivering voltages to and from nearby neurons. 9 shows an empty MEA, 10 shows an MEA used by the department of neuroscience with a live neuron culture.

2) Headstage: The electrodes of the MEAs are measured and stimulated by the headstage which contains the necessary high precision electronics needed for microvolt range readings. 11 shows the same type of headstage used in this paper along with an MEA.

3) Interface board: The interface board connects to up to two head-stages and is responsible for interfacing with the data acquisition computer, as well as auxiliary equipment such as temperature controls. The interface board has two modes of operation. In the first mode the interface board processes and filters data from up to two headstages as shown in 4 which can then be acquired on a normal computer connected via USB. In the second mode of operation a Texas instruments TMS320C6454 digital signal processor is activated which can then be interfaced with using the secondary USB port as shown in 5

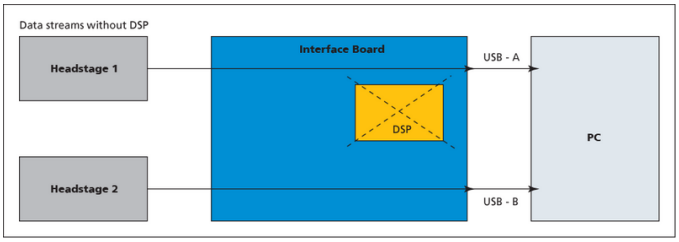

Fig. 4. Standard operating mode for the interface board

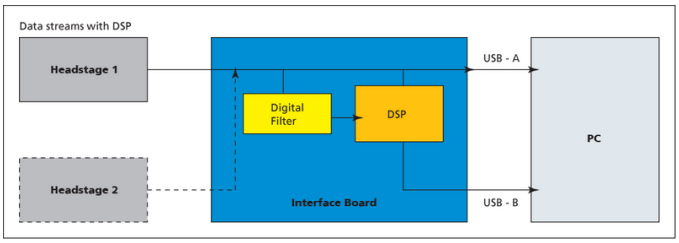

Fig. 5. Activating the digital signal processor allows real-time processing and feedback at the cost of a headstage slot.

IV. Results

A working implementation of the system has been run, using data from an empty micro electrode array to simulate real conditions. The random data produced by the empty micro

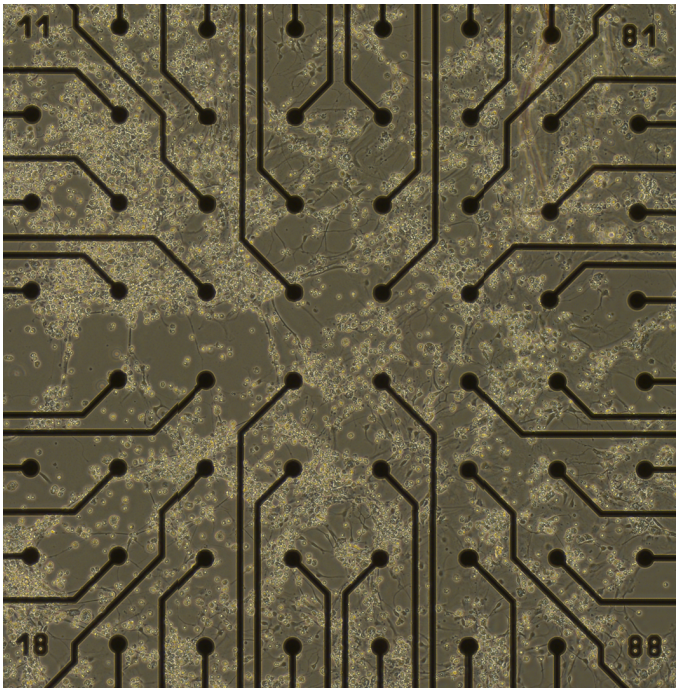

Fig. 6. Microscopic image on neural networks with visible structures in neuron culture grown by the department of neuroscience

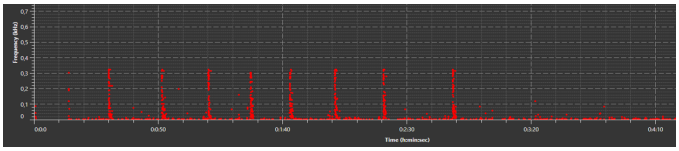

Fig. 7. A recording of spikes in a neural culture made at the department of neuroscience

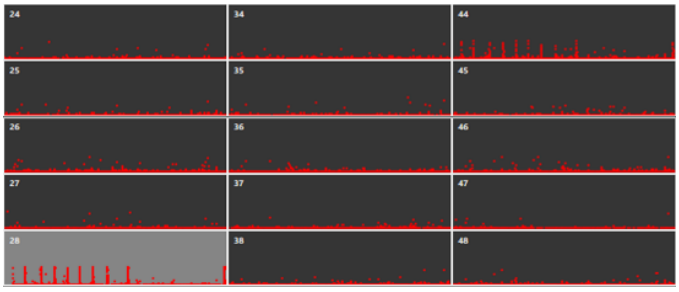

Fig. 8. The grayed out cell in the bottom left shows the readouts in 7. Each cell corresponds to one of the electrodes as seen in 6. Even though electrode 28 and electrode 44 are far away from each other there seems to be a clear correlation between these two sites.

electrode array was received at remote computer via TCP/IP and translated into actions for a simulated agent by a simple untrained feed forward artificial neural network acting as a stand in for a filter trained to interact with a neural reservoir. 13 shows a series of screenshots visualizing the running agent responding to this random data. The movement of the agent is random just like the input data, while it looks like it might be

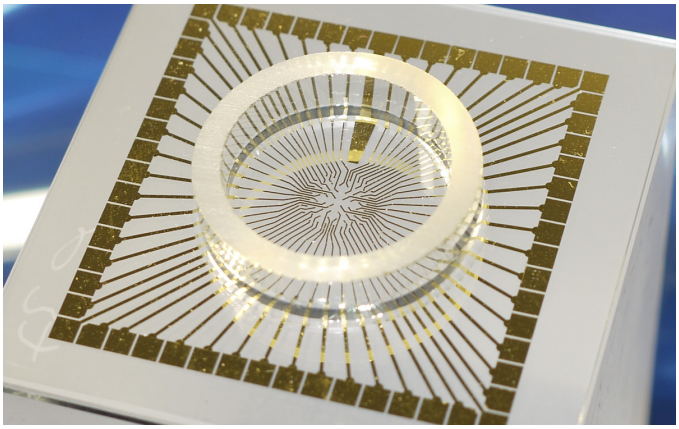

Fig. 9. A generic MEA

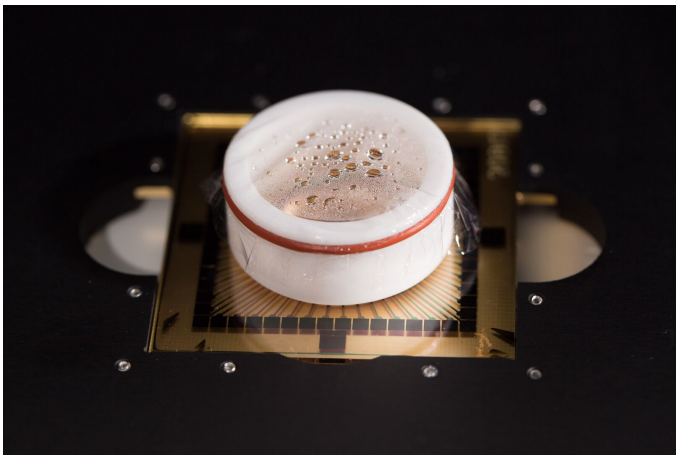

Fig. 10. A MEA with a live culture from the department of neuroscience

Fig. 11. The headstage. This device contains the highly sensitive electrical measurement equipment needed to get accurate readings from neural cultures.

Fig. 12. The interface board. This device is a link between headstages and lab computers. It can support two headstages in normal operation, or one headstage in DSP mode.

attempting to avoid a wall this is due to chance. In the test the sensory data perceived by the agent is sent back via TCP/IP to the computer interfacing with the empty micro electrode array which is stimulated. The result of the test is that a closed loop has been successfully run, showing that the architecture of the cyborg project is capable of handling the practical challenges involved. The visualizer shown in 13 runs in real time in a web browser, however it is not yet available on the internet.

V. An RC Cyborg Platform

Introductory the connection between neural networks and complex systems guided the reservoir computational approach. This fundamental similarity between seemingly unrelated systems has guided the design process of the software system to allow for accommodating a wide variety of reservoirs in contrast to previous approaches such as [10]. Rather than focusing simply on connecting in-vitro cultures to robots, the architecture which is currently in development aims to provide a highly modular system which allows for many different reservoirs on many different platforms. By broadening the concept of a cyborg to a hybrid of machine and reservoir we can study differences and similarities between different reservoirs, increasing our understanding of the underlying properties of complex systems.

A. Areas Of Concern

The guiding principle for the RC cyborg architecture is to separate different concerns and provide simple interfaces between them. An important consequence of this choice is that it allows us to reuse much code between a cyborg using a neural network as a reservoir and one using a random boolean network. To clarify it is important to introduce the concept of *application specific* and *reservoir specific* data processing. The former is the data processing stage performed in context

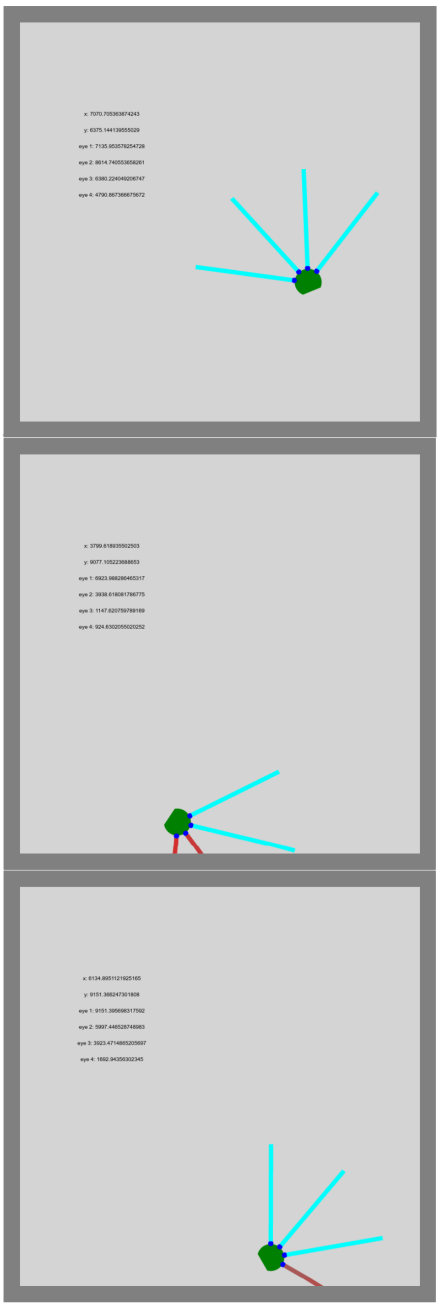

Fig. 13. A simulated agent responding to random data. In the first image the agent does not see any obstacle, while in the two lower images the agent sees the wall, visualized as the sensor indicator turning red.

of reservoir computing used to interpret the dynamics of the reservoir to perform a specific task. The latter describes data processing done outside the context of reservoir computing, from simple noise reduction to more transformative filters, such as spike detection for neural networks. From this perspective the same neural network can act as two different reservoirs,

one delivering raw waveform data, the other delivering spike data. The following sections describe the areas of concern, and where they fit in context of reservoir computing.

1) Data Acquisition and Interfacing: Data acquisition (DAQ) is the task of configuring and interfacing with a specific reservoir, essentially providing the wrapper for an actual reservoir which represents the idealized shown in 1. Finally, the data acquisition and interfacing module should expose an interface via TCP/IP allowing configuring the reservoir, accessing reservoir output and accepting requests to stimulate the reservoir.

2) Data Processing: This part of the architecture is responsible for the reservoir computing processing of data which is shown in the idealized model of reservoir computing in 1. Implementations of the data processing stage must be able to filter input from a reservoir received from the DAQ module into actions, as well as filtering sensor data received from the agent control module to a format compatible with the reservoir. Note that two different reservoirs may provide the same interface, allowing the same data processing component to utilize different reservoirs as long as the format remains the same. As with the DAQ module, data processing communicates via TCP/IP to both the DAQ module and the agent control module.

3) Agent Control: This module implements the robot part of the RC-cyborg, which can range from being a simple simulated agent in an idealized game to a fully fledged physical robot interacting with the real world. The agent control interfaces with the data processing module, receiving agent specific output from the reservoir which it may use (in a physical robot the reservoir does not have the final word as a safety measure). It also transmits sensor data back to the data processing module, providing feedback to the reservoir. As with the DAQ module, some sensory processing may be done by agent control, but only reservoir agnostic filtering.

B. Implementation

The following sections describe the software currently implemented for controlling the NTNU cyborg and where they fit into the general RC-cyborg framework. This represents the bulk of work performed in this paper.

1) MEAME: MEAME [11] implements data acquisition and interfacing for using neural cultures as a reservoir. It is written in C# and interfaces to the MEA2100 system using an API provided by multichannel systems allowing it to interface with the interface board 12. MEAME also implements the optional DSP which is used to stimulate the neurons and can be used to process the waveform data from the neural cultures in real time, and even implement more advanced techniques for timing stimulation to maximize impact. By configuring the DSP the MEAME system may act as several reservoirs in context of an RC-cyborg system, even though it uses the same underlying neural culture for each mode. Many configurations are possible for the DSP not only for processing outbound data. In [12] machine learning is employed in order to select optimal timings for stimulating a neural network to maximize response. If this technique is implemented on the DSP the resulting system would be a reservoir with different characteristics even though it uses the same underlying neural network as its physical reservoir.

2) SHODAN: SHODAN [13] is a framework for composing reservoir computing experiments written in scala. In contrast to MEAME which is written specifically for interfacing with MEA2100, SHODAN is intended to facilitate general purpose input and output processing for reservoir computing. The main focus of SHODAN is to perform task specific processing of reservoir data, but it also comes with tools for extending reservoir specific processing such as spike detection. At the core SHODAN is a framework containing a library of tools used for processing functional streams of data such as feed forward artificial neural networks and methods for encoding and decoding data to facilitate implementing different protocols for TCP/IP communication. SHODAN also comes equipped with a built in simulator that currently consists of a simple wall avoidance game which acts as an agent control module which includes a real time visualizer that can be accessed in a browser.

VI. Discussion

A. Reservoir Compatibility

With the RC cyborg framework using different reservoirs in the same harness is made possible, however this requires some modification to the reservoirs themselves. As an example we consider the difficulties involved with training the reservoir computing system on neural networks. These networks are constantly changing and growing, even small topological changes may move an important cluster of neurons outside the range of an electrode. A possible approach is to use a random boolean network as a stand in reservoir for a neural culture. A random boolean network is a graph where each node is assigned a boolean function of their inbound edges. Since these nodes will have either the value 1 or 0 we can select a few of them and let nodes with state 0 act as no activity, and nodes with the value 1 act as a spiking neuron. Similarly sensor stimuli can be used to switch input nodes on or off, "perturbing" the system. Note that this is only one of many possible reservoir specific filters as defined in the RC cyborg framework altering the functionality of the reservoir. With this translation two very different underlying reservoirs can now be interacted with using the same input and output, allowing a system to be trained on different underlying reservoirs while sharing a majority of code. Simply making the input and output between two reservoirs compatible will of course not cause them to behave anything like each other. The problem of utilizing random boolean networks as a viable stand-in for neurons is very far from being solved, however compatibility is an important first step.

B. Adaptivity

In the RC cyborg framework adaptivity and learning is not elaborated on, but this aspect will become crucial when the implementations mature. Reservoir computing does not add any constraints on how the task specific filters should be

constructed, thus typically the filters are created using general purpose machine learning kits. For reservoirs such as random boolean networks this is a sensible solution, however for neural networks that evolve, changing their behavior with time the RC cyborg must be capable of adapting along with the reservoir. As with reservoir compatibility this is a complex task, but by considering the learning system to be a part of the RC-cyborg we may lessen the compatibility gap between different reservoirs.

VII. Conclusion

A working solution for the architectural challenges for interfacing a robot with a neural cultures has completed a proof of concept test, showing that the remaining efforts for the cyborg project is harnessing the computations performed by the neural network. By proposing the RC cyborg concept as a broader definition of a cyborg the foundation has been laid for exploring differences between neural networks and other reservoirs used in the same contextual "harness" such that the differences between reservoirs may be studied quantitatively as well as qualitatively. With the theoretical and practical framework needed to interface with neural network the cyborg project is now equipped to explore the fundamental aspects of neural computation.

VIII. Further Work

A. Exploring Reservoirs

Only neural networks and random boolean networks have been used as reservoirs in this project. One of the stated goals of the RC cyborg platform is to facilitate the use of a wide range of reservoirs ranging from recurrent artificial neural networks, to more exotic physical reservoirs. By comparing performance of different reservoirs for the same task different characteristics impacting performance of RC systems may be explored.

B. Investigating Neuron Growth Rules

Neural cultures change their own structure in response to stimuli, which in turn is a result of their structure. This interplay between self-organization and current dynamics of the neural network is not well understood, but it may have a profound impact on creating artificial neural networks imitating their biological counterparts. One experiment that might shed light on this process is to observe the differences in regularity between neural networks who have been subjected to random stimuli compared to cultures who have experienced regular stimuli.

C. Integrating Machine Learning

In order for SHODAN to support on-line adaptivity tools for machine learning is required. A framework for genetic algorithms in scala, scalapagos [14], will be integrated with SHODAN allowing exploration of the parameter spaces both for reservoirs themselves and for data processing filters.

References

[1] "Moore's law on wikipedia." https://en.wikipedia.org/wiki/Moore's$_law$.

[2] "Hipeac vision 2015." https://www.hipeac.net/static/files/hipeac-vision-2015.0bde1ff5d0b5.pdf.

[3] M. Sipper, "The emergence of cellular computing," vol. 32, no. 7, pp. 18–26.

[4] G. Tufte, "From Evo to EvoDevo: Mapping and Adaptation in Artificial Development,"

[5] "Ntnu cyborg webpage." https://www.ntnu.edu/cyborg.

[6] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," vol. 42, no. 1, pp. 12–37.

[7] B. Schrauwen, D. Verstraeten, and J. M. V. Campenhout, "An overview of reservoir computing: Theory, applications and implementations," in *ResearchGate*, pp. 471–482.

[8] H. Jaeger, "Adaptive Nonlinear System Identification with Echo State Networks," in *Advances in Neural Information Processing Systems 15* (S. Becker, S. Thrun, and K. Obermayer, eds.), pp. 609–616, MIT Press.

[9] T. Natschlger, W. Maass, and H. Markram, "The "Liquid Computer" A Novel Strategy for Real-Time Computing on Time Series," vol. 8, no. 1, pp. 39–43.

[10] Y. Li, R. Sun, B. Zhang, Y. Wang, and H. Li, "Application of Hierarchical Dissociated Neural Network in Closed-Loop Hybrid System Integrating Biological and Mechanical Intelligence," vol. 10, no. 5, p. e0127452.

[11] "Meame github repository." https://github.com/PeterAaser/MEAME.

[12] S. S. Kumar, J. Wlfing, S. Okujeni, J. Boedecker, M. Riedmiller, and U. Egert, "Autonomous Optimization of Targeted Stimulation of Neuronal Networks," vol. 12, no. 8, p. e1005054.

[13] "Shodan github repository." https://github.com/PeterAaser/SHODAN.

[14] "Scalapagos github repository." https://github.com/PeterAaser/Scalapagos.