Axel Tidemann

A Groovy Virtual Drummer: Learning by Imitation Using a Self-Organizing Connectionist Architecture

Thesis for the degree of Philosophiae Doctor

Trondheim, August 2009

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

NTNU
Norwegian University of Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

ISBN 978-82-471-1734-7 (printed ver.)
ISBN 978-82-471-1735-4 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2009:171

Printed by NTNU-trykk

To rock'n'roll, girls and computer music.

ABSTRACT

The research in this thesis aims to enable robots to imitate humans. Learning by imitation is a fundamental part of human behaviour, since it allows humans to acquire motor skills simply by demonstration; seen from a robotic viewpoint you can easily "program" your fellow humans by showing them what to do. Would it not be great if the same mechanism could be used to program robots? A robot is programmed by specifying the torque of its motors. The torque can be regarded as the force or strength that is the result of muscles contracting or relaxing. Typical approaches to determine motor torques that will lead to a desired behaviour include setting them manually, i.e. on a trial-and-error basis, or specifying them by mathematical equations. Neither of these are intuitive to most humans, so most robot behaviours are programmed by engineers. However, if an engineer was to design a preprogrammed housekeeping robot, it would be very hard to program all the possible behaviours the robot could be expected to perform, even in such a limited domain. It is much more cost-efficient to make the robot *learn* what to do. This would allow the robot to adapt to its human owner, and not the other way around. Since humans easily learn new behaviours by imitating others, it would be ideal if humans could use the same technique to transfer motor knowledge to robots. I believe research in this area could be of great help to bridge the human-robot interaction gap that currently exists, so that you could have truly intelligent robots that could assist people in everyday life.

To understand imitation learning, knowledge of psychology and neuroscience is required. The research in this thesis has taken an interdisciplinary approach, studying the desired mechanism on both a behavioural and neuroscientific level. I have focused on imitation in a musical setting. The system can both see and hear, and seeks to imitate the perceived behaviour. The application has been to create an intelligent virtual drummer, that imitates both the physical playing style (i.e. the movement of the arms) as well as the musical playing style (i.e. the groove) of the teacher. The virtual drummer will then both look and sound like a human drummer. The research in this thesis presents a multi-modal architecture for imitation of human movements. I have been working on simulated robots due to limits of time and money,

however the principles of my research have been developed in a platform-independent way, so it should be applicable to real robots as well.

PREFACE

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfillment of the requirements for the degree *philosophiæ doctor*, and is organized as a collection of papers. The papers are in Part II, in their original publication format. The research has been conducted at the Department of Information and Computer Science, NTNU under the supervision of Associate Professor Pinar Öztürk. I have also been a guest researcher at the Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, University of London, enjoying a productive collaboration with Senior Lecturer Yiannis Demiris.

ACKNOWLEDGEMENTS

First I would like to thank my supervisor Pinar Öztürk. Her constant drive and motivation has kept me working hard over the years, a key factor for the completion of this thesis. Her supervision has felt more like a collaboration; exploring ideas and research paths together. When doing research, I have greatly appreciated the freedom she has given me.

I would also like to thank Yiannis Demiris at Imperial College, London, where I spent a productive year as a guest researcher. He also believed strongly that I should follow my own research ideas, and let me do so under his guidance. I am very grateful for our fruitful and rewarding collaboration.

My co-supervisors Keith Downing and Ruud van der Weel have also helped my research through discussions and insights, bringing forward my work. A special thanks goes to Ruud, who has let me use his motion tracking equipment several times.

My colleagues and friends at IDI and Imperial have also been a source for inspiration, help and guidance to my research. These are Rune Sætre, Lester Solbakken, Rikke Amilde Løvlid, Boye Annfelt Høverstad, Martin Thorsen Ranang, Diego Federici, Richard Blake, Jörg Cassens (my LaTeX-guru), Magnus Lie Hetland, Anthony Dearden, Tom Carlson, Juan Camilo Moreno, Paschalis Veskos, Simon Butler and Matthew Johnson.

However, this thesis would never have been finished if I was not surrounded by so many wonderful people with whom I share a passion for music, both creatively, performance-wise and artistically. These people are Tor-Morten "Roy Albert" Kvam, Dagfinn "Big D/Randy Royce" Ritland, Luis Della Mea Delucchi, Julie Rokseth (and the rest of the Rokseth clan), Frode "The Package" Thomassen, Ola Nordal, Sven-Arne Skarvik, Magnus "The Magulator" Ritland, Margaret Berger, Jørgen Assar Mortensen, Carl-Gustaf "Sege" Lundholm, Tony Håndstad, Tony André Søndbø, Erik Støre, Eiliv Brenner, Annette Hoff, Rebecca Ludvigsen, Øistein Refseth, Henrik "Mönrik" Sahlin-Pettersen, Anne Guro Hukkelaas Gaustad, Ole Petter "Bobby Overtone" Berg, Kari Røssland, Ian "B-Dawg" Butcher, Tore "T-Bone" Meberg, Rupert Taylor, Marte-Helene Bjørk, my Krambugata crew and my Nonnegata crew. Furthermore, I am very grateful for having such a wonderful family: my mother Miriam and father Svein who put me on this earth and have been showering me

with their love ever since, my sister Kine Louise “Kinelus” Tidemann Mårvik who is probably the person I look up to the most on this planet, my inspiring brother-in-law Marius Tidemann Mårvik, my lovely little brothers Trym and Magnus, their mother Hildegunn Gran Tidemann, her sister Ingvild “Evil I” Kari Gran, her husband Øystein “Brølstein” Vold and the rest of my extended family. I am also very happy to have had such wonderful grandparents as Lill-Margrete and Walter Johan Tidemann, with whom I spent a significant part of my childhood, giving my parents some well-deserved time off.

And last, but not least (but still *très petite et mignonne*): my wonderful girlfriend Pernille Monstad Aga.

CONTENTS

Part I

Research Overview

1 INTRODUCTION

1.1 MOTIVATION

The work presented in this thesis seeks to diminish the human-robot interaction gap. Whereas some Hollywood film-makers tend to draw a dystopian future where robots pose a threat to mankind, I think they can help and augment our way of life. Dangerous tasks could be performed by machines that you could repair if an accident occurred, without loss of life or health. Robots could aid people in their daily life, performing chores so that more free time could be spent doing fun and meaningful things. Hospitals and nursing homes could have robots that would ensure better treatment for the patients. These are but a few aspects where robots could be of great benefit to society.

There are two challenges that must be met before robots can fulfill their potential in human society. Firstly, the way robots are built must be improved. Robots today are brittle, break easily and nowhere near the dexterity of even simple animals. They must become cheaper to build and easier to fix. Preferably, robot technology would employ some biological principles, such as the ability to grow and heal by itself. These obstacles are indeed significant, but focuses mainly on technique and mechanics. The second challenge is (in my opinion) harder and with a potentially higher reward: *Robots must be easier to program*. By programming them, I mean *make them do what you want*. Robots will not permeate human society as long as you need an engineering degree in order to achieve a desired behaviour. You should be able to teach robots motor skills as easily as you do with other human beings. The robot should be able to *imitate* your actions. This is what I have chosen to focus on in my PhD: programming imitative behaviour in robots. In this thesis, the focus has been to implement a virtual drummer, that imitates the playing style of human drummers.

Imitative behaviour can be observed on (at least) two levels. At the low level, you imitate a desired trajectory. This consists of imitating a movement, following the same trajectory through space as that of the demonstrator. Learning to dance is an example; you observe the moves made by the teacher, and do the same. On a higher level you have goal-directed imitation.

Goal-directed imitation manifests itself when the observer is able to imitate the intention of the observed action, even if the goal is not achieved. For instance, if a teacher fails to stack an object on top of another, the observer understands what the intention was, and imitates accordingly by placing the object on top of the other object. Another example would be achieving a goal regardless of the differences in physical appearance. Imagine a robot with one arm that is able to imitate a human movement with two arms. When the human lifts a cup with both arms and puts it on the table, the robot would then be able to imitate the same goal using only one arm, even if the trajectories of the two actions are different.

The research presented in this thesis incorporates both trajectory-based and goal-directed imitation, which will be explained in the next section.

1.2 Overview

The research focus of this thesis is to create a virtual drummer, that is able to imitate the playing style of human drummers. This task requires the use of both visual and audial information, i.e. when seeking to imitate the playing style the imitator must both see and hear the teacher. The fusion of modalities is crucial to the system, it requires an active understanding of how one modality relates to the other to achieve the imitative behaviour. To achieve this goal I have implemented a system that models human musical expressiveness, see figure 1.1.

The system has two external sensory input channels: audio and vision. Like a human, it can both see and hear the motor actions needed to produce the desired sound. The system has two output channels: sound and motor commands. A simplification is revealed: the system will produce both the sound itself *as well* as the motor commands required to produce the sound. This is done so that the physical environment need not be simulated. A simulation of the drums requires physical models of the drums, i.e. properties relating to the tension of the skin, dimension of the drum, which material is used to make the shell, and so on. Instead of focusing on modeling drums, the system produces sound in itself. For instance, when hitting a drum, the system will imitate the arm movement and the resulting sound of hitting the drum itself. This requires a way to represent musical signals, but alleviates the need to simulate the physical properties of the drum. In order to create an intelligent virtual drummer that can both be seen and heard, two subsystems will be developed: a sound system that is able to imitate the drum patterns, and a motor system that is able to imitate the corresponding arm movements.

This simplification makes the architecture capable of imitation of both trajectories and goals. The imitation of trajectories is based on visual input; the architecture imitates the movements that it sees. However, the imitation of movements is intrinsically linked to what the system *hears* since the move-

ment results in sound. In other words, the architecture performs goal-directed imitation as well: the goal is the sound that is the result of the movement. How will these goals differ? Recall that the artificial drummer is designed to be a *groovy* drummer. This implies that the dynamics and timing of hitting a drum will vary over time, since these two factors determine the groove. The architecture must therefore issue motor commands that will yield the trajectory corresponding to the goal of the movement (for instance a certain accentuation of hitting the drum).

In the virtual domain, the simplification of having separate motor and sound systems make sense: instead of employing computationally expensive physical simulators (as would be in the case of generating sound), the usage of a dedicated sound system is a lot more efficient. The movements are still the same, and in a virtual world the end user will most likely not be able to tell the difference between a physically simulated drum and a dedicated sound system, when the drumming sounds the same and the drummer moves in the same way.

This can be taken further: in computer games, perceived realism is more important than implementing a simulator that is 100% true to the real world. However, it is still important to have systems that are capable of human behaviour. The approach in this thesis is to use models of human behaviour, but to make simplifications to facilitate the implementation of the system.

The independence of the motor and sound systems allows these to be developed and tested on their own, i.e. the motor system can be tested on imitating movements (without any corresponding sound to be made) and the sound system can be tested on imitating sound (without any corresponding movement), before combining the two. Since the systems are developed independently, the system depicted in figure 1.1 does not explicitly deal with cases where the simulated robot should only imitate one of the modalities. For instance, if the robot is to imitate only arm movements, it will use only the motor system, and vice versa; this will be known a priori to the experiment by the designer, and is not something that will be determined by the system from one experiment to the next.

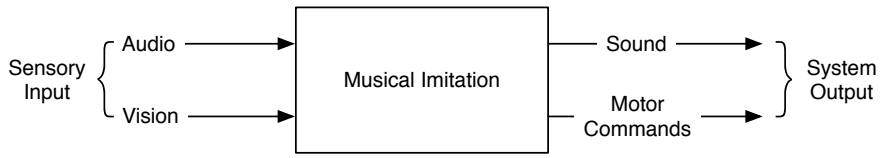


Figure 1.1: Approach to model and imitate human musical expressiveness.

1.3 Research Context

This thesis is focused on imitating human movements in which music is a crucial element, i.e. through dance and drumming. However, the work is not limited to this particular application. The main point is the fusion of two modalities in the system. The two sensory input signals used in the system (audio and vision) could be replaced with other sensory signals - the crucial point being that the two sensory signals are in different reference frames. The difference in the signal streams presents both a challenge and an advantage: the challenge is to fuse the two modalities so that the combined sensory input provides more information than the sum of its parts (i.e. a Gestalt view on the sensory fusion). The advantage is the increased robustness when using two modalities - should there be noise or missing signals in one modality, there is still another signal that you can use to guide your movements. For instance, if one of the loudspeakers should suddenly fail when you are dancing, you can continue dancing (at least for a short period of time) simply by watching the others dance. In the case of dancing, the sound has a crucial role to provide the context for the dancer, i.e. making it aware which part of the movement is supposed to be executed next. Choreographed dances are made up of sequences of body movements that are performed synchronized to music. The music then serves as a time keeper for the dancer, and an indicator of which part of the choreographed movement is up next.

I have chosen to work with vision and sound signals since these modalities are easy to implement on a robot (i.e. by the use of a camera and a microphone) and are crucial sensory streams for a robot that mimics human behaviour. A robot designed to mimic a bat, for instance, would most likely employ the audial and olfactory senses, since these are the two most predominant external sensory inputs to the bat.

In a wider AI research context, this work progresses the field of control systems for situated agents. Situated agents (i.e. autonomous agents that operate in an environment) require a control mechanism that can deal with the challenges of operating in an unpredictable environment. An agent is built to perform one or more tasks. These tasks themselves might be easy to specify, however a control system should be able to deal with sudden changes in the state of both the agent and its surroundings, and do so intelligently, in order to avoid damage to the agent. Review of related work is in section 1.5, the next section describes the research goals in this thesis.

1.4 Research Goals

The research goals represent milestones. The research hypotheses indicate what was predicted to be a result of achieving the research goals.

Research goal 1 *Design and develop an architecture for musical imitation*

that is able to model and imitate human expressiveness (i.e. the groove) when playing the drums.

The overall research goal is to create an artificial intelligent groovy virtual drummer, as explained in the previous sections. To achieve this goal, two subsystems will be developed; one for imitation of arm movements (i.e. a motor system) and one for imitation of drum patterns (i.e. a sound system).

1.4.1 Research Goals Related to the Motor System

Research goal 2 *Design and develop an architecture for motor control and learning that can be used to imitate human movements.*

A crucial aspect of the architecture is how it fuses two modalities: vision and audio. The command fusion happens in the motor subsystem, the sound system operates on its own. Not only is the fusion of vision and audio required for the system to work, it is also hypothesized that this is an *advantage* to the system.

Research hypothesis 1 *Fusing high-level audio signals with visual signals improves the performance of the motor system.*
Test: *Remove the audio signal, and examine the performance of the system.*

In addition to learning sequences of movements, the system should be able to use the acquired motor knowledge in novel situations. In other words, if the system learns a sequence of movements *ABC* it should be able to form novel sequences that contain these movements but in a different order, e.g. *CAB*. The motor system should be able to extract the segments of movements and make use of these segments, regardless of the order of segments that was present in the original sequence.

Research hypothesis 2 *The motor system will internalize motor knowledge regardless of the sequence of movements that was used to train the system.*
Test: *Test the motor system with random permutations of the sequence of movements that was used to train the system, and compare the performance of the system when testing with the original training sequence.*

1.4.2 Research Goals Related to the Sound System

Research goal 3 *Design and develop an architecture for imitation of human drum patterns that capture the human expressiveness of playing.*

The sound system is designed to model and imitate the human quality of *groove*. Such a quality must therefore be evaluated by humans, to verify whether the system is capable of producing human-like drum tracks.

Research hypothesis 3 *The musical output of the system should produce imitations that will be perceived similar to the originals.*
Test: *Use human listeners to compare the original audio with the imitated audio and classify which served as training signal to the imitated audio.*

1.5 Related Work on Imitation Learning

This section describes related work to imitation learning. Since the approach in this thesis is to develop two subsystems (one motor system and one sound system), the related work will be discussed in separate sections.

1.5.1 Imitation of Motor Actions

Developing architectures for motor action starts with an understanding of imitation learning as it happens in nature; imitation learning has been studied extensively before the AI community started implementing the imitative capability. This section is therefore divided in two: the first section gives a brief background on imitation learning in developmental psychology and neuroscience, which has inspired many of the control architectures described in the second section.

Imitation in Developmental Psychology and Neuroscience

Previously, imitation was regarded as a simple mechanism, i.e. "monkey see, monkey do". Over the last decades, imitation has gained more attention as an important form of human cognition. Piaget [1962] was a pioneer on research on imitation in infants. He regards imitation as an ongoing process of adjusting sensory-motor schemas (comprising both motor and perception stimuli) to the external world, stored through repetition. These schemas are then maintained through production and recognition of the same behaviour. Piaget claims these mechanisms (adjusting and maintaining schemas) are what makes an individual intelligent. He defines several stages of development of the imitative capability, and that it is in the late stages that the infant can imitate movements invisible to itself. However, Meltzoff and Moore [1977] discovered that neonates can imitate facial gestures, suggesting that there is an innate mechanism that allows visual and proprioceptional information to be transformed into a matching representation of the child's motor system. Meltzoff and Moore [1997] explains this process called *active intermodal mapping* (AIM), which they believe unifies perception and action. The process is able to detect discrepancies between the observed and produced behaviour, and correct the produced behaviour accordingly. The infant learns to control its body through motor babbling; in robotics this would be similar to sending random motor commands and observing the result. This process begins in utero, and equips the infant with the necessary motor knowledge to be able to correct

itself when it observes action that it wants to imitate. Imitation has also been considered an important aspect of human society, since it allows people to feel that other people are "like me" [Meltzoff, 1999].

Rizzolatti et al. [1996b] discovered neurons in monkeys that would fire both when observing and performing the same action, and called them *mirror neurons*. The same neural activity was found in humans [Rizzolatti et al., 1996a; Grafton et al., 1996] and is by some researchers regarded as possible a neural implementation of the imitative mechanism. This activity is partially found within Broca's area, which sends motor commands to produce speech. Arbib [2002] suggests the mirror neurons enable language in humans, since the ability to imitate each other allows for the development of language through word-games. The use of language must have been a great advantage, since our narrow throats required to produce sounds increase the risk of choking on food. Kohler et al. [2002] brought further evidence to this claim when they discovered mirror neurons to be active also when hearing the sound associated with the action. The mirror neurons respond to both audio and visual cues, but interestingly not when an action is seen and not heard. If the monkey saw a peanut cracked open, but did not *hear* it crack, the mirror neurons would not fire, since it would be a sign that the action had not been successful.

One of the interesting reasons to examine mirror neurons in monkeys is the possibility that there is an equivalent system in humans. However, recent fMRI studies have shown that there is not a direct mapping of activity when observing and performing the same action in humans [Dinstein et al., 2007, 2008a; Lingnau et al., 2009]. The results suggest that even though mirror neurons might exist, they are not the only neurons active during observation and execution of movement, calling for a more nuanced view of a potential mirror neuron system.

However, understanding a possible mirror neuron capability is important, since it is believed that a dysfunctional mirror neuron system is the cause for autism spectral disorders [Williams et al., 2001]. Gallese and Goldman [1998] believe mirror neurons enable *mind-reading* in humans, i.e. attributing beliefs, hopes, desires etc. to other humans, based on how you would react if you were in the same situation. Since the mirror neurons allow for an inner simulation of the actions you observe (i.e. the same firing pattern as when observing an action), they can also provide an inner simulation of another person. Later studies seemed to confirm the hypothesis that a dysfunctional mirror neuron system caused autism spectral disorder [Oberman et al., 2005; Dapretto et al., 2005], however this has also been met with skepticism due to lack of evidence [Dinstein et al., 2008b]. It is safe to say that the current concept of a mirror neuron system analogous to that found in the monkey is controversial in the neuroscientific community.

However, modeling the imitative capability is clearly of great interest to computational neuroscientists. Wolpert et al. [2003] describe a model-based approach for explaining imitative behaviour in humans, using inverse models

(i.e. controllers or behaviours) to produce the desired action, and forward models (i.e. predictors or estimators) to predict their outcome, an ordering that has been suggested to exist in the brain based on fMRI studies [Imamizu et al., 2004]. This will be further discussed in the next section.

Imitation Architectures in AI

Imitation learning has gained considerable interest from the artificial intelligence community over the last years. The work on imitation learning is closely related to that of building a control architecture for an embodied agent. In fact, they can be seen as the exact same thing - it is the approach to *learn* that separates the two (albeit very closely connected) fields. Architectures for imitation learning explicitly state that the agent learns by observing others, producing the same motor actions. Imitation learning can thus be seen as a more specialized branch of research on architectures for motor control and learning. This section will describe both fields.

Brooks [1986] describes the subsumption architecture; a hierarchical modular approach where the modules are designated with different tasks. The low-level modules are responsible for short-term critical tasks, such as obstacle avoidance, whereas the high-level modules implement goal-directed behaviour. The low-level modules can override the high-level behaviours, since they represent behaviours that keep the robot safe from harm. The designer of the system determines how the modules are able to override each other, as well as the capabilities of each module. Maes [1990] distributes behaviours using competence modules. These competence module are connected and spread activation throughout the network, based on fulfilling certain pre- and postconditions that are defined by the designer of the network. Some modules only activate other modules, whereas some issue motor commands. Gaussier et al. [1998] describe a modular approach where both the structure and functionality of each module is predefined. The modules represent the different sensorimotor processing stages, such as perception, recognition and action selection. Similar to this architecture is the work of Matarić [2002], which proposes an architecture for learning by imitation that explicitly deals with visual processing and motor behaviour in separate modules.

Jordan and Rumelhart [1992] tackle another important aspect of learning actions: namely those where only the outcome of the actions are available to the learner, not the actions themselves. Jordan and Rumelhart use the following example: a basketball player that learns to shoot can only use the visual information of the ball flying through the air to correct its own muscle commands. Supervised learning is dependent on a teacher signal, and Jordan and Rumelhart showed how a supervised approach could be used when there is no *explicit* teacher signal, only consequences of the actions. By using an internal model of the consequences of the actions (i.e. a forward model), the forward model can be used to transform the error signals in sensory space

to error signals of the motor space. This allows for particular solutions to be found in the action space, as opposed to direct inverse modeling (i.e. not using a forward model, only an inverse model) where the input/output relationship of the environment is reversed to train the inverse model - this is not necessarily a correct inverse model, since there are many ways to achieve a desired goal. This work forms the basis for Wolpert's model (see Haruno et al. [2001] for an implementation). Similarly, Demiris and Hayes [2002] employ an architecture with paired inverse and forward models. What separates these architectures from Jordan and Rumelhart is that they use *multiple* paired inverse and forward models to imitate motor actions. The inverse models learn behaviours, and the forward models predict the outcome of these behaviours. Based on these predictions, the best suited inverse model for the task is chosen. The approach of using multiple paired inverse/forward models differ from the approaches of Brooks and Maes since the latter uses inhibition as a control mechanism among modules, whereas a multiple paired inverse/forward approach coordinates which inverse model is best suited to control the robot based on the performance of the forward models. Another important feature of using forward models is the focus on prediction, which is not present to the same extent in the architectures of Brooks and Maes.

The mixture of experts (MOE) architecture [Jacobs et al., 1991] is related to the inverse/forward model approach. It divides responsibility between competing neural networks using a gating network to distribute the responsibility between the competing networks.

This section has so far described modular approaches for motor control and learning. Some researchers focus solely on neural network architectures. Tani et al. [2004] implement mirror neuron activity in a recurrent neural network. Special neurons (dubbed "parametric bias") self-organize to specific activations both when producing and observing an action. It is also possible to manually activate the parametric bias neurons, and the network will then generate the corresponding activity. Cangelosi and Riga [2006] uses a neural network to ground symbols from actions; the student learns actions and corresponding symbols by observing the teacher. The robots play language games, using imitation to ground the symbols of the actions, an area of research which is also heavily influenced by Steels [2003]. Billard and Hayes [1999] has a similar approach, where a single recurrent associative network (with no hidden units) learns to label symbols to sensory perception. A student robot imitates a teacher robot, acquiring sensory data and symbols at the same time, storing the information in the network using Hebbian learning.

1.5.2 Imitation of Sound

This section will discuss approaches to modeling user-specific variations of musicians. This is an active field of study within the AI community, and particularly on the playing style of pianists. Saunders et al. [2008] model the

playing style by looking at changes in tempo and loudness, using string kernels to form the representations. Tobudic and Widmer [2005] use first-order logic to represent tempo and dynamics, and then a clustering algorithm to group similar phrases. This representation describes how a pianist would play a classical piece of music. Pachet [2002] uses Hidden Markov Models to create an instrument called "The Continuator" that plays with musicians in real time. The human pianist plays a phrase, and the Continuator plays a second phrase that is a *continuation* of the first phrase. The Markov Models represent the probability that a note will follow the other, focusing on *melodic signature* instead of tempo and dynamics. de Mantaras and Arcos [2002] use case-based reasoning to imitate the expressive qualities of how the music should *feel*, i.e. joyful or sad, and to change the tempo of the song but still maintain the musical expressiveness [Grachten et al., 2006]. Raphael [2003] has a system called "Music Plus One" that models user-specific variations in tempo when playing a piece of classical music. Classical musicians interpret the score and vary the intensity and tempo of the notes. This is why MIDI[1] playback of classical pieces sound very machine-like. This system allows soloists to practice along with the computer, and the system is then able to follow the score of the soloist during changes in tempo and dynamics. Hoover et al. [2008] use an evolutionary approach to generate drum tracks based on melodic input. The user of the system selects which rhythm figures should be allowed to evolve, and new rhythm figures are created in an evolutionary manner. The evolved networks vary dynamics as well as beats, but not timing, which is an important part of human expressiveness. Hoover et al. does not *model* human expressiveness, the system only tries to mimic a natural feel by introducing variations. The user directs the randomized search (which lies at the core of all evolutionary algorithms) towards a desired result.

1.5.3 Robot Drummers

This thesis has a focus of modeling human expressiveness and using these models to imitate human behaviour. Implementing a virtual drummer is the field of application for the research. This section will review other robot drummers that have been built for research purposes.

Hajian et al. [1997] built a robotic hand that was able to play drum by modulating the grasp force in the hand to overcome bandwidth limitations of the control system. This was done as a way to demonstrate how human drummers are able to play faster than the delays of the neuromuscular system, showing how impedance modulation can be used to overcome limitations of slow actuators. The control system was set up as a lumped-element second order model, using springs and dampers to represent the fingers, drum stick and head. The focus was not on building an intelligent model of the drumming

[1] Musical Instrument Digital Interface, a protocol for electronic music equipment to communicate in real-time.

behaviour, but to mimic how humans effectively vary the stiffness of the drum stick by modulating grasp force, increasing the frequency of the drumming.

Schaal [2003] has developed a theoretical framework for specifying movement trajectories for robots, based on the idea that movement is built up of dynamic primitives. These primitives can then be combined to create rhythmic and discrete movements. To demonstrate rhythmic movements, a robot was set to perform a drumming task. The primitives are made up of nonlinear differential equations, and provide a mathematical approach to generating motor control. Related to this work is the research by Degallier et al. [2006], who use switching and superposition between discrete and rhythmic movements. By using oscillators, movement trajectories are generated on-line for a humanoid robot. The humanoid robot plays the drums according to a score given to the system, and the corresponding arm trajectories are produced from the score using the oscillators. This allows the system to be robust towards perturbations of the trajectories.

The Haile drummer created by Weinberg and Driscoll [2006] plays the Native American Pow-wow drum. It has both a database of learned patterns, as well as the ability to imitate and modify patterns when interacting with human drummers. The control mechanism consists of using an environment called Max/MSP, which makes it easy to implement beat detection, frequency analysis and other operations necessary for musical interaction and performance. The Haile robot was later extended to play the xylophone, using a genetic algorithm to generate novel melodies while playing [Weinberg et al., 2008].

Crick et al. [2006] focus on the social interaction aspect of drumming. Their robot, Nico, synchronizes to a conductor and another drummer. It fuses auditory and visual perception to keep time with the conductor and the human drummer. The conductor and drummer can then change the tempo of their drumming, and the robot will change accordingly. The patterns played by the drummer are very simple; one beat for each gesture made by the conductor. However, the generation of complex drum patterns is not the focus of research, it is instead studying how robots can synchronize in social tasks. This work is an extension of the research by Williamson [1999], which employs oscillators in a drumming task to exploit the natural dynamics of the robot system when performing rhythmic movement. This adaptation was made possible since the oscillators could make use of feedback sensor signals (i.e. hearing the sound produced by the drum).

Konno et al. [2007] focus on how impulse forces can be used to overcome torque limitations of the actuators. A humanoid robot was used to play a Japanese drum that requires large impulse forces. Their goal is to formulate an impact dynamics that can deal with the various issues that occur when large impulse forces are used (as opposed to static forces), which are crucial for stability and security of robot systems. Trajectories for the arm movements were calculated using cubic splines, and the robot was able to play along to a

musical score.

How do these robots compare to the research presented in this thesis? All these robot drummers have two things in common: 1) their expressiveness is limited since they are implemented using physical robots. It is obviously not fair to compare physical robots to simulated ones in terms of dexterity, but it is one of the advantages of using a simulated robot. 2) These robot drummers are not concerned with the cognitive aspect of imitation, i.e. seeking to implement a biologically inspired learning mechanism, as is the case with the research presented in this thesis. The robot drummers mentioned here have more practical solutions to generating trajectories. The research presented in this thesis is concerned with the biological motivation for implementing artificial intelligence, and therefore more emphasis is put on the underlying principles, instead of choosing a more practical approach.

RESEARCH DESCRIPTION

2

This chapter elaborates upon the research done throughout the PhD period. This work is supported by the publications in part II, and based on the research goals and hypotheses in section 1.4. This chapter will present a deeper description of the research; the finer details are left in the supporting papers.

Figure 1.1 shows the overall approach to implementing an imitative architecture. It receives audial and visual sensory inputs, and produces sound and motor commands to imitate the perceived stimuli. In order to build such a system, it is decomposed into a motor system (section 2.4) and a sound system (section 2.5). These systems were developed to function on their own, so that their particulars could be studied on a low level. After the development of these separate subsystems, they were combined to achieve the overall goal of creating an imitative agent in a musical setting. This final step is described in section 2.6.

2.1 RESEARCH APPROACH

The first AI systems were based on symbol manipulation and logic. Even though such systems have had some success in implementing artificial intelligence, there is now a strong tendency amongst AI researchers to investigate and be inspired by how nature implements intelligence [Clark, 2001; Beer et al., 1997; Izhikevich, 2006]. Debating *what is intelligence* is beyond the scope of this thesis, but to implement a biological phenomenon (i.e. intelligence) it makes sense to use techniques that are based on the same principles. Artificial neural networks inspired by the neural networks present in our brains, have been used extensively to implement artificial intelligence in this PhD (in addition to other techniques, as described in section 2.2). Common to all these techniques is that they learn from low-level stream of data, i.e. they are *machine learning* approaches. These techniques have been chosen because of their robustness to noise, their strong history in similar applications, and the appealing idea that the world can be its own model. Instead of manually specifying the sensor space in which the system operates, this can be discovered by the system itself, and it is able to extract salient features

that might even be hidden to the designer of the system.

Another important biological principle is also used: namely that of *self-organization*. Both the motor and sound systems are developed so that the system should self-organize the way it works. The designer lays out the overall architecture, but the system is also designed to self-organize when solving a problem. This design choice is made to relieve the designer of possible problems that might not be obvious before the system is deployed. For instance, certain parameters such as size (i.e. neural resources) might not be evident before the system is tested. Instead of hard-coding such constraints, the system is allowed to sort out such parameters on its own. The system will incorporate several modules with the same functional areas (i.e. sensorimotor perception and action), and self-organize how these modules control the robot. Another advantage with a modular approach is its ability for redundancy, which is crucial for robust intelligent systems [Pfeifer and Scheier, 2001]. The machine learning techniques (section 2.2) are similar but not identical; they are able to model the perceived sensory stream without being told *how* to do it, but they do not change their own organization. The principle of self-organization is thus at a higher level; i.e. how clusters of these low-level modules are organized to solve a problem.

2.2 AI TOOLS

The desire to create intelligent machines has probably existed since humans started using tools to make their lives easier. It was in 1956 that John McCarthy coined the term "artificial intelligence"[1]. The field had tremendous success in its first years, and predictions were made of how AI would surpass human intelligence within years. However, these predictions failed to materialize. Most AI research at the time worked in a purely *symbolic* fashion, where the use of logic and rules were the framework for implementing AI. It became clear that these approaches, albeit efficient and suitable in certain situations, had its computational limitations when the size of the problems grew. Focus shifted more towards using systems inspired by biology, since humans easily solve many of the tasks effortlessly that were computationally intractable for computers. This divide is roughly what separates "Good Old Fashion AI" from "nouvelle AI", occurring at the late 70s/early 80s. The techniques used in my research stem from the latter category. I mention only the techniques I have used - this is by no means an exhaustive list of machine learning algorithms.

2.2.1 Neural Networks

Neural networks are inspired from how the brain is organized into many small and simple processing units, that together make up a powerful process-

[1] http://news.cnet.com/Getting-machines-to-think-like-us/2008-11394_3-6090207.html, retrieved 11th of March, 2009.

ing element. Each unit receives input signals, and uses a *transfer function* to compute its output. The transfer function determines how the units respond to its input signals; there are many variations, but most commonly used are non-linear transfer functions such as the sigmoid function. Rosenblatt [1958] developed the perceptron, which had some success but was limited due to having only an input layer and an output layer; this simple organization of neurons can only solve problems that are linearly separable. Werbos [1974] developed the backpropagation algorithm for perceptrons with multiple layers, increasing the popularity of neural networks, since they now could be trained easily. (Note: there exists a plethora of neural network architectures. I will present the ones I have used in my research.) I have used neural networks to implement the different models in my work on motor and sound imitation.

Recurrent Networks

Recurrent neural networks (RNNs) are similar to multi-layer perceptrons, but they also have recurrent connections at the hidden layer, also known as Elman networks [Elman, 1990]. This introduces sequential memory to the network, since it has an internal state that influences the output of the hidden layer. Multi-layer perceptrons (also known as feed-forward networks) do not have an internal state, and therefore cannot model sequences in the same way as RNNs. However, since the RNNs are trained using the backpropagation algorithm, they tend to be less stable during training compared to feed-forward networks.

Used in: An earlier implementation of the motor system, see figure 2.1 and paper A.

Echo State Networks

Echo State Networks (ESNs) utilize the same basic principles as those of multi-layer perceptrons, with one defining difference: the input layer weights are generated at random, and remain fixed throughout the training of the neural network [Jaeger and Haas, 2004]. The training of the output layer is then reduced to a simple regression problem, which is quickly solved (most easily by calculating the pseudo-inverse matrix of the hidden layer states and the desired output states throughout a sequence). The randomly generated input layer transforms the input sequence into a higher dimension, where it becomes easier to separate the different data points. To achieve this, the hidden layer is typically bigger than what is used for RNNs. In applications where you would use 30-50 neurons in the hidden layer of an RNN (the number is determined by intuition, experience and how long you are willing to wait for the training process to end), you would start out with hundreds or

thousands of neurons in the hidden layer of the ESNs. This would be prohibitive with RNNs, since the training time would simply be too long. With ESNs, the training time is greatly reduced, so the cost of adding more nodes to the hidden layer is negligible. In fact, this is one of the reasons ESNs work so well - adding more neurons leads to a higher dimensionality in which the problem can be decomposed. The increased computational power and reduced training times greatly facilitates the development and testing phase.

Used in: The motor system, see figure 2.1 and paper B. ESNs replaced the RNNs used in earlier versions of the motor system. ESNs were also used in the sound system, see figure 2.7 and paper D. Subsequently, ESNs were used in the combination of the motor and sound systems, see figure 2.11 and papers E and F.

2.2.2 Hidden Markov Models

Hidden Markov Models (HMMs) are used to model sequences of actions, when you do not know their underlying probabilities [Baum, Leonard E. et al., 1970]. It is these probabilities that are *hidden*. Once these probabilities have been calculated, the model can be used to generate new output sequences that will be similar (but not necessarily identical) to the observed sequences.

Used in: The sound system, in figure 2.7 it would be in place of the "Sequence ESN". See paper C.

2.2.3 String matching

Searching for supermaximal repeats is a string matching algorithm used in computational biology to find sequences of genes [Gusfield, 1997]. A supermaximal repeat is a recurring substring that is not a substring of any other pattern. For instance, in the string (2.1) **abcq** is a supermaximal repeat, whereas **abc** is *not* since it is a substring of **abcq**.

$$S = eg\mathbf{abcq}xw\mathbf{abc}zi\mathbf{abcq}lm \tag{2.1}$$

Used in: The sound system to search for recurring drum patterns. In figure 2.7, it is part of the "SHEILA analysis". This was used in the papers on the sound system alone (papers C and D) as well as when the motor and sound systems were combined (papers E and F).

2.2.4 Why these Techniques Were Chosen

As mentioned in section 2.2, choosing approaches to implement intelligence inspired from how nature implements intelligence (i.e. neural networks) seems

like a good idea. However, this was not the sole reason neural networks were used. Neural networks were chosen also based on their long history in robot control systems. Being tolerant to noise is a crucial ability when designing robot controllers, in addition to being able to model sequences by having memory. However, it should be noted that this approach is not completely different from the symbolic approaches it sought out to replace. The synaptic weights can be seen as encoding rules and logic, albeit on a more distributed level. Neural networks are designed to discover these relationships on its own, instead of having to specify them. This is an apparent strength, but there are still many variables that need to be set in order for the neural network to learn, e.g. the size of the hidden layer, the learning rate, what transfer function to use, how fast the internal memory of the nodes should dissipate, and so on. Setting these variables requires experience; neural networks are not a silver bullet for all modeling problems. Indeed, other approaches would also be well suited to implement imitation learning. Case Based Reasoning (CBR) is one such technique which has already been used to generate expressive music performances [Arcos et al., 1999]. CBR is an AI problem solving approach where previous similar solutions are reused to solve new problems. The approach includes learning from solved problems, expanding the knowledge base for each solved problem. Arcos et al. analyze expressive musical performances which are then stored as cases. The cases contain high-level musical descriptions of the performance, and these descriptions are then reused to generate musical performance in a particular expressive manner. In the research presented in this paper, CBR could have been applied; a case could have represented information about how to plan certain movements. However, the cases are not discovered by searching previous experiences, since they are not defined explicitly, but instead distributed across several neural networks. The cases are reused and retained though, so a CBR-approach should be feasible. CBR was not chosen initially because of an interest in combining neural networks with inverse/forward models (see section 2.3), and not immediately recognizing the possible link, the CBR route was not taken.

Using HMMs and string searching algorithms were chosen for more pragmatic reasons. HMMs are designed to model sequences of discrete events, and were therefore well suited to be used in the groovy drum machine. The string searching algorithm was a necessity for the system to be able to extract the recurring patterns.

2.3 Concepts from Control Theory

The motor system of the architecture employs concepts from control theory. The most important notions from control literature will now be explained.

State The state is a description of the system at an instance of time. The

current state at time t of the system is written as x_t. The desired state is the goal state of the system. It is one timestep ahead of the current state, written as x'_{t+1}. The tick ($'$) indicates that it is a desired state. Predictions of the state (i.e. what will state x be at time $t+1$) are written as $\hat{x}_{t+1}$, where the hat symbol ($\hat{}$) indicates that it is a prediction of the state.

Motor commands The motor commands are signals sent to the motor actuators (also called plant in control theory) at time t, written as u_t.

Inverse model An inverse model is a controller or a behaviour. It achieves a desired state x'_{t+1} given a current state x_t by outputting motor commands u_t.

Forward model A forward model is a predictor or an estimator. It predicts the next state $\hat{x}_{t+1}$ based on the current state x_t and the motor commands u_t applied to the system.

The inverse and forward model are *paired*, so that the output of the inverse model is fed into its paired forward model. The forward model is then able to predict the outcome of the motor commands, before sensory input about the actual outcome is received. Such an arrangement can compensate for delays in the sensor system. Also, when there are multiple paired inverse/forward models, the performance of the forward models can be used to arbitrate between which inverse models are best suited to control the robot. This will be further elaborated upon in section 2.4.1.

Used in: The motor system as building blocks of the architecture, see figure 2.1. These concepts were thus used in the papers that focused solely on the motor system (papers A and B) and when the motor system was combined with the sound system (papers E and F).

2.4 Imitation of Human Movement: Dancing

The development of an architecture for motor control and learning was done to achieve research goal 2. The architecture will self-organize the control of the robot using a modular approach, as mentioned in section 2.1. The system will see movements and issue motor commands that will yield the same behaviour in an embodied agent. It is the motor part of the overall approach seen in figure 1.1. Imitation of human dance movements has been selected as the area of focus. Imitation learning is evident when students learn to dance from teachers. Another important aspect is the multimodal nature of the problem: the student both sees the teacher and hears the accompanying music. The imitator uses both visual and audial clues in order to generate the intended behaviour of its motor system. This research is published in papers A and B.

2.4.1 The Multiple Paired Models Architecture (MPMA)

The motor architecture developed in this thesis is based on the inverse/forward model pairing by Jordan and Rumelhart [1992], used in multiple instances as in the research by Wolpert et al. [2003] and Demiris and Hayes [2002]. A crucial aspect is how the pairs of inverse/forward models self-organize when controlling the robot. The architecture is called the Multiple Paired Models Architecture, abbreviated MPMA. The reasons for choosing this approach will now be explained. There is neuroscientific evidence that such an organization exists in the brain [Imamizu et al., 2004]. Schaal [1999] regards model-based approaches as those of Wolpert and Demiris as the most suitable way to implement imitation learning in agents. It is also based on an approach that is well understood in the control literature [Jordan and Rumelhart, 1992]. Furthermore, it addresses the design choice of building an architecture that self-organizes the control of the robot, using a modular approach. How does such an approach compare to the other solutions mentioned in section 1.5.1? The subsumption architecture of Brooks [1986] has a heavy reliance on the designer, both to define the capabilities of each module, and how they are able to override each other. In this regard, it does not offer a self-organizing approach. The architecture of Maes [1990] uses several modules with specific competences which can be regarded as similar to a multiple paired approach, but the modules do not self-organize how they influence each other - the lists of pre- and postconditions necessary for activation must be specified by the designer. The architectures of Gaussier et al. [1998] and Matarić [2002] are "holistic" solutions to learning motor skills, with separate modules for attention, recognition, learning and motor activation. Such an approach was not chosen due to the view that sensory perception and action are closely related [Demiris and Hayes, 2002; Wolpert et al., 2003]. Therefore, sensory perception and action should not be separated into different modules. The works of Tani et al. [2004], Cangelosi and Riga [2006] and Billard and Hayes [1999] employ a central neural network to learn motor skills and label them accordingly. These approaches are not self-organizing on a modular level, since they use only one neural network (not to be confused with the self-organization that occurs at the neuronal level). Tani et al. [2004] is an exception since he adds another network as a linguistic module, and connects it to the motor module. These can be said to be self-organizing on a modular level, however the self-organization occurs between two modules of different modalities (language versus motor control). The architectures of Wolpert et al. [2003], Demiris and Hayes [2002] and Jacobs et al. [1991] employs self-organization between modules that work in the same coordinate space. This is an advantage, since more neural resources can be allocated to learning new concepts in an adaptive manner, and it also allows for redundant coding of motor knowledge. For this reason, approaches such as that of Tani et al. [2004] (and also Gaussier et al. [1998] and Matarić [2002]) are not chosen.

The details of how the MPMA is formed by using principles from Wolpert et al. [2003] (named MOSAIC) and Demiris and Hayes [2002] (named HAMMER) will now be explained. Both architectures use multiple instances of inverse/forward pairings, and the performance of the forward model is used to choose which inverse model should control the robot. However, the predictions of the forward model can only be measured once the predicted state has happened. MOSAIC introduces a *responsibility predictor* (RP) as a remedy to this problem. There is one RP for each inverse/forward pairing (from now on referred to as a module), which predicts the suitability of the module to control the robot based on contextual information. The context is a supplementary signal, that is more high-level than the current state of the system. Wolpert use the example of an empty or full cup as contextual information; if you know that it is full it will help you select the correct inverse model to lift the cup, before any action has been taken. In the research presented in this thesis, the context signal represents high-level musical signals. It can be thought of as a sequential musical marker; like the melody in a song.

In ambiguous situations, a system relying only on the forward models would have to try out several solutions before finding the correct inverse model. The combination of the RP and forward model allows the formation of a *confidence signal* λ_t^i to consist of both a *prior* prediction and a *posterior* measurement of suitability of the inverse model to control the robot. The lack of this mechanism in HAMMER is a drawback when it comes to building a predictive architecture.

Structurally, HAMMER and MOSAIC have two modes of operation. HAMMER has the passive mode, in which novel movements are acquired, and the active mode, where behaviours are tried out in parallel to find the one best suited to match a desired trajectory. In MOSAIC, the ordering of inverse/forward models are different depending on whether it is recognizing an action (the output of the inverse model is given directly to its paired forward model) or producing an action (the sum of the outputs of the inverse models are used as input to each forward model). To me, it is not clear *why* you would need an active and a passive mode for an architecture for motor learning by imitation. Grezes and Decety [2001] point to several fMRI studies reporting activation of the primal sensorimotor cortex and the dorsal premotor cortex during mental simulation, as well as action production (albeit the activity was higher during action production compared to mental simulation). Grezes and Decety suggest that action observation and action simulation requires inhibition of motor output, since the motor systems are active during action observation and simulation. Piaget [1962] suggests a similar mechanism, he discovered that when children were about 16 months old, they would often imitate without knowing they were imitating, i.e. when they observed an action, they produced motor commands that immediately tried to achieve the same behaviour.

In view of the architectures of HAMMER and MOSAIC, this argues for a

unified architecture for both action production and recognition. When observing an action, the motor system could be regarded as working as when producing the same action, the only difference being that the motor system is inhibited. This is why the MPMA, see figure 2.1, has the same organization both for action production and recognition. In the research presented in this PhD, the architecture has only been used in *active* mode, but it is designed to be used in passive mode as well, with the difference being that the motor output is inhibited. The details of the MPMA will now be described.

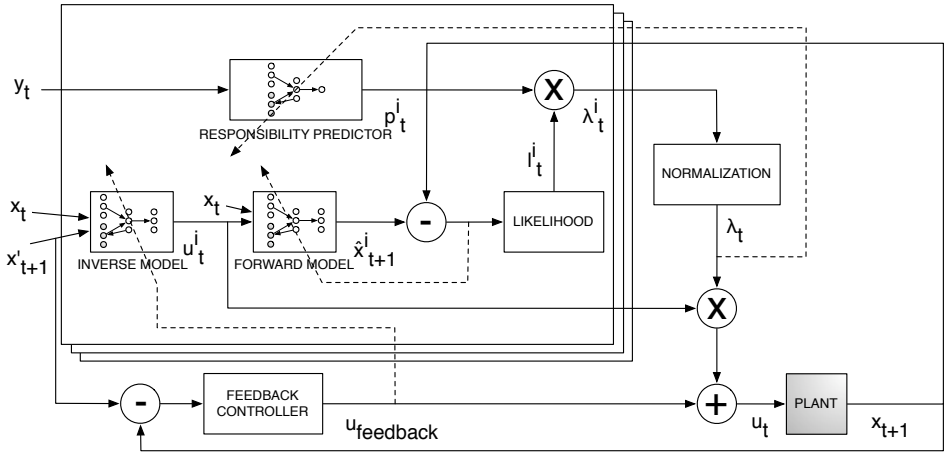


Figure 2.1: The MPMA, inspired from Wolpert et al. [2003] and Demiris and Khadhouri [2006]. Details described in the text.

The Inverse Model

The inverse model is a *controller*, or a *behaviour* in behaviour-based AI terminology [Bryson, 2003]. It has two input signals: the desired state x'_{t+1} and the current state x_t. It learns to output the motor commands u^i_t to achieve the desired state.

The Forward Model

The forward model is a *predictor* or an *estimator*. The forward model predicts the next state of the system $\hat{x}^i_{t+1}$ based on the current state x_t and the output from its paired inverse model, u^i_t.

The Responsibility Predictor

The responsibility predictor (RP) is another *predictor*. The RP uses context information y_t to predict p^i_t how well the module is able to control the robot.

In situations where the current state is ambiguous, the context information helps the architecture choose the best module without having to try out several modules and converge towards the correct one. This would lead to less performance, since non-optimal modules would influence the control of the robot.

The Likelihood Model

The likelihood model estimates how well the forward model predicts. The predictive ability of the forward model is easy to compute; it is simply the difference between what the forward model predicted and what was the actual state at the next timestep, i.e. $|\hat{x}^i_{t+1} - x_{t+1}|$. The likelihood model expresses this difference as a *scalar*, which allows it to be computed with p^i_t to form the final responsibility signal. It assumes the presence of gaussian noise. If $|\hat{x}^i_{t+1} - x_{t+1}|$ is small, the output of the likelihood model will be high. When this difference increases, the output will be low (i.e. it is not very likely that a good prediction was made). The likelihood l^i_t is calculated according to equation (2.2).

$$l^i_t = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-|x_t - \hat{x}^i_{t+1}|^2/2\sigma^2} \tag{2.2}$$

Calculation of λ

The calculation of the normalized λ vector is what distributes control between the modules. The λ signal can be seen as a combination of a *prior* prediction of how well the module performs (i.e. the RP signal) and *posterior* knowledge of how well the forward model performs (represented by l^i_t). Equation (2.3) describes how each element (denoted by superindex i, i.e. λ^i_t) in the λ vector is calculated. The final λ vector is multiplied with the motor commands from the inverse models, which is how distribution of control is effectuated. The λ signal also gates the learning of the models, and serve as a training signal to the RPs.

$$\lambda^i_t = \frac{p^i_t l^i_t}{\sum_j p^j_t l^j_t} \tag{2.3}$$

The Feedback Error Motor Controller

In order to train the inverse model, good error signals are crucial. The feedback error motor command is based on the differences between x'_{t+1} and x_{t+1}, i.e. the goal state the robot was trying to achieve at time t compared to the state it actually ended up in at time $t+1$ [Kawato, 1990]. This pulls the system towards the correct state when bad motor commands are issued, and is used as training signals to the inverse model. In other words, differences in *state*

is used to issue *motor commands*. Finding a solution to training an inverse model is a hard problem [Jordan and Rumelhart, 1992], and this approach guarantees a solution. The motor error feedback command is added to the final motor commands. With increasing performance of the inverse models, the influence of the feedback error controller will decrease.

2.4.2 Simplifications

I have not been working with real robots, relying on simulators as a sufficient substitute. When using sensory input to guide your own motor system, there is the inherent problem of transforming the externally perceived signals to an internal reference frame, called the *correspondence problem* [Nehaniv and Dautenhahn, 2002]. This can be seen as an instance of the *grounding problem* [Harnad, 1990], describing how symbols only become meaningful when they are grounded in the sensory capacities a given system. Given that the drummer is virtual, this simplifies the grounding problem. I have chosen to simplify the transformation of visual input so that it is directly mapped to the motor system of the imitator. Schaal [1999] describes how approaches to implement imitative behaviour are divided into two groups: 1) those who work with the transformation of visual input into a meaningful representation to the agent, and 2) those who assume this transformation has already been done, and that the perception stimuli is ready to be given to a perception-action system. The research presented in this thesis belongs to the latter category. This is an approach taken by other researchers as well [Demiris and Hayes, 2002; Cangelosi and Riga, 2006]. Furthermore, Torres and Zipser [2004] suggest that such a transformation occurs in the brain.

2.4.3 Experimental setup

The experiments were carried out in order to fulfill research goal 2 to test research hypothesis 1. Dance movements were recorded using a motion tracking system called Pro Reflex[2]. Fluorescent markers were attached to the body of the dancer, and five infrared cameras spanned a volume in which 3D coordinates could be recorded. The Pro Reflex system is pictured in figure 2.2.

The robot does not see the stick figure as drawn in figure 2.2, but instead perceives the joint positions mapped to its own motor system, as mentioned in the previous section. The recorded signals can then be directly matched to the imitating robot. The experiment consisted of imitating the dance to the song YMCA by the Village People, a 70s disco group from New York, as shown in figure 2.3. A simulated robot was implemented, and the MPMA served as the intelligence of the simulated robot. Paper A describes the first implementation of the system, where it was set to imitate the YMCA dance. In paper B, the importance of the RPs were examined, both during training

[2] www.qualisys.com

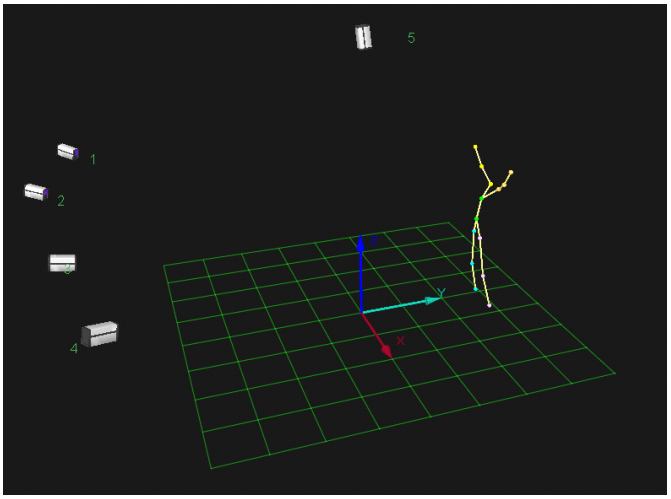


Figure 2.2: The Pro Reflex motion tracking system. The fluorescent markers are attached to the wrist, elbow, shoulder, torso, hip, knee and foot of the dancer. These markers are shown as the colored balls, the stick figure is then drawn based on these balls. The five infrared cameras (4 in front, one hanging from the roof) are able to record the 3D coordinates of the fluorescent markers.

and testing, by repeating the desired movement of the teacher. This paper performs the testing of research hypothesis 1, since the removal of the RPs will make the context signal irrelevant.

The Context Signal

For this experiment, the context signal was a binary coding of the melody of the song. The binary coding can be seen in figure 2.3. The context signal codes high-level melodic structure, like the ones dancers use as cues to a choreographed dance. The context signal was defined to follow the melody of the song, since the melody also indicated the division of movement segments (i.e. the letters), which could be regarded as motor primitives of the dance.

2.4.4 Results

The self-organizing capabilities of the MPMA are reported in both paper A and B. In the latter paper, the desired trajectory was repeated, to see to what extent the λ activations would repeat themselves. Figure 2.4 shows how the system would self-organize during repetition of the movement, and how the modules switch between controlling the robot during the run of the movement. Notice how the λ values switch in accordance with the context

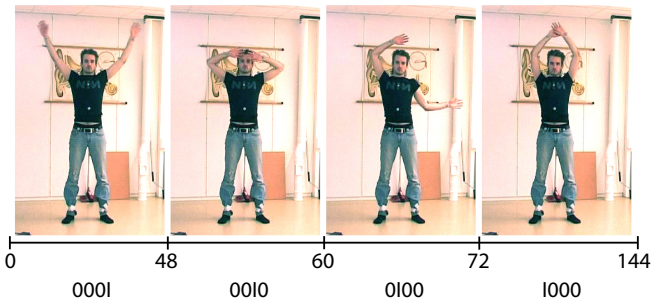


Figure 2.3: YMCA: spelling the letters *Y M C A* using the arms. The numbers show at which timestep the switch from one letter to another occurred, as designated by me. This corresponds to the context signal, seen as the binary vectors below.

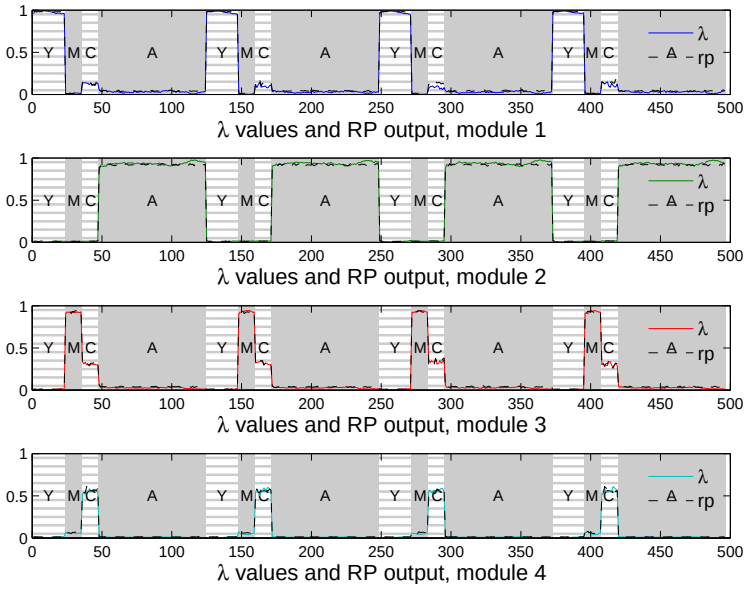


Figure 2.4: The figure shows one of the results of paper B; how the system self-organizes the decomposition of the movement into specific modules, and how they compete and collaborate during control of the movement. The background shows the recurring YMCA context signal, corresponding to figure 2.3.

signal. Furthermore, paper B found the importance of the RPs to be crucial to the system.

2.4.5 Summary, experiment 1

Research goal Create a motor system that can imitate human movements. Research goal 2.

Research hypothesis Using two modalities (vision and audio) will improve the performance of the motor system. Research hypothesis 1.

Results The motor system imitated human dance movements, captured using motion tracking. By removing the sound input to the architecture, performance was found to be less than when both modalities were present.

2.5 Imitation of Recurring Patterns: Drumming

The system for learning and imitating sound was developed and designed in order to meet research goal 3 and to test research hypothesis 3. Drumming is an area where imitation learning also is evident, and a literature search revealed that modeling user-specific variations and imitating them in the drumming domain had not been done before. Current drum sample software (e.g. Toontrack EZDrummer, FXPansion BFD, Reason Drum Kits, Native Instruments Battery, DigiDesign Strike) contain sophisticated software with gigabytes of sound samples. Many of these systems allows for tweaking of parameters that add noise to the system, i.e. as randomization of beats and onset time. This is then supposed to be perceived as human-like, but they have no *intelligent* way of generating human-like drum tracks. The approach to *model* these variations and the ability to generate drum tracks using these models would then be a cost-effective way to having human-like drum tracks in the studio. The main publications of this research are in papers C and D.

2.5.1 Software for Hierarchical Extraction of Drum Patterns in a Learning Agent (SHEILA)

One of the key insights when it comes to modeling drumming patterns, is that there are only two parameters that represents the groove of a pattern (keeping in mind that the pattern itself must be represented): velocity and onset time. Velocity is how hard a drum is struck. Onset time is how much the drum was struck before or after the metronome. A drum machine will keep these parameters constant, which results in a machine-like feel. A human drummer will always vary the velocity and onset time during playing, i.e. introduce *small-scale variations*, see figure 2.5.

Additionally, the drummer will vary the pattern itself, such as playing a break or adding or removing beats. These variations are called *large-scale variations*, see figure 2.6. One of the reasons why drumming is well suited for modeling is due to its repetitive nature. When a drummer plays a certain

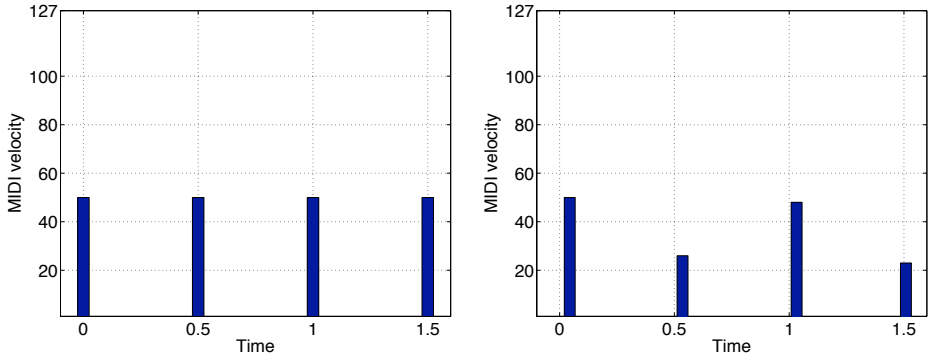


Figure 2.5: An example of small-scale variations. Compare the four drum strokes generated by a computer (a) and that of a human drummer (b). Notice how the drummer (plot b) introduces variations in velocity (seen as vertical displacements) and timing (seen as horizontal displacements from the grid). These small-scale variations constitute the groove of the drummer. The data in plot (b) is from the experiment reported in paper D.

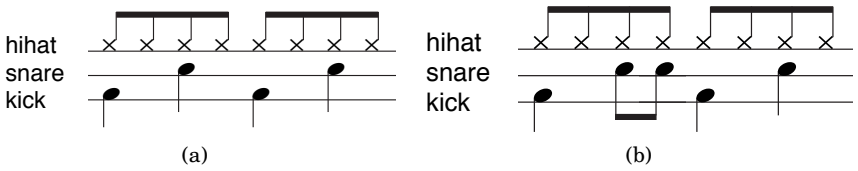


Figure 2.6: An example of a large-scale variation. Plot (b) shows a large-scale variation of pattern (a); another snare drum hit is added. In papers C and D, plot (a) is one of the core patterns that the system learns.

pattern repeatedly, it becomes possible to model these changes in velocity and onset time, in addition to learning the patterns themselves.

SHEILA learns drum patterns recorded from a drum session in a hierarchical fashion, see figure 2.7. The system models both small- and large-scale variations of the groove of the drummer. Once the models of the patterns are acquired, they can be used to imitate the playing style of the drummers. An important point is that the imitations are similar but not *identical*. The model of the drummer can then be used in the studio on different tracks, and the user will know the resulting groove. Instead of hiring the drummer itself, it could simply buy a software plug-in that had a representation of the drummer.

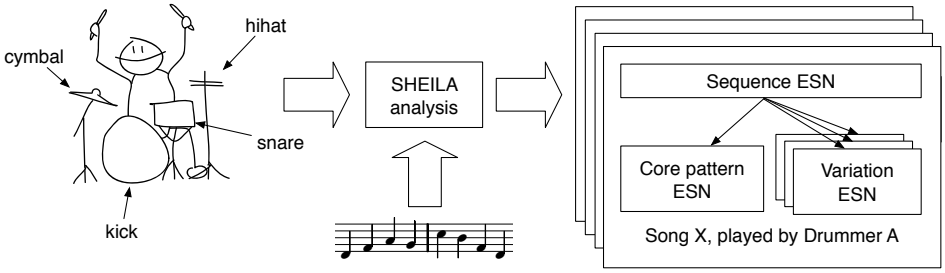


Figure 2.7: The SHEILA architecture. The playing style is extracted form recorded drum patterns, and modeled and stored in the architecture.

SHEILA Analysis

The melody of the song is used to form segments of the song, i.e. verse/chorus/bridge. However, the sound system does not know beforehand what constitutes a certain melodic part (e.g. a verse). This is discovered by searching for supermaximal repeats. Recall that a supermaximal repeat is a recurring pattern that is not a subpattern of any other pattern (see also section 2.2.3). The patterns that remain after analyzing the melody of the song are then the "building blocks" of the song, i.e. the distinctly different melodic parts. After the different melodic segments are revealed, the same search for supermaximal repeats is performed on the drumming data in the song. The most commonly played pattern within each melodic segment is then referred to as the *core pattern* C_x of that part. The patterns that differ from the core pattern are the large-scale *variations* C_xV_y of that pattern. SHEILA uses a bottom-up strategy; from a low-level stream of MIDI data, a high-level segmentation of the song (both in terms of melody and drumming patterns) is found. After all the recurring patterns of the MIDI stream have been identified, the similar patterns are grouped together. These groups can now be used to model small-scale variations of each beat in the rhythm figure.

Storing the Groove

After the data analysis, the sequence of patterns are modeled. In an early version of SHEILA (paper C), the sequence of core patterns and variations were modeled using HMMs (see section 2.2.2). In later versions (paper D), this sequence was modeled using ESNs, see figure 2.7. The same figure also shows how ESNs are used to model core patterns and variations of the core patterns. After learning, the models can be used to generate drum tracks that are similar to those of the teacher. There is one entry in the SHEILA library for each core pattern. The end user specifies for how many bars a specific core pattern should be played, and the Sequence ESN then "starts" the corresponding core pattern and variations, generating the final MIDI output.

2.5.2 Simplifications

In order to facilitate analysis, the drumming was recorded using MIDI instead of audio. A velocity sensitive electronic drum kit will equally well capture timing and velocity of a drummer's groove as an acoustic drum kit. Timing and velocity can be extracted from an acoustic signal, but it would require a more expensive setup with an acoustic drumkit, microphones, audio recording software and hardware, as well as data analysis to extract the timing and velocity of each drum strike.

2.5.3 Experimental Setup

The experiments were carried out to test research hypothesis 3. Using a Roland TD-3, which is a velocity sensitive electronic drum kit, drum patterns were recorded using five amateur drummers, see figure 2.8. Amateur drummers were used because of budget restraints; there was no money available to pay professional session drummers. However, the reader should note that the term "professional" and "amateur" are somewhat misleading. Some of the drummers participating in the experience had been drumming for more than 20 years, and are very skilled drummers. What separates them from professional drummers is that they do not make a living off playing the drums. Being a drummer myself, I do not see how the basic movement trajectories when playing a given pattern would be very different in complexity when comparing a professional to an amateur drummer. In fact, the trajectories of a professional drummer could even be easier to model, since the trajectories could be more stable and well-defined due to the increased hours spent practicing.

The drumming was recorded in MIDI, to simplify data analysis. The recorded drum patterns served as input to SHEILA, and using the learned models, the system was able to generate groovy drum tracks in the style of the original drummers.

Figure 2.8: One of the drummers playing on the Roland TD-3.

2.5.4 Results

Both papers C and D reports successful results from the system, and the produced drum tracks were similar to the original ones. Examples (taken from paper D) can be seen in figures 2.9 and 2.10, and there are MP3 files available[3] that demonstrate the imitative capabilities, confirming research hypothesis 3. Table 2.1 shows how the drummers played the song differently in terms of large-scale variations.

2.5.5 Summary, experiment 2

Research goal Create a groovy drum machine that can imitate the playing style of human drummers. Research goal 3.

Research hypothesis The output of the sound system should be similar to the originals, based on human evaluation. Research hypothesis 3.

Results The sound system generated drum tracks that were similar to the original, but not identical. Both human evaluation (paper C, also performed with different listeners in the auxiliary paper G.1) and statistical analysis (paper D) showed the similarity between original and imitated drum tracks.

[3] http://www.idi.ntnu.no/~tidemann/sheila

Table 2.1: How each drummer played the song in terms of core patterns (C_x) and variations (C_xV_y), indicating the different playing styles of the drummers with respect to large-scale variations. C_xB_y stands for a recurring variation; this is indicated as a *break* (hence the letter *B*). EB is *end break*, i.e. a small variation done at the end of the song. This is the recorded drum data used in papers C - F.

Drummer A	Drummer B	Drummer C	Drummer D	Drummer E
C1V2	C1V3	C1V3	C1	C1V1
C1	C1	C1	C1V2	C1
C1	C1	C1	C1	C1
C1	C1	C1V7	C1	C1V5
C1	C1	C1	C1	C1
C1	C1	C1V4	C1	C1
C1	C1	C1	C1	C1
C1B1	C1B1	C1B1	C1B1	C1B1
C1V2	C1V2	C1V3	C1V1	C1V1
C1V1	C1	C1	C1	C1
C1	C1	C1	C1	C1
C1V5	C1	C1V6	C1	C1
C1V4	C1	C1	C1	C1V1
C1	C1	C1	C1	C1
C1	C1	C1	C1	C1V3
C1B1	C1B1	C1B1	C1B1	C1B1
C2V4	C2V2	C2B1	C2	C2V2
C2	C2	C2	C2	C2V1
C2	C2	C2	C2	C2
C2V6	C2	C2V8	C2	C2V11
C2	C2	C2	C2	C2V10
C2V10	C2	C2	C2	C2
C2	C2	C2	C2	C2V1
C2V9	C2	C2V7	C2	C2V9
C1V2	C1V3	C1V3	C1	C1V1
C1V1	C1	C1	C1	C1
C1	C1	C1	C1	C1
C1V3	C1	C1V5	C1	C1V4
C1	C1	C1V3	C1	C1V1
C1V1	C1	C1V4	C1	C1
C1	C1	C1	C1	C1
C1B1	C1B1	C1B1	C1B1	C1B1
C1V2	C1V2	C1V3	C1	C1V1
C1V1	C1	C1	C1	C1V3
C1	C1	C1	C1	C1V3
C1V1	C1V1	C1V2	C1	C1V2
C1	C1	C1	C1	C1V1
C1V1	C1	C1	C1	C1
C1	C1	C1	C1	C1
C1B1	C1B1	C1V1	C1B1	C1B1
C2V8	C2V1	C2B1	C2V1	C2V6
C2V7	C2	C2	C2	C2
C2	C2	C2	C2	C2V4
C2V6	C2	C2V6	C2	C2V8
C2V4	C2V1	C2V1	C2	C2V2
C2	C2	C2	C2	C2
C2	C2	C2	C2	C2
C2V5	C2	C2V5	C2	C2
C3V6	C3V1	C3	C3	C3V5
C3	C3	C3V5	C3	C3V4
C3V5	C3	C3V4	C3	C3
C3V4	C3V2	C3V3	C3	C3V3
C3V3	C3V1	C3	C3	C3V2
C3V2	C3	C3V2	C3	C3
C3	C3	C3	C3	C3
C3V1	C3	C3V1	C3	C3V1
C2V4	C2V1	C2B1	C2	C2V6
C2	C2	C2	C2	C2V1
C2	C2	C2	C2	C2
C2V3	C2	C2V4	C2	C2V7
C2	C2V1	C2	C2	C2V6
C2	C2	C2	C2	C2
C2	C2	C2	C2	C2
C2V2	C2	C2V3	C2	C2V5
C2	C2V1	C2V1	C2	C2V2
C2	C2	C2	C2	C2V4
C2	C2	C2	C2	C2
C2V1	C2	C2V2	C2	C2V3
C2	C2V1	C2V1	C2	C2V2
C2	C2	C2	C2	C2
C2	C2	C2	C2	C2V1
EB	C2	EB	C2	EB
	EB			

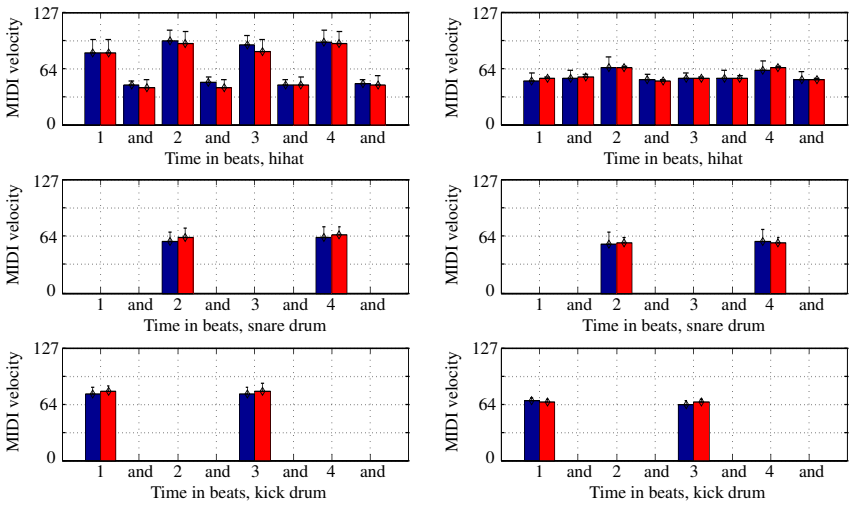


Figure 2.9: Velocity plots for two of the drummers who participated in the experiment in paper D. The blue bar indicates the original data, the red bar indicates the imitated data. Their similarity shows that the system reproduces the same velocity profile.

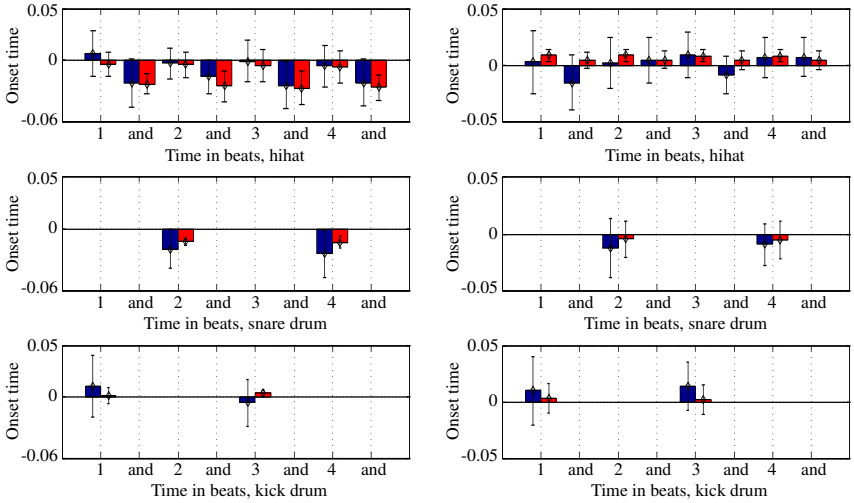


Figure 2.10: Onset time plots, same as in figure 2.9. The similarity shows that the temporal profiles of the drummers have been learned.

2.6 SHEILA: An Animated Drummer

The final stage of research in this PhD was to combine the work with the motor system with the groovy drum machine (both based on learning by imi-

tation) to create a groovy animated drummer. The groovy drum machine will be augmented by being visualized, meaning it can be both seen and heard. The research was done to meet research goal 1. The work on a groovy drum machine focused solely on the sound part; the motor imitation work focused on the arm movements. A drummer also moves its legs, but movement of the arms is what is clearly visible from behind a drum kit. The idea is that SHEILA can be used in a live setting, visualized using a laptop and a projector. Since the drum machine incorporates the motor system previously developed, the combination of the two systems retained the name (SHEILA) of the groovy drum machine. The research is presented in paper E and in paper F. The latter paper is submitted, its status is not known at the time of publication.

2.6.1 Combining Motor and Sound Systems

The combination of the motor and sound systems can be seen in figure 2.11. When combining the groovy drum machine with the motor imitation architecture, the drum machine was designed to be the neural center that defines the groove the agent is to play. The output of the neural groove center was thus the *desired state* of the motor system. In the previous research on the motor system (see section 2.4.1), the desired state had been in the same coordinate frame as the robot itself, so the inverse models had to learn the motor commands to move the arms from x_t to x'_{t+1}. Now, the desired state represents the actual *sound* that the movement is supposed to produce, denoted x'_{sound}. It is still a perfectly viable desired state, only on a higher level compared to a desired state expressed in terms of intrinsic body coordinates, as was previously the case. The desired state would now be (in human language) "hit the drum so hard that this particular sound is produced" instead of "move your arm from this position to that position".

2.6.2 The Predictive Error Controller

When changing the representation of the desired state from body coordinates to desired sound outcome, there was still a need to issue corrective motor commands and train the inverse models. The motor error feedback controller was essential, but it relied on a match between the desired state x'_{t+1} and the actual state x_{t+1}. Now, the desired sound state x'_{sound} could not be used for this purpose. The desired *arm* state could be used to guide the error controller. This could be thought of as a memory of how to move the arm in order to achieve the desired sound output. This desired *arm* state is referred to as x'_{t+1}, since it is the same state used in previous experiments. The difference is that it is not given to the inverse models. Inspired by the cerebellum, the error controller could be a *predictive* error controller instead of being reactive. The cerebellum receives high-level signals of the intentions of the motor actions,

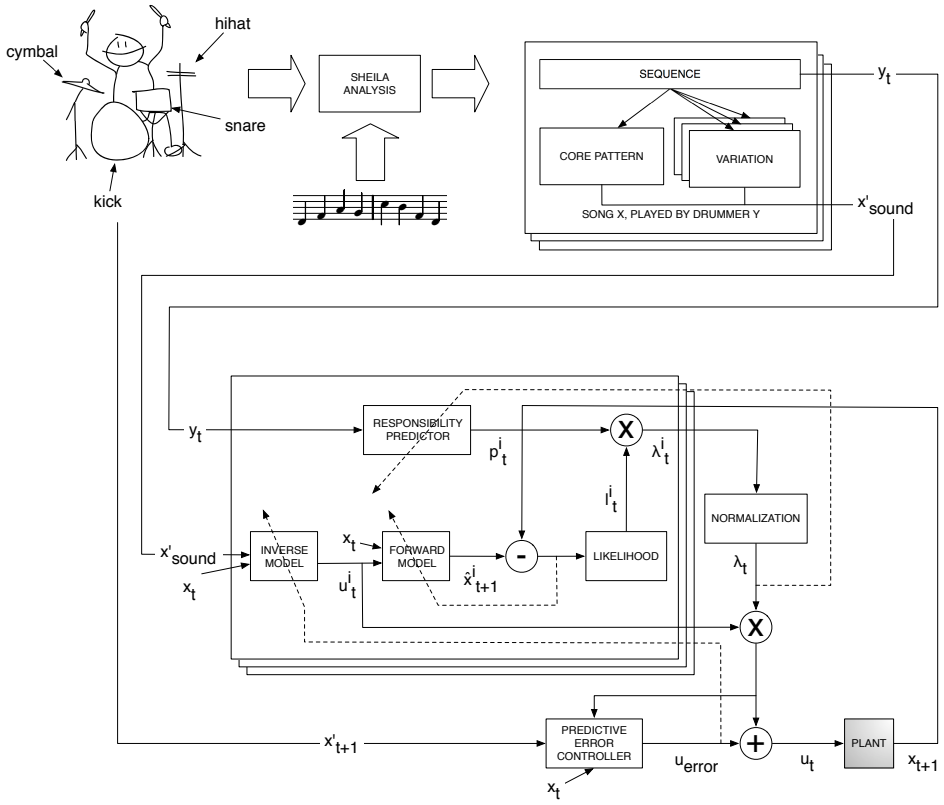


Figure 2.11: SHEILA: The animated groovy drummer. The sound part on the top drives the motor part on the bottom.

and evaluates discrepancies between the intended movement and the current movement and makes last-minute corrections of the motor signals before they are sent further down the spinal cord [Kandel et al., 2000]. By making the error controller predictive, better training *and* corrective signals were generated. In addition to the current state x_t, the desired arm movement x'_{t+1}, it also receives the motor commands from the models u_t, that are supposed to achieve x'_{t+1}. The predictive error controller (PEC) thus predicts the outcome of the motor commands u_t on the current state x_t, compares it to the desired state x'_{t+1} and issues corrective motor commands u_{error} if necessary.

2.6.3 Simplifications

The x'_{sound} signal was not an acoustic representation of the sound it should produce; it was the output of the neural network modeling the rhythm pattern. These neural signals are converted to MIDI, which subsequently produces the actual sound. It is referred to as x'_{sound} since it codes for the actual sound, only on a higher level.

2.6.4 Experimental Setup

During the MIDI recordings for the groovy drum machine, the movements of the arms of the drummers were recorded using the Pro Reflex tracking system mentioned earlier. This provides the desired arm state x' necessary for the PEC to operate. To use the output of the groovy neural center, the target sound state x'_{sound} is linked to the time line of the desired arm state x'. Since the output of the groovy drum machine is simply the beats and how to play them, the desired sound is linked to the time it takes to execute them. Furthermore, the desired state precedes the arm movement with 1/4 note, this is to emulate the behaviour that a drummer thinks about hitting the snare drum in a particular fashion before moving the arm to do so. Since the drumming was recorded at 120 beats per minute (BPM), the sampling frequency of Pro Reflex was 20Hz, the length (in samples) of a quarter note $l_{1/4}$ (i.e. one beat) is thus

$$l_{1/4} = \frac{60 seconds/minute}{120 beat/minute} 20 samples/second = 10 samples/beat \qquad (2.4)$$

The signal for hitting the snare drum will thus be repeated 10 times when linked to the "body timing" time frame. The x'_{sound} signal can be seen in figure 2.12. The system was now started with 10 modules, and modules that were not active were pruned during the experiment. In the research published in papers A and B, the designer had decided (somewhat arbitrarily) how many modules should be present in the architecture, the new approach had the intention that only the *necessary* number of modules would be left after training.

To examine if the order of the target state was important for the system to be able to learn the desired state (research hypothesis 2), the trained multiple models presented in paper E were run (without learning) on a target state that was different from what it was trained on. This was done the following way: a context signal lasts one bar. The sequence of the bars were then permutated randomly to form a new sequence. Ten permutations were performed for each of the trained multiple models. Afterwards, a statistical analysis was run to see if the λ values were recurring to the same extent as was present in paper E. The results of this experiment are in paper F.

The Context Signal

In this experiment, the gating signals of the Sequential ESNs were used as context γ (see figure 2.11). This represents which core pattern and variation of core pattern should be played at any time, i.e. a high-level musical description of what movement pattern (i.e. drum pattern) should be played. This is similar to the context signal used in the YMCA experiment (see section 2.4), where the context signal coded for the melody of the song. However, the context signal is now derived from the low-level MIDI data, i.e. it is *data driven*, instead of being specified by the designer of the experiment. Furthermore, the sequence to be imitated is longer than the YMCA experiment (consisting of 144 timesteps, the imitation of drumming lasted for 1960 timesteps). This leads to a more complicated context signal; compare figure 2.13 which shows the context signal for the current experiment, to the context signal of the YMCA experiment (figure 2.3).

2.6.5 Results

The combination of the motor architecture and the groovy drum machine fulfilled research goal 1, the results are in paper E. The imitative qualities of the sound produced by the groovy neural center was published in paper D; paper E focuses more on the combination of the two architectures and learning a long and complex sequence of movements. Figure 2.14 shows how the RPs repeatedly activate with repeating context signals. How well the system performed in terms of motor actions can be seen in figures 2.15 and 2.16. The introduction of the PEC ensured much more accurate training and corrective signals, increasing the performance compared to previous work (see papers A and B).

Paper E describes how recurrent λ values are calculated. The algorithm considers only strict recurrences of λ activation values, and is therefore a good measure of how well the λ values is repeated in accordance with the context signal. Paper F demonstrated that the activations of the modules would be the same when the target states were randomly permutated, i.e. the modules do specialize on specific parts of the movement. This fulfilled research hypothesis 2.

2.6.6 Summary, experiment 3

Research goal To combine the motor and sound systems to create an artificial groovy drummer. Research goal 1.

Research hypothesis The system learns sequences of motor knowledge regardless of the original training signal. Research hypothesis 2.

Results The motor and sound systems were succesfully combined, and an artificial groovy drum agent was created. The groovy drum agent was both

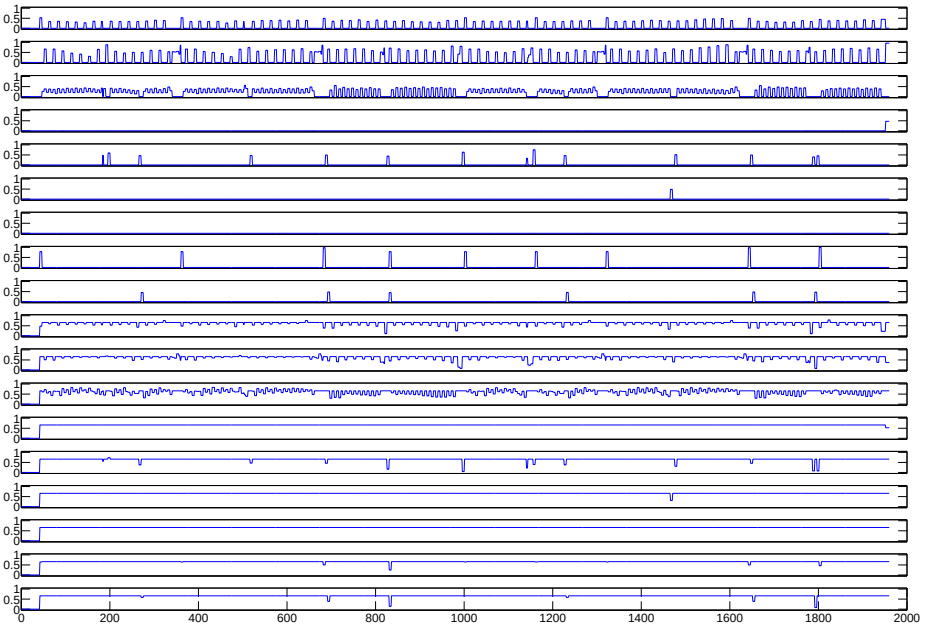


Figure 2.12: The target music state, x'_{sound}, linked to the time frame of the arms. The output of the neural groove center are in spikes, but each signal is replicated 10 times (as shown in equation (2.4)) to make it coincide with the desired arm movement. The upper 9 signals represent velocities of the different drums to hit, starting with the kick drum, snare drum, hihat, ride, toms and cymbals. The lower 9 signals correspond to the onset times for each beat, i.e. how much the beat should be before or after the metronome.

visualized and had sound output. By creating random permutations of the target sequence, the system showed that it had specialized on certain parts of the movement.

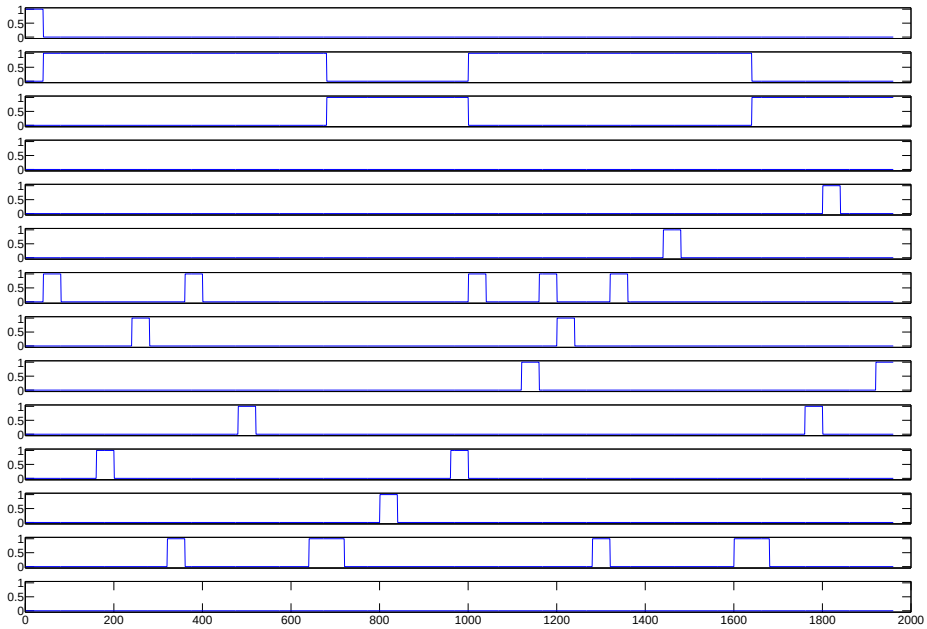


Figure 2.13: The context signal, y. This is the output of the Sequential ESN, as seen in figure 2.11. The first row correspond to the count-in. The three subsequent rows correspond to the core patterns present in the library for this drummer, and the rows below are the variations of that pattern.

2.7 Discussion

What has been the contribution to the field of AI in this thesis? First of all, an architecture for motor control and learning has been developed and tested on imitating human movements. Secondly, a novel way of modeling human drum patterns has been developed and designed, and shown to produce results that are similar to the original drum tracks that were used to train the system. Finally, these two systems have been combined to form a groovy artificial drummer. The introduction of the predictive error controller makes it possible to use desired state signals that are in different reference frames than that of the robot itself. The neural drum machine can therefore be replaced with any other module generating high-level signals of how a goal should be achieved. The research demonstrates how goal-directed imitative behaviour can be realized using biologically inspired techniques. However, there is no requirement that neural networks must be used to implement the different models of the architecture; this has been a design choice due to an interest of solving AI

problems by biological inspiration. In other words, the research presented in this PhD is a contribution to the literature of control approaches for embodied robots. The approach is influenced by research in developmental psychology, neuroscience and previous work on approaches to imitation learning.

The research on implementing a groovy drum machine has contributed to the field of modeling human expressiveness in music. Through a novel approach, human drum patterns have been successfully modeled and imitated. Furthermore, this research solves a problem not solved by the music software industry. However, for it to be a more cost-effective way of modeling drum tracks, the system should be able to learn from audio recordings, instead of using MIDI files. By "de-mixing" the audio recordings (i.e. extracting the melody and drum patterns from the composite sound file), the system could easily learn the drum signatures of famous drummers. Extracting these data from audio recording is no trivial task (some solutions are outlined in Dixon [2004]), but possible. This could make it possible to buy a software plug-in that plays like John Bonham of Led Zeppelin, or Ringo Starr of the Beatles.

The use of musical imitation has also been on the rise in the computer game industry lately, with popular games such as Guitar Hero[4] and Rock Band[5]. These games let users play MIDI-equipped controllers corresponding to the guitar, bass and drum tracks of popular rock songs. The users themselves are imitating the playing of the famous musicians; more importantly from a computer science perspective is how the avatars (i.e. the animated musicians that appear on the screen as you play along) are set to imitate the behaviours of the real-world musicians. Lars Ulrich of Metallica used motion capture technology (similar to the research in this thesis) when recording his drumming movements[6]. However, these avatars do not respond to user input, i.e. if the user makes an error, this is not reflected at the graphical representation of the player. The avatars are there to enhance the feel of the game. The research presented in this paper could be used to enhance the graphic visualization of these games; the avatars could imitate the playing style of the corresponding musicians, but move according to how the user plays. This is basically how the system works in papers E and F, the neural drum centre sends drum signals and the motor system must issue the motor commands that will create the corresponding sound. It should be noted that this has not been tested to the extent suggested here. The output of the neural drum centre varies very little from the drum tracks it was trained on - this is why the PEC uses the "memory" of the actual arm movements to correct the motor system in case of errors. Why was this not tested as part of the PhD? The problem lies at the motor system's reliance on the PEC to make corrections. If the motor system would be able to completely learn the relationship between x'_{sound} and the re-

[4] www.guitarhero.com

[5] www.rockband.com

[6] www.blender.com/LarsUlrichRevealstheSecretsBehindGuitarHeroMetallica/Blender-Blog/blogs/1168/63718.aspx, retrieved 2009-04-02

quired motor commands, the PEC could be disengaged, and the system could be used in a fashion suitable for Guitar Hero: the motor system would receive drum signals (with errors) and estimate how the drummer would move its arms in order to achieve the desired sound, even if it is a way of playing it was not trained on. On the other hand, having such a last-minute error corrector as the PEC (and the cerebellum) is of great benefit to the system (and to us humans), in terms of robustness. Another approach would be to keep the PEC, and estimate what the arm movement would look like given $\dot{x}'_{sound}$ (not to be confused with *how* to achieve such a movement, which is precisely what the inverse models do), and use this estimation as input to the PEC. Due to time limits, this last problem was not solved, and is left for future work.

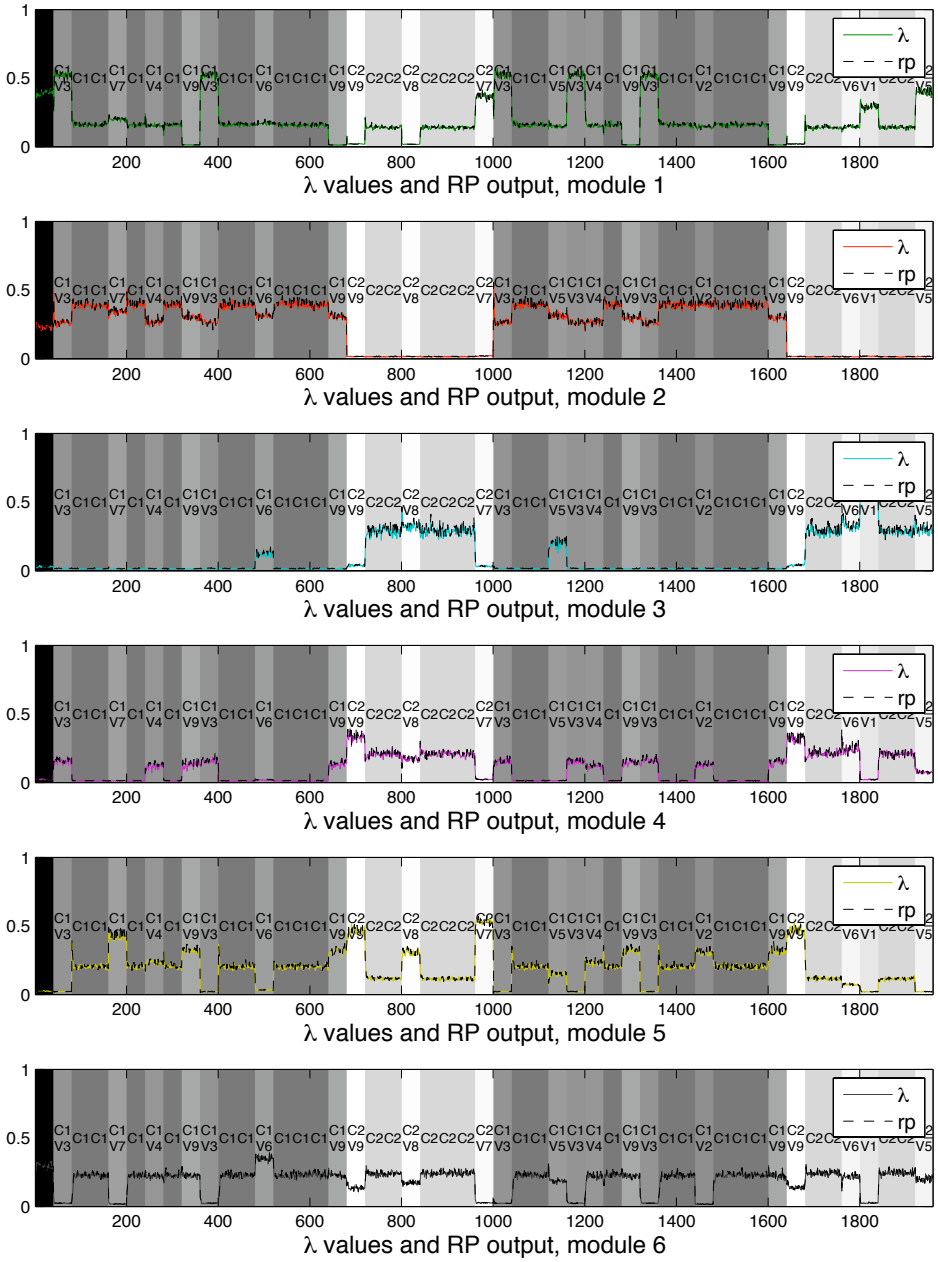


Figure 2.14: An experiment showing λ and RP output, and how they overlap. The gray letters and corresponding color in the background shows the context signal, see also figure 2.13. The overlap between λ and RP indicate stability. The figure shows how the system self-organizes the control of the robot, and how the modules both compete and collaborate when controlling the robot.

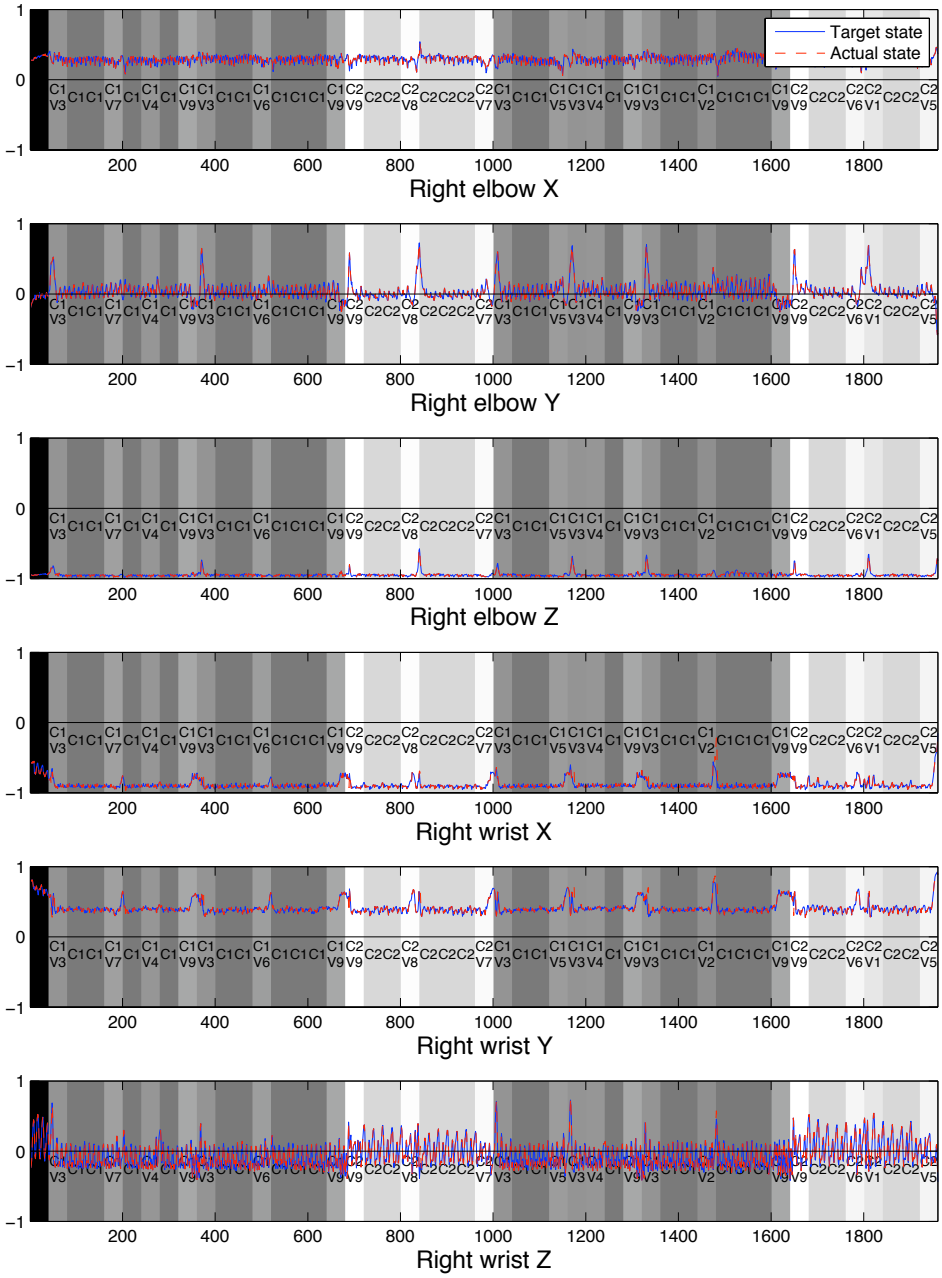


Figure 2.15: Typical performance of the system after training. The close match between target and actual state shows that the system successfully learns and executes the desired arm movements. This is the right arm of the drummer.

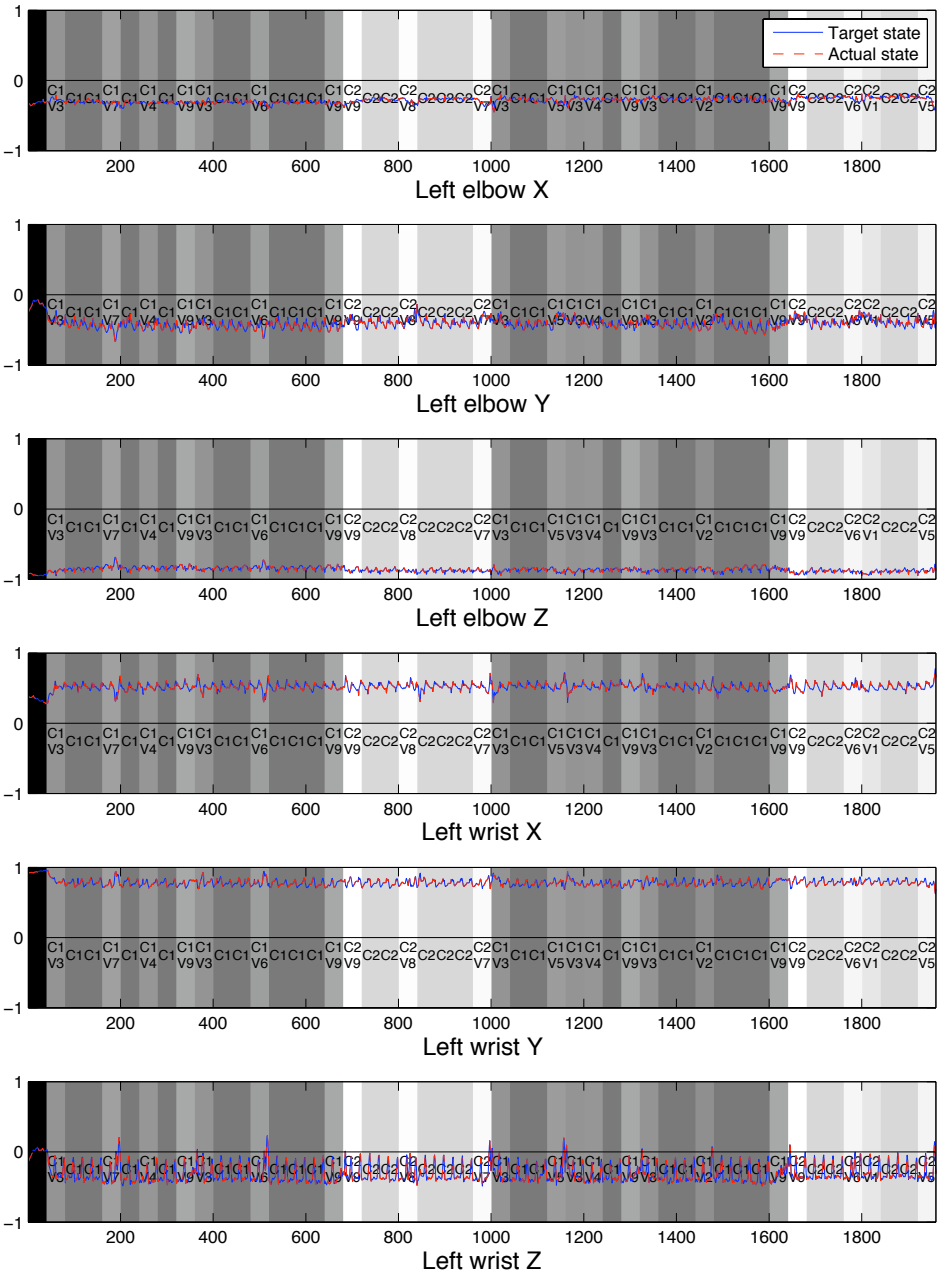


Figure 2.16: Same experiment as in figure 2.15, this shows the left arm of the drummer.

CONCLUSIONS AND FUTURE WORK

3

This PhD has focused on implementing imitative behaviour on simulated robots. The overall goal has been to create a groovy virtual drumming agent. To achieve this, both motor and sound systems have been created and tested independently. Throughout the thesis, the focus has been on using machine learning techniques, so that the models themselves are learned and not specified beforehand by the designer. The research goals and hypotheses in section 1.4 were shown to be satisfied through the published (and submitted) articles that make up this PhD. The architecture has been developed and tested on human behaviour, with good results. Although progress has been made, I see three areas in need of further research for the architecture to be fully complete. 1) Solving the correspondence problem. Currently, the transformation from an extrinsic coordinate system to an intrinsic one is done by the designer of the system. In a real-life situation, the robot would need to take raw multimodal sensory input and transform it into a meaningful representation for the robot. This would not affect the architecture itself, however. This will constitute a pre-processing stage of signals that are fed into the architecture. Thus, research from other fields (such as computer vision) should in theory be easy to integrate; the computer vision system would then do the pre-processing of visual input into intrinsic coordinates. This is a complex task in itself, which is why this has not been the focus of this thesis. 2) To better understand the relationship between the PEC and the neural models. Currently, the PEC controls a significant amount of the total motor signals sent to the robot, as outlined in the discussion. Although this can be regarded as a weakness of the architecture, it is also one of its strengths: it is more robust to noise and perturbations. Studying *what* parts of the motor control space the PEC influences could reveal more about its relation to the motor system. I speculate that general knowledge is contained in the modules, and that the PEC issues corrective error commands for the details, based on the generalizing capacity inherent in neural networks. The problem is *how* to analyze the motor signals sent from the motor system, in order to assess this speculation. Future work would require deeper understanding of the separation between the motor system and the PEC in the motor space. 3) Employ the architecture on

a real robot. Although the simulations are added noise, it is only when run on a real robot platform that the architecture can be fully tested. If given the opportunity, this would be my main priority before any further development. The noise and brittleness would put an extra stress test on the system, and would hopefully highlight the roles of the various models. This would allow the artificial drummer to be a part of a live setting in an interactive manner.

Part II

Publications

ORDERING OF THE PAPERS

The papers are ordered logically, instead of chronologically. For instance, both papers on the groovy drum machine were published before paper B, but the idea is that the reader will appreciate the progression of similar topics, before the convergence of the two fields of research at the end of this part.

SELF-ORGANIZING MULTIPLE MODELS FOR IMITATION: TEACHING A ROBOT TO DANCE THE YMCA

A

Authors:
Axel Tidemann and Pinar Öztürk

Abstract:
The traditional approach to implement motor behaviour in a robot required a programmer to carefully decide the joint velocities at each timestep. By using the principle of learning by imitation, the robot can instead be taught simply by *showing* it what to do. This paper investigates the self-organization of a connectionist modular architecture for motor learning and control that is used to imitate human dancing. We have observed that the internal representation of a motion behaviour tends to be captured by more than one module. This supports the hypothesis that a modular architecture for motor learning is capable of self-organizing the decomposition of a movement.

Main Result:
This was the first implementation of the self-organizing connectionist architecture for motor learning and imitation that has been developed throughout the PhD studies. The architecture was inspired by Demiris' HAMMER and Wolpert's MOSAIC architectures. The experiment consisted of imitation human dance movements, using a robot simulator. The results showed that switching of the modules controlling the robot occurred during a breaking point in symmetry of the movement to be imitated, indicating that the breaking of symmetry is hard to learn, and thus requires more modules.

Published in:
IEA/AIE 2007, volume 4570 of *Lecture Notes in Computer Science*, Springer (2007) 291-302

A. Self-organizing Multiple Models for Imitation: Teaching a Robot to Dance the YMCA

My main contributions to the paper:

- Recording human dance movements using motion tracking
- Programming the system and running the experiments
- Writing the paper

The co-author contributed to the following areas:

- How to combine HAMMER and MOSAIC
- Discussion of how the system self-organizes in relation to motor primitives

Self-organizing Multiple Models for Imitation: Teaching a Robot to Dance the YMCA

Axel Tidemann and Pinar Öztürk

IDI, Norwegian University of Science and Technology
tidemann@idi.ntnu.no

Abstract. The traditional approach to implement motor behaviour in a robot required a programmer to carefully decide the joint velocities at each timestep. By using the principle of learning by imitation, the robot can instead be taught simply by *showing* it what to do. This paper investigates the self-organization of a connectionist modular architecture for motor learning and control that is used to imitate human dancing. We have observed that the internal representation of a motion behaviour tends to be captured by more than one module. This supports the hypothesis that a modular architecture for motor learning is capable of self-organizing the decomposition of a movement.

Keywords: Human Robot Interaction, Machine Learning, Neural Networks.

1 Introduction

Learning by imitation is an important part of human cognitive behaviour, and this approach has gained considerable interest in the artificial intelligence community [1]. Dancing is an area where imitation learning is easily observable. When learning to dance, students imitate the movement of the teacher with only visual input to guide their own motor system.

This paper presents a humanoid robot simulation that mimics this phenomenon. Our model is based on an on-line imitation learning algorithm for acquisition of dance movements. As our research agenda includes the study of the mechanisms for composing complex movements from simpler motor primitives we have a special focus on how motor primitives are acquired. Towards this end we implemented a modular architecture where a module learns and represents such a primitive. A module may be envisioned as an expert that specializes on one or several such primitives. We investigate whether an architecture with several such expert modules can self-organize learning and execution of motor primitives. The underlying architecture uses multiple paired inverse/forward models, implemented using recurrent neural networks. The research question of this paper reads: *How and to what extent will multiple paired inverse and forward models self-organize when used as controllers for a humanoid robot?*

2 Imitation in Psychology, Neuroscience and AI

Developmental psychologists have long focused on how infants learn by imitating their parents and the people in their surroundings. Piaget [2] describes imitation as the continuation of *accommodation* (i.e. adjustment or adaptation) of sensory-motor schemas (i.e. motor and perception stimuli) to the external world, an important part of intelligent human behaviour. Meltzoff and Moore suggest that imitation in newborns is due to a process they call *active intermodal mapping* (AIM) [3]. The intermodal mapping relies on a representational system capable of uniting perception and production of human acts, which can be used to correct the child's behaviour so that it will match that of the demonstrator.

The discovery of mirror neurons [4] has caught the interest of researchers in several disciplines. Mirror neurons, which are observed to activate both when observing and performing a certain action, are suggested to play an important role in imitation [1], language [5] and mind reading [6].

Research in the field of imitation learning in computer science is coarsely divided in two groups: those focusing on transforming visual information to a meaningful representation for the agent (also called the *correspondence problem*) or the motor part (all perceptual information is already present, ready to be input to a perception-action system) [1]. Schaal believes that the most interesting approach to imitation learning is *model-based learning* (note that this brief review focuses on *multiple model* approaches to imitation learning), where an inverse model will output a motor command to achieve a desired state, given the current state and the desired state. The output of the inverse model is given as input to a forward model that will predict the outcome of the motor commands acting on the environment. It is because of the predictive aspect that he favors this approach. Schaal also thinks the modular approach fits nicely with the simulation theory of mind-reading, as well as a possible way to implement the AIM framework of Meltzoff and Moore.

Demiris [7,8] and Wolpert [9] have worked on multiple inverse/forward models for imitation learning (note that this is in principle similar to Jacobs' mixtures of experts, [10]). Wolpert argues that the cerebellum has several forward and inverse models [11], and that they correspond to different motor primitives. Wolpert believes the inverse/forward models should be implemented as neural networks, whereas Demiris uses different techniques such as PID controllers and Bayesian belief networks to represent the models.

Matarić has a holistic modular approach to imitation; her architecture also addresses the visual processing of imitation [12] (it is one of the modules, along with attention, motor primitives and learning modules). She uses different techniques for implementation of the modules, and has tested the architecture on several robots testbeds.

3 A Multiple Paired Model for Dance Imitation

We have implemented a multiple paired models architecture (abbreviated MPMA) inspired from Demiris' HAMMER [8] and Wolpert's MOSAIC [9,13]

architectures. Both MOSAIC and HAMMER use multiple paired inverse and forward models. Why use multiple paired models as a control architecture? The modularity of the brain serves as a good inspiration for modeling artificial intelligence. Even though multiple paired inverse/forward models may not be the actual organization of the brain, it is a computational approach that is well understood in the control literature [14]. In addition, multiple models can be seen as a solution to the problem of trying to compress too much information into one network and when learning of new concepts destroys what has previously been learnt, i.e. *catastrophic forgetting* [15]. Furthermore, modular architectures can implement redundancy which is an important feature of intelligent systems [16], and hence ensure robustness. Modular architectures can also expand their capabilities by adding new modules (although this is not done in our implementation). Such a modular system enables the exploration of whether and how a movement is decomposed into smaller submovements and how and why these can be represented by different modules.

The MPMA can also exploit the advantages of both a localist and distributed representation, a localist representation (i.e. a module[1]) makes it easy to tell where a certain concept is stored, whereas a distributed representation (i.e. the neural networks in the models) is noise-tolerant and can still function even if some of the nodes of the network become destroyed.

Figure 1 illustrates the architecture. The forward model and responsibility predictor are *predictors* of the next state and the suitability of the module, respectively. The inverse model can be thought of as a *controller* or *behaviour*. The different models will now be explained in more detail.

3.1 The Models

The forward model learns to predict the next state $\hat{x}^i_{t+1}$ based on the motor command u^i_t applied to the current state x_t. The error signal (i.e. dashed arrow) used for training[2] the forward model is the difference between the predicted and the actual next state.

The inverse model learns how to produce the motor commands u^i_t to achieve the target state $x^{'}_t$ given the current state x_t. The error signal of the inverse model is the *feedback motor error command*, u_{feedback}. The feedback controller [17] computes the difference between the target state at time t with the actual state at time $t+1$. The difference is multiplied with a constant K, and used as motor commands to pull the system in the right direction, i.e. the error in joint *angles* is used to compute joint angle *velocities*. Training an inverse model is a hard problem, since there are many ways to achieve a certain desired state [14]. The feedback error motor signal represents a simple way to find a trajectory since it will coarsely pull the system in the right direction when the system issues bad motor commands, with decreasing influence as the system performs better.

[1] The term *module* is used in this paper to group three components together: an inverse and forward model, and a responsibility predictor, which will be explained shortly.

[2] All the recurrent neural networks are trained using the backpropagation through time algorithm.

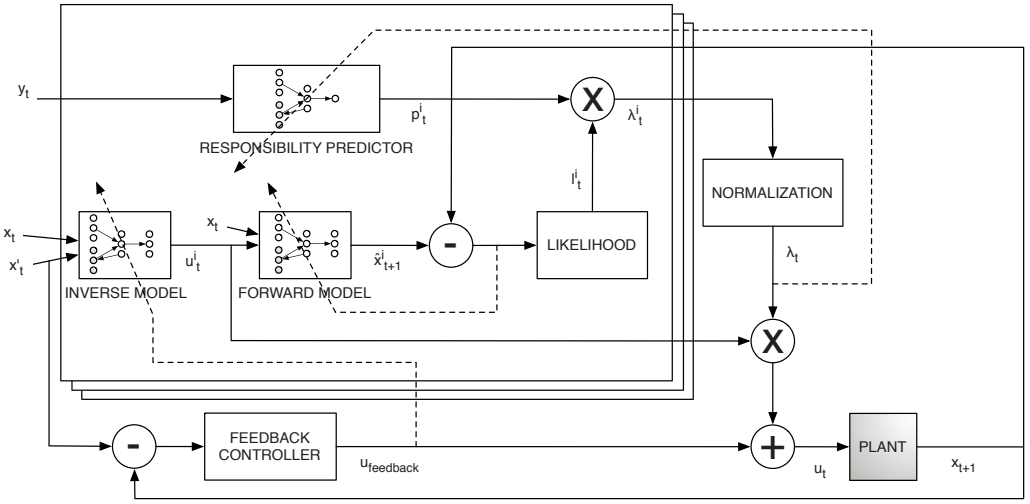


Fig. 1. The multiple paired models architecture. See the text for explanation of the different variables. Inspired from [13] and [8].

The responsibility predictor learns to predict the suitability p_t^i of a module *prior* to movement, based on the context signal y_t. Its error signal is λ_t^i. The λ_t^i value is calculated based on two factors: 1) the likelihood l_t^i calculates how well the forward model predicts the next state, based on the difference between the predicted next state and the actual next state, assuming it is influenced by gaussian noise with a standard deviation σ, and 2) the predicted responsibility p_t^i. The λ_t^i values are normalized across the modules, resulting in the final λ_t vector. The λ_t vector corresponds to which of the modules is most suitable to control the motor commands sent to the plant (i.e. robot). The output of the inverse models u_t^i is multiplied with the normalized λ_t vector. Modules with a high λ value will then achieve more influence over the total motor command u_t sent to the plant than modules with a low λ value. This is how switching between modules is realized. In addition, the λ value is used to gate the learning of the inverse and forward models, by multiplication with the error signal. If a module makes bad predictions and receives a low λ value, it will not influence the final motor command, nor will it receive much of its error signal. Modules that make good predictions will gain more control over the total motor command, and also receive more of its error signal to further fine-tune its predictions.

The difference between λ_t^i and p_t^i was used as a performance measure during training. When the difference was below a certain threshold the training would stop, since the system then correctly predicts which module is more suitable to control different parts of the movement.

3.2 Input/Output of the MPMA

The input to the MPMA was the joint angles of the demonstrator. This was done to overcome the correspondence problem, i.e. the transformation from an

extrinsic to an intrinsic frame of reference [18]. Demiris and Hayes [7] take the same approach. It is questionable whether this is a biologically plausible approach, however studies in neuroscience anticipate the existence of a geometric stage where sensory input is transformed to postural trajectories which in turn can be used by the motor system, as argued by Torres and Zipser [19]. Furthermore, experiments done by Desmurget and Prablanc [20] show that humans actually use estimation of joint angles when imitating.

The other input to the system is the context signal. As an example of a context signal, Wolpert uses whether a cup is empty or full [9] in order to select between two inverse models that both will generate motor commands to lift the cup. We let the context signal correspond to the music playing while dancing, i.e. it can be thought of as a representation of the melody the imitator hears while it is dancing.

The output of the system are motor commands to be applied on the plant. In this case, the robot is the plant, and the motor commands are the *joint angle velocities* of the robot. Joint angle velocities are not directly equal to motor commands, since motor commands are expressed in *forces* of the actuators. We define that the inverse models output joint angle velocities because the simulator (see section 4) takes joint angle velocities as input, *not* forces. The simulator will calculate the forces that will yield the desired joint angle velocity. This approach is similar to [21], which calculates desired trajectories and relies on an inverse dynamics controller for the motor commands. Still, issuing joint angle velocities is closer to motor commands than a desired trajectory, and it is a simpler problem to solve for the inverse dynamics controller. To resume the argument above by Torres and Zipser; there is some indication that this is how the brain plans. After application of the motor commands, the plant returns the actual next state.

3.3 Details of the MPMA

In MOSAIC [9] there is a difference between action production and observation; when *producing* an action, each forward model is fed the *sum* of all the inverse motor commands. When *observing* an action for imitation, the output of *each* inverse model is fed into the paired forward model, this is similar to the HAMMER architecture. The MPMA does not have two modus operandi, the architecture design is the same as MOSAIC in *action observation mode*, since we consider this best models the action/prediction-relationship of having multiple paired inverse/forward models. This design choice is also related to the fact that there is no "observation phase" with inhibition of the motor commands as the imitator is watching the teacher, with a subsequent selection of appropriate modules that will imitate the action (as is the case in both MOSAIC and HAMMER). The imitator tries to imitate the perceived movement as it is being demonstrated. Children often imitate what they are observing without being aware that they are imitating (Piaget defines this as the sixth level of imitation [2]). The MPMA uses a divide-and-conquer technique to learn a movement. A movement is divided

into meaningful pieces (which may also be called motor primitives), each of which is learned by a module. The modules can split movements amongst themselves as they see fit.

Our implementation seeks to combine the best of both HAMMER and MOSAIC, i.e. the inverse/forward pairing of the former (note that MOSAIC also has inverse/forward pairing, but this was only in *action observation mode*), and the responsibility predictor and the focus on self-organization from the latter.

4 Experimental Setup

The breve simulator[3] (an open-source graphical simulator, using the Open Dynamics Engine[4]) was used to implement the humanoid robot. The MPMA was implemented in MatLab, and the communication between MatLab and breve was done via sockets. MatLab sent the motor commands to the breve simulator, and breve returned the state of the simulator after the application of the motor commands for 13 timesteps. This required the forward models to predict the state of the robot for 13 timesteps into the future.

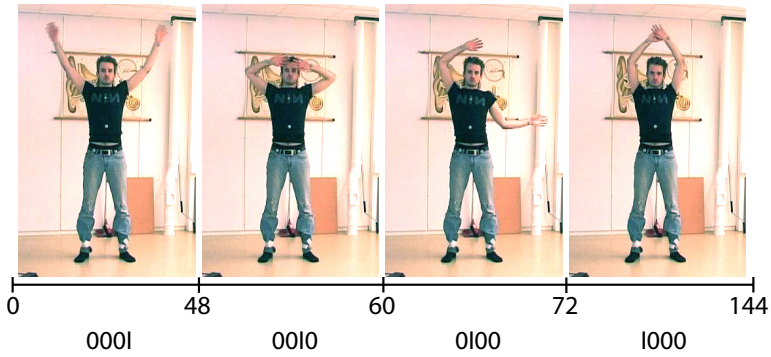


Fig. 2. The YMCA movement. The letters *Y M C A* are spelled out using arm movements. The numbers directly below the figure show how many timesteps are spent forming each letter. The context signal is shown below the time scale.

Two conditions were examined for the experiment: 1) the system was allowed to completely self-organize, and 2) the responsibility predictors were trained on the context signals *before* the training of the system as a whole, i.e. biasing the system to learn specific parts of the movement in each of the modules. This was done to see whether the system would self-organize differently, and how it would affect the performance of the system. Note that the responsibility predictors would still be trained during the training of the entire system in the second condition, retaining its self-organizing capabilities. It is the perspective of the designer that one module should be allocated to each of the context signals, since they represent different parts of the melody and also different movements.

[3] http://www.spiderland.org/breve/
[4] http://ode.org/

The experiment consisted of imitating the dance from a song called YMCA by the Village People, a 70s disco group from New York, see figure 2. Training data was collected by using the Pro Reflex tracking system at the Developmental Neuropsychology lab at the Institute of Psychology, NTNU. The system is able to track the position of fluorescent balls within a certain volume by using five infrared cameras. The tracking of the balls yields Cartesian coordinates of the balls, which can be used to calculate the joint angles of the demonstrator, which were scaled to the range $[0, 1]$. No averaging or smoothing of the data was done. The noisy data was added small amounts of Gaussian noise with mean 0 and standard deviation 0.01 at each training epoch, to render it slightly different for each training epoch. Only the arm movements are imitated, so the robot has four degrees of freedom (one at each shoulder and elbow joints). The learning rate of the inverse, forward and responsibility predictor networks were $\delta_{inv} = \delta_{forw} = 0.1$, $\delta_{rp} = 0.3$ (the learning rate of the responsibility predictor is higher than the other networks to make it adapt quickly to λ_i^t), $\sigma = 0.25$ (of the likelihood function), the feedback error motor constant was $K = 3$, and the error proportion $ep = 1.1$ was used to bias the learning of the winning module (as is done in [22]). The biasing was done as follows: after normalization of the λ_t vector, the highest λ value is multiplied with ep, and then the λ_t vector is normalized again. The stopping criterion was the following: the mean difference between p_i^t and λ_i^t throughout an epoch had to be lower than 0.0045 for all modules.

5 Results

Each of the conditions was run 20 times. The results are presented in table 1. The robot imitating the human dancer can be seen in figure 3. Figure 4 shows how the MPMA switches between different modules for both the conditions after training by plotting λ_t^i. p_t^i is also plotted in the same figure. Note that they overlap. This is an indication of stability in the system, and was used as a stopping criterion, as discussed above. Figure 5 shows the performance of the system regarding the target trajectory versus the actual trajectory. The background color correspond to λ_t^i, making it easy to see which module controls which part of the movement.

Fig. 3. Imitation of the YMCA, demonstrator on the left, imitator on the right

Table 1. Results of the experiments. Transitions mean the changing between modules for controlling the robot. If there are 0 transitions, only one module controlled the robot through the entire last epoch. If there is one transition, two modules controlled the robot, and so on. However, multiple transitions does not ensure different modules were in control for each transition, therefore the number of unique modules that controlled the robot are also listed. u_{feedback}/u_t says how strong the feedback error motor signal was relative to the total motor command at the last epoch. The performance error p_e is the accumulated difference between the desired state and the actual state of the system at the last epoch.

	min/max/avg epochs	min/max/avg p_e	min/max/avg u_{feedback}/u_t	Transitions				Unique modules			
				0	1	2	3	1	2	3	4
Cond. 1	667/1864/1121	10.4/19.0/12.6	0.05/0.11/0.07	7	12	0	1	7	13	0	0
Cond. 2	491/1857/1013	9.8/21.6/14.7	0.05/0.12/0.08	1	5	7	7	1	12	1	6

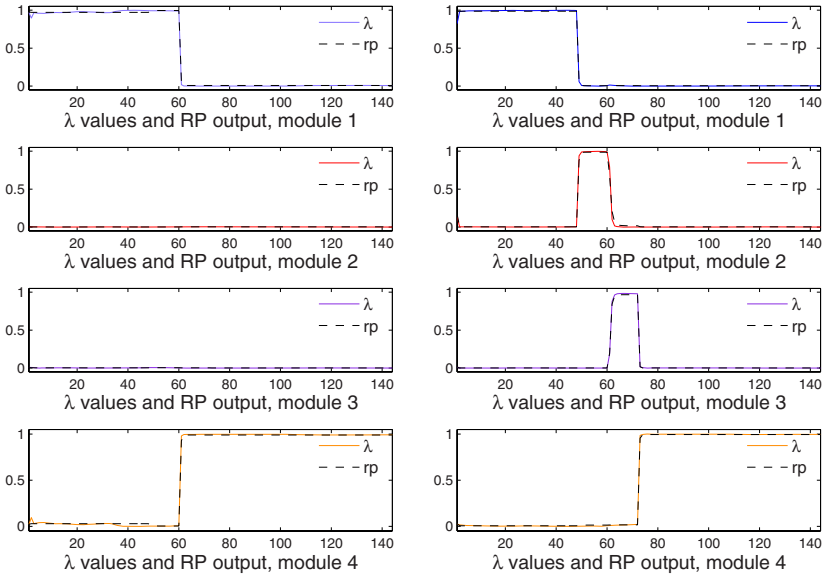


Fig. 4. The switching of control between the modules. The graph on the left shows one run in the first condition, with two different modules (1 and 4) in control. The graph on the right shows one run in the second condition, with all four modules in control at different timesteps through the imitation.

6 Discussion

From the results, the system more often than not makes use of more than one module indicating that a movement is divided into smaller submovements. In cases where the responsibility predictor is not trained beforehand, usually two modules are involved in the representation of the movement. In the second condition, the system uses more than two modules for capturing the movement.

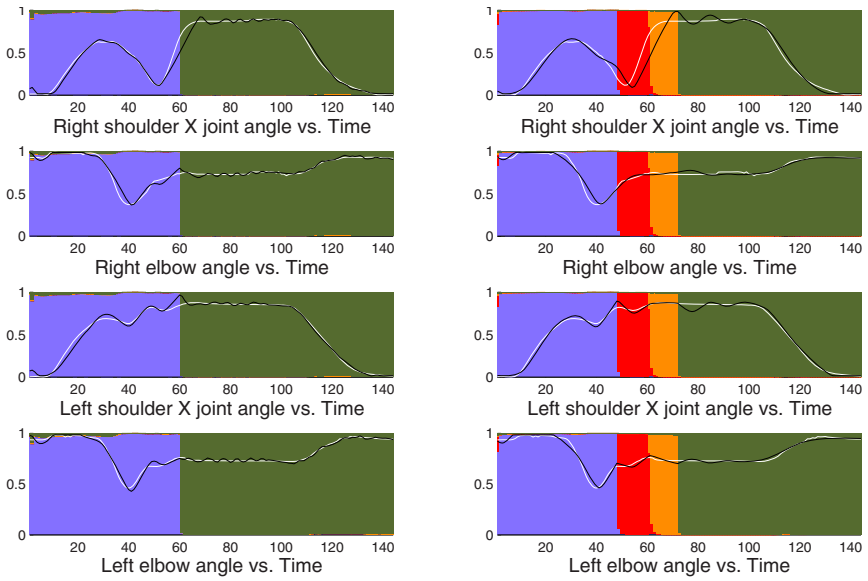


Fig. 5. How well the trajectory of the system (white line) matched the target trajectory (black line). The background color shows which module was in control, condition 1 on the left and condition 2 to the right. The graphs are from the same experiment as in figure 5.

These results indicate that the system promotes self-organized decomposition of movement when learning motion behaviours.

Representation of small and reusable motions, under different names such as 'motion segments', 'motor primitives' or 'symbols', has attracted researchers interest across different disciplines. Some of these researchers predefine the motor primitives and make the robot learn these [12]. Others, similar to us, adopts the idea of self-decomposition and aims at determining the global, invariant structures and criteria underlying motor primitives. For example, [23] illustrates that a humanoid robot can successfully learn such global structures through self-exploration. The learned structures are represented as attractors in a neural network. In our case we have several neural networks each representing different primitives. It is in our future work agenda to investigate how the modules of the MPMA may possibly be capturing attractors.

One interesting question is what qualifies as a motor primitive. Matarić also points out this is hard to define [12]. From figure 5, it seems like the modules learn some motor primitives, but instead of learning the motor primitives according to the context signal (which is how the designer of the system would define a motor primitive), the modules find distinctions in the sensory flow to define them separately. The hypothetical (i.e. designer's) switch points in the YMCA movement at timesteps 48, 60 and 72 are not always obeyed. The system's own motor primitives are usually bigger than what the designer had thought as a meaningful motor primitive. In fact, the system has a lower average performance error when the responsibility predictors are *not* trained beforehand, indicating that only a few modules are necessary to control the movement to be imitated.

On the other hand, the differences in p_e and the number of epochs spent to train the system are quite small, so performance-wise a clear preference cannot be made.

The main point of the MPMA is the decomposition of a movement into segments and simultaneously learning these segments in different modules. Our goal is to explore the possible criteria the system might be using for the decomposition. For this purpose, the coordinate system of the movement is first transformed into an internal referential frame. The idea of such a conversion is adopted by researchers in both neuroscience and robotics [19,21], where planning of a movement is done in form of postural trajectories, independent of motor commands. Our research has not been focused on *how* this geometrical transformation is done, but these ideas and solutions can be integrated into the MPMA.

One drawback of the YMCA movement is the symmetry of the Y, M and A movements, only the C movement breaks this symmetry. Closer examination reveals that for all the experiments (for both conditions) that have one or more transition, a switch between modules occurs at around the 60th timestep, which is exactly when the C movement begins, see figure 2. This may imply that not having symmetrical movements would help the separation of the modules, probably because it is more difficult to imitate and therefore needs to be stored in a separate module. Research in neuroscience supports this claim maintaining that nonsymmetric action in bimanual movements leads to interference and takes longer time compared to symmetrical ones [24]. It is interesting to see that the system agrees on the switch point around the 60th timestep in both conditions. This suggests that the human guesses of what makes up a motor primitive somehow aligns with what the system decides to be a motor primitive as well. The architecture and the experiments presented in this paper are the early and promising results of our work on motor imitation learning.

7 Future Work

We are currently examining how differences in speed and direction might affect the self-organization of the system. Research related to human motor learning suggests the changes in direction and velocity as determining factors in learning motion behaviour. It has also been suggested that specific muscle synergies are shared across different motor behaviours [25] and by various animals. We are also investigating how repetition of the target movement will affect the self-organization. Preliminary results indicate that repetition affects the way the modules will self-organize, in the way that more modules are being used. In addition, we are looking at making the movements to be imitated more complex, both in term of having longer patterns to imitate and more degrees of freedom. This will hopefully reveal if we can find a level of complexity where the system will be saturated, i.e. all the neural resources are used and new modules must be added in order to achieve the desired imitative behaviour. As mentioned above, we are also interested in how the MPMA captures attractors, for example it will

be interesting to see how the attractor landscape would be for a module that seems to represent several primitives.

References

1. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences 3(6), 233–242 (1999)
2. Piaget, J.: Play, dreams and imitation in childhood. W. W. Norton, New York (1962)
3. Meltzoff, A.N., Moore, M.K.: Explaining facial imitation: A theoretical model. Early Development and Parenting 6, 179–192 (1997)
4. Rizzolatti, G., Fadiga, L., Gallese, V., Fogassi, L.: Premotor cortex and the recognition of motor actions. Cognitive Brain Research 3, 131–141 (1996)
5. Arbib, M.: The Mirror System, Imitation, and the Evolution of Language. In: Imitation in animals and artifacts, pp. 229–280. MIT Press, Cambridge (2002)
6. Gallese, V., Goldman, A.: Mirror neurons and the simulation theory of mind-reading. Trends in Cognitive Sciences 2(12) (1998)
7. Demiris, Y., Hayes, G.: Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In: Imitation in animals and artifacts, pp. 327–361. MIT Press, Cambridge (2002)
8. Demiris, Y., Khadhouri, B.: Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems 54, 361–369 (2006)
9. Wolpert, D.M., Doya, K., Kawato, M.: A unifying computational framework for motor control and social interaction. Philosophical Transactions: Biological Sciences 358(1431), 593–602 (2003)
10. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation 3, 79–87 (1991)
11. Wolpert, D.M., Miall, R.C., Kawato, M.: Internal models in the cerebellum. Trends in Cognitive Sciences 2(9) (1998)
12. Matarić, M.J.: Sensory-Motor Primitives as a Basis for Learning by Imitation: Linking Perception to Action and Biology to Robotics. In: Imitation in animals and artifacts, pp. 392–422. MIT Press, Cambridge (2002)
13. Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Networks 11, 1317–1329 (1998)
14. Jordan, M.I., Rumelhart, D.E.: Forward models: Supervised learning with a distal teacher. Cognitive Science 16, 307–354 (1992)
15. Ans, B., Rousset, S., French, R.M., Musca, S.: Self-refreshing memory in artificial neural networks: learning temporal structures without catastrophic forgetting. Connection Science 16(2), 71–99 (2004)
16. Pfeifer, R., Scheier, C.: Understanding Intelligence (Illustrator-Isabelle Follath). MIT Press, Cambridge (2001)
17. Kawato, M.: Feedback-error-learning neural network for supervised motor learning. In: Eckmiller, R. (ed.) Advanced neural computers, pp. 365–372 (1990)
18. Nehaniv, C.L., Dautenhahn, K.: The Correspondence Problem. In: Imitation in Animals and Artifacts, pp. 41–63. MIT Press, Cambridge (2002)
19. Torres, E.B., Zipser, D.: Simultaneous control of hand displacements and rotations in orientation-matching experiments. J Appl Physiol 96(5), 1978–1987 (2004)
20. Desmurget, M., Prablanc, C.: Postural Control of Three-Dimensional Prehension Movements. J Neurophysiol 77(1), 452–464 (1997)

21. Ijspeert, A., Nakanishi, J., Schaal, S.: Trajectory formation for imitation with nonlinear dynamical systems. In: Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2001), pp. 752–757 (2001)
22. Haruno, M., Wolpert, D.M., Kawato, M.: MOSAIC model for sensorimotor learning and control. Neural Comp. 13(10), 2201–2220 (2001)
23. Kuniyoshi, Y., Yorozu, Y., Ohmura, Y., Terada, K., Otani, T., Nagakubo, A., Yamamoto, T.: From humanoid embodiment to theory of mind. In: Pierre, S., Barbeau, M., Kranakis, E. (eds.) ADHOC-NOW 2003. LNCS, vol. 2865, pp. 202–218. Springer, Heidelberg (2003)
24. Diedrichsen, J., Hazeltine, E., Kennerley, S., Ivry, R.B.: Moving to directly cued locations abolishes spatial interference during bimanual actions. Psychological Science 12(6), 493–498 (2001)
25. d'Avella, A., Bizzi, E.: Shared and specific muscle synergies in natural motor behaviors. PNAS 102(8), 3076–3081 (2005)

A Self-Organizing Multiple Model Architecture for Motor Imitation

B

Authors:
Axel Tidemann and Pinar Öztürk

Abstract:
Learning by imitation allows humans to easily transfer motor knowledge between individuals. Our research is aimed towards equipping robots with imitative capabilities, so humans can simply show a robot what to do. This will greatly simplify how humans program robots. To achieve imitative behaviour, we have implemented a self-organizing connectionist modular architecture on a simulated robot. Motion tracking was used to gather data of human dance movements. When imitating the dance movements, the architecture self-organizes the decomposition of movements into submovements, which are controlled by different modules. The modules both collaborate and compete for control during the movement. The trajectory recorded during motion tracking was repeated, revealing recurrent neural activation patterns of the inverse models (i.e. controllers), indicating that the modules specialize on specific parts of the trajectory.

Main Result:
The target state is repeated, and the recurring activation values of the modules show how they specialize on certain parts of the movement. The importance of the responsibility predictors is also examined. The performance of the system was a lot worse when training and testing without the RPs. The performance was better when training with RPs and disabling them during testing, but still inferior to training and testing with the RPs. The results show that the RPs are crucial for the modules to specialize in specific parts of the trajectory.

To appear in:
International Journal of Intelligent Information and Database Systems, Volume 4, No. 1, 2009

My main contributions to the paper:

- Programming the system and running the experiments
- Statistical analysis
- The idea of testing the importance of the RPs, and how to do it
- Writing the paper

The co-author contributed to the following areas:

- The idea of using repetition as a way to demonstrate if the modules became specialists on certain parts of the movement
- Writing the discussion and conclusion

Is not included due to copyright

A Drum Machine that Learns to Groove

C

Authors:
Axel Tidemann and Yiannis Demiris

Abstract:
Music production relies increasingly on advanced hardware and software tools that makes the creative process more flexible and versatile. The advancement of these tools helps reduce both the time and money required to create music. This paper presents research towards enhancing the functionality of a key tool, the drum machine. We add the ability to *learn how to groove* from human drummers, an important human quality when it comes to drumming. We show how the learning drum machine overcomes limitations of traditional drum machines.

Main Result:
A drum machine that models both small- and large-scale variations of drum patterns recorded using amateur drummers. The large-scale models are represented as Hidden Markov Models, whereas the small-scale variations are represented using a normal distribution. MIDI was used as both input and output to the system, enabling the creation of a low-cost groovy drum machine. The imitated drum patterns were found to be similar to the original ones by human evaluation.

Published in:
31st Annual German Conference on AI, Volume 5243 of *Lecture Notes in Computer Science*, Springer (2008) 144-151

My main contributions to the paper:

- Coming up with the idea of creating a groovy drum machine that learns user-specific variations
- Gathering drumming data using an electronic drum kit and five amateur drummers (one of them being the first author)
- Programming the system and running the experiments
- Analysis of both the recorded and generated data patterns
- Writing the paper

The co-author contributed to the following areas:

- Suggesting that several drummers should participate in the experiment to better evaluate the results
- Pointing out relevant references
- Writing the discussion

A Drum Machine that Learns to Groove

Axel Tidemann[1] and Yiannis Demiris[2]

[1] IDI, Norwegian University of Science and Technology
tidemann@idi.ntnu.no
[2] BioArt, ISN, EEE, Imperial College London
y.demiris@imperial.ac.uk

Abstract. Music production relies increasingly on advanced hardware and software tools that makes the creative process more flexible and versatile. The advancement of these tools helps reduce both the time and money required to create music. This paper presents research towards enhancing the functionality of a key tool, the drum machine. We add the ability to *learn how to groove* from human drummers, an important human quality when it comes to drumming. We show how the learning drum machine overcomes limitations of traditional drum machines.
Keywords: Imitation Learning, Machine Learning, Music

1 Introduction

Digital audio workstations allow users to quickly program drums, but the drawback is the machine-like sound of the programmed drums. Our research goal is to make a drum machine that has the groove of a human drummer, by modeling the user-specific variations of human drummers. The groove of a drummer is made up of its large-scale variations (i.e. playing a break) and the small-scale variations (i.e. varying tempo and how hard a beat is played). The approach of drumming software so far has been to add random noise to the produced drum patterns, to make them sound more human. Our approach is to learn drum patterns from human drummers and *model* the variations that make a drummer groovy to make an *intelligent* drum machine that sounds like a human drummer.

2 Background

Saunders et al. [1] and Tobudic and Widmer [2] focus on variations of tempo and dynamics when modeling the style of pianists, albeit with different techniques (string kernels and clustering of similar phrases, respectively). Pachet uses Markov models in his Continuator system [3] to model the probability that one note will follow the other. The Continuator learns the tonal signature of the pianist, and has been implemented in a real-time system. Mantaras and Arcos focus on imitating expressiveness, such as joyful or sad, by using case-based reasoning [4]. Raphael [5] focuses on the problem that arises when soloists do not have the opportunity to play with a real orchestra, but must do with a static

recording. His system dubbed "Music Plus One" lets soloists practice along with an orchestra played by a computer, and the system then learns to follow the variations in tempo introduced by the soloist.

Current drumming software (e.g. FXpansion BFD, Toontrack EZdrummer, DigiDesign Strike, Reason Drum Kits, Native Instruments Battery) consist of huge sample libraries (typically in the gigabytes range), where meticulous precision has gone into sampling the different dynamics when playing drums (i.e. from soft to hard). Still, the user must program the drum tracks. The user can typically select which pattern to play and adjust some parameters to add random noise to both onset time and velocity, to make the patterns sound more human. The lack of intelligent ways to generate human-like drum patterns in these applications forms the motivation for our research.

3 Architecture

Our architecture is called "Software for Hierarchical Extraction and Imitation of drum patterns in a Learning Agent" (SHEILA), see figure 1. SHEILA learns drum patterns and playing style from human drummers. The drum patterns and the models of how they are played are stored in a library. The repetitive nature of playing drum patterns makes it suitable for machine learning techniques to extract user-specific variations. After the learning process, SHEILA can be used as an *intelligent and groovy* drum machine that has the ability to imitate the style of the drummers that served as teachers.

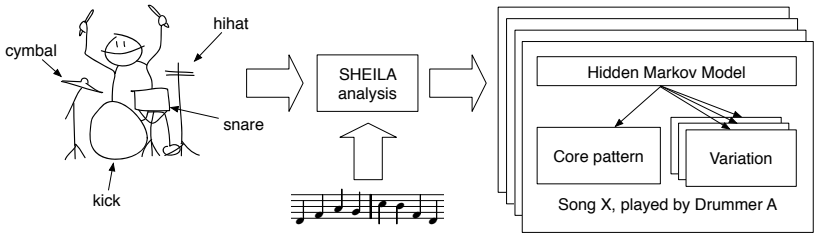


Fig. 1. The SHEILA architecture. Drum patterns are learned, accompanied with melody. In addition to learning drum patterns, SHEILA models the *style* of each drummer that served as a teacher. After learning, SHEILA uses the acquired models to imitate a drum pattern in the style of a given drummer.

3.1 Modeling the Groove

There are two inputs to SHEILA: drum patterns and melody, presented in the MIDI[3] format. SHEILA models both small- and large-scale variations of drum

[3] Musical Instrument Digital Interface, a protocol for electronic music equipment to communicate in real time.

patterns. The *large-scale variations* are discovered the following way: the most frequently played patterns are defined as *core patterns*. To find the core patterns, the MIDI drum pattern sequence is transformed into a string, and examined for *supermaximal repeats*, a technique used in computational biology to find sequences of genes [6]. A supermaximal repeat is a recurring pattern that is not a substring of any other pattern. The same approach is taken on the melody MIDI sequence; the supermaximal repeats are used to divide the song into different parts (e.g. the verse/chorus/bridge of the song), thus the melodic input serves as musical context. For each of these parts, the most commonly played pattern is then considered to be the core pattern of that part, yielding several core patterns for a song. Patterns that differ from the core pattern within the part are defined as the large-scale variations of the core pattern. The sequence of core patterns and variations within similar parts is modeled using a Hidden Markov Model (HMM). The *small-scale variations* are defined as the variations in timing and velocity for each beat that occurs when a drummer plays a pattern. We model the small-scale variations by calculating the mean (μ) and standard deviation (σ) of both the onset time (i.e. how much the beat differs from the metronome) and velocity (i.e. how hard a note is played) of each beat across similar patterns. Our pilot study [7] showed that the Gaussian distribution was an appropriate model. One of the leading software samplers on the market, FXpansion BFD, use the same approach to model human variations in its "Humanize panels", however the user must specify the μ and σ using a graphical interface[4]. Each entry in the SHEILA library consists of a core pattern and variations of the core pattern and the HMM modeling their sequence. Each beat in all the patterns is assigned the normal distribution parameters μ and σ of both velocity and onset time.

3.2 Imitating the Groove

The user of SHEILA presents the desired pattern in the MIDI format to SHEILA. SHEILA presents a list of drummers who have played the desired pattern, along with the name of the song this pattern was played on (indicating what the imitation will sound like, presuming the user knows the song). The user then selects which drummer SHEILA should imitate when playing the desired pattern, and for how many bars. The HMM for each core pattern is used to generate output sequences (i.e. core patterns and variations). This is how *large-scale variations* are introduced. To create the actual beats, random numbers determining the velocity and onset time are drawn from the normal distribution assigned for each beat, introducing *small-scale variations* to the imitated pattern.

The generated onset times are averaged locally based on a 10 beat window (five beats before and five beats after the current beat). A human drummer will vary the onset time of the beats, but this variation will not occur randomly within a close timeframe. The local averaging makes beats played at the same

[4] See page 118 of the user manual (accessed 2008-04-08), www.fxpansion1.com/resourceUploads/BFD_Manual_English.pdf

time sound more coherent, whereas the onset time is still allowed to drift over time. Averaging over ten beats was found to be the value that sounded most natural; less than 10 yielded too random-sounding onset times, whereas more than 10 made the onset times sound too rigid.

The combination of large- and small-scale variations in the imitated drum patterns will result in a pattern that is different from the original, but still in the *style* of drummer X. The output is in the MIDI format, ready to be imported into music production software with high quality drum samples.

4 Experimental Setup

SHEILA was implemented in MatLab. *vmatch*[5] was used to find the supermaximal repeats. Propellerheads Reason 3.0[6] loaded with Reason Drum Kits was used for recording MIDI signals and for generating sound from MIDI files. Recording MIDI was done with a Roland TD-3[7] velocity sensitive electronic drum kit. Five male amateur drummers recorded drum tracks to a melody written by the authors, and were told to play specific patterns for the verse (shown in figure 2), chorus and bridge, and a specific break at each 8th bar of the verse. The drummers were free to introduce large-scale variations according to what felt natural to them. The overall structure of the song was verse/chorus/verse/chorus/bridge/chorus/chorus. The tempo was 120 beats per minute, the length of the song was 2:30 minutes.

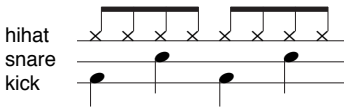


Fig. 2. One of the three core patterns played in the experiment.

5 Results

Figures 3 and 4 show how SHEILA models drummers A and E playing the pattern shown in figure 2. [7] provides more examples. The figures show the mean and standard deviation for both velocity and onset time, and how SHEILA models the unique style of each drummer. Most easily observable is the difference in accentuation of the hihat between drummers A and E. Accents are defined as periodic variations in velocity. This is demonstrated by the strong regular variations in velocity that can be seen in the hihat bar plots of drummer A, whereas for drummer E these variations are not so prominent.

[5] www.vmatch.de
[6] www.propellerheads.se
[7] www.roland.com

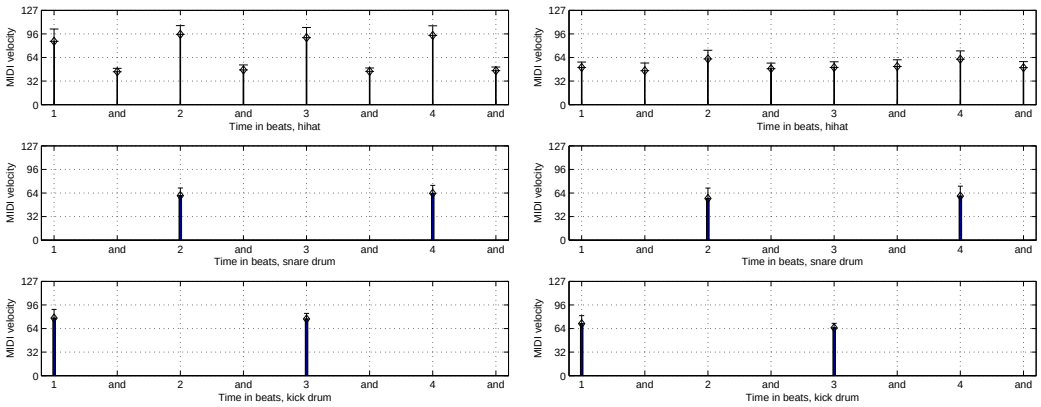


Fig. 3. The plots show the velocity profiles of drummer A (left) and drummer E (right) playing the pattern in figure 2. The MIDI resolution [0−127] is on the Y axis. The X axis represent the beats in a bar. The bar plots show how the velocity varies periodically over time (standard deviation shown as the error bar). This is what makes up the drummer's *groove* in terms of small-scale variations, including the onset times (shown below). The accentuation (i.e. variation of velocity) on the hihat is more pronounced for drummer A compared to drummer E, providing a visual confirmation that the drummers have a different groove when playing the same pattern. The acquired models makes SHEILA capable of imitating the playing style of the drummers.

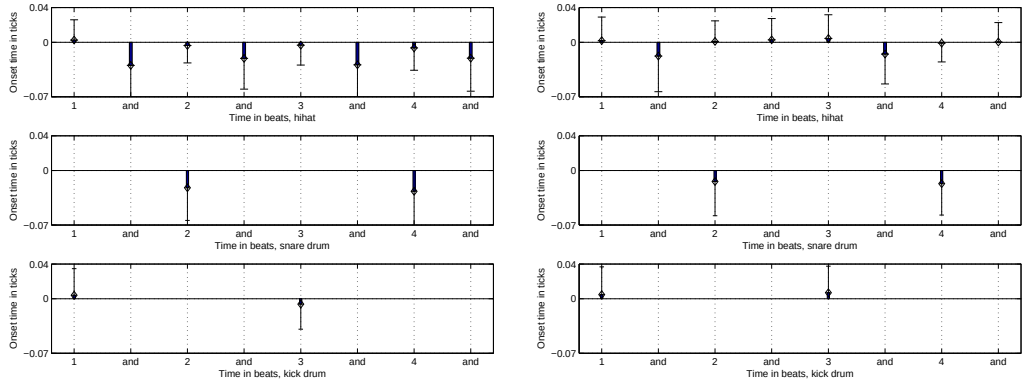


Fig. 4. The plots show the onset time profiles for drummer A (left) and E (right) playing the pattern in figure 2. The Y axis shows the onset time, there are 120 ticks in range $[0. - 0.99]$ between each quarter note. The X scale is the same as in figure 3. The differences in the onset time profiles is further evidence that the drummers have different grooves when playing the same pattern.

The onset time reveals the temporal qualities of the drummers, i.e. how aggressive (ahead of the metronome) or relaxed (behind the metronome) he is. Figure 4 shows how the onset time profiles are different for drummers A and E. The plots show how the playing style of drummer A is slightly more aggressive than that of drummer E. The reader can listen to available MP3[8] files to get a better sense of these terms.

After the learning process, SHEILA was set to imitate the same song used in the training phase to evaluate the quality of the imitation. Since both the large-scale and small-scale variations are introduced randomly using the acquired models (HMMs and normal distributions, respectively), the output will have the human quality that it is different each time a drum track is generated. Figure 5 shows how the local averaging of onset times allows the tempo to drift over time, making the output sound more human.

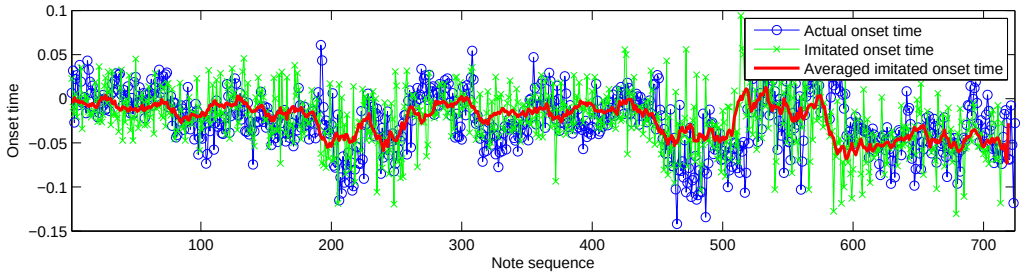


Fig. 5. The circle plots show the tempo drift of drummer A throughout the song. The cross plots show the generated onset times; they are locally averaged since they are drawn from a normal distribution. The local averaging will make beats that are close temporally sound more coherent, whereas the tempo can still drift over time, as shown by the thick line, adding realism to the generated patterns.

Each of the drummers participating in the experiment was presented both the original drum tracks and the imitations in random order and asked to classify which drummer SHEILA imitated (each song lasting 2:30 minutes, the available MP3 files include both originals and imitations). All the tracks used the same set of samples and were mixed the same way, i.e. the sonic qualities of the drumming were the same from track to track. Only the playing style of generated tracks were different. It should be noted that normally other qualities will help identify a drummer (e.g. the sound of the drums, the genre of music, the band the drummer is playing in, etc.). We have removed all these factors, making the recognition task harder than what first might be expected. Table 1 shows that the imitated drum tracks were often perceived as being similar to the teacher (average of 76% correct classification).

[8] http://www.idi.ntnu.no/∼tidemann/sheila/ki08/

Table 1. How often the imitated SHEILA output was correctly classified as being imitated from the corresponding human drummer. Each drummer that participated in the experiment evaluated the imitation, and tried to match which drummer served as a teacher.

Drummer	A	B	C	D	E
Classification	80%	60%	80%	80%	80%

6 Discussion and Conclusion

This paper presents a more sophisticated version of SHEILA than the pilot study in [7], which does not model large-scale variations, does not employ the local averaging of onset time for added realism, and was trained on fewer and simpler datasets. The were two reasons for using amateur drummers playing on a MIDI drumkit: 1) budget (i.e. professional drummers were too expensive) and 2) simplicity of data gathering. We have bypassed the data acquisition stage of human drummers, that consist of seeing and hearing drums being played. We believe these simplifications allow focus on the core problem, which is learning drum patterns.

Our software drummer learns how a human plays drums by modeling small- and large-scale variations. The intelligent drum machine is capable of mimicking its human teacher, not only by playing the learned patterns, but also by always varying how patterns are played. The imitative qualities of SHEILA was shown to be good by evaluation from the different drummers that served as teachers. Our drum machine is an example of how a tool can be augmented by applying AI techniques in an area where incorporating human behaviour is of essence. Creating music is an area where human qualities such as "feel" and "groove" are important, but hard to define rigorously. By enhancing our software drummer with the ability to learn and subsequently imitate such human qualities, the drum machine evolves from being a static rhythm producer to become closer to a groovy human drummer. The added advantage is the ease and cost-effective way of creating human-like drum tracks.

7 Future Work

A more biologically plausible approach of learning drum patterns would be to make SHEILA listen to music (i.e. using audio files as input) and extract melody lines and drum patterns from low-level audio data. For reasons of simplicity, this was not done in the current implementation, but possible solutions are described in [8].

We also aim to visualize SHEILA, by providing imitative capabilities of upper-body movements. SHEILA could then be used in a live setting, playing with other musicians. Our previous work on dance imitation using motion tracking and imitating arm movements [9] forms the basis for this area of research. This approach uses multiple forward and inverse models for imitation [10].

8 Acknowledgements

The authors would like to thank the drummers who participated in the experiment (Inge Hanshus, Tony André Søndbø, Daniel Erland, Sven-Arne Skarvik).

References

1. Saunders, C., Hardoon, D.R., Shawe-Taylor, J., Widmer, G.: Using string kernels to identify famous performers from their playing style. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: ECML. Volume 3201 of Lecture Notes in Computer Science., Springer (2004) 384–395
2. Tobudic, A., Widmer, G.: Learning to play like the great pianists. In Kaelbling, L.P., Saffiotti, A., eds.: IJCAI, Professional Book Center (2005) 871–876
3. Pachet, F. In: Enhancing Individual Creativity with Interactive Musical Reflective Systems. Psychology Press (2006)
4. de Mantaras, R.L., Arcos, J.L.: AI and music from composition to expressive performance. AI Mag. **23**(3) (2002) 43–57
5. Raphael, C.: Orchestra in a box: A system for real-time musical accompaniment. In: IJCAI workshop program APP-5. (2003) 5–10
6. Gusfield, D.: Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York, NY, USA (1997)
7. Tidemann, A., Demiris, Y.: Imitating the groove: Making drum machines more human. In Olivier, P., Kay, C., eds.: Proceedings of the AISB symposium on imitation in animals and artifacts, Newcastle, UK (2007) 232–240
8. Poliner, G.E., Ellis, D.P.W., Ehmann, A.F., Gomez, E., Streich, S., Ong, B.: Melody transcription from music audio: Approaches and evaluation. IEEE Transactions on Audio, Speech and Language Processing **15**(4) (2007) 1247–1256
9. Tidemann, A., Öztürk, P.: Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA. In: IEA/AIE 2007. Volume 4570 of Lecture Notes in Computer Science., Springer Verlag (2007) 291–302
10. Demiris, Y., Khadhouri, B.: Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems **54** (2006) 361–369

GROOVY NEURAL NETWORKS

D

Authors:
Axel Tidemann and Yiannis Demiris

Abstract:
The drum machine has been an important tool in music production for decades. However, its flawless way of playing drum patterns is often perceived as mechanical and rigid, far from the groove provided by a human drummer. This paper presents research towards enhancing the drum machine with learning capabilities. The drum machine learns user-specific variations (i.e. the groove) from human drummers, and stores the groove as attractors in Echo State Networks (ESNs). The ESNs are purely generative (i.e. not driven by an input signal) and the output is used by the drum machine to imitate the playing style of human drummers, making it a cost-effective way of achieving life-like drums.

Main Result:
This is a neural network-based implementation of the groovy drum machine presented in paper C, with a deeper statistical analysis of the performance of the system.

Published in:
18th European Conference on Artificial Intelligence, Volume 178, IOS Press (2008) 271-275

My main contributions to the paper:

- Coming up with the idea of using attractors in generative ESNs to store the groove of the drummers
- Programming the system and running the experiments
- Statistical analysis
- Writing the paper

The co-author contributed to the following areas:

- Highlighting the need for a deeper statistical analysis
- Evaluation of the results
- Helping to formulate key ideas of the research

Groovy Neural Networks

Axel Tidemann[1] and **Yiannis Demiris**[2]

Abstract. The drum machine has been an important tool in music production for decades. However, its flawless way of playing drum patterns is often perceived as mechanical and rigid, far from the groove provided by a human drummer. This paper presents research towards enhancing the drum machine with learning capabilities. The drum machine learns user-specific variations (i.e. the groove) from human drummers, and stores the groove as attractors in Echo State Networks (ESNs). The ESNs are purely generative (i.e. not driven by an input signal) and the output is used by the drum machine to imitate the playing style of human drummers, making it a cost-effective way of achieving life-like drums.

1 INTRODUCTION

The research in this paper is aimed to enhance the cost-effective and easy way of creating drum tracks that is possible with current music production software (e.g. Logic, Pro Tools, Cubase) with the *groove* of a human drummer. The drum machine is a much cheaper alternative to recording live drums. However, the drum machine plays patterns without flaws, which makes it sound rigid and machine-like. Human drummers vary the way patterns are played, both on a small-scale level (varying dynamics and tempo) and on a large-scale level (changing the pattern to be played altogether, such as playing a break). These variations constitute the *groove* of the drummer. Current music production software have parameters that can be tweaked to achieve a human-like effect, however these variations add random noise with the intention that these variations will be perceived as human - the software has no understanding of what makes a drum pattern groovy. The research presented in this paper models these user-specific variations with recurrent neural networks, and demonstrates that the networks are able to represent both the small and large-scale variations that make a drummer groovy. These networks are then used to imitate the playing style of the drummers that served as teachers. The result is a groovy drum machine.

2 BACKGROUND

Modeling user-specific variations in playing style has been a field of study for years within the AI community. Saunders et al. [12] use string kernels to identify the playing style of pianists, by looking at changes in beat-level tempo and beat-level loudness. However, imitating the *style* of the pianists was not attempted. Tobudic and Widmer [15] also consider variations in tempo and dynamics as the two most important parameters of expressiveness. To learn the playing style of a pianist, they use first-order logic to describe how the pianist would play a certain classical piece, and then a clustering algorithm to group similar melody lines (i.e. phrases) together. They use the models to play back music in the style of given pianists. Pachet's Continuator uses Markov models to create a system that allows real-time interactions with musicians [8], however his focus is more on replicating the *tonal* signature of a musician; the Markov model represents the probabilities that a certain note will follow another. A musician plays a phrase and the Continuator will then play another phrase which is a *continuation* of that phrase. Mantaras and Arcos use case-based reasoning to generate expressive music performance by imitating certain expressive styles, such as joyful or sad [2]. Raphael [10] has implemented a real-time system for accompanying soloists, "Music Plus One". The system allows soloists to play along with an orchestra played by a computer, after the soloist has "practiced" along with the system. The idea is to model how a soloist tends to vary the tempo when playing a classical piece of music, making the orchestra (i.e. the computer) follow the soloist.

Current sophisticated drum sample software (e.g. FXpansion BFD, Toontrack EZdrummer, DigiDesign Strike, Reason Drum Kits, Native Instruments Battery) contains gigabytes of samples that closely match the acoustic response to playing dynamics. However, the drum libraries still need to be programmed, since they provide no intelligent way to generate human-like drum patterns. This must be done by the user, either by programming the pattern himself or choosing a rhythm template. The drum software contains parameters that can be tweaked to enhance the realism of the produced tracks (typically a "groove engine" where it is possible to increase randomization of beats and/or adjust timing and velocity), in addition to manually changing programmed patterns. Still, the user needs to know how to achieve the desired result, since the software has no understanding of how to generate human-like drum patterns. If the user could buy a "drummer in a box" that had a *model* of how a real drummer plays a certain pattern, it would greatly reduce the cost of having life-like drums. We believe this could be an important tool for musicians, since the programming of the drums would be easier and the user could select the drummer of his preference to perform on his tracks.

3 ARCHITECTURE

The architecture for learning and imitation of drum patterns is called "Software for Hierarchical Extraction and Imitation of drum patterns in a Learning Agent" (SHEILA), see figure 1. SHEILA learns drum patterns and the playing style of human drummers, and stores them in a library. After training, SHEILA can be used as a groovy artificial drummer, capable of imitating the playing style of the drummers that provided the drum patterns used as training data. The drumming domain is suited for time-dependent sequential modeling due to its repetitive nature, since the groove of a drummer manifests over time.

[1] IDI, NTNU, Norway, email: tidemann@idi.ntnu.no
[2] ISN, Imperial College London, UK, email: y.demiris@imperial.ac.uk

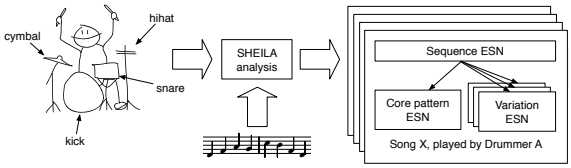


Figure 1. The SHEILA architecture. Playing style is learned in a hierarchical fashion by learning large-scale variations (i.e. variations of a pattern), as well as small-scale variations (variations of dynamics and tempo). All the models are represented by Echo State Networks (ESNs). The melody is used to group similar drum patterns together. After the training phase, SHEILA can imitate the playing style of the drummers that served as teachers.

3.1 Modeling the groove using neural networks

The inputs to SHEILA are drum patterns and the accompanying melody, both represented in the MIDI[3] format. SHEILA models both small- and large-scale variations of drum patterns using Echo State Networks (ESNs). An ESN is a recurrent neural network characterized by a large sparsely connected hidden layer, where only the output layer weights are modified during learning [6]. The input layer weights are generated at random, and left unchanged during training. By only modifying the output layer weights, training the network is reduced to a linear regression problem, which is a lot cheaper computationally compared to the backpropagation through time algorithm. In SHEILA, the input layer is not used. The ESNs learn the sequences by teacher forcing, i.e. by writing the desired sequence of output states into the output nodes during the training phase. The only inputs to the hidden layer comes from the output nodes. After training, the teacher forcing stops, and the network runs on its own. The ESN continues to generate the desired output sequence due to the reverberations of the hidden layer dynamics. In other words, the desired sequence of output states is stored as an attractor in the ESN. How this is used in SHEILA will be explained in the following sections. Why use ESNs to represent the groove sequence? Apart from the obvious advantage of using a dynamical system with memory capabilities, it also draws on biological inspiration; neuroscientific findings suggest specific areas of the brain process temporal musical information [11]. To model a human quality such as groove, using a technique that is modeled on how the brain works seems like a step in the right direction.

The name of the drummer and the song is also stored in the SHEILA library. Different drummers play the same pattern, but in their own style. The name of the drummer and the song will then help the user of SHEILA to decide which style he wants on the imitated drum track when SHEILA is used to imitate a drummer.

3.1.1 Large-scale variations

Large-scale variations are defined as changes in the actual pattern played, such as playing a break instead of a certain pattern. The most commonly played drum patterns are denoted *core patterns*. Core patterns are found by a recursive process: first the different parts of the melody are found (what in common music terms would be referred to as the verse/chorus/bridge of the song). This is achieved by transforming the melody into a string, and searching for *supermaximal repeats*, an approach used in computational biology to discover sequences of genes [4]. A supermaximal repeat is defined as a recurring pattern that is not a substring of any other pattern. Similar parts are grouped together, and the core patterns are the most commonly played drum pattern within the similar parts. To find the most commonly played drum patterns, the same search for supermaximal repeats is performed. Patterns correspond to one bar of the MIDI note sequence (i.e. 4 quarter notes). Patterns that differ from the core pattern within a melodic segment is considered to be a large-scale variation of the core pattern. For each song, there will be several core patterns, corresponding to the melodic segments. Core patterns are written as C_x, whereas variations are written as $C_x V_y$. From the low-level MIDI data a high-level representation of the song is found, namely the sequence of drum patterns. The sequence of patterns within a melodic segment is represented by an ESN (from now on referred to as ESN_{seq}). The string sequence is transformed into a binary matrix where one row corresponds to one bar, and the location of the high bit indicates which pattern (core or variation) to play. This sequence is then teacher forced to the ESN_{seq}, which produces the same output sequence after the training phase. The design choice to have one ESN_{seq} for similar melodic segments was made because it is the intention that SHEILA will later be used in a different setting, where only specific core patterns (and their variations) are to be played. If the ESN_{seq} learned the large-scale sequence of the entire song, it would only be suited to play back that particular song.

3.1.2 Small-scale variations

Small-scale variations are defined as variations in timing (i.e. how much the drummer is before or after the metronome) and dynamics (i.e. how hard a beat it played, also referred to as velocity) that occur when a drummer is playing a pattern. After defining the core patterns and variations, the similar drum patterns are grouped together, and the MIDI data is transformed into a target matrix where one row represents one timestep of the MIDI data. Quantizing the raw MIDI data allows for calculation of how much a note was before/after the metronome. The placements of the velocity and onset time data also code for which note was played (for instance, hihat, snare drum or kick drum). The velocity and onset times were scaled to the range [0,1]. One ESN represent the sequence of velocities (denoted ESN_{vel}), one ESN represent the sequence of onset times (ESN_{ons}). Early experiments tried to combine both onset time and velocity in one ESN, but finding a stable solution was difficult. Closer examination of the data revealed that the onset times had a variation that was on a slower timescale than that of the velocities. The onset times varies over several bars, whereas the velocity varies greatly from one note to the next (this will be elaborated in section 4). The spectral radius of the ESN describes the speed of the internal dynamics of the ESN, and is the most important parameter to tune [5]. It was therefore crucial that the velocity and onset times were represented on different networks, since this parameter needed to be different for each network. After the division was made, finding stable solutions became a lot easier.

The grouped patterns are then used to train ESNs that represent their variations in velocity and timing, resulting in specific ESNs for the core pattern and for each of the variations of the core pattern.

3.2 Imitating the groove

When the training is completed, the user of SHEILA presents a desired pattern in the MIDI format. Upon recognition, the user can choose which drummer should play the desired pattern. The name

[3] Musical Instrument Digital Interface, a protocol for electronic music equipment to communicate in real time.

of the drummer and which song the pattern was played on will aid the user to decide. The user then specifies how many bars the desired pattern should be played. SHEILA then runs the ESN_{seq} of the desired pattern for the desired number of bars, outputting the sequence of core patterns and variations. The corresponding ESN_{vel} and ESN_{ons} are then run for the desired number of bars; the output results in MIDI data. The ESN_{seq} introduces large-scale variations, and the ESN_{vel} and ESN_{ons} introduce small-scale variations. Recall that the ESNs are all purely generative, they are not driven by input at all. However, they need to be given a starting state, which is the last state of the hidden and output layer of the network during training.

4 EXPERIMENTAL SETUP

The SHEILA system was implemented in MatLab. To find the supermaximal repeats, *vmatch*[4] was used, which is implemented using the algorithms described in [1]. Propellerheads Reason 3.0[5] loaded with Reason Drum Kits was used for recording MIDI signals and for generating sound from MIDI files. Recording MIDI was done with a Roland TD-3[6] velocity sensitive electronic drum kit. Five male amateur drummers (average age 27.8) recorded drum tracks to a melody written by the authors, and were told to play specific patterns for the verse (shown in figure 2), chorus and bridge. At each 8th bar of the verse, there was a break that they had to play the same way. Apart from these directions, the drummers were free to introduce large-scale variations as they saw fit. The overall structure of the song was verse/chorus/verse/chorus/bridge/chorus/chorus. The tempo was 120 beats per minute (BPM), yielding the length of the song 2:30 minutes. The ESN_{seq} had 50 hidden nodes, and a spectral radius $\alpha = .99$. The spectral radius describes the speed of the internal dynamics of the ESN; $\alpha = .99$ specifies slow dynamics. This was chosen because the timescale of the ESN_{seq} were often long and highly irregular. The ESN_{vel} and ESN_{ons} on the other hand, represent faster dynamics since their role is to capture a short cycle in a long stream of data. By examining the training data, the timescale of which variations occur was discovered to be different on velocity and timing data. This can be seen in figure 3, which shows the velocity and timing data for the hihat sequence that correspond to the pattern in figure 2. Observe how the velocity data vary greatly from one note to the next, whereas the onset time varies more slowly.

To account for these observations, $\alpha = .1$ for the fast ESN_{vel}, $\alpha = .4$ for the slower ESN_{ons}. These values are found by experimentation, as recommended by Jaeger [5]. The networks started out with 10 nodes in the hidden layer. Finding a stable solution in a purely generative ESN was not guaranteed in every trained ESN. The training error would be low for every network, but once left to run on its own, some networks tended to oscillate in an unstable manner. To overcome this problem, each ESN was run for the same length as the training data, and if the resulting output pattern differed more than 10% from the training pattern, it was discarded and another ESN was created, trained and tested. If five consecutive ESNs had an error greater than 10%, the number of nodes in the hidden layer was increased by 1. In practice, this simple heuristic would guarantee that a solution would be found rather quickly, with less than 25 nodes in the hidden layer. Recall that the training of the ESN is a simple linear regression task; training and testing an ESN on some of the longer sequences (e.g. a 144 x 3 matrix) takes less than a second on an 1.8GHz iMac G5 running MatLab 7.

[4] www.vmatch.de
[5] www.propellerheads.se
[6] www.roland.com

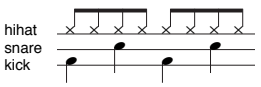


Figure 2. One of the three core patterns the drummers were required to play in the experiment.

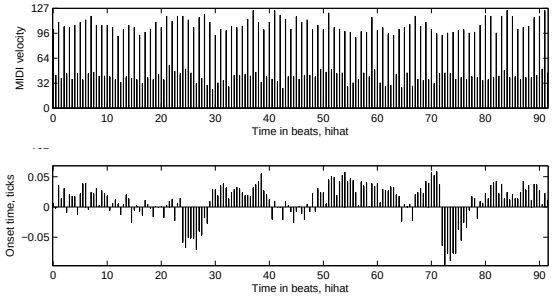


Figure 3. The plots show the training sequence of hihat velocity and onset time when drummer B played the pattern in figure 2. The difference from one velocity to the next is much larger than the difference from one onset time to the next, which fluctuates on a much slower timescale.

5 RESULTS

To evaluate the imitation performance of the SHEILA architecture, it was set to play back the same song structure used during training. Table 1 shows how many large-scale variations a drummer would introduce when playing a core pattern in addition to how often variations were played instead of the core pattern, calculated from the original training data. The table shows how some drummers tends to introduce many variations and play them often, whereas other drummers tend to play just the pattern they were told to play. This indicates the complexity of the sequence the ESN_{seq} had to learn, and the complexity of the imitated sequence.

Table 1. The tuples represent how many unique variations the drummers introduced when playing a core pattern, and how often variations in total were played instead of a core pattern (keeping in mind that a particular variation can be repeated throughout the sequence). This indicates the complexity of the sequence of large-scale variations and core patterns.

Drummer	A	B	C	D	E
C_1	5, 54%	3, 18%	7, 43%	2, 7%	5, 46%
C_2	10, 41%	2, 22%	8, 41%	1, 3%	11, 63%
C_3	6, 75%	2, 38%	5, 63%	0, 0%	5, 63%

The small-scale variations of both velocity and onset time can be modeled using a Gaussian distribution, as described in [13]. One of the leading software samplers on the market, FXpansion BFD, use the same approach to model human variations in its "Humanize panels"[7]. This allows for a simple way to show the small-scale variations present in both the original and imitated data. Figures 4 and 5 show how SHEILA models the small-scale variations of drummers A and E playing the pattern shown in figure 2 (due to space limits,

[7] See page 118 of the user manual (accessed 2008-05-26), www.fxpansion1.com/resourceUploads/BFD_Manual_English.pdf

the graphs for all drummers cannot be shown). Figure 4 shows how drummer A strongly accentuates (i.e. periodically varies the velocity of) the hihat beats, whereas drummer E has a more even velocity profile for the hihat beats.

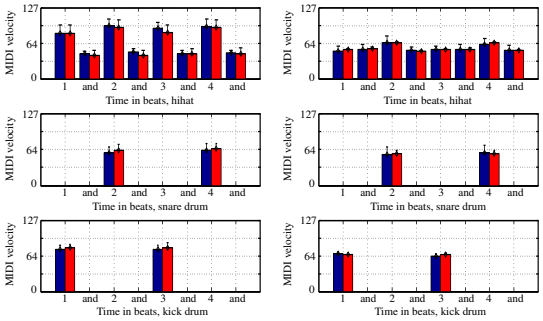


Figure 4. To the left is the velocity profile for drummer A, playing the pattern shown in figure 2. The Y scale is $[0 - 127]$, corresponding to the MIDI resolution. The X scale corresponds to the beats in the measure, which is a common way to count when playing music. The blue bar stems from the training data, the red bar from the output of SHEILA, when instructed to play the same song as that of the training input. The similarity between the blue and red bars indicate that the ESN_{vel} successfully captures the small-scale variations of the training data. Notice also how the velocity profile differs from that of drummer E (to the right). Most easily seen is how the accentuation (i.e. variation of velocity) on the hihat is not as pronounced as for drummer A; this is a manifestation of the different grooves of drummers A and E.

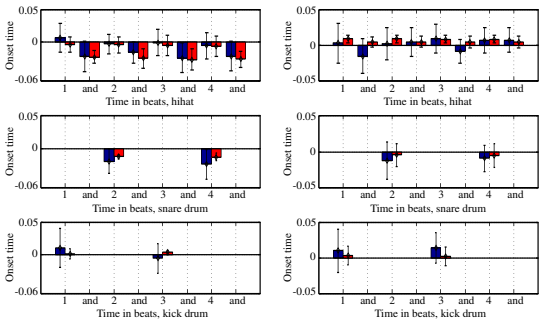


Figure 5. To the left is the onset time profile for drummer A, playing the pattern shown in figure 2. The Y scale is onset time in *ticks*. There are 120 ticks in the range $[0 - 0.99]$ between each quarter note. The X scale corresponds to the beats in the measure, similar to figure 4. As in figure 4, the blue bar is the statistics from the training data, the red bar is the analysis performed on the imitation done by SHEILA, showing that the output of the ESN_{ons} resembles that of the training data. The plot shows how drummer A tends to be ahead of the metronome when playing the pattern in figure 2. To the right is the onset time plot for drummer E. The onset times tend to be more centered around the metronome for the hihat beats, distinctively more than for drummer A, which contributes to the difference of groove between drummers A and E.

The onset time plays an important role in how aggressive/relaxed drum patterns are perceived, depending on whether the onset time is before or after the metronome. Figure 5 reveals that drummer A tends to be ahead of the metronome (yielding a more aggressive feel), whereas drummer E tends be more centered around the metronome, for a more "tight" feel. The authors are aware that these terms are vague but acoustically distinct; we encourage the reader to listen to available MP3 files[8] that better demonstrate these differences (included are imitations performed by SHEILA). Figures 4 and 5 show the mean and standard deviation for both velocity and onset time, both for the original data and the imitated output. The similarity between the plots shows how SHEILA successfully models the small-scale variations, in addition to demonstrating that drummers A and E plays the same pattern with a different groove.

To assess both the large- and small-scale differences between original and imitated drum tracks, as well as between drummers, a sequence similarity metric was implemented as described in [7]. The cost function was adapted to account for differences in velocity as well as timing of events, e.g. by adding the normalized difference in velocity between two events. The similarity metrics can be seen in table 2. The metrics show that imitations are similar to originals, and that the drummers have different styles when compared to another. The difference when comparing originals to imitations and drummers to each other is generally an order of magnitude. However, note that the metrics only have value as relative *comparisons* between the MIDI sequences. They do not represent an absolute difference. Yuijan and Bo have recently developed a *normalized* metric [16], however it does not account for timed series; this appears to be an open research issue, and beyond the scope of this paper. Still, the similarity metrics indicate a strong degree of similarity between original drum tracks and imitations (which is further backup up by figures 4-5), and that each drummer has a playing style different from the others.

Table 2. (a) shows the similarity metric described in [7] when comparing original drum tracks to SHEILA's imitations, (b) compares drummers to other drummers. The metrics indicate that the originals and imitated drum tracks are similar, and that the different drummers have different playing styles.

Original	A	B	C	D	E
Imitation	0.46408	0.37102	0.37176	0.60169	0.37995

(a)

	A	B	C	D	E
A	0	5.185	5.8272	6.1193	6.9911
B	5.185	0	5.4271	1.944	5.4166
C	5.8272	5.4271	0	6.0649	6.4713
D	6.1193	1.944	6.0649	0	6.135
E	6.9911	5.4166	6.4713	6.135	0

(b)

Another important aspect of the onset time is the tempo drift that occurs over time. A drummer will constantly be before or after the metronome, which will make the tempo fluctuate over time, as can be seen in figure 3. Figure 6 shows how the output of SHEILA induced the same drift in tempo over time as that of the original drum sequence. To examine how the ESN store the grooves as attractors, plots were made of hidden layer nodes during a run where the ESN was generating output. Figure 7 shows plots for some hidden nodes of the ESN_{vel} of the pattern in figure 2 for drummer A. The ESN_{vel} was run for 240 timesteps (double what it was trained on). The figures show that the activation patterns have stable attractor shapes, but with deviations. This is a further testament to how small-scale variations are introduced when imitating a certain pattern; these deviations will make the output slightly different over time. But since

[8] www.idi.ntnu.no/∼tidemann/sheila

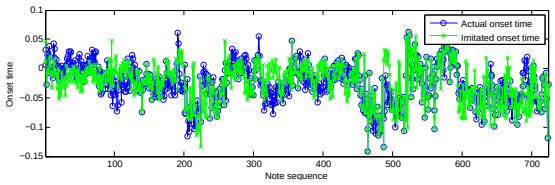


Figure 6. Tempo drift throughout the song, drummer A. The circle plots show the tempo drift present in the recorded drum patterns. The cross plots show the onset times during imitation. Observe how both the original and imitated note sequence drift over time in a similar fashion.

the attractors are modeled from the patterns from a human drummer, the fluctuations will be *similar* to that of the human drummer.

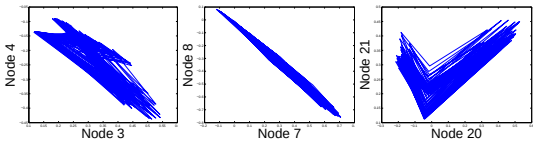


Figure 7. Attractor plots, for some randomly selected hidden layer nodes of the ESN_{vel} of the pattern in figure 2, drummer A. The ESN_{vel} was run for 240 timesteps (twice the training length). The plots are stable, but with deviations implying that the output will be slightly different over time.

6 DISCUSSION AND CONCLUSION

The choice of using MIDI was based on simplicity regarding the gathering and analysis of data. Another advantage with MIDI is that it will disregard the sound of the drums, which often will help to identify a drummer. The MIDI data allows focusing on the playing style of the drummer, which is the aim for our research.

By using ESNs, SHEILA is able to model the human quality that is *groove* by using a biologically inspired computational model. The model encompasses the quality of varying the output like that of a human drummer, making the output different from the original but still *recognizable*. The research presented in this paper enables the drum machine to become closer to a groovy human drummer. Effectively, the results will make it cheaper and easier to create human-like drum tracks when making music.

7 FUTURE WORK

SHEILA depends on MIDI information gathered using a MIDI drum kit. Acquiring the data is expensive; extracting the drum patterns and melody line directly from sound files would give access to vast amounts of training data; possible approaches are described in [9]. Apart from ease of computation, the reason for recording live drummers was an interest in making SHEILA learn the *physical* playing style of human drummers, i.e. the movement of the arms, upper torso and head. This would be used to visualize SHEILA. During the experiments conducted for this paper, motion tracking was also done. The goal is to be able to use SHEILA in a live setting as an *accompanying musician*, interacting with humans playing other instruments. This work will continue research done with motion tracking and subsequent imitation of arm movements as described in [14], using multiple forward and inverse models as building blocks for a motor control architecture [3].

ACKNOWLEDGEMENTS

The authors would like to thank the referees who helped improve the paper, and the drummers who participated in the experiment (Daniel Erland, Inge Hanshus, Sven-Arne Skarvik, Tony André Søndbø).

REFERENCES

[1] M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch, 'Replacing Suffix Trees with Enhanced Suffix Arrays', *Journal of Discrete Algorithms*, **2**, 53–86, (2004).
[2] Ramon Lopez de Mantaras and Josep Lluis Arcos, 'AI and music from composition to expressive performance', *AI Mag.*, **23**(3), 43–57, (2002).
[3] Yiannis Demiris and Bassam Khadhouri, 'Hierarchical attentive multiple models for execution and recognition of actions', *Robotics and Autonomous Systems*, **54**, 361–369, (2006).
[4] Dan Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*, Cambridge University Press, New York, NY, USA, 1997.
[5] Herbert Jaeger, 'Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network"', Technical report, German National Research Institute for Information Technology, (2005).
[6] Herbert Jaeger and Harald Haas, 'Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication', *Science*, **304**(5667), 78–80, (2004).
[7] H. Mannila and P. Ronkainen, 'Similarity of event sequences', *TIME*, 136–139, (1997).
[8] Francois Pachet, *Enhancing Individual Creativity with Interactive Musical Reflective Systems*, Psychology Press, 2006.
[9] G. E. Poliner, D. P. W. Ellis, A. F. Ehmann, E. Gomez, S. Streich, and B. Ong, 'Melody transcription from music audio: Approaches and evaluation', *IEEE Transactions on Audio, Speech and Language Processing*, **15**(4), 1247–1256, (May 2007).
[10] Christopher Raphael, 'Orchestra in a box: A system for real-time musical accompaniment', in *IJCAI workshop program APP-5*, pp. 5–10, (2003).
[11] Séverine Samson and Nathalie Ehrlé, *The cognitive neuroscience of music*, chapter Cerebral substrates for musical temporal processes, Oxford University Press, 2004.
[12] Craig Saunders, David R. Hardoon, John Shawe-Taylor, and Gerhard Widmer, 'Using string kernels to identify famous performers from their playing style.', in *ECML*, eds., Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, volume 3201 of *Lecture Notes in Computer Science*, pp. 384–395. Springer, (2004).
[13] Axel Tidemann and Yiannis Demiris, 'Imitating the groove: Making drum machines more human', in *Proceedings of the AISB symposium on imitation in animals and artifacts*, eds., Patrick Olivier and Chris Kay, pp. 232–240, Newcastle, UK, (April 2007).
[14] Axel Tidemann and Pinar Öztürk, 'Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA', in *IEA/AIE 2007*, volume 4570 of *Lecture Notes in Computer Science*, pp. 291–302. Springer Verlag, (June 2007).
[15] Asmir Tobudic and Gerhard Widmer, 'Learning to play like the great pianists.', in *IJCAI*, eds., Leslie Pack Kaelbling and Alessandro Saffiotti, pp. 871–876. Professional Book Center, (2005).
[16] Li Yujian and Liu Bo, 'A normalized levenshtein distance metric', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(6), 1091–1095, (2007).

A GROOVY ARTIFICIAL DRUMMER

E

Authors:
Axel Tidemann, Pinar Öztürk and Yiannis Demiris

Abstract:
This paper presents an architecture for an intelligent virtual agent that imitates human drumming behaviour. Through imitation, the agent models the user-specific variations that constitute the "groove" of the drummer. The architecture comprises a motor system that imitates arm movements of a human drummer, and a sound system that produces the sound of the human playing style. The presence of a sound system alleviates the need to use physical models that will create sound when a drum is struck, instead focusing on creating an imitative agent that booth looks and sounds similar to its teacher. Such a virtual agent can be used in a musical setting, where its visualization and sound system would allow it to be regarded as an artificial musician. The architecture is implemented using Echo State Networks, and relies on self-organization and a bottom-up approach when learning human drum patterns.

Main Result:
Combining my previous work on imitation of dance movements with the groovy drum machine. The neural network-based drum machine now drives the motor system of a groovy artificial drummer. The introduction of the Predictive Error Controller is an incremental improvement to the architecture.

To appear in:
9th International Conference on Intelligent Virtual Agents, IVA2009, 14-16 September 2009

My main contributions to the paper:

- Programming the system and running the experiments
- Coming up with the idea of the Predictive Error Controller, inspired from how the cerebellum works. This drastically improves performance of the system
- Data gathering and analysis
- Writing the paper

The co-authors contributed to the following areas:

- Writing the abstract, introduction and discussion

A Groovy Virtual Drumming Agent

Axel Tidemann[1], Pinar Öztürk[1], and Yiannis Demiris[2]

[1] IDI, NTNU, Sem Sælands vei 7-9, 7491 Trondheim, Norway
axel.tidemann@idi.ntnu.no
[2] BioART, EEE, Imperial College, Exhibition Road, SW7 2BT London, UK

Abstract. This paper presents an architecture for an intelligent virtual agent that imitates human drumming behaviour. Through imitation, the agent models the user-specific variations that constitute the "groove" of the drummer. The architecture comprises a motor system that imitates arm movements of a human drummer, and a sound system that produces the sound of the human playing style. The presence of a sound system alleviates the need to use physical models that will create sound when a drum is struck, instead focusing on creating an imitative agent that booth looks and sounds similar to its teacher. Such a virtual agent can be used in a musical setting, where its visualization and sound system would allow it to be regarded as an artificial musician. The architecture is implemented using Echo State Networks, and relies on self-organization and a bottom-up approach when learning human drum patterns.
Keywords: Embodied cognitive modeling, Architectures for virtual agents, Artistic application

1 Introduction

The research focus of this paper is to create a virtual drumming agent that plays drum patterns in an intelligent way. The intelligence lies within its ability to *model* and *imitate* drum patterns from human drummers. The agent learns drum patterns from human drummers, and is able to create new drum patterns that are similar but not identical. The sound system can be regarded as an intelligent drum machine in itself. Drum machines play patterns perfectly, however this flawless way of playing is also what makes it sound rigid and mechanical. Human drummers will always introduce small variations in both tempo and dynamics when they play; this is what makes the drummer sound *groovy*. The sound system models these variations to create a groovy drum machine; it can be used in a live setting on itself, however a visualization of the drumming agent would increase its appeal as an artificial *drummer*. For this reason, the system incorporates a motor system in addition to the sound system that is able to imitate arm movements of human drummers. The imitative drumming agent both sees and hears the teacher, and the architecture fuses these two modalities. The sound system is based on previous work on groovy drum machines [1], and the motor system is based on previous work on imitation of arm movements [2]. The research goal in this paper is to combine these two areas of research,

to create an intelligent virtual drumming agent. The agent relies on the human cognitive function of imitation, and its implementation draws on biological inspiration, with self-organizing neural networks being an important feature. Both the sound system and the motor system are discussed in the paper, but due to space limitations the self-organization of the motor system is investigated in slightly more detail. Two important distinctions need to be clear: the sound system models user-specific variations, and is able to imitate them to create new groovy drum tracks. The motor system tries to *accurately* produce the arm movements that will yield the desired groovy drum track, i.e. it "remembers" the arm movement that produced the desired sound. Therefore, it is evaluated by how similar the produced arm movement is to the original trajectory.

2 Background

Learning motor skills by imitation has long been regarded as an important cognitive function, and has been studied extensively in developmental psychology [3, 4]. The discovery of neurons firing both when performing and observing the same action (dubbed *mirror neurons* [5]) has been suggested as a neural implementation of the imitative capability [6], language [7] and mind reading [8]. In computer science, model-based learning is regarded as the most suitable approach to implement imitation of motor actions [6], an approach well-known in the control literature [9]. This approach pairs an inverse model (i.e. controller) with a forward model (i.e. a predictor), and has been used in architectures for imitation learning [10, 11]. It has been argued that the cerebellum contains inverse/forward model pairings, and it is therefore a suitable approach for an architecture for motor control and learning in an imitative setting [12].

Modeling user-specific variations (i.e. "modeling virtuosity") in the cross-section of AI and music has been the focus of several studies (predominantly on piano music), where various techniques ranging from string kernels [13], first-order logic and clustering [14], Markov models [15] and case-based reasoning [16] have been used to model and subsequently generate music that sound like they were made by humans. The most sophisticated drum machines today (e.g. FXpansion BFD, Toontrack EZdrummer, DigiDesign Strike, Reason Drum Kits, Native Instruments Battery) are made in software, since they allow large sample libraries (some are in the gigabyte range) and fine-tuned control over various parameters to tweak the sound of the drums. However, there is no *intelligent* way to generate human-like drum tracks, apart from adding random noise with the intention that the noise can be perceived as "human". The approach in this paper is to *model* how human drummers play, and use these models to imitate the style of human drummers. With an ability to imitate arm movements, the groovy drum machine would become closer to a groovy artificial drummer, suitable to be part of a band in a live setting. The Haile robotic percussionist [17] has some similarities, but it is limited to playing with one arm (albeit in real-life, not only as a simulation). It replays and modifies drum patterns during interaction with other players, but it does not *learn* the patterns.

3 Architecture

The architecture presented in this paper is called "Software for Hierarchical Extraction and Imitation of drum patterns in a Learning Agent" (SHEILA), see figure 1. It combines techniques from imitation of drum patterns [1] as well as imitation of body movements [11, 2, 18]. The two subsystems that make up the architecture will now be described.

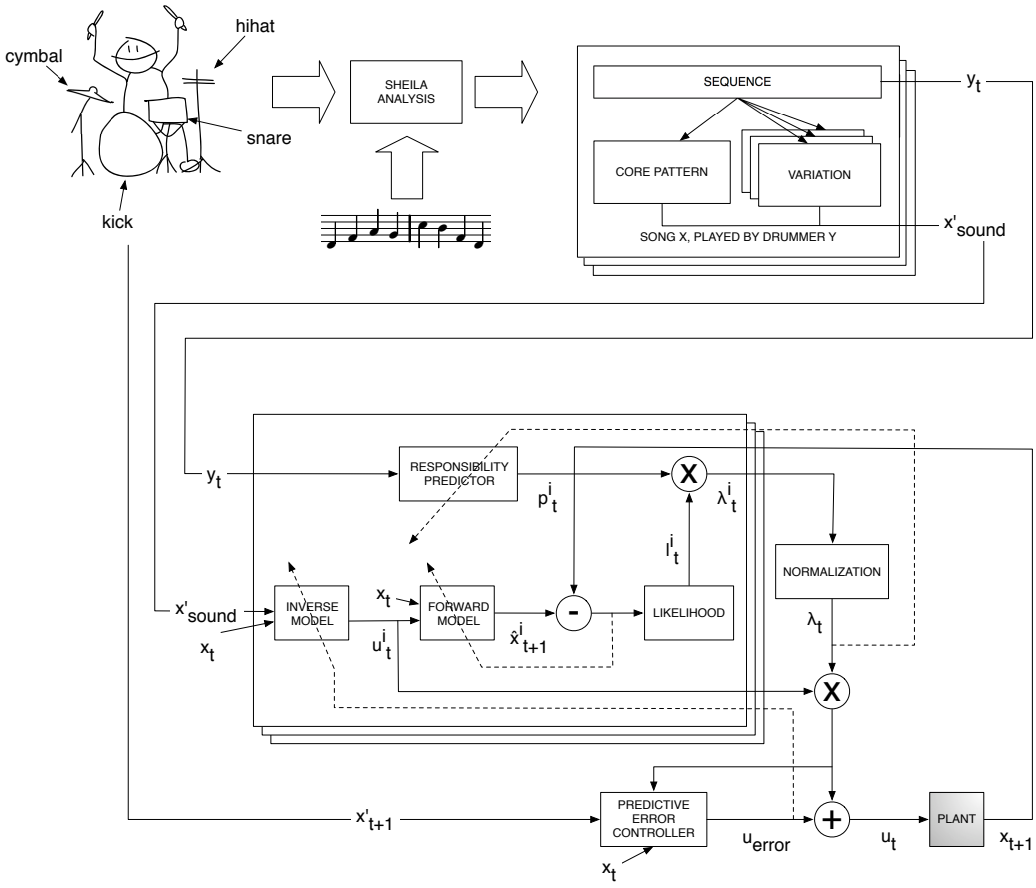


Fig. 1: The SHEILA architecture. On the top is the sound generative part, driving the motor part at the bottom.

3.1 Sound System

The sound generative part is on the top of figure 1. It consists of several Echo State Networks (ESNs) [19] that learn user-specific variations from human drum-

mers. MIDI[3] recordings facilitate the analysis of drum patterns. The resulting drum patterns are analyzed in a hierarchical manner: first, the MIDI drum sequence is transformed into a string. Similar patterns are then found by looking for supermaximal repeats, a technique used to find sequences of genes [20]. The melody of the song is used to divide the song into different parts (in common music terms, these would constitute the verse/chorus/bridge of the song). The most commonly played pattern within a part is then regarded as a core pattern C_x, where a pattern has the length of one bar (i.e. 4 quarter notes). Patterns that differ are regarded as *large-scale variations* of the core pattern, i.e. C_xV_y. Similar patterns are grouped together. From the similar patterns, the *small-scale variations* are extracted. These consist of variations in tempo and dynamics that the drummer introduces when playing a certain pattern. These are found in the MIDI data by looking at two parameters: 1) the onset time, which says how much a beat was offset relative to the metronome, and 2) the velocity, which describes how hard a note was played.

Both the small- and large-scale variations are stored in ESNs, to exploit the memory capacity and fast training algorithm. The sequence of core patterns and variations are transformed into a bit matrix, where each row encodes a core pattern and its variation (if any), and stored in an ESN called ESN_{seq}, one for each melodic segment. The onset time and velocity are scaled to $[0, 1]$ and the resulting matrix is used to train ESNs that model the small-scale variations of a pattern. The ESN_{seq} thus gates the corresponding ESNs modeling the small-scale variations of a pattern, and the output is scaled back to the MIDI format. All these ESNs are self-generating networks, there is no input signal to drive them; the ESNs use the feedback connections from the output layer to reverberate around the desired state. To use the sound system as a groovy drum machine alone, the user must decide which core pattern should be played, and for how long. The system then lets the specified ESN_{seq} run for the desired length of time. The output of the ESN_{seq} then gates the output of the corresponding ESNs that represent the actual core pattern and the variations of the core pattern. The output of these ESNs are what creates the actual sound, the ESN_{seq} merely governs when large-scale variations are to be introduced.

3.2 Motor System

An important aspect of this research is the idea that in order to achieve a life-like visualization, it is crucial to have an underlying motor architecture based on biological inspiration. For this reason, the research in this paper focuses more on the underlying principles that will yield a control architecture capable of generating arm movements similar to that of the teacher; the implementation of the visualization is simple compared to the effort that went into the control architecture. The research is performed more from an AI perspective than an approach with a focus on the animation of the agent. The latter approach could

[3] Musical Instrument Digital Interface: a protocol for real-time communication between electronic instruments and computers.

simply use the recorded motion tracking data to create the desired trajectories. The approach in this paper is to *learn* the arm movements that will generate the same trajectories; this is intrinsically linked to the research goal of creating an intelligent drumming agent. The motor part of SHEILA can be seen at the bottom of figure 1. The motor architecture consists of a number of paired inverse and forward models, and a responsibility predictor (explained below). It is inspired by Demiris' HAMMER [10] and Wolpert's MOSAIC [11] architectures, combining the best of both: the consistent inverse/forward ordering of HAMMER and the responsibility predictor of MOSAIC. The core of the architecture is the pairs of inverse and forward models. An inverse model is a controller that issues motor commands u_t^i based on the current state x_t and a desired state $x_t^{'}$[4]. A forward model is a predictor, that predicts the next state $\hat{x}_{t+1}^i$ based on the current state x_t and the motor commands u_t^i from its paired inverse model. The inverse and forward model, along with the responsibility predictor are grouped together in a *module*. Each module learns and represents a different motor skill (or a set of skills). The motor control problem then comes down to the selection of the module(s) that best produces the desired movement. The responsibility predictor (RP) predicts p_t^i, signifying how suitable a module is to control the robot prior to movement, based on contextual information y_t. In [11] the following example is given: if the robot is to lift a cup, contextual information would be whether it is empty or full, so the proper inverse model can be chosen. The likelihood (l_t^i) is a model that expresses how well the forward model predicts the next state (see [18] for details), which is multiplied with p_t^i. The result is λ_t^i, representing how well the module is performing *and* a prediction of how suited the module is to control the robot. All the λ_t^i from the different modules are normalized into the final λ_t vector. The motor output u_t^i from each module is multiplied with its corresponding λ_t^i value, and all the motor commands are summed to form the final motor command u_t, see equation (1).

$$u_t = \sum_i \lambda_t^i u_t^i \tag{1}$$

λ_t thus enables switching of control between different modules. This also gates the learning of the models; modules that perform well will receive more of their error signal than modules with bad performance. This way, the motor system self-organizes the control of the robot. Figure 4 (which will be explained in the Results section) illustrates how the λ value of each module changes when playing different sound patterns; the motor system self-organizes what motor knowledge each module will capture.

The inverse models need good error signals in order to converge. This is supplied by the predictive error controller (PEC), which is inspired by the human

[4] In this paper, the following notation is used: a desired state is indicated by a prime symbol, e.g. $x^{'}$. An estimated state is noted by the hat symbol, e.g. $\hat{x}$. A superscripted i indicates that the signal comes from one of the modules, e.g. u^i, otherwise it is system-wide, for instance the desired state $x^{'}$ is the same for all modules. A subscripted t indicates time, so the current state at time t is x_t.

cerebellum. Like the cerebellum, it is able to predict how well motor commands will achieve the goal of the movement, and make adjustments before the final motor commands are sent to the motor system. The inputs are thus the current state x_t, the desired state x'_{t+1} and the motor commands u_t sent from the motor system. It then uses a model to predict the outcome of the motor commands on the system, and if there are any discrepancies between the goal and the predicted outcome, it issues motor commands u_{error} that will correct the situation, which are added to u_t before sent to the robot. This approach was initially inspired by the universal feedback controller [21]; however it is only *reactive* because it issues corrective motor commands based on the performance of the system at the previous timestep. The PEC is *predictive*, and thus more able to issue good corrections to the final motor command. u_{error} is used to train the inverse models of the motor system. The forward models are trained on the actual next state x_{t+1}, whereas the RPs are trained on the final λ_t vector.

3.3 Combining Sound and Motor Systems

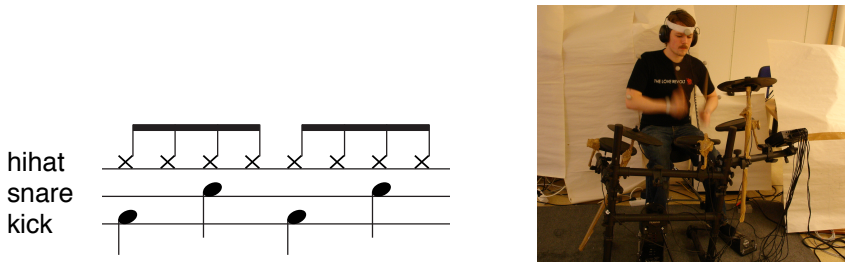
The sound generative part was previously used solely as a neural network-based drum machine [1], now it is the neural centre that drives the motor part of SHEILA; it provides both the context input to the motor system, as well as the desired state input. The sequence of core patterns and variations serve as context input y_t to the motor part. In previous work [2, 18] y_t was defined by the designers of the experiment, now it is extracted from the low-level data recorded from the drummer, i.e. the context signal is *data driven*. The actual sound output is used as the desired state x'_{sound} to the motor system. In previous work, the desired state was the same as the desired state of the robot at the next timestep. Now, the inverse models receive the current state x_t of the robot and the desired *sound* x'_{sound} which is what the *effect* of moving the arm should sound like; i.e. a target state that is in a different coordinate system than that of the robot. This makes it harder for the inverse models to issue the correct motor commands u_t^i, since it must model the different coordinate systems used for the two input signals. The desired state of the robot x' is used as input to the PEC, which then is able to correct erroneous motor commands sent from the motor part of SHEILA. This can be thought of as a memory of what the resulting arm movement should look like.

It should be noted that a more realistic simulation of a drummer would include a physical simulation of the drums that the robot is hitting. In the current implementation, the imitation of sound is done regardless of the movement of the arms. The current simplification, however, made it possible to have special focus on the actual imitation of movement and sound, which allowed the creation of a virtual agent that will look and sound similar to the original drummer.

4 Experiment

The five drummers that participated in the experiment played drum patterns to a melody written by the first author. The drummers had to play specific drum

patterns (see figure 2a), but were free to introduce variations that felt natural to them. The task of the imitative architecture was then to imitate both the sound and arm movements of the drummer. To achieve this, two experiments were conducted: 1) the training of the sound system and 2) the training of the motor system. These experiments will now be explained.



(a) (b)

Fig. 2: (a) One of the patterns the drummers were told to play. (b) A drummer playing on the Roland TD-3 velocity sensitive drum kit, which was also captured using motion tracking.

4.1 Experiment 1: Training the Sound System

MIDI was recorded using a Roland TD-3 velocity sensitive electronic drum kit, see figure 2b. The software for recording MIDI was Propellerhead Reason 3.0. After analyzing the melodic structure and finding the corresponding core patterns and the variation of core patterns, the ESNs of the sound system was trained. The size of the ESNs depended on the complexity of the recorded drum patterns. The high-level ESN_{seq} learned the sequence of core patterns and the variations, and the low-level ESNs learned to model the sequences of velocities and onset times. All the ESNs in the sound system were self-generative, i.e. they were not driven by an input signal. The ESNs were teacher-forced [19], using the reverberations of the hidden layer dynamics to generate the correct output. Since the complexity and size of the target matrices differed from each drummer, the size of the ESNs were not determined beforehand, but searched for by the system itself (described in detail in [1]). After training the sound system, it was set to imitate the same sequence that had been used for training. The output of the trained sound system served as input to the motor system.

4.2 Experiment 2: Training the Motor System

Movement data was gathered using a Pro Reflex 3D motion tracking system, which uses five infrared cameras to track fluorescent markers. The markers were put on the shoulders, arms and wrists of the drummer in the experiment. The song consisted of two alternating patterns with corresponding melody, i.e.

verse/chorus/verse/chorus, lasting 98 seconds. The Pro Reflex sampling rate of 20Hz made the models predict 0.5 seconds into the future. The noisy motion data was the desired state $x^{'}$ used by the PEC. The elbow coordinates were normalized to the range $[-1, 1]$ for all three dimensions, with the shoulder as origin. The wrist coordinates were normalized to the same range with the elbow as origin. The robot was defined in the same way, to overcome the correspondence problem [22]. Neuroscientific data suggest that such a transformation of visual input from an external to an intrinsic coordinate frame occurs in the brain [23]. To simulate a robot with human-like arms, a four degree of freedom (DOF) model of a human arm was implemented [24]. The model has a three-dimensional spherical shoulder joint, and a one-dimensional revolute elbow joint. The entire simulated robot was described by 8DOF.

The inverse models had 30 input signals. 12 represented the current state x_t of the robot, corresponding to the 3D coordinates of the elbow and wrist of both arms. The remaining 18 inputs corresponded to the $x^{'}_{sound}$ signal, i.e. the velocity and onset time of the various elements of the drums, i.e. snare drum, kick drum, hihat and so on. There were 8 outputs in the range $[-1, 1]$ which made up the motor commands u^i_t to the robot. The forward model had 20 inputs, 12 stemming from x_t and 8 from u^i_t, and 12 outputs to predict the next state $\hat{x}^i_{t+1}$. The RPs had 14 input signals, coding the core pattern and variation to be played. The output was a prediction of the suitability of the module to control the robot, p^i_t, in the range $[0, 1]$. The motor system was tested with different sizes of the hidden layer of the ESNs. All networks of the motor system had spectral radius $\alpha = 0.9$ which determine the length of the internal memory (range $[0, 1]$, with increasing memory as α increases) and noise level $v = 0.2$ which adds 10% noise to the internal state of the network. The PEC implements the same model as the simulated robot, which enables it to make accurate predictions and therefore good error signals u_{error} for the inverse models, crucial for such a high-dimensional system to converge. The motor system started out with 10 modules in each experiment. For every second epoch the activity of the modules was examined: a module had to be at least 20% active (i.e. $\lambda > 0.2$) for at least 10% of the time, otherwise it was pruned. The check was done every other epoch to allow the system to stabilize before pruning again. There were three stopping criteria: 1) the performance error p_e had to be less than 1%, 2) the RP/λ error had to be less than 5%. If the output of the RPs correspond to the final λ value of a module, it correctly predicts how well suited the module is, indicating stability in the system, 3) the u_{error} had to be less than 50% of the total motor command, so that the inverse models control most of the robot.

5 Results

5.1 Experiment 1

In order to test the imitative quality of the sound system, the groovy drum machine was set to play back the same sequence that it was trained on. Performing the same statistical analysis on both the original and generated data sets reveals

that the system is able to model and generate the learned drum patterns, see figure 3. More details about the imitative qualities of the sound system, including statistical analysis of the imitative performance of the system, can be found in [1].

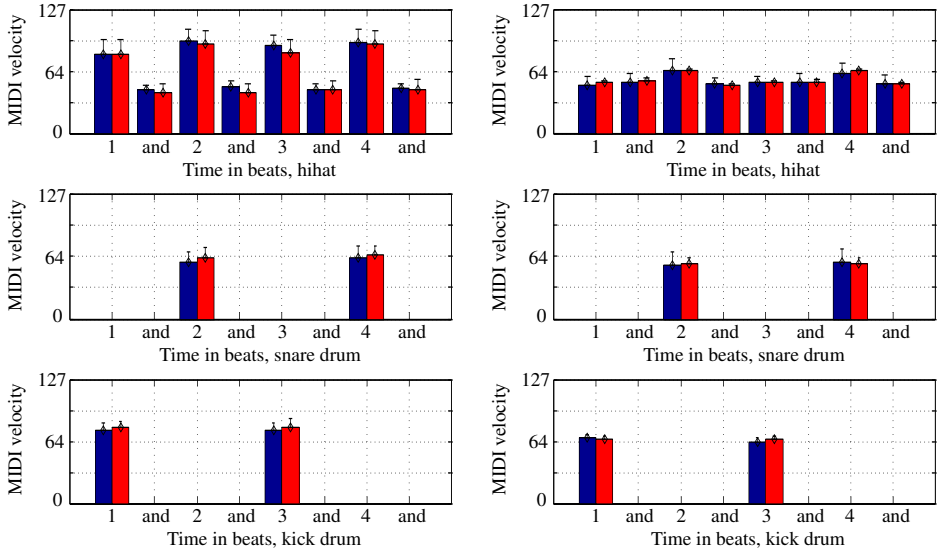


Fig. 3: An example of the imitative quality of the sound system; *two* different drummers are shown, one in each plot. The sound system was used to generate a drum sequence similar to that of the original training data. These similarities are shown in bars, they are all organized in pairs within each plot. The bar on the left of each pair (blue) shows the training data, and the bar on the right of each pair (red) shows the generated sequence. This similar pairs of bars shows how the sound system learned the same drum pattern as in figure 2a for each of the two drummers.

5.2 Experiment 2

Note that the experiment reported in this paper focuses on *one* case of motion tracking, whereas the previous work on the sound system alone [1] featured several drummers. The motor system was tested with five different sizes of the hidden layer: 100, 250, 500, 750 and 1000 nodes. Each network configuration was run 20 times. The results from the experiments can be seen in table 1. The motor system distributed the movement knowledge across different modules, as can be seen in figure 4, which also shows the complexity of the context signal. Figure 5 shows how the system matches the target trajectory when imitating.

6 Discussion

Figure 3 shows how the sound system successfully models and imitates the playing style of different drummers. The sound system learns user-specific variations

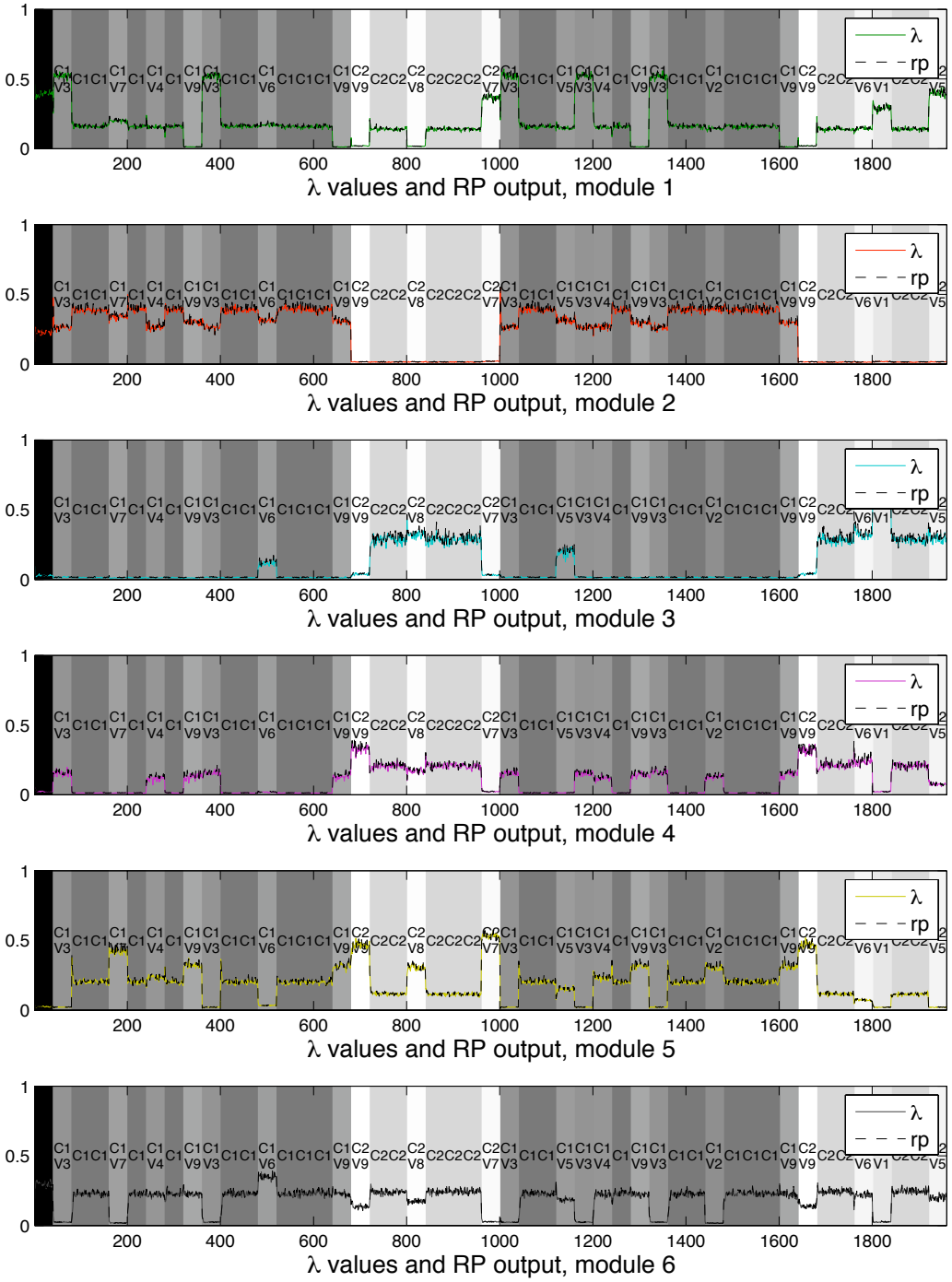


Fig. 4: An example of λ and RP output, 500 nodes in the hidden layer. The shades of gray in the background shows the boundaries of the context signal. The letters indicate which core pattern and corresponding variation the context signal was made up from, making it easier to see recurring context signals. The black column to the far left signify the count-in. In accordance with table 1, it allows for a visual inspection of how the system self-organizes the decomposition the control of the target movement into different modules, and how they collaborate when controlling the robot.

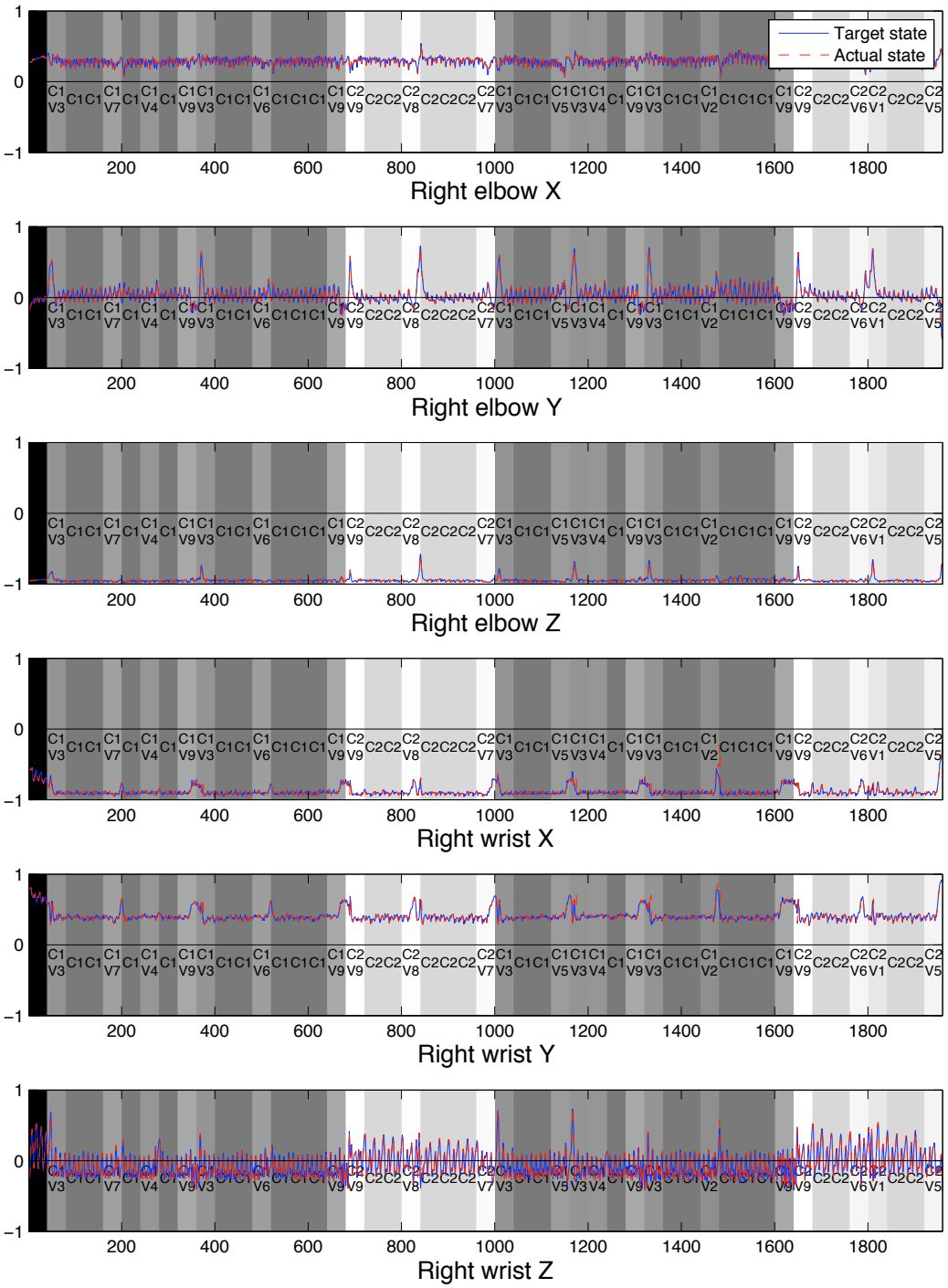


Fig. 5: Performance of the motor system. Note how the two lines depicting the actual state versus the desired state overlap. In the background the context signal is shown, as in figure 4 (same experiment). This shows the right arm; due to space limits the left arm is not shown. The left arm plots show a similar match between desired and actual state, a p_e around 0.03% is typical (see table 1).

Nodes	Modules (μ, σ)	Rec. activation (μ, σ)	Epochs (μ, σ)	Perf. error (p_e) (μ, σ)	u_{error} ratio (μ, σ)	Conv. exp.
100	4 ; 0	99.58% ; 0.83%	15 ; 0.82	0.0285% ; 0.0036%	49.72% ; 3.3%	20%
250	4.47 ; 0.70	98.57% ; 1.58%	14 ; 2.87	0.0273% ; 0.0037%	48.64% ; 1.05%	95%
500	5.20 ; 0.83	93.74% ; 4.21%	12 ; 1.81	0.0349% ; 0.0103%	47.47% ; 1.88%	100%
750	5.05 ; 0.87	91.39% ; 5.94%	12 ; 1.84	0.0341% ; 0.0104%	45.82% ; 1.88%	100%
1000	5.15 ; 0.81	88.46% ; 6.11%	12 ; 1.63	0.0358% ; 0.0091%	44.91% ; 1.84%	100%

Table 1: Results from the experiments of the motor system. The "Modules" column shows how many modules the system was using on average after training. "Rec. activ." is short for *recurrent activation* and tells to what extent the λ value of each module was recurring when the context signal was repeated. The recurring activation value was calculated as follows: for repeating context signals, the mean and standard deviation of λ was calculated for all modules. If the standard deviation was more than 5% of the mean during one part of the recurring context signal, it was counted as *not* being recurrent. Within those λ values within the 5% limit, only those that differed less than 1% from the mean counted towards the recurrent activation percentage. A high recurrent activation value indicates that modules specialized on certain parts of the movement, since modules had the same λ value (i.e. influence) over the robot during recurring context signals. "Perf. error" is short for *performance error* (p_e), representing how much the imitated trajectory differed from the desired trajectory. The u_{error} ratio indicates how much the PEC influenced the final motor command after training. "Conv. exp" is short for *converged experiments*, showing how many of the 20 experiments converged (if the experiment had not converged within 40 epochs, it was terminated).

of drumming patterns, and stores the grooves in the hidden layer dynamics of the ESNs. The sound system is able to generate drum patterns that are similar to the training data, but not identical. The sound system then drives the motor system: the sound system produces a sequence of drum patterns similar to the original training data, which the motor system receives as a target state representing what the end result of the arm movement should be (i.e. the sound). Since the sound system has been thoroughly discussed in [1], the discussion will now focus on the motor system. The results show that the motor system produces the correct arm movements based on the produced sound patterns, see table 1 (the column for *performance error*) and figure 5. Since the research focus of this paper is to create an intelligent agent, there has been an emphasis on developing a motor architecture based on biological principles. Table 1 and figure 4 reveal that the motor system successfully distributes control of the movement to be imitated between the different modules. Table 1 indicates that the smaller networks (100 nodes) are the most efficient networks, when solutions are found (only 20% of the experiments converged). These networks have the highest recurrent activation value, meaning that the modules actively repeat their activation for repeating context signals. This is less for the biggest networks, which could indicate an excess in neural resources allows for modules to have overlapping motor knowledge.

The sound and motor systems are both based on biological principles of self-organization, implemented with neural networks, and are designed to be *intelligent* systems. The sound system drives the motor system, which is where the fusion of modalities happens - the motor system "hears" what the end result should be, and issues motor commands that will result in that particular sound. The motor system is able to transform a desired state in a different reference frame (i.e. sound) into actions that will lead to that sound; the sound system operates at a higher level than the motor system since it outputs *consequences* of arm movements. The fusion of modalities is therefore not limited to sound: the sound system could be replaced with any other centre that issues desired states in different reference frames from that of the motor system.

An agent that understands the link between sound and movement could also be used in the gaming industry. Current popular games such as Rock Band and Guitar Hero receive musical input from the player (typically through a guitar-like interface), but the avatar on the screen does not respond to this input. A possible use of the SHEILA architecture could be to generate a visualization of an avatar that would move in accordance with the performance of the player, for greater visual feedback when playing.

7 Future Work

For all experiments, the u_{error} ratio is relatively high on average (ranging from 44.91% to 49.72%). The architecture controls most of the motor output, but the PEC is crucial for the system to function well. However, this resembles how the brain works: high-level motor commands are sent from the dorsolateral frontal cortex to the posterior parietal and premotor areas, specifying the spatial characteristics of the desired movement. Details of the motor signals are defined in the motor circuits of the spinal cord [25]. Future work will show if the motor system of SHEILA works in a similar fashion.

SHEILA does not imitate drum tracks that are unknown to the system. However, it should be fairly trivial to implement this feature in the sound system. Based on the already learned models of drum patterns, the knowledge of similar drum patterns could be used to generalize to unknown patterns. Once this mechanism is in place for the sound system, the motor system would require some way of estimating the arm movement required for the novel drum patterns. A model that learned drum patterns and the corresponding trajectories of the arm could then be employed to create predictions of what trajectories would be the result of an unknown drum pattern, required for the PEC to function.

References

1. Tidemann, A., Demiris, Y.: Groovy neural networks. In: 18th European Conference on Artificial Intelligence. Volume 178., IOS press (July 2008) 271–275
2. Tidemann, A., Öztürk, P.: Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA. In: IEA/AIE. Volume 4570 of Lecture Notes in Computer Science., Springer (June 2007) 291–302

3. Piaget, J.: Play, dreams and imitation in childhood. W. W. Norton, New York (1962)
4. Meltzoff, A.N., Moore, M.K.: Imitation of facial and manual gestures by human neonates. Science **198** (October 1977) 75–78
5. Rizzolatti, G., Fadiga, L., Gallese, V., Fogassi, L.: Premotor cortex and the recognition of motor actions. Cognitive Brain Research **3** (1996) 131–141
6. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences **3**(6) (1999) 233–242
7. Arbib, M.: The Mirror System, Imitation, and the Evolution of Language. In: Imitation in animals and artifacts. MIT Press, Cambridge (2002) 229–280
8. Gallese, V., Goldman, A.: Mirror neurons and the simulation theory of mind-reading. Trends in Cognitive Sciences **2**(12) (1998)
9. Jordan, M.I., Rumelhart, D.E.: Forward models: Supervised learning with a distal teacher. Cognitive Science **16** (1992) 307–354
10. Demiris, Y., Khadhouri, B.: Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems **54** (2006) 361–369
11. Wolpert, D.M., Doya, K., Kawato, M.: A unifying computational framework for motor control and social interaction. Philosophical Transactions: Biological Sciences **358**(1431) (2003) 593–602
12. Wolpert, D.M., Miall, R.C., Kawato, M.: Internal models in the cerebellum. Trends in Cognitive Sciences **2**(9) (1998)
13. Saunders, C., Hardoon, D.R., Shawe-Taylor, J., Widmer, G.: Using string kernels to identify famous performers from their playing style. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: ECML. Volume 3201 of Lecture Notes in Computer Science., Springer (2004) 384–395
14. Tobudic, A., Widmer, G.: Learning to play like the great pianists. In Kaelbling, L.P., Saffiotti, A., eds.: IJCAI, Professional Book Center (2005) 871–876
15. Pachet, F. In: Enhancing Individual Creativity with Interactive Musical Reflective Systems. Psychology Press (2006)
16. de Mantaras, R.L., Arcos, J.L.: AI and music from composition to expressive performance. AI Mag. **23**(3) (2002) 43–57
17. Weinberg, G., Driscoll, S.: Robot-human interaction with an anthropomorphic percussionist. In: CHI 2006 Proceedings. (April 2006) 1229–1232
18. Haruno, M., Wolpert, D.M., Kawato, M.: MOSAIC model for sensorimotor learning and control. Neural Comp. **13**(10) (2001) 2201–2220
19. Jaeger, H., Haas, H.: Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science **304**(5667) (2004) 78–80
20. Gusfield, D.: Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York, NY, USA (1997)
21. Kawato, M.: Feedback-error-learning neural network for supervised motor learning. In Eckmiller, R., ed.: Advanced neural computers. (1990) 365–372
22. Nehaniv, C.L., Dautenhahn, K.: The Correspondence Problem. In: Imitation in Animals and Artifacts. MIT Press, Cambridge (2002) 41–63
23. Torres, E.B., Zipser, D.: Simultaneous control of hand displacements and rotations in orientation-matching experiments. J Appl Physiol **96**(5) (2004) 1978–1987
24. Tolani, D., Badler, N.I.: Real-time inverse kinematics of the human arm. Presence **5**(4) (1996) 393–401
25. Kandel, E.R., Schwartz, J.H., Jessell, T.M.: Principles of neural science. McGraw-Hill, New York (2000)

USING MULTIPLE MODELS TO IMITATE DRUMMING

F

Authors:
Axel Tidemann and Pinar Öztürk

Abstract:
The ability to learn motor skills by imitating other individuals is an important part of human cognition. The research presented in this paper aims to implement this cognitive ability in robots. The goal is to facilitate programming of robots in a way that is natural to humans. To achieve this goal, a multi-modal architecture for learning motor skills by imitation is implemented. The system imitates human drumming behaviour. The goal is to imitate both the playing style (i.e. the "groove") as well as the arm movements. The virtual drummer can then be used in a musical setting, as it will both look and sound human. Echo State Networks are used to implement the architecture, and it self-organizes the control of the simulated robot. How the system self-organizes the control of the robot is evaluated.

Main Result:
The sequence of sound and movements were randomly perturbed to show that the architecture learns different parts of the movement, and it is not the sequence alone that determines the performance of the system.

Submitted to:
Journal of Artificial Intelligence Research

F. Using Multiple Models to Imitate Drumming

My main contributions to the paper:

- Programming the system and running the experiments
- Writing the paper

The co-authors contributed to the following areas:

- Writing the paper, particularly the discussion

Using Multiple Models to Imitate Drumming

Axel Tidemann TIDEMANN@IDI.NTNU.NO
Pinar Öztürk PINAR@IDI.NTNU.NO
IDI, Norwegian University of Science and Technology
Sem Sælands vei 7-9, 7491 Trondheim, Norway

Abstract

The ability to learn motor skills by imitating other individuals is an important part of human cognition. The research presented in this paper aims to implement this cognitive ability in robots. The goal is to facilitate programming of robots in a way that is natural to humans. To achieve this goal, a multi-modal architecture for learning motor skills by imitation is implemented. The system imitates human drumming behaviour. The goal is to imitate both the playing style (i.e. the "groove") as well as the arm movements. The virtual drummer can then be used in a musical setting, as it will both look and sound human. Echo State Networks are used to implement the architecture, and it self-organizes the control of the simulated robot. How the system self-organizes the control of the robot is evaluated.

1. Introduction

Learning by imitation enables easy transfer of motor skills between individuals. By simply watching another individual demonstrate a motor skill, you are able to reproduce the same behaviour. It is not hard to imagine how this has been a tremendous advantage to the development of human society. Our goal is to enable the imitative capability in robots. With the increase of elderly in many developed nations, robots can play a very important part in their daily care, performing tasks for the nurses (Pineau, Montemerlo, Pollack, Roy, & Thrun, 2003) or as interactive pets (Wada, Shibata, Saito, & Tanie, 2004). This is a field that is in growth due to the predicted lack of people to care for the aging population. Robots could also play an important part in the daily life of people without need for assistance, performing repetitive menial chores. The designer of a robot cannot know in advance all the behaviours desired by the end user. Besides, different users may want to own a personalized robot that adapts to the personality and special needs of its owner. Therefore, robots must be able to learn new behaviours in different environments. Equipping robots with the ability to learn by imitation would be a convenient way to address the varied and changing needs of the end user. An adaptive robot should be able to imitate as easily as other humans in order to become truly useful to its owner. This paper presents an architecture that is able to imitate human drum movements. The goal is to create an artificial groovy drummer, that imitates the playing style of human drummers. This requires a multi-modal approach, since the imitating agent needs to both hear and see what it should imitate.

2. Imitation in Nature

Learning by imitation is a multi-disciplinary field of research. In developmental psychology, Piaget (1962) describes imitation learning as an ongoing process where the infant adjusts internal sensory-motor schemas to what it perceives in the outside world. Piaget believed the child could not imitate actions invisible to itself before it was about one year old. Meltzoff and Moore (1977) found that newborn infants are able to imitate facial gestures shortly after birth, suggesting that imitation is an innate mechanism that combines vision and proprioception in a way that can be used to guide the motor system of the infant. Meltzoff and Moore (1997) call this process *active intermodal mapping*.

Imitation was put in a neuroscientific context when Rizzolatti, Fadiga, Gallese, and Fogassi (1996b) discovered neurons that were active both when observing and performing the same action; they were named "mirror neurons". The initial discovery was in monkeys, Rizzolatti, Fadiga, Matelli, Bettinardi, Paulesu, Perani, and Fazio (1996a) found the same activity in human brains as well. These neurons were then hypothesized to be a "neural implementation" of the imitative capability (Schaal, 1999). Because of the placement of the mirror neurons coincide with the language areas of the brain (Broca's area, specifically), Arbib (2002) consider the mirror neurons crucial for our ability to have language. Since humans can imitate the sounds produced by other individuals, language-games necessary for the development of language became possible. Kohler, Keysers, Umilta, Fogassi, Gallese, and Rizzolatti (2002) found that mirror neurons were active also when *hearing* the sounds of an action, not only seeing the action being performed, i.e. the sound of cracking open a peanut elicited the same neural activity as when seeing or opening the peanut by the monkey itself. However, only seeing the peanut being opened without hearing the sound itself did not trigger the same activity, showing the importance of hearing the associated sound when understanding the action performed - if there was no sound of the opening of the peanut, the action was not successful.

Gallese and Goldman (1998) link the mirror neurons to our ability to empathize with others; the mirror neurons allow for a transformation of oneself into the situation of another person. It is the lack of this ability that identifies (amongst other symptoms) autism spectral disorder (ASD), and Williams, Whiten, Suddendorf, and Perrett (2001) hypothesized that people with ASD might suffer from a dysfunctional mirror neuron system. This was later confirmed by EEG (Oberman, Hubbard, McCleery, Altschuler, Ramachandran, & Pineda, 2005) and fMRI (Dapretto, Davies, Pfeifer, Scott, Sigman, Bookheimer, & Iacoboni, 2005) studies, however these findings have been met with skepticism (Dinstein, Thomas, Behrmann, & Heeger, 2008). Neuroscientists have questioned the direct link between the mirror neuron system in monkeys and humans (Dinstein, Hasson, Rubin, & Heeger, 2007; Dinstein, Gardner, Jazayeri, & Heeger, 2008; Lingnau, Gesierich, & Caramazza, 2009). If there is a mirror neuron system in humans, they are not the only neurons active during observation and execution of the same action. The concept of a mirror neuron system in humans remains controversial.

3. Imitation in AI

This section discusses imitation approaches in the field of artificial intelligence. Since the research in this paper has a multi-modal approach (i.e. imitation of both motor actions and sound), each will be discussed in separate subsections.

3.1 Imitation of Motor Actions

Imitation learning has gained considerable interest in the AI community over the last decade. Schaal (1999) divides AI research on imitation learning in two groups: 1) trying to solve the *correspondence problem*, which consists of transforming external sensor stimuli into an intrinsic reference frame (Nehaniv & Dautenhahn, 2002) or 2) assuming that sensor input has already been mapped to the motor system of the observing agent. Schaal considers model-based approaches as most suitable to implement imitative behaviour in robots, consisting of pairing an inverse model (i.e. behaviour or controller) with a forward model (i.e. predictor). This is an established approach in the control literature (Jordan & Rumelhart, 1992), which also has been implemented in AI architectures for imitation learning (Demiris & Khadhouri, 2006; Wolpert, Doya, & Kawato, 2003). Wolpert, Miall, and Kawato (1998) argue that inverse/forward models are present in the cerebellum, leading to an architecture based on those principles (Wolpert & Kawato, 1998). fMRI studies suggest that such an ordering is present in the brain (Imamizu, Kuroda, Yoshioka, & Kawato, 2004).

Gaussier, Moga, Banquet, and Quoy (1998) and Matarić (2002) propose architectures for motor imitation that have predefined modules for the various stages of sensorimotor processing, i.e. perception, recognition and action selection.

There are other approaches to imitation learning that focus on neural network architectures instead of a modular solution. (Tani, Ito, & Sugita, 2004) has developed a novel neural network architecture that implements the mirror neuron activity. This is achieved by adding specific neurons (called "parametric bias") that can be trained to have the same activation both when performing an observing an action. These neurons can also be activated manually, which will elicit the same behaviour. (Cangelosi & Riga, 2006) uses a neural network to ground symbols; the student learns the association between actions and words by imitating the actions of the teacher. (Billard & Hayes, 1999) employs a recurrent associative network to connect symbols to sensory perception. The student learns the association between labels and motor actions by imitating the teacher, using Hebbian learning to store the knowledge.

3.2 Imitation of Musical Expressiveness

Artificial intelligence is also applied when modeling human qualities of musical performances. To most people, the difference between a music played by a human compared to that of a machine is clear. Humans always introduce small errors when playing, which gives the performance its natural feel. Machines excel in playing without errors which results in a perfect performance, but this often is perceived to lack the human quality of "feel" or "groove". Many different techniques are used to model the human qualities of a performance. Saunders, Hardoon, Shawe-Taylor, and Widmer (2008) use string kernels to classify pianists based on audio recordings, modeling tempo and dynamics. The kernels are used

to generate a "performance worm", a 2D representation of the performance of the pianist. Alphabets are formed from the worms, and the resulting string representation is used to identify the pianists. Case-based reasoning has also been used to model expressiveness, such as the mood of a musical piece (de Mantaras & Arcos, 2002) and how the tempo of a piece of music can be changed and still maintain the expressive qualities perceived at the original tempo (Grachten, Arcos, & de Mantaras, 2006). The Continuator uses Hidden Markov Models to predict note sequences, and is a system that can be used in real-time to interact with musicians (Pachet, 2002). The focus of the Continuator is not to model the expressiveness in terms of tempo or dynamics, but the creativity that manifests in tone sequences of the musician. The system has also been used to help children improvise when learning to play the piano (Benghi, Addessi, & Pachet, 2008). Raphael (2003) has a system "Music Plus One" that allows soloists to practice with a computer that plays the score of the accompanying orchestra. The system models the expressiveness of the soloist, and adjusts the playback of the score accordingly (i.e. changes in tempo is followed by the system and missed notes does not confuse the score-following of the system). Tobudic and Widmer (2005) use first-order logic to describe the changes in tempo and dynamics that occurs when a pianist plays a piece of music. The system learns how the pianists play by clustering similar phrases together, and the acquired model can identify pianists based on playing style.

4. The Predictive Nature of Imitation

It is agreed that anticipation is a vital component of human intelligence. Neuroscientists contend that in order to exhibit skilled motor behavior, the brain needs to learn not only to control the body but also to predict the consequences of this control (Flanagan, Vetter, Johansson, & Wolpert, 2003; Kawato, 1999). The role of prediction has probably been best formulated in the *simulation hypothesis* (Decety, Jeannerod, & Prablanc, 1989; Jeannerod, 1994; Decety, 1996; Jeannerod, 2001; Hesslow, 2002; Barsalou, Barbey, Simmons, & Santos, 2003; Grush, 2004).

According to simulation hypothesis, the same neural mechanisms are activated when an action/movement is internally simulated without executing the action, and when the action is actually executed. For example, imagination of a finger movement and its physical execution activates the same primary motor area in the frontal lobe (Ehrsson, Geyer, & Naito, 2003).

Behavioral and neurophysical findings provide convincing evidence in favor of the hypothesis. It was shown that human subjects use equal time when they imagine themselves walking on a familiar path compared to when they actually walk there (Decety et al., 1989). The subjects took longer time when they imagined walking through a narrower gate than a larger gate at the same distance. This has been interpreted as an indication that subjects build up mental representations of the targets and retrieve these memories during motor imagery. Similar observations was done when the subjects are required to estimate the feasibility of grasping an object placed in different locations. The response time of the subjects was a function of the location indicating that the subjects must be mentally simulating the arm and finger movements (Frak, Paulignan, & Jeannerod, 2001). Neurophysiological findings also support the simulation hypothesis. fMRI studies, for example, demonstrated that

the primary sensorimotor cortex (which is active during motor execution) was activated during motor imagery and that mental and physical performance of hand movements share common functional circuits (Stippich, Ochmann, & Sartor, 2002). More specifically, imagination of movements using different parts of the body activates somatotopical sections of the motor cortices (Ehrsson et al., 2003). These findings lead to the use of motor imagery in sports training and neurological rehabilitation (Robin, Dominique, Toussaint, Blandin, Guillot, & Le Her, 2007; Zimmermann-Schlatter, Schuster, Puhan, Siekierka, & Steurer, 2008).

It has been conceived that cognitive tasks involving another person, such as mind reading, may also employ a similar internal simulation. Jeannerod argues that mind reading, which is the "ability for normal people to understand and predict the behavior of their conspecifics, ... represents an attempt to replicate and simulate the mental activity of the other agent" (Jeannerod (2001), p.104). Another such cognitive task is imitation. Evidence indicating that executing a particular action by oneself and observing another person performing the same action activates the same brain area. This was explained by a direct matching of the demonstrator's action onto an internal simulation of that action (Rumiati, Weiss, Tessari, Assmus, Zilles, Herzog, & Fink, 2005), making imitation a first class instance of the simulation hypothesis. Taken together, these findings suggest that mental simulation of a movement (either self-generated or triggered by a demonstrator) and executing it may be sharing the same neural circuits where there is an inhibitory element in the latter case, avoiding the overt action.

Importance of imitation for artificial intelligence, therefore, is twofold: 1) imitation in itself as a cognitive capability is one of the key elements of intelligence and is a requirement for intelligent agents/robots, 2) main components of imitation such as prediction, the involved models, and mechanisms of the activation of appropriate models are shared across different cognitive tasks and in different domains. Imitation therefore gains a special role in research investigating the architectural commonalities underlying the process of doing something and thinking or observing something.

5. Motor Behavior Underlying Imitation

The key element of the simulation hypothesis is a mapping from motor commands onto their consequences. The motor system underlying simulation and execution of an action rests on a coupling of two models (in control theory terminology); an inverse and a forward model (Flanagan et al., 2003; Wolpert & Kawato, 1998; Kawato, 1999).

A forward model is the component of the motor system that uses the current state of the motor system and motor commands to predict the next state (Miall, 2003); it implements a state transformation function that computes an estimation of the next sensory state of the motor system. The inverse model on the other hand computes the motor command that brings the body from its current state to the desired state. Hence, inverse model implements a controller while the forward model is a predictor.

It is envisaged that an efference copy of the output of the inverse model, the motor command, is sent to the forward model which, in turn, estimates what the next state of the body would be when the motor executes the command.

Forward models play an important role in prediction in general, which underlies a large range of cognitive tasks at various levels of intelligence. Existence of such models in the brain has been discussed for a while in neuroscience and evidence is provided supporting the existence of such models (Blakemore & Sirigu, 2003). For example, according to Blakemore, Wolpert, and Frith (2000)), it is the prediction feature/capability that explains why we cannot tickle ourselves: "During self-generated movement it is postulated that an efference copy of the descending motor command, in conjunction with an internal model of both the motor system and environment, enables us to predict the consequences of our own actions" (Blakemore, Goodbody, and Wolpert (1998), p. 7511). Based on this anticipation the motor system prepares itself and cancels the consequences of the command, explaining why we cannot tickle ourselves. Blakemore et al. (2000) provides evidence suggesting that the cerebellum might be involved in the generation of such predictions. Already in 1970, Ito and colleagues had suggested that forward models of the limbs may reside in the cerebellum (Ito, 1970).

Different roles have been attained to the forward models, all relating to some kind of prediction, such as state estimation, internal feedback to overcome time delays, distal supervised learning and prediction of the controller that is most appropriate in the current situation (Miall & Wolpert, 1996; Wolpert & Flanagan, 2001; Miall & King, 2008; Jordan & Rumelhart, 1992). Agents need the knowledge of current state in order to produce the next motor commands properly. For example, during a grasping movement, the agent needs information about the hand location for accuracy of the movement. Sensory input (e.g., visual feedback) provides the current state information. However, it alone cannot provide a perfect state information because they are subject to delays induced by the central nervous system. In addition, the sensor information is affected by noise. Therefore, sensory information, when used alone as the current state, can lead to instability in the movement. Nature's solution is to use another source of information to better estimate the current state: the prediction of the next state by a forward model. Another consequence of the delay is that the controller may need to adjust the motor command, a phenomenon called in neuroscience "feedback control". In this control strategy, the controller uses the discrepancy between the sensed state of the motor apparatus and the intended state as a negative feedback for its corrective operation. Forward states are needed here again, because of the delay in sensory input, and hence in the sensor feedback. The last example we gave above on the roles of forward model is the selection of controllers. This approach entertains the idea of multiple representations, i.e., the brain has different representations for the same type of movements in different contexts. A simplified example is given in figure 1. It shows two inverse models each coupled with a forward model. The performance of the forward models aids the system to select which inverse model should have more influence over the motor system.

Forward models, based on the estimated performance (i.e., appropriateness) of the candidate model, takes role in the selection of the best inverse model (i.e., the controller). Neuroscientists postulate that the brain stores experienced movements in a context-sensitive manner. More specifically, cerebellum hosts multiple internal models that compete to learn new objects and tools (Wolpert et al., 1998; Kawato, 1999). That is, reaching and grasping a full cup or an empty one requires different motor commands and the information about fullness/emptiness context helps the motor system to behave accordingly. Why would such an approach be an advantage? Consider the case where the context information is not

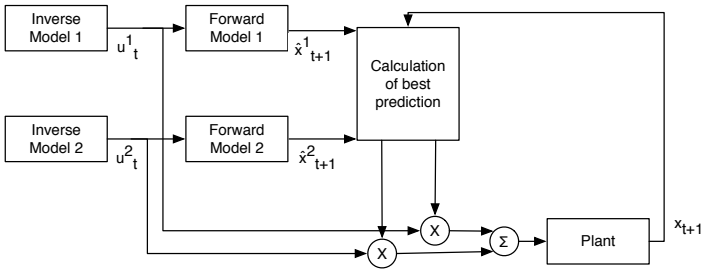


Figure 1: A simplified version of an action selection architecture with two pairs of inverse/forward models. An efference copy of the output of each inverse model is given to its paired forward model, which predicts what the outcome of the motor command will be. The forward model that correctly predicts the outcome of the motor commands will give its corresponding inverse model more influence over the motor system. A forward model that correctly predicts the outcome is more likely to have a suitable inverse model. This will bias the system towards the best inverse model, and relaxes the need for the system to try out both inverse models.

present: all the motor programs for lifting a cup would be considered equally suitable, and their appropriateness could only be established after trying them out. This would clearly lead to sub-optimal performance compared to an approach where the most suitable motor program can be chosen before any movement is initiated. Forward models, together with the context information, provide the basis for a proper selection.

It is this role of forward models we focus on in this paper. We present an architecture for imitative learning of groovy drumming that rests on multiple internal models, and a prediction-based controller selection mechanism.

6. Combining Imitation of Motor and Sound to Create A Groovy Drummer

Our aim is to create an artificial groovy drummer, which provided an excellent platform to investigate imitation architectures and the role of prediction in cognition. Building a robot with the required dexterity would be very expensive. Time and budget limits force us to use of computer simulation to implement the groovy drummer. The advantage of this approach is not only in terms of cost, but it allows an important simplification to be made: the simulated robot does not have to employ a physical simulation of the drums. Instead, it can imitate the arm movements required to *create* the sound. The actual sound generation can be done by another part of the architecture. In other words, the imitation of sound is separate from the imitation of arm movements. The imitation of sound can be thought of as an intelligent groovy drum machine. It models user-specific variations and uses the learnt models to play drum patterns with the same groove as the human drummer. This can be used as a groovy drum machine in its own right. But to create an artificial groovy *drummer*,

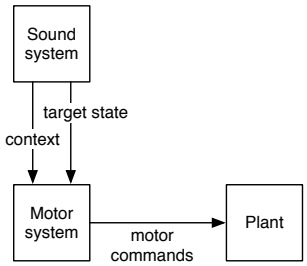


Figure 2: How sound and motor systems are combined. The output of the sound system drives the motor system, which uses the input to issue to motor commands to achieve the target state.

visualization is needed. This is where the motor system comes into play. By adding the ability to imitate motor actions, the architecture implements both the sound and action part of the groovy drummer. This simplification also allows each separate subsystem to be studied in detail.

However, there is one important distinction between the motor and sound system. The sound system is independent of the motor system; it produces sound output on its own. The motor system depends on the output of the sound system to know which arm movements to produce. The motor system produces the arm movements corresponding to the sound being created. The output of the sound system provides the *context* and the *target state* to the motor system; an overview of the architecture can be seen in figure 2. The motor system incorporates the inverse/forward approach outlined in section 4.

7. SHEILA: Software for Hierarchical Extraction and Imitation of Drum Patterns in a Learning Agent

This section presents the groovy drum architecture and explains how it implements imitative behaviour, with a special focus on the role of context and the forward models. The architecture (see figure 3) is divided in two parts: 1) a neural groovy drum machine, generating the drum patterns to be played and 2) a motor system, that produces the arm movements required to achieve the desired drum patterns. These will now be explained.

7.1 The Sound System: A Neural Drum Centre

The top part of figure 3 shows the neural drum centre of SHEILA. This part was developed as a groovy drum machine implemented with neural networks; a brief overview of the system will be presented in this section. Tidemann and Demiris (2008) describes the system in detail. The groovy drum machine was developed because there existed no intelligent way of generating human-like drum tracks in modern software, apart from adding random noise. The groovy drum machine *models* the expressiveness of human drummers, and can use the acquired models to generate groovy drum tracks. Five human drummers played to a piece of music written by the first author on an electronic drum kit, see figure 4. MIDI data was

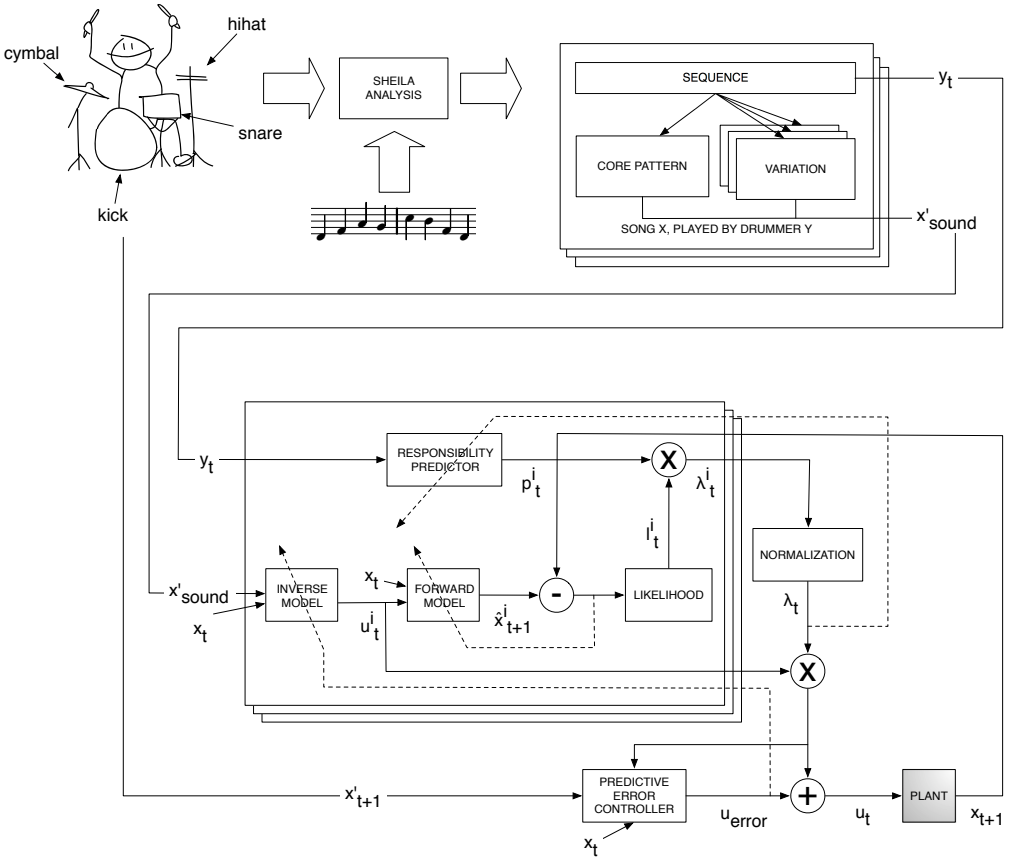


Figure 3: The SHEILA architecture. The neural drum centre on the top drives the motor system on the bottom. The full lines indicate sensory streams, the dashed lines are training signals. This figure is the detailed version of figure 2. See the text for details.

recorded from each drummer. From the low-level MIDI data, a high-level representation of the piece of music was formed. Similar patterns were grouped together, and the system learns these patterns and their inherent *groove*, i.e. the variations in tempo and dynamics. The system discovers the core patterns of the system, i.e. the pattern that was played the most for each part of the song (e.g. the verse and chorus). The patterns that deviate from the core pattern are the *large-scale variations* of that pattern; these are typically called *breaks* (i.e. adding or removing beats from the standard rhythm figure). The approach was to model both the sequence of core patterns and their variations, and the small-scale variations of the patterns. The learned model could then be used to produce new drum tracks, in the same playing style as the human drummer that served as training data.

It was conceived to use Echo State Networks (ESNs) to model the groove because of their strength in learning sequences. ESNs are characterized by a fixed input layer (randomly generated upon creation of the network) and a large hidden layer (Jaeger & Haas, 2004). The training of the network is then reduced to a linear regression problem, greatly reducing the training time. By arranging ESNs in a hierarchical manner, each ESN could operate on different timescales. A high-level ESN was trained on the sequence of core patterns and their variations, and low-level ESNs were trained on small-scale variations. The high-level ESNs would then gate the output of the low-level ESNs accordingly, as can be seen in figure 3. In the groovy drum machine, the ESNs are all self-generative, i.e. there is no input signal to drive them. Instead, it is the dynamics of the hidden layer that allow them to reverberate around the desired output sequence (technically, this is achieved by having connections from the output layer into the hidden layer). To generate drum patterns, the user needs only to let the sequence ESN run for the desired length, which gates the corresponding ESNs representing the patterns themselves. These ESNs subsequently generate a low-level stream of drum patterns that can be transformed into a MIDI sequence and played back using a software sequencer. The output of the drum centre drives the motor system.

7.2 The Motor System

The motor architecture uses the approach of Jordan and Rumelhart (1992), where an inverse model is paired with a forward model. Movement control rests on multiple pairs of inverse/forward models, and the switching between these pairs. This approach is inspired by two other architectures; HAMMER (Demiris & Khadhouri, 2006) and MOSAIC (Wolpert et al., 2003). The motor system will now be explained in more detail.

7.2.1 The Controller

The motor system consists of multiple controllers. A controller is equivalent to an inverse model, or a behaviour. It receives the current state of the system x_t and a desired state x'_{t+1}, and issues the motor commands u_t^i to achieve the desired state. In the current implementation, the desired state is the resulting *sound* that the arm movements should produce, denoted x'_{sound} to separate it from the desired state of the arm, i.e. x'_{t+1}. Previously, the desired state was expressed in the same coordinate system as that of the current state x_t (Tidemann & Öztürk, 2007). The difference in reference frames between x_t and x'_{sound} makes it harder for the inverse models to find the correct solution. The training signal

Figure 4: One of the drummers playing on the Roland TD-3.

comes from the predictive error controller, explained in section 7.2.3. The controller is implemented using Echo State Networks.

7.2.2 Selection of Controller

The advantages of using multiple controllers are many: it codes for redundancy, which is crucial for robust systems (Pfeifer & Scheier, 2001). It allows the network to scale with increasingly more complex tasks to be learned (i.e. the requires more neural resources). By growing the size of the controllers, problems such as catastrophic forgetting (where new concepts presented to the network destroys previously learned concepts) can be avoided (Ans, Rousset, French, & Musca, 2002). However, having more controllers requires a switching mechanism in order to select the controller best suited for the task at hand. The coordination of controllers is essential for the motor system to function. The action selection (i.e. selection of controller) is performed on the basis of two *predictors*, the forward model and the responsibility predictor (RP).

Note that for the rest of this text, the word *module* is used to group an inverse model, forward model and responsibility predictor. Such a modular approach is similar to Jacobs, Jordan, Nowlan, and Hinton's (1991) mixture of experts, allowing for redundant coding of motor skills, an important feature of robust systems (Pfeifer & Scheier, 2001). Farrar and Zipser (1999) claims a modular approach to motor control can best deal with the complexity of controlling a robot with the possibility of moving in 3D.

The Forward Model The forward model predicts the next state $\hat{x}^i_{t+1}$ based on the current state x_t and the motor commands applied from its corresponding inverse model u^i_t. The predicted state is compared to the actual state at the next timestep. This comparison can only be done after action has been taken, and this is why the system uses RPs to

determine which inverse model should influence the robot *prior* to movement. The difference between the predicted and actual next state is used together with the responsibility predictor (explained in the next section) to determine the appropriateness of the module it is a part of.

The Responsibility Predictor The responsibility predictor (RP) uses context input y_t to predict p_t^i, an indication of how much weight its corresponding inverse model should have upon the final motor command sent to the robot. The context signal in the current experiment is the sequence of core patterns and their variations. The sequence ESN in the drum centre outputs which core pattern and which variation (if any) to be played. If the system is in a state that is similar for many patterns, the context signal will help to determine the most suitable inverse model. Even though the forward model and the RP are both predictors, there is one notable difference: the output of the forward model can only be assessed at the next timestep, since the prediction of state $\hat{x}_{t+1}^i$ can only be verified when the actual state x_{t+1} is available to the system. The RP facilitates a *predictive* approach to selecting the best controller. This mechanism helps the system save resources, since bad solutions to a problem need not be examined.

Combining the RP and Forward Model Outputs The weight (or influence) of a controller is quantified in the final λ vector. It is calculated based on both the output of the RP and the *performance* of the forward model. The comparison of the predicted state $\hat{x}_{t+1}^i$ to the actual next state x_{t+1} is a vector, since $\hat{x}_{t+1}^i$ is subtracted from x_{t+1}. In order to express this vector as a scalar, a *likelihood model* is used, see Haruno, Wolpert, and Kawato (2001) for details. It assumes the presence of Gaussian noise, and outputs a scalar l_t^i that indicates how well the forward model predicted the next state. It is calculated according to equation (1).

$$l_t^i = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-|x_t - \hat{x}_t^i|^2}{2\sigma^2}} \tag{1}$$

The l_t^i is multiplied with p_t^i to form the λ_t^i. All the λ_t^i values are then normalized to form the final λ_t vector. The elements of λ_t now specifies how much each module should influence the control of the robot. This is effectuated by multiplying u_t^i with the corresponding value in the λ_t vector, and summing all the u_t^i into the final motor command u_t, similar to MOSAIC (Haruno et al., 2001). The λ_t vector gates the error signals of the models as well; models that perform well receives more of the error signal, enabling them to improve.

7.2.3 Online Correction and Learning: The Predictive Error Controller

A crucial aspect of any self-organizing system is how it is trained. The approach to training the forward model is straightforward: its output (a prediction of the next state, $\hat{x}_{t+1}^i$) can be compared to the actual next state x_{t+1}, and the difference between the predicted and actual next state can be used as a training signal. Similarly, the RP predicts how much influence its corresponding controller should have over the robot. If the RP makes a perfect prediction, there will be no difference between its output p_t^i and the corresponding value in the λ vector. The training signal of the RP will be the difference between the predicted influence p_t^i and the actual influence (i.e. λ).

However, finding the error signal for the controller is more difficult. For example, there is an infinite amount of ways to move your arm from your hip to your head. To remedy this problem, we introduce the predictive error controller (PEC). It is inspired by the feedback error controller (Kawato, 1990). The feedback error controller works in a simple, but effective way: it uses the difference between the desired state x'_{t+1} and the actual next state x_{t+1} to issue corrective motor commands. Since there are infinitely many ways to achieve a desired state from any position in a redundant system (Jordan & Rumelhart, 1992), this approach guarantees a solution will be found. We used this in previous work (Tidemann & Öztürk, 2007), but realized that its function could be improved if it was *predictive* instead of *reactive*. The feedback error controller is reactive because it can only issue corrective error signals *after* based on the difference between a previous prediction and an actual state that has just happened, similar to how the performance of the forward model is measured. The idea of the PEC is to overcome this limitation, and issue motor correction signals based on a prediction of the outcome of the motor commands from the motor system.

Since the PEC acts like a last-minute corrective facility before issuing the motor commands to the robot, it resembles the role of the cerebellum in the human brain. The cerebellum receives motor commands, information about the intention of the motor commands and proprioceptive information. The cerebellum is able to predict the consequences of intended motor actions, and issue corrective signals accordingly (Kandel, Schwartz, & Jessell, 2000; Wolpert & Kawato, 1998). This forms the basis for the PEC. Its inputs are the current state x_t, the desired state x'_{t+1} and the total motor command u_t. The PEC then predicts how well the motor commands will achieve the desired state. If necessary, the PEC issues a motor correction signal u_{error}, which is added to the motor command u_t before sent to the robot. u_{error} is also used to train the inverse models.

7.3 Combining the Sound System and the Motor System

As mentioned in section 6, the sound system can be used as a stand-alone neural network drum machine (Tidemann & Demiris, 2008). In the SHEILA architecture, this is the driving force behind the motor system. The sound system produces the sound output of the system as a whole, but it also provides the *target state* and *context* to the motor system. The sound output of the sound system is used as the target state of the motor system, denoted x'_{sound}. In previous work, the current state of the system x_t and the desired state x'_{t+1} was in the same coordinate frame (Tidemann, 2008; Demiris & Hayes, 2002). In SHEILA, the inverse models receive a target state representing what the *effect* of moving the arm should sound like, as well as the current state x_t of the system. The inverse models must now model the different coordinate systems for the two input signals, which is clearly harder than if they were in the same coordinate system.

The context signal y_t to the motor system represents the sequence of core patterns and variations. Previously (Tidemann & Öztürk, 2007; Haruno et al., 2001), y_t was hand-crafted. In the current experiment, the context signal comes from the low-level MIDI data recorded during the drum session.

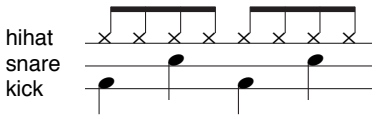


Figure 5: One of the patterns the drummers were told to play; a common pattern in pop/rock music.

The desired state $x^{'}$ recorded with the Pro Reflex motion tracking is used as input to the PEC, required for corrective control of the motor commands. This is similar to having a memory of what the arm movement looks like.

In this paper, the area of focus has been on modeling human drumming behaviour. But it should be noted that the SHEILA architecture is designed to be more general. In fact, the neural drum centre could be replaced with another module that sends out high-level commands to the motor system. This ability makes the system capable of *goal-directed* behaviour, since the goal of the action can be expressed in a higher level than the reference frame of the robot. The independence between the neural drum centre and the motor system in terms of coordinate frames (i.e. their inputs and outputs) is a testament to this capability. The output of the sound system is in principle no different than a speech synthesis module outputting spoken commands of what to do. Another example would be a high-level centre that issues a target location in form of latitude and longitude, and the motor system would then have to achieve this desired state (i.e. the given location).

8. Experimental Setup

The task of the system consisted of imitating the drumming patterns of a human drummer. The drummers played to a melody written by the first author, and were told to play specific patterns for each of the melodic segments; one such pattern can be seen in figure 8. The drummers were free to introduce variations. The neural drum centre imitated the rhythm patterns, whereas the motor system imitated the arm movements. The experiments performed were done in three stages; first the neural drum machine was trained. Then, the motor system was trained with input from the sound system, to see how the system self-organizes the control of the robot. The third experiment consisted of creating random permutations of the desired states that were given to the motor system (i.e. x, $x^{'}_{sound}$ and y), to examine if the modules learned the sequences of motor actions independently of the sequence in which they were trained, or if they were depending on the original sequence of motor actions.

8.1 Experiment 1: Training the Sound System

Drumming data was recorded using a Roland TD-3 velocity sensitive electronic drum kit, see figure 4. The use of an electronic drum kit facilitated data analysis, since it became possible to record MIDI signals which are far easier to analyze than recorded audio. The MIDI stream was then analyzed to find recurring patterns. The melody was used to segment

the MIDI stream into different parts (i.e. verse/chorus/bridge). The different segments were found by searching for supermaximal repeats, a technique used to discover sequences in genes (Gusfield, 1997). A supermaximal repeat is a string that is not a substring of any other string. Within each melodic segment (for instance, the verses) the same search for supermaximal repeats was performed on the MIDI stream from the drummer. The pattern that was most frequently played was labeled the *core pattern*, C_x. Patterns that differed from the core pattern were labeled *variations* of the core pattern, C_xV_y. Note that the designer of the experiment determined which pattern the drummer should play for each melodic segment (i.e. a core pattern), the system did not know what was a core pattern or a melodic segment beforehand. From the low-level MIDI stream a high-level representation of the song was then found; namely the sequence of core patterns and variations. For each of these patterns (either core or variation of a core pattern), the similar patterns are grouped together. The system then models the *small-scale variations* that constitute the groove of the drummer. The small-scale variations consists of changes in velocity (i.e. how hard a drum is struck) and timing (i.e. how much the beat was before or after the metronome). Both the sequence of core patterns and variations, as well as these small-scale variations were modeled using Echo State Networks (ESNs). More details of this system can be are described by Tidemann and Demiris (2008).

8.2 Experiment 2: Training the Motor System

At the same time as MIDI data was recorded, the movement of the arms of the drummer was recorded using a Pro Reflex motion tracking system. Five infrared cameras were used to track the position of the fluorescent markers on the drummer, this can be seen as the grey balls on the drummer in figure 4. The sampling frequency was $f = 20Hz$. No smoothing was performed on the noisy data; it was used as the desired state. This required the forward models to predict 0.05 seconds into the future. The use of motion tracking also solved the correspondence problem (Nehaniv & Dautenhahn, 2002): the recorded 3D positions of the human drummer could now serve directly as the desired state x'_{t+1} required by the PEC. The desired state consisted of four data points: 3D coordinates for each wrist and elbow. The wrist coordinates were normalized to the range $[-1, 1]$ with the elbow as the origin. The elbow coordinates were normalized to $[-1, 1]$ with the shoulder as origin. The normalization enabled a direct correspondence with the coordinate frame of the simulated robot. (Torres & Zipser, 2004) suggests that such a transformation occurs in the brain based on neuroscientific findings. This direct mapping of target and intrinsic coordinate systems has been done in previous work (Demiris & Hayes, 2002; Tidemann & Öztürk, 2007; Pastor, Hoffmann, Asfour, & Schaal, 2009). Tolani and Badler (1996) describe a four degree of freedom (DOF) model of a human arm that was implemented and used as a robot simulator. The motor system was only used to imitate the arm movements of the drummer. The arm model had a 3DOF spherical shoulder joint and a 1DOF revolute elbow joint. The entire simulated robot was described by 8DOF. The output of the inverse models are joint angle velocities, as opposed to forces which are used in real robots. This is because the simulator uses joint angle velocities instead of forces. However, since the direction and speed of the motor commands are specified, it should be trivial to replace the simulated robot with an inverse dynamics controller that would calculate forces for a real-world robot.

The current joint angle velocity is not used as part of the state x_t of the robot, even though this is common in many robot system. This is omitted since the neural networks used to implement the models are recurrent neural networks (i.e. they have memory), which can represent the changes in coordinates internally.

Both the inverse model, forward model and responsibility predictor were implemented using ESNs. The inverse model had 30 input signals; 12 from the current state of the robot in the range $[-1, 1]$ and 18 from the x'_{sound} (trained in the first experiment) in the range $[0, 1]$ coming from the neural drum centre. The inverse model had 8 output signals in the range $[-1, 1]$ to control the degrees of freedom for the robot.

The forward model had 12 input signals from the current state x_t and 8 signals from the inverse model output, u^i_t. The output of the forward model had 12 inputs to predict the next state $\hat{x}'_{t+1}$, in the range $[-1, 1]$.

The RP had 14 input signals, stemming from the sequence ESN of the neural drum centre, all in the range $[0, 1]$. The RP had one output signal in the range $[0, 1]$.

The system was tested with different sizes of the hidden layer; 100, 250, 500, 750 and 1000 nodes. The spectral radius α determining the length of the internal memory (range $[0, 1]$) was held constant at $\alpha = 0.9$ for all configurations. The internal noise level was $v = 0.2$, which effectively adds 10% noise to the internal state of the ESNs. All the network configurations started out with 10 modules. Every second epoch the λ activations were examined, and modules that had been less than 20% active (i.e. $\lambda^i < 0.2$) for less than 10% of the time were removed from the architecture. This check was only done on epochs of even numbers to allow the system to stabilize before pruning it again.

The model used to implement the simulated robot is also used in the PEC, which makes it able to produce very accurate predictions about the consequences of the planned actions. This ensures good training signals for the inverse models as well, which is crucial in a system with such high dimensionality.

The likelihood estimator l^i_t indicates how well the forward model predicted the next state, and expresses this as a scalar. It assumes the presence of Gaussian noise σ. What value should σ be? We found through trial and error that σ should be in the range of 10-15% of the maximum error signal. Since l^i_t follows the Gaussian distribution, an error of σ will be rewarded fairly high: $N(\sigma, \sigma)/N(0, \sigma) = 0.6065$. It is desirable that a prediction error of 10-15% receives a high likelihood of being correct, which is why σ should be 10-15% of the maximum error signal. In the current experiment, the current state x_t has 12 signals in the range $[-1, 1]$. The maximum summed error is 24; a hypothetical situation could be where the predicted state were all -1, whereas the actual state were all 1. In the experiments described in this paper, we used $\sigma = 3$, which is 12.5% of the maximum error. It is important to find the proper σ value; if it is too small only near perfect predictions will receive a high likelihood, yielding the architecture unstable. If it is too large, the likelihood will be high for all predictions of the forward model, rendering it useless for determining the performance of the module.

There were three stopping criteria for the system: 1) p^i could not differ from the final λ value with more than 5%, 2) the system have had to finished pruning modules and 3) the u_{error} had to be less than 50% of the total motor command, forcing the models to control most of the robot. If the experiment did not converge to these criteria within 40 epochs, it was aborted. In order to examine how much the introduction of the PEC aided the learning

of the system, this experiment was also run with the exact same parameters, but using the feedback error controller (Kawato, 1990) instead.

8.3 Experiment 3: Testing How the Modules Self-Organize

In experiment 2, the system self-organizes the control of the simulated robot. Experiment 3 consists of examining to what extent the motor system self-organizes the control of the arm movements. The testing was done the following way: after training, the target states (i.e. x'_{sound}, y and x') were split into regions of 40 timesteps each (corresponding to one bar of musical length). The regions were then randomly permutated ten times. These random permutations were given as input to the motor system. After running each trained instance on ten such random permutations, the performance and λ activations can be compared to experiment 2.

9. Results

The results will be presented in accordance with the different experiments performed.

9.1 Experiment 1

Experiment 1 created a groovy neural network drum machine that learns from human drummers. The system successfully modeled the playing style of five different drummers that participated in the experiment. This was revealed by using the drum machine to play back the same patterns as it had been trained on, comparing the output with the original training sequence. Performing the same statistical analysis on both the original and imitated drum patterns, revealed that the neural drum machine produces drum patterns that are similar but not identical to the teachers. An example is given in figure 6. More details about this experiment are described by Tidemann and Demiris (2008).

9.2 Experiment 2

The results from the training of the motor system (following the training of the sound system) can be seen in table 1. The reader should note that five drummers participated in experiment 1, whereas only one of the trained sound systems was used in experiment 2. This is because experiment 1 dealt with the ability of the sound system to imitate the playing style of different drummers, whereas experiment 2 investigates the properties of a self-organizing motor system. Figure 7 gives an example of self-organization of control of the robot by plotting the RP output as well as the λ vector. The system is stable when the RPs correctly predict how much the module will influence the robot; this can be seen as the overlap between RP output and λ activation. The differences in how many modules were active during the experiment with respect to network configurations can be seen in figure 8. This indicates that the movement C_1 is harder to learn, since more modules are used to collaborate during C_1. Figures 9 and 10 show the performance of the simulated robot with respect to the desired trajectory. Note the close match between actual and desired state. Table 2 shows the same experiment run with the feedback error controller instead of the PEC. Note the improved performance when using the PEC (Table 1) to train the inverse models, compared to the feedback error controller (Table 2).

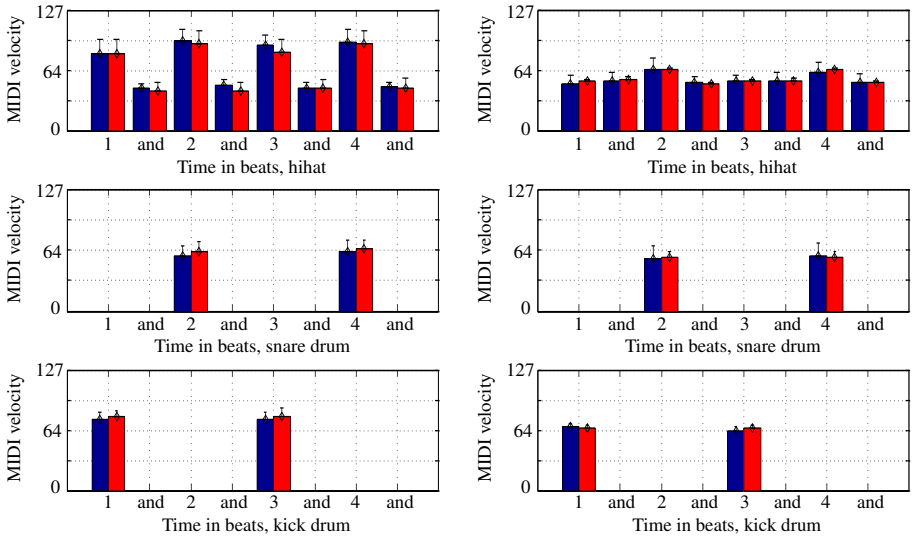


Figure 6: Two drummers and imitations of their playing style is shown, to the left and right. The blue bar represents the training data, the red bar the imitated sequence. The similar bars reveal that the sound system is able to model and imitate the playing style of the drummers. The plots show the velocity profile when playing the same drum pattern as in figure 8.

9.3 Experiment 3

The third experiment was to test the trained networks from experiment 2 on random permutations of the target states. This was done to examine to what extent the modules became experts on certain parts of the trajectory or if it was the sequence of context signals that determined which module was in control. The result is shown in table 3, indicating that modules indeed become experts. This is demonstrated in figure 11, which shows the RP output and λ value of permutation run on a trained instance of the architecture.

10. Discussion

The sound system presented good results regarding its capability to imitate the playing style of human drummers, with a focus on analyzing the motor part of SHEILA (which will be the focus for the remainder of this discussion). For more discussion regarding the sound system, see (Tidemann & Demiris, 2008). The recurrent activation value is the key element to evaluation of the self-organizing properties of SHEILA. In the second experiment (Table 1), the high recurrent activation value indicated that the modules specialize on certain parts of the movement. The recurrent activation value was also high when each of the trained instances were tested on 10 random permutations of the target states. This is further evidence that the modules become specialized on specific segments of the target trajectory, and that it is not the sequence of target states that determine which modules dominate and collaborate upon controlling the robot. Looking closer at when modules are active throughout the sequence to be imitated (Figure 8) the number of active modules tend to *drop* during *variations* of core patterns. Look for instance at C_1V_7 following C_1 just before the 200th timestep; for all the network configurations there is considerably less modules active. This is a tendency that can be observed on most of the variations, indicating that the modules learn the generic motor capabilities of the core patterns, but further specialize on the variations. The notable exception is the SHEILA instance with 100 nodes in the hidden layer. It is also the instance with the highest number of collaborating modules. During some of the variations (e.g. C_1V_4 and C_2V_1), it has the highest number of active modules, going against the trend of fewer modules during variations. This could be due to C_1V_4 and C_2V_1 being extra difficult to learn, requiring cooperation between more modules. Previous work has suggested that increase in complexity requires more neural resources (Tidemann & Öztürk, 2007), which could also explain why there is in general fewer modules active during C_1 compared to C_2. The only difference between C_1 and C_2 is that the right hand of the drummer moves at twice as fast during C_1 compared to C_2. Humans reduce accuracy with increased speed (Kandel et al., 2000), the increase in speed could then require more neural resources because the architecture struggles to stay accurate. The instance of SHEILA with 100 nodes in the hidden layer is also the configuration with the least modules on average (Table 1). It appears than when it manages to find a solution, it needs less neural resources. ESNs depend on a randomly initiated hidden layer. When only 100 nodes are being used in the hidden layer, it is clear that not all randomly created input layers possess the necessary properties to ensure convergence. For larger networks, convergence is not a problem; networks with 500 or more nodes in the hidden layer even converge at about the same number of epochs, with minimal changes in performance error and the number of modules used on average. Two quantities change with increasing network

Nodes	Modules (μ, σ)	Recur. activ. (μ, σ)	Epochs (μ, σ)	Perf. err. (p_e) (μ, σ)	u_{error} ratio (μ, σ)	Conv. exp.
100	4 ; 0	99.58% ; 0.83%	15 ; 0.82	0.0285% ; 0.0036%	49.72% ; 3.3%	20%
250	4.47 ; 0.70	98.57% ; 1.58%	14 ; 2.87	0.0273% ; 0.0037%	48.64% ; 1.05%	95%
500	5.20 ; 0.83	93.74% ; 4.21%	12 ; 1.81	0.0349% ; 0.0103%	47.47% ; 1.88%	100%
750	5.05 ; 0.87	91.39% ; 5.94%	12 ; 1.84	0.0341% ; 0.0104%	45.82% ; 1.88%	100%
1000	5.15 ; 0.81	88.46% ; 6.11%	12 ; 1.63	0.0358% ; 0.0091%	44.91% ; 1.84%	100%

Table 1: Results from the initial learning phase. The column titled "Modules" displays how many modules were used after training (recall that all experiments started with 10 modules). "Recur. act." is short for "Recurrent activation", which shows to which degree the modules had the same λ activation during recurrent context signals. It was calculated in 3 steps: 1) For recurring context signals, the mean and standard deviation (μ, σ) of λ was calculated for all modules. The mean and standard deviation was *not* calculated for the whole sequence of recurring context signals; it was calculated for *each* of the segments that had a recurring context signal. 2) If the *standard deviation* was larger than 5% during a recurrent context signal, it was counted as *not* being recurrent, focusing the result on λ values that were stable throughout the recurring context signal. 3) Among the remaining segments with a standard deviations less than 5% of the λ values, the median was calculated. The remaining segments were examined again, and only those where the mean differed less than 1% were counted as being recurrent. The filtering in step 2 excludes segments with variations, and in step 3 only those that had very similar mean λ activations were counted. In other words, the criteria for being counted as recurrent was very strict. A high recurrent activation value is an indication that the modules specialize on certain parts of the movement, since the modules have the same activation levels for similar context signals. "Perf. err." is an abbreviation of "Performance error" p_e telling how much the imitated trajectory differed from the target trajectory. A low p_e is desirable. The u_{error} ratio shows to what extent the PEC influenced the final motor command after training. Ideally, this ratio would go towards zero as the modules learn more. However, the length and complexity of the current experiment requires the online correction facility of the PEC. This leads to robustness of the system as well. The results show that the modules control most (i.e. more than 50%) of the motor commands sent to the robot. The column named "Conv. exp." is short for "Converged experiments", showing the percentage of converged experiments (experiments were aborted if they had not converged within 40 epochs).

Nodes	Modules (μ, σ)	Recur. activ. (μ, σ)	Epochs (μ, σ)	Perf. err. (p_e) (μ, σ)	u_{error} ratio (μ, σ)	Conv. exp.
100	4.50 ; 1.05	99.77% ; 0.57%	17 ; 9.54	22.81% ; 2.22%	49.81% ; 0.13%	30%
250	4.40 ; 1.52	98.57% ; 2.18%	23 ; 12.16	20.88% ; 3.06%	49.64% ; 0.39%	25%
500	4.67 ; 0.71	97.29% ; 3.20%	20 ; 9.88	20.69% ; 2.81%	49.57% ; 0.31%	45%
750	4.75 ; 0.89	89.36% ; 10.66%	17 ; 8.77	21.54% ; 3.21%	49.61% ; 0.32%	40%
1000	4.67 ; 0.58	86.93% ; 3.33%	14 ; 2.08	22.85% ; 2.40%	49.81% ; 0.19%	15%

Table 2: The same columns as in table 1, this shows how the system performs when using the feedback error controller instead of the PEC introduced in this paper. Particularly, the performance error is higher than when using the PEC, and also there are fewer converged experiments.

Nodes	Modules (μ, σ)	Recur. activ. (μ, σ)	Perf. err. (p_e) (μ, σ)	u_{error} ratio (μ, σ)
100	4 ; 0	99.28% ; 0.92%	0.0308% ; 0.0041%	50.13% ; 3.6%
250	4.47 ; 0.70	95.96% ; 3.33%	0.0293% ; 0.0056%	48.94% ; 1.0%
500	5.20 ; 0.81	89.79% ; 4.64%	0.0366% ; 0.0084%	48.0% ; 1.85%
750	5.05 ; 0.87	87.36% ; 7.95%	0.0348% ; 0.0094%	46.28% ; 1.72%
1000	5.15 ; 0.79	83.57% ; 6.74%	0.0384% ; 0.0154%	45.45% ; 1.80%

Table 3: Results from the random permutations of the target states (i.e. x'_{sound}, y and x'). The results show a somewhat worse performance compared to table 1, but the difference is small. This indicates that the modules do become specialists on specific parts of the movement.

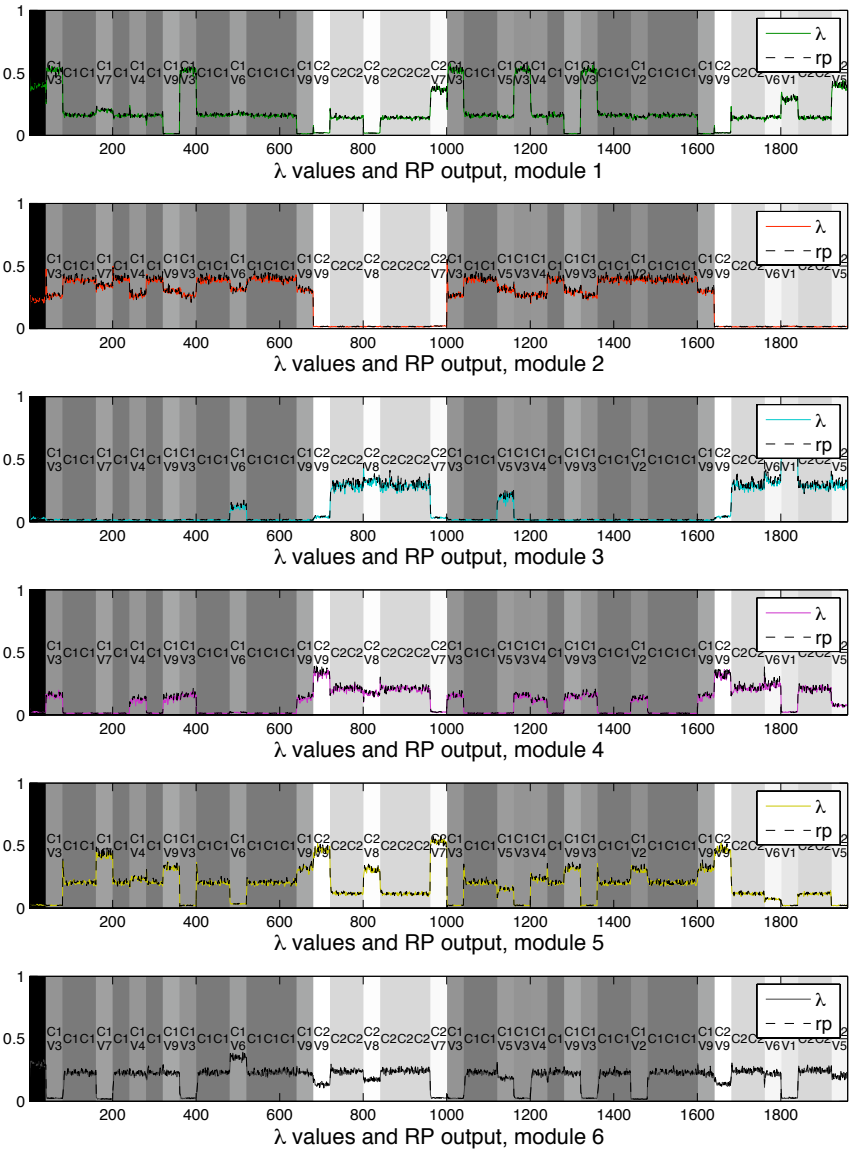


Figure 7: This shows λ and RP output from an experiment with 500 nodes in the hidden layer. The background shows the context signal in terms of core patterns and variations. The all-black segment to the far left signifies the count-in (one bar). The background allows visual inspection of recurrent context signals and the corresponding λ activations for the modules. The close match between RP output and λ values indicate stability in the system, since the RPs correctly predicted how much the module would influence the robot.

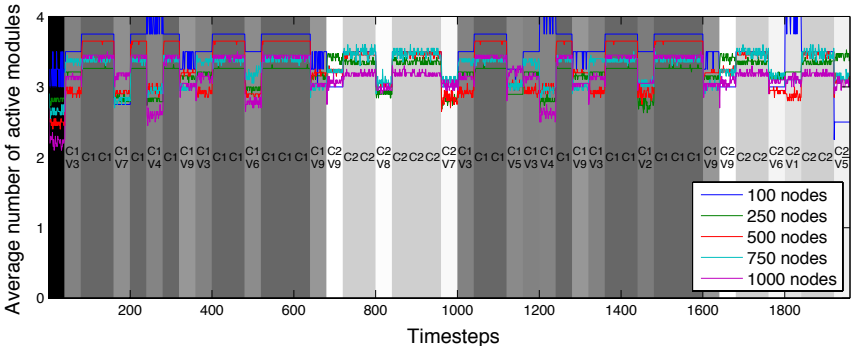


Figure 8: The plot shows the average number of active modules for each of the motor experiments with different number of nodes in the ESNs of the motor system. A module was considered active if $\lambda_t > 0.1$, i.e. its activation was bigger than 10%. Note how the number of active modules during variations of core patterns tends to go down, indicating that the modules specialize on variations (e.g. see the transition from C_1 to C_1V_7 just before timestep 200).

sizes; the recurrent activity value *decreases* as well as the u_{error} ratio. The decrease in recurrent activity might indicate that as neural resources increase, the modules do not have to specialize to the same extent as when neural resources are scarce, i.e. there could be more redundant coding between the modules. This coincides with the number of modules used for instances of SHEILA with only 100 nodes in the hidden layer; it uses fewer modules with lesser nodes. It seems as though the smallest networks may capture more general parts of the motor control space, since the u_{error} ratio is larger for the smaller networks than the bigger ones. It is still not entirely clear what is learned where in the architecture. Figures 7 and 11 show how the modules compete and collaborate when controlling the robot. For cases where one module dominate the entire output this is not an issue, but for areas with shared control this is less clear. Since the inverse models output motor commands in a system with infinite many ways to achieve a desired state, it is not easy to visualize what each inverse model does. The drawback of using a self-organizing system is that it is not always easy to understand *why* it happens the way it is observed. The rules are laid down by the designers of the system, but exactly how the system searches out its solution in the sensory-motor space needs further study.

11. Conclusion

This paper has presented an architecture for motor control and imitative learning, applied to the drumming domain. The experiments in this paper have revealed how the modules self-organize into specialists for parts of the movement to be imitated. Trained instances were tested on random permutations of the target states, demonstrating that the self-organization is consistent even when the sequence of target state changes. This paper has focused on drumming as an imitative task, but the neural drum centre can be exchanged

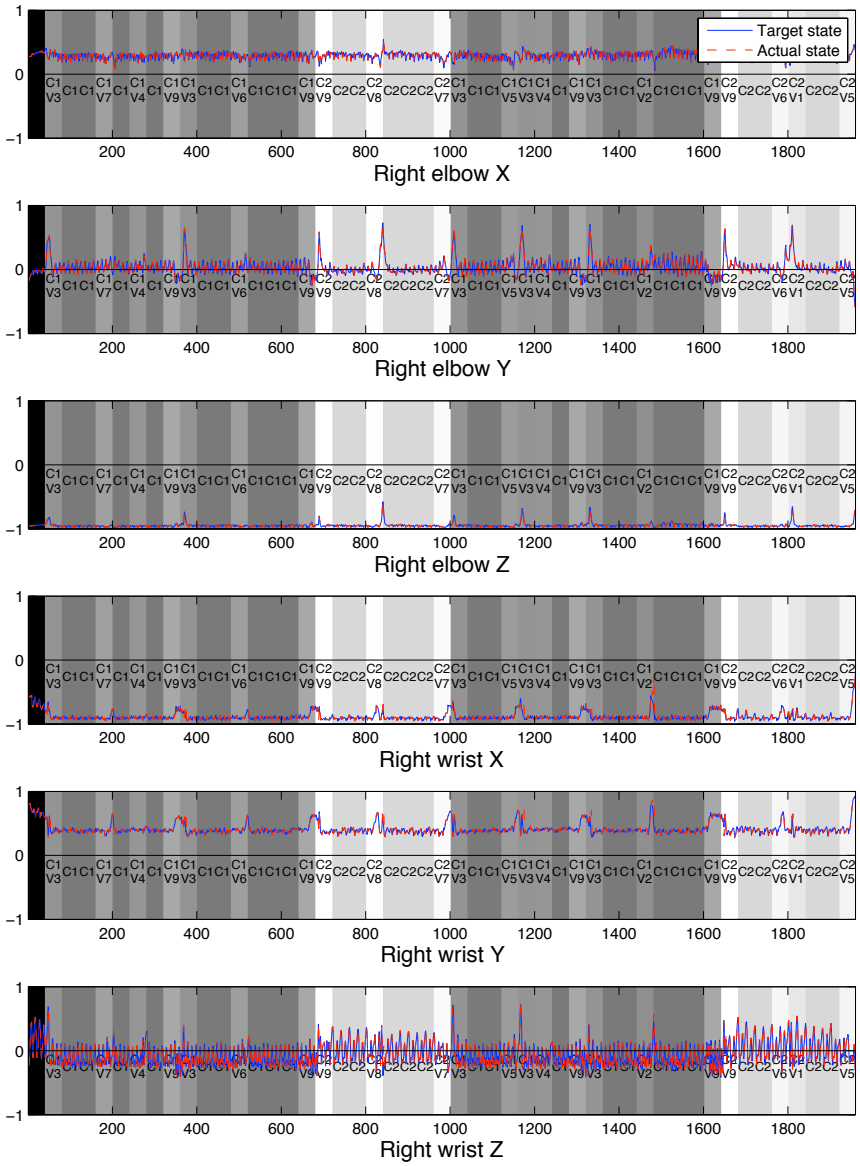


Figure 9: The red and blue lines show how well the system performs, blue line being the desired trajectory and the red dashed line the actual trajectory produced by the system. Same experiment as in figure 7. This plots the right arm, see figure 10 for the left arm. Note the low error rate, a performance error of around 0.03% is typical for the experiments, see table 1.

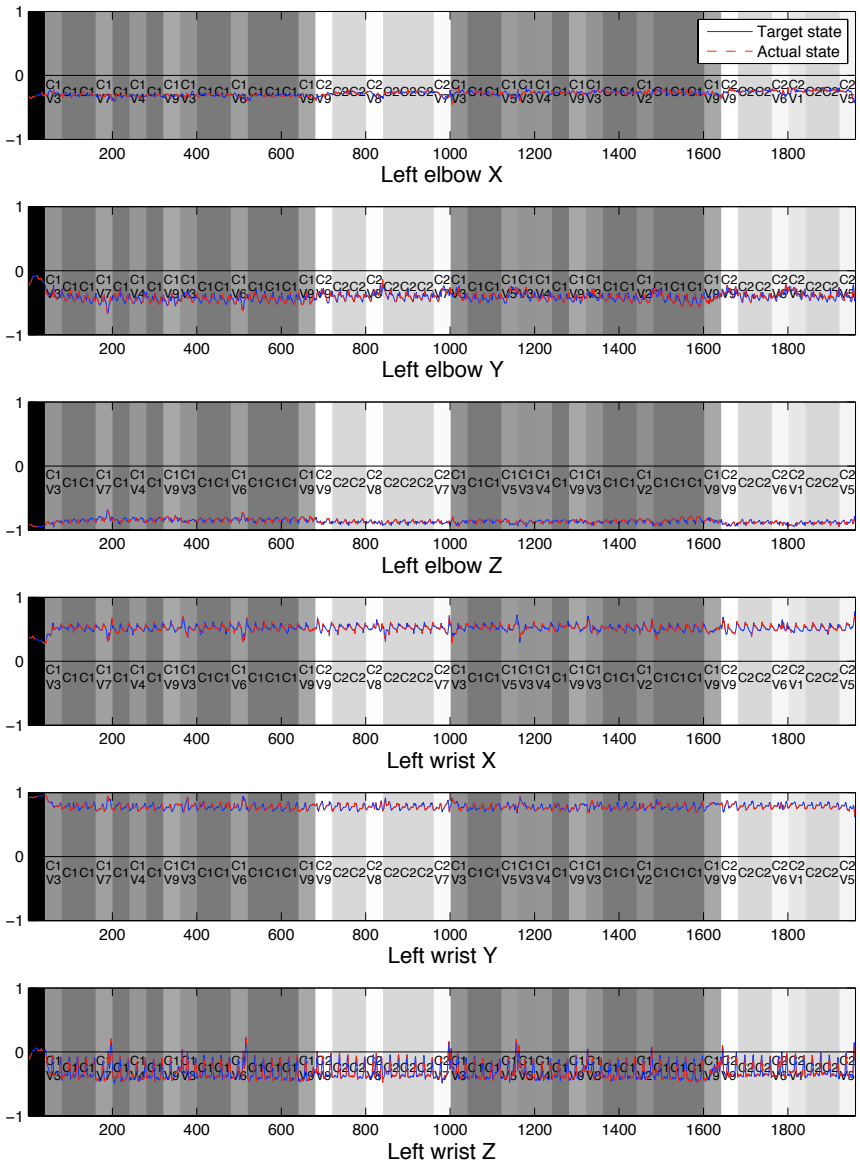


Figure 10: Same as in figure 9, but for the right arm of the simulated robot. The similar close match between actual and desired state is observed.

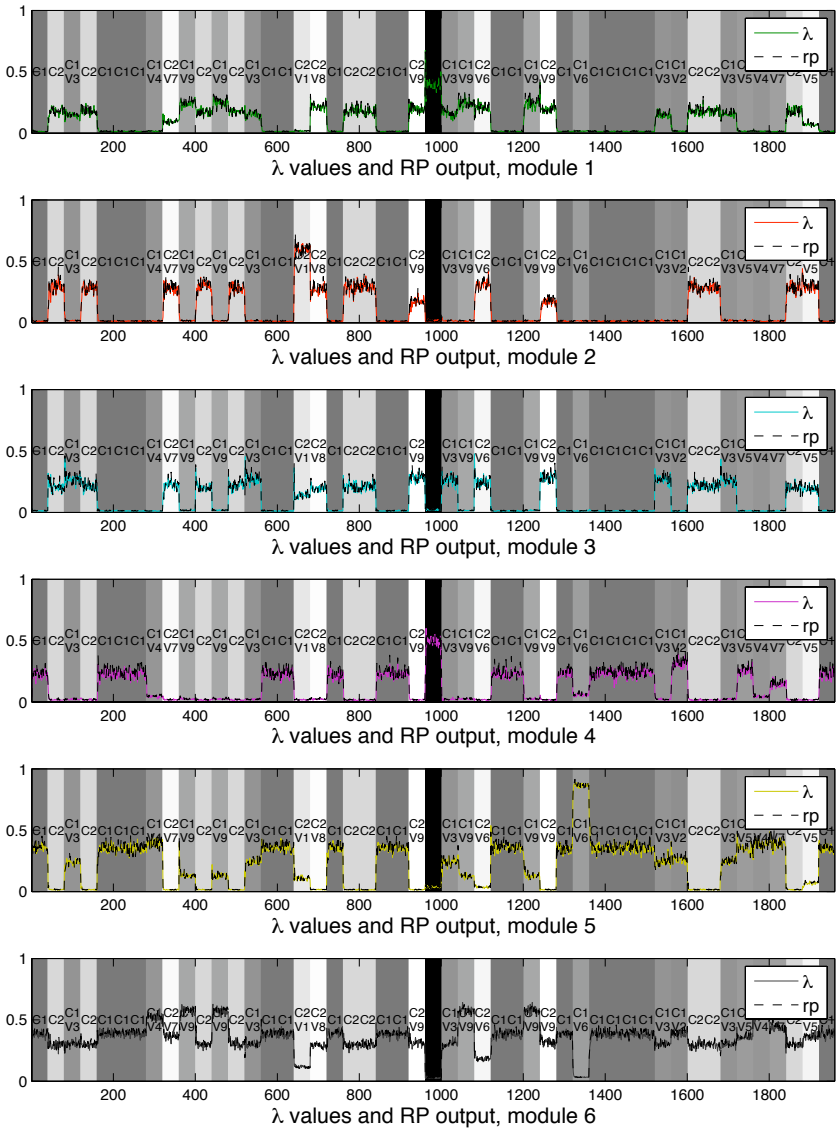


Figure 11: A random permutation of the target states, 1000 nodes in the hidden layer. See how the λ values are recurring in accordance with the repeating context signals.

with anything else that provides high-level target states for the motor system to perform. We wish to better understand the self-organizing process, also with respect to properties of the sensory stream, i.e. to see if there is some correlation with the input/output space and its inherent complexity. This could be an aid for setting some of the variables of the architecture in future applications, such as number of modules, network size and α value.

References

Ans, B., Rousset, S., French, R. M., & Musca, S. (2002). Preventing catastrophic interference in multiple-sequence learning using coupled reverberating networks. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, pp. 71–76.

Arbib, M. (2002). *Imitation in animals and artifacts*, chap. The Mirror System, Imitation, and the Evolution of Language, pp. 229–280. MIT Press, Cambridge.

Barsalou, L., Barbey, A., Simmons, W., & Santos, A. (2003). Social embodiment. In Ross, B. H. (Ed.), *The psychology of learning and motovation*, pp. 43–92, San Diego. Academic Press.

Benghi, D., Addessi, A.-R., & Pachet, F. (2008). Teaching to improvise with the continuator. In *Proceedings of the 28th International Society for Music Education World Conference*, Bologna, Italy.

Billard, A., & Hayes, G. (1999). DRAMA, a connectionist architecture for control and learning in autonomous robots. *Adaptive Behavior*, *7*(1), 35–63.

Blakemore, S. J., Wolpert, D., & Frith, C. (2000). Why can't you tickle yourself?. *Neuroreport*, *11*(11).

Blakemore, S. J., Goodbody, S. J., & Wolpert, D. M. (1998). Predicting the consequences of our own actions: The role of sensorimotor context estimation. *Journal of Neuroscience*, *18*, 7511–7518.

Blakemore, S.-J., & Sirigu, A. (2003). Action prediction in the cerebellum and in the parietal cortex. *Experimental Brain Research*, *153*(2), 239–245.

Cangelosi, A., & Riga, T. (2006). An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots. *Cognitive Science*, *30*(4), 673–689.

Dapretto, M., Davies, M. S., Pfeifer, J. H., Scott, A. A., Sigman, M., Bookheimer, S. Y., & Iacoboni, M. (2005). Understanding emotions in others: mirror neuron dysfunction in children with autism spectrum disorders. *Nature Neuroscience*, *9*, 28–30.

de Mantaras, R. L., & Arcos, J. L. (2002). AI and music from composition to expressive performance. *AI Mag.*, *23*(3), 43–57.

Decety, J. (1996). Do imagined and executed actions share the same neural substrate?. *Cognitive Brain Research*, *3*, 87–93.

Decety, J., Jeannerod, M., & Prablanc, C. (1989). The timing of mentally represented actions. *Behavioural Brain Research*, *34*, 35–42.

Demiris, Y., & Hayes, G. (2002). *Imitation in animals and artifacts*, chap. Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model, pp. 327–361. MIT Press, Cambridge.

Demiris, Y., & Khadhouri, B. (2006). Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, *54*, 361–369.

Dinstein, I., Gardner, J., Jazayeri, M., & Heeger, D. (2008). Executed and observed movements have different distributed representations in human aIPS. *Journal of Neuroscience*, *28*(44), 11231–11239.

Dinstein, I., Hasson, U., Rubin, N., & Heeger, D. (2007). Brain areas selective for both observed and executed movements. *Journal of neurophysiology*, *98*(3), 1415–1427.

Dinstein, I., Thomas, C., Behrmann, M., & Heeger, D. (2008). A mirror up to nature. *Current Biology*, *18*(1), 13–18.

Ehrsson, H. H., Geyer, S., & Naito, E. (2003). Imagery of voluntary movement of fingers, toes, and tongue activates corresponding body-part-specific motor representations. *Journal of Neurophysiology*, *90*(5), 3304–3316.

Farrar, D., & Zipser, D. (1999). Neural network models of bilateral coordination.. *Biol Cybern*, *80*(3), 215–25.

Flanagan, R. J., Vetter, P., Johansson, R. S., & Wolpert, D. M. (2003). Prediction precedes control in motor learning. *Current Biology*, *13*(2), 146–150.

Frak, V., Paulignan, Y., & Jeannerod, M. (2001). Orientation of the opposition axis in mentally simulated grasping. *Experimental brain researc*, *136*(1), 120–127.

Gallese, V., & Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, *2*(12).

Gaussier, P., Moga, S., Banquet, J. P., & Quoy, M. (1998). From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. *Applied Artificial Intelligence*, *1*(7), 701–727.

Grachten, M., Arcos, J., & de Mantaras, R. (2006). A case based approach to expressivity-aware tempo transformation. *Machine Learning*, *65*(2), 411–437.

Grush, R. (2004). The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and Brain Sciences*, *27*(03), 377–396.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Comp.*, *13*(10), 2201–2220.

Hesslow, G. (2002). Thinking as simulation of be-haviour: an associationist view of cognitive function. http://www.mphy.lu.se/avd/nf/hesslow/philosophy/ShortSimulation.htm. Retrieved June 2009 from http.

Imamizu, H., Kuroda, T., Yoshioka, T., & Kawato, M. (2004). Functional magnetic resonance imaging examination of two modular architectures for switching multiple internal models. *Journal of Neuroscience*, *24*(5), 1173–1181.

Ito, M. (1970). Neurophysiological aspects of the cerebellar motor control system. *Int J Neurol*, pp. 162–176.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87.

Jaeger, H., & Haas, H. (2004). Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, *304*(5667), 78–80.

Jeannerod, M. (1994). The representing brain. neural correlates of motor intention and imagery. *Behavior and Brain Sciences*, *17*, 187–245.

Jeannerod, M. (2001). Neural simulation of action: A unifying mechanism for motor cognition. *Neuroimage*, *14*(1), S103–S109.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, *16*, 307–354.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (2000). *Principles of neural science*. McGraw-Hill, New York.

Kawato, M. (1990). Feedback-error-learning neural network for supervised motor learning. In Eckmiller, R. (Ed.), *Advanced neural computers*, pp. 365–372.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, *9*, 718–727.

Kohler, E., Keysers, C., Umilta, M., Fogassi, L., Gallese, V., & Rizzolatti, G. (2002). Hearing sounds, understanding actions: action representation in mirror neurons..

Lingnau, A., Gesierich, B., & Caramazza, A. (2009). Asymmetric fMRI adaptation reveals no evidence for mirror neurons in humans. *Proceedings of the National Academy of Sciences*, *106*(24), 9925–9930.

Matarić, M. J. (2002). *Imitation in animals and artifacts*, chap. Sensory-Motor Primitives as a Basis for Learning by Imitation: Linking Perception to Action and Biology to Robotics, pp. 392–422. MIT Press, Cambridge.

Meltzoff, A. N., & Moore, M. K. (1977). Imitation of facial and manual gestures by human neonates. *Science*, *198*, 75–78.

Meltzoff, A. N., & Moore, M. K. (1997). Explaining facial imitation: A theoretical model. *Early Development and Parenting*, *6*, 179–192.

Miall, R. C. (2003). Connecting mirror neurons and forward models.. *Neuroreport*, *14*(17), 2135–2137.

Miall, R. C., & King, D. (2008). State estimation in the cerebellum.. *Cerebellum*, *7*(4), 572–576.

Miall, R. C., & Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural Networks*, *9*(8), 1265–1279.

Nehaniv, C. L., & Dautenhahn, K. (2002). *Imitation in Animals and Artifacts*, chap. The Correspondence Problem, pp. 41–63. MIT Press, Cambridge.

Oberman, L., Hubbard, E., McCleery, J., Altschuler, E., Ramachandran, V., & Pineda, J. (2005). EEG evidence for mirror neuron dysfunction in autism spectrum disorders. *Cognitive Brain Research*, *24*(2), 190–198.

Pachet, F. (2002). Interacting with a musical learning system: The continuator. In *IC-MAI '02: Proceedings of the Second International Conference on Music and Artificial Intelligence*, pp. 119–132, London, UK. Springer-Verlag.

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *International conference on robotics and automation (icra2009)*.

Pfeifer, R., & Scheier, C. (2001). *Understanding Intelligence*. MIT Press, Cambridge, MA, USA. Illustrator-Isabelle Follath.

Piaget, J. (1962). *Play, dreams and imitation in childhood*. W. W. Norton, New York.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., & Thrun, S. (2003). Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, *42*(3-4), 271–281.

Raphael, C. (2003). Orchestra in a box: A system for real-time musical accompaniment. In *IJCAI workshop program APP-5*, pp. 5–10.

Rizzolatti, G., Fadiga, L., Matelli, M., Bettinardi, V., Paulesu, E., Perani, D., & Fazio, F. (1996a). Localization of grasp representations in humans by PET: 1. observation versus execution.. *Experimental Brain Research*, *111*(2), 246–252.

Rizzolatti, G., Fadiga, L., Gallese, V., & Fogassi, L. (1996b). Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, *3*, 131–141.

Robin, N., Dominique, L., Toussaint, L., Blandin, Y., Guillot, A., & Le Her, M. (2007). Effects of motor imagery training on returning serve accuracy in tennis: the role of imagery ability. *International Journal of Sport and Exercise Psychology*, *2*, 177–188.

Rumiati, R. I., Weiss, P. H., Tessari, A., Assmus, A., Zilles, K., Herzog, H., & Fink, G. R. (2005). Common and differential neural mechanisms supporting imitation of meaningful and meaningless actions. *J. Cognitive Neuroscience*, *17*(9), 1420–1431.

Saunders, C., Hardoon, D., Shawe-Taylor, J., & Widmer, G. (2008). Using string kernels to identify famous performers from their playing style. *Intelligent Data Analysis*, *12*(4), 425–440.

Schaal, S. (1999). Is imitation learning the route to humanoid robots?. *Trends in Cognitive Sciences*, *3*(6), 233–242.

Stippich, C., Ochmann, H., & Sartor, K. (2002). Somatotopic mapping of the human primary sensorimotor cortex during motor imagery and motor execution by functional magnetic resonance imaging. *Neuroscience letters*, *331*(1), 50–54.

Tani, J., Ito, M., & Sugita, Y. (2004). Self-organization of distributedly represented multiple behavior schemata in a mirror system: Reviews of robot experiments using RNNPB. *Neural Networks*, *17*, 1273–1289.

Tidemann, A. (2008). Using multiple models to imitate the YMCA. In *Agent and Multi-Agent Systems: Technologies and Applications*, Vol. 4953 of *LNAI*, pp. 783–792. Springer.

Tidemann, A., & Demiris, Y. (2008). Groovy neural networks. In *18th European Conference on Artificial Intelligence*, Vol. 178, pp. 271–275. IOS press.

Tidemann, A., & Öztürk, P. (2007). Self-organizing multiple models for imitation: Teaching a robot to dance the YMCA. In *IEA/AIE*, Vol. 4570 of *Lecture Notes in Computer Science*, pp. 291–302. Springer.

Tobudic, A., & Widmer, G. (2005). Learning to play like the great pianists.. In Kaelbling, L. P., & Saffiotti, A. (Eds.), *IJCAI*, pp. 871–876. Professional Book Center.

Tolani, D., & Badler, N. I. (1996). Real-time inverse kinematics of the human arm. *Presence*, *5*(4), 393–401.

Torres, E. B., & Zipser, D. (2004). Simultaneous control of hand displacements and rotations in orientation-matching experiments. *J Appl Physiol*, *96*(5), 1978–1987.

Wada, K., Shibata, T., Saito, T., & Tanie, K. (2004). Effects of robot-assisted activity for elderly people and nurses at a day service center. *Proceedings of the IEEE*, *92*(11), 1780–1788.

Williams, J., Whiten, A., Suddendorf, T., & Perrett, D. (2001). Imitation, mirror neurons and autism. *Neuroscience and Biobehavioral Reviews*, *25*(4), 287–295.

Wolpert, D. M., & Flanagan, J. R. (2001). Motor prediction.. *Current biology*, *11*(18).

Wolpert, D. M., Doya, K., & Kawato, M. (2003). A unifying computational framework for motor control and social interaction. *Philosophical Transactions: Biological Sciences*, *358*(1431), 593–602.

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*, 1317–1329.

Wolpert, D. M., Miall, R. C., & Kawato, M. (1998). Internal models in the cerebellum. *Trends in Cognitive Sciences*, *2*(9).

Zimmermann-Schlatter, A., Schuster, C., Puhan, M. A., Siekierka, E., & Steurer, J. (2008). Efficacy of motor imagery in post-stroke rehabilitation: a systematic review. *Journal of NeuroEngineering and Rehabilitation*, *5*(8), 8–18.

ADDITIONAL PUBLICATIONS

G

This chapter lists additional publications relevant to the thesis, but which are not included in the main text for two reasons: 1) portions of their content are available in the main papers, and 2) some of these papers demonstrate pilot studies that are elaborated upon in the main papers.

G.1 Imitating the Groove: Making Drum Machines More Human

Authors:
Axel Tidemann and Yiannis Demiris

Abstract:
Current music production software allows rapid programming of drum patterns, but programmed patterns often lack the *groove* that a human drummer will provide, both in terms of being rhythmically too rigid and having no variation for longer periods of time. We have implemented an artificial software drummer that learns drum patterns by extracting user specific variations played by a human drummer. The artificial drummer then builds up a library of patterns it can use in different musical contexts. The artificial drummer models the groove and the variations of the human drummer, enhancing the realism of the produced patterns.

Main Result:
This was the pilot study of SHEILA, basically a more primitive version of the system presented in paper C.

Published in:
AISB symposium on imitation in animals and artifacts, (2007) 232-240

My main contributions to the paper:

- Coming up with the idea of the groovy drum machine that models human drumming
- Programming the system and running the experiments
- Gathering data and analysis
- Writing the paper

The co-author contributed to the following areas:

- Pointing out references
- Writing the paper
- Supervision of the research

G.2 Using Multiple Models to Imitate the YMCA

Author:
Axel Tidemann

Abstract:
Learning by imitation enables people to program robots simply by showing them what to do, instead of having to specify the motor commands of the robot. To achieve imitative behaviour in a simulated robot, a modular connectionist architecture for motor learning and control was implemented. The architecture was used to imitate human dance movements. The architecture self-organizes the decomposition of the movement to be imitated across different modules. The results show that the decomposition of the movement tends to be both competitive (i.e. one module dominates the others for a part of the movement) and collaborative (i.e. modules cooperate in controlling the robot).

Main Result:
This was in incremental improvement of the system presented in paper A: instead of joint angles, coordinates was used as input to the robot. This makes the input/output vectors of the neural networks much bigger. The relationships are not only harder to learn because the increase in size, but also because the input/output of the inverse models are now in different coordinate systems; the input is in cartesian coordinates, whereas the output is motor commands (i.e. joint angle velocities). In paper A the input consisted of joint angles, and the output was joint angle velocities, which is simply the derivative. Furthermore, this was the first time Echo State Networks were used to implement the different models.

Published in:
Agent and Multi-Agent Systems: Technologies and Applications, volume 4953 of *Lecture Notes in Computer Science*, Springer (2008) 783-792 (Best Student Paper Award)

G.3 Learning Dance Movements by Imitation: A Multiple Model Approach

Authors:
Axel Tidemann and Pinar Öztürk

Abstract:
Imitation learning is an intuitive and easy way of programming robots. Instead of specifying motor commands, you simply show the robot what to do. This paper presents a modular connectionist architecture that enables imitation learning in a simulated robot. The robot imitates human dance movements, and the architecture self-organizes the decomposition of movements into submovements, which are controlled by different modules. Modules both dominate and collaborate during control of the robot. Low-level examination of the inverse models (i.e. motor controllers) reveals a recurring pattern of neural activity during repetition of movements, indicating that the modules successfully capture specific parts of the trajectory to be imitated.

Main Result:
This is mostly an earlier version of paper B. This is the first time repetition of movement is analyzed with respect to the specializing of modules.

Published in:
31st Annual German Conference on AI, Volume 5243 of *Lecture Notes in Computer Science*, Springer (2008) 380-388

My main contributions to the paper:

- Programming the system and running the experiments
- Data gathering and analysis
- Writing the paper

The co-author contributed to the following areas:

- The idea of repeating the movement to see how it would affect the self-organization of modules
- Writing the paper, mostly the discussion

Part III

Postscript

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

The following bibliography lists references in Part 1 (Research Overview).

BIBLIOGRAPHY

Michael Arbib. *Imitation in animals and artifacts*, chapter The Mirror System, Imitation, and the Evolution of Language, pages 229–280. MIT Press, Cambridge, 2002.

J. L. Arcos, D. Canamero, and R. Lopez de Mantaras. Affect-driven cbr to generate expressive music via imitation of human performances. In *ICCBR'99: 3d International Conference on Case-Based Reasoning*, pages 1–12, 1999.

Baum, Leonard E., Petrie, Ted, Soules, George, and Weiss, Norman. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, feb 1970. ISSN 0003-4851.

R.D. Beer, R.D. Quinn, H.J. Chiel, and R.E. Ritzmann. Biologically inspired approaches to robotics: what can we learn from insects? *Communications of the ACM*, 40(3):30–38, 1997.

Aude Billard and Gillian Hayes. DRAMA, a connectionist architecture for control and learning in autonomous robots. *Adaptive Behavior*, 7(1):35–63, 1999.

Rodney Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA. ISBN 1-55860-160-0.

Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal of Robotics and Automation*, 2(1):14–23, 1986.

Joanna J. Bryson. The behavior-oriented design of modular agent intelligence. In *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, pages 61–76. Springer, 2003.

Angelo Cangelosi and Thomas Riga. An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots. *Cognitive Science*, 30(4):673–689, 2006.

Andy Clark. *Mindware: an introduction to the philosophy of cognitive sciences*. Oxford University Press, New York, 2001. ISBN 0-19-513856-2 (ib.), 0-19-513857-0 (h.).

C. Crick, M. Munz, and B. Scassellati. Synchronization in social tasks: Robotic drumming. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 97–102, 2006.

Mirella Dapretto, Mari S. Davies, Jennifer H. Pfeifer, Ashley A. Scott, Marian Sigman, Susan Y. Bookheimer, and Marco Iacoboni. Understanding emotions in others: mirror neuron dysfunction in children with autism spectrum disorders. *Nature Neuroscience*, 9:28–30, 2005.

Ramon Lopez de Mantaras and Josep Lluis Arcos. AI and music from composition to expressive performance. *AI Mag.*, 23(3):43–57, 2002. ISSN 0738-4602.

S. Degallier, C. P. Santos, L. Righetti, and A. Ijspeert. Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS06)*, 2006.

Yiannis Demiris and Gillian Hayes. *Imitation in animals and artifacts*, chapter Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model, pages 327–361. MIT Press, Cambridge, 2002.

Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54:361–369, 2006.

I. Dinstein, U. Hasson, N. Rubin, and D.J. Heeger. Brain areas selective for both observed and executed movements. *Journal of neurophysiology*, 98(3): 1415–1427, 2007.

I. Dinstein, J.L. Gardner, M. Jazayeri, and D.J. Heeger. Executed and observed movements have different distributed representations in human aIPS. *Journal of Neuroscience*, 28(44):11231–11239, 2008a.

I. Dinstein, C. Thomas, M. Behrmann, and D.J. Heeger. A mirror up to nature. *Current Biology*, 18(1):13–18, 2008b.

Simon Dixon. Analysis of musical content in digital audio. *Computer Graphics and Multimedia: Applications, Problems, and Solutions*, pages 214–235, 2004.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Vittorio Gallese and Alvin Goldman. Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, 2(12), 1998.

P. Gaussier, S. Moga, J. P. Banquet, and M. Quoy. From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. *Applied Artificial Intelligence*, 1(7):701–727, 1998.

M. Grachten, J.L. Arcos, and R.L. de Mantaras. A case based approach to expressivity-aware tempo transformation. *Machine Learning*, 65(2):411–437, 2006.

S.T. Grafton, M. A. Arbib, L. Fadiga, and G. Rizzolatti. Localization of grasp representations in humans by positron emission tomography. 2. observation compared with imagination. *Experimental Brain Research*, 112(1):103–111, November 1996.

J. Grezes and J. Decety. Functional anatomy of execution, mental simulation, observation, and verb generation of actions: A meta-analysis. *Human Brain Mapping*, 12(1):1–19, 2001.

Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997. ISBN 0-521-58519-8.

AZ Hajian, DS Sanchez, and RD Howe. Drum roll: increasing bandwidth through passive impedance modulation. In *1997 IEEE International Conference on Robotics and Automation, 1997. Proceedings.*, volume 3, 1997.

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.

Masahiko Haruno, Daniel M. Wolpert, and Mitsuo Kawato. MOSAIC model for sensorimotor learning and control. *Neural Comp.*, 13(10):2201–2220, 2001.

A.K. Hoover, M.P. Rosario, and K.O. Stanley. Scaffolding for interactively evolving novel drum tracks for existing songs. *Lecture Notes in Computer Science*, 4974:412, 2008.

H. Imamizu, T. Kuroda, T. Yoshioka, and M. Kawato. Functional magnetic resonance imaging examination of two modular architectures for switching multiple internal models. *Journal of Neuroscience*, 24(5):1173–1181, 2004.

Eugene M. Izhikevich. Polychronization: Computation with spikes. *Neural Comput.*, 18(2):245–282, 2006. ISSN 0899-7667.

Robert A. Jacobs, Micheal I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

Herbert Jaeger and Harald Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80, 2004. doi: 10.1126/science.1091277.

Michael I. Jordan and David E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell. *Principles of neural science*. McGraw-Hill, New York, 2000. ISBN 0-8385-7701-6 (ib.), 0-07-112000-9 (ib.).

Mitsuo Kawato. Feedback-error-learning neural network for supervised motor learning. In R. Eckmiller, editor, *Advanced neural computers*, pages 365–372, 1990.

E. Kohler, C. Keysers, M.A. Umilta, L. Fogassi, V. Gallese, and G. Rizzolatti. Hearing sounds, understanding actions: action representation in mirror neurons, 2002.

Atsushi Konno, Takaaki Matsumoto, Yu Ishida, Daisuke Sato, and Masaru Uchiyama. *Humanoid Robots: New Developments*, chapter Drum Beating and a Martial Art Bojutsu Performed by a Humanoid Robot. I-Tech Education and Publishing, Vienna, Austria, 2007.

A. Lingnau, B. Gesierich, and A. Caramazza. Asymmetric fMRI adaptation reveals no evidence for mirror neurons in humans. *Proceedings of the National Academy of Sciences*, 106(24):9925–9930, 2009.

Patti Maes. Situated agents can have goals. In Patti Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.

Maja J. Matarić. *Imitation in animals and artifacts*, chapter Sensory-Motor Primitives as a Basis for Learning by Imitation: Linking Perception to Action and Biology to Robotics, pages 392–422. MIT Press, Cambridge, 2002.

Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of artificial neural networks*. MIT Press, Cambridge, Mass., 1997. ISBN 0-262-13328-8 (ib.).

A.N. Meltzoff. Origins of theory of mind, cognition and communication. *Journal of Communication Disorders*, 32(4):251–269, 1999.

Andrew N. Meltzoff and M. Keith Moore. Imitation of facial and manual gestures by human neonates. *Science*, 198:75–78, October 1977.

Andrew N. Meltzoff and M. Keith Moore. Explaining facial imitation: A theoretical model. *Early Development and Parenting*, 6:179–192, 1997.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

Chrystopher L. Nehaniv and Kerstin Dautenhahn. *Imitation in Animals and Artifacts*, chapter The Correspondence Problem, pages 41–63. MIT Press, Cambridge, 2002.

L.M. Oberman, E.M. Hubbard, J.P. McCleery, E.L. Altschuler, V.S. Ramachandran, and J.A. Pineda. EEG evidence for mirror neuron dysfunction in autism spectrum disorders. *Cognitive Brain Research*, 24(2):190–198, 2005.

Francois Pachet. Interacting with a musical learning system: The continuator. In *ICMAI '02: Proceedings of the Second International Conference on Music and Artificial Intelligence*, pages 119–132, London, UK, 2002. Springer-Verlag. ISBN 3-540-44145-X.

Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. MIT Press, Cambridge, MA, USA, 2001. ISBN 026266125X. Illustrator-Isabelle Follath.

Jean Piaget. *Play, dreams and imitation in childhood*. W. W. Norton, New York, 1962. ISBN 0-393-00171-7.

Christopher Raphael. Orchestra in a box: A system for real-time musical accompaniment. In *IJCAI workshop program APP-5*, pages 5–10, 2003.

G. Rizzolatti, L. Fadiga, M. Matelli, V. Bettinardi, E. Paulesu, D. Perani, and F. Fazio. Localization of grasp representations in humans by PET: 1. observation versus execution. *Experimental Brain Research*, 111(2):246–252, September 1996a.

Giacomo Rizzolatti, Luciano Fadiga, Vittorio Gallese, and Leonardo Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive Brain Research*, 3:131–141, 1996b.

F. Rosenblatt. The perceptron, a probabilistic model for information storage and organization in the brain. *Psych. Rev*, 65(6):386–408, 1958.

C. Saunders, D.R. Hardoon, J. Shawe-Taylor, and G. Widmer. Using string kernels to identify famous performers from their playing style. *Intelligent Data Analysis*, 12(4):425–440, 2008.

S. Schaal. Dynamic movement primitives: A framework for motor control in humans and humanoid robotics. In *2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM)*. Springer, 2003.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

L. Steels. Evolving grounded communication for robots. *Trends in cognitive sciences*, 7(7):308–312, 2003.

Jun Tani, Masato Ito, and Yuuya Sugita. Self-organization of distributedly represented multiple behavior schemata in a mirror system: Reviews of robot experiments using RNNPB. *Neural Networks*, 17:1273–1289, 2004.

Asmir Tobudic and Gerhard Widmer. Learning to play like the great pianists. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 871–876. Professional Book Center, 2005. ISBN 0938075934.

Elizabeth B. Torres and David Zipser. Simultaneous control of hand displacements and rotations in orientation-matching experiments. *J Appl Physiol*, 96(5):1978–1987, 2004. doi: 10.1152/japplphysiol.00872.2003.

G. Weinberg and S. Driscoll. Toward robotic musicianship. *Computer Music Journal*, 30(4):28–45, 2006.

G. Weinberg, M. Godfrey, A. Rae, and J. Rhoads. A real-time genetic algorithm in human-robot musical improvisation. *Lecture Notes In Computer Science*, 4969:351–359, 2008.

P.J. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Harvard University, 1974.

JHG Williams, A. Whiten, T. Suddendorf, and DI Perrett. Imitation, mirror neurons and autism. *Neuroscience and Biobehavioral Reviews*, 25(4):287–295, 2001.

M.M. Williamson. *Robot arm control exploiting natural dynamics*. PhD thesis, Citeseer, 1999.

Daniel M. Wolpert, Kenji Doya, and Mitsuo Kawato. A unifying computational framework for motor control and social interaction. *Philosophical Transactions: Biological Sciences*, 358(1431):593–602, 2003.