Effect of Internal Models on Homeokinetic Controled Autonomous Robots

Athanasios Polydoros

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2013

Abstract

This master thesis will present the results of my work into the reanimation of lifeless human tissues.

Acknowledgements

Many thanks to my mummy for the numerous packed lunches; and of course to Igor, my faithful lab assistant.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Athanasios Polydoros*)

Table of Contents

Chapter 1

Introduction

The need for robots that can be used in real world drives researchers towards the development of intelligent and autonomous machines. Such machines should be capable to operate in complex environments that depend to many factors, thus the derivation of control rules that would achieve a desired result is challenging. The more widely used approaches belong to the class of evolutionary techniques (Nolfi and Floreano, 2000) but they have serious drawbacks.

In the introductory chapter, will be presented, the attributes that robots should have in order to be characterized as intelligent and autonomous. Furthermore, will be mentioned, the main reasons that make the evolutionary strategies insufficient for real world applications and how their drawbacks can be overcome by adopting an approach inspired by the process of self-organization.Also, the definition of an internal model, will be presented, its significance at self-organized control and the research made in this field. Finally, will be mentioned, the motivation and the hypothesis of this thesis.

1.1 Intelligent Robotics

Nowadays robots are used in a wide range of applications. Companies that are operating on the fields of automation, space, defence are using robots for accomplishing tasks that are dangerous or hard for people, examples of such tasks are the assembly of heavy machine parts, underwater inspection of oil and gas pipes, diffusion of hazardous explosive devices, space exploration etc.

Those tasks are complex and taking place in extreme dynamically changing environment. This fact arises the need of robots that could cope with environmental changes, be easily adapted in different environments, perform slightly different tasks

each time and operate without human intervention. This functionality can be achieved by developing intelligent autonomous robots.

In robotics, the notion of intelligence can be interpreted as the ability of a robot to perform tasks that cannot be preprogrammed due to the uncertainty for the world state, thus an intelligent robot should be capable to improve its performance based on its experience and also has the ability of planning low-level actions for achieving a goal without being explicitly programmed. Finally, intelligent robots can adapt to environmental changes.

Autonomy is also an important feature of robots. An autonomous robot is not directly controlled by a human operator, and is not attached to an external power supply or an external processing unit. Among the most interesting principals are the self-regulation, self-organization and self-governing. Self-regulated robots can maintain a desired functionality despite the disturbances from their environment. Self-organization makes the robot capable to generate its own ability to perform actions and self-governing corresponds to the creation of own goals. Thus, autonomous robots are able to choose and change their behaviour by their own.

Those principals are part of evolutionary strategies for robot control (Nolfi and Floreano, 2000) the common characteristic of evolutionary strategies is that they are biologically inspired thus the robots are developed based on a process that imitates the physical evolution. An initial population is initialized, the best individuals of this population are selected according to a performance criterion (fitness function), from those individuals the next generation is formed through evolutionary processes such as crossover and mutation. The next generation is also evaluated and the best individual are selected. The process terminates when the performance criterion is met.

The serious drawbacks of the evolutionary strategies are that they are time consuming and computationally demanding tasks (Nolfi and Floreano, 2002) since the performance of every single member of a population should be evaluated according to an evaluation criterion. Furthermore, the selection of this criterion is difficult because of the need to encode the possible states of the environment and also the desired low-level goals in a criterion, which could be impossible for a real, complex environment and task. Finally, at the application of evolutionary algorithm should be considered the limitations that are arising from the architecture of the robots and from the environment, such limitations have to be introduced as mathematical formulas, which could be a very complex task.Thus, modifications of the fitness function are required in case of changing environment or using a robot with different architecture.

1.2 Self-organization

A control scheme based on the principle of self-organization, deals with the described drawbacks of evolutionary strategies. According to Haken (2008) self-organization is defined as:*" The spontaneous often seemingly purposeful formation of spatial, temporal, spatiotemporal structures or functions in systems composed of few or many components"*. Thus, in a system which is initially disordered, through the self-organization process, arises some form of order or coordination between the agents that consist the system. This process is not triggered and does not depend on a single agent.

A crucial characteristic of self-organization is that there is not a central control of the process, on the contrary the control is distributed in the entire system, which corresponds to a decentralized control strategy. Thus all the agents of a self-organized system contribute to the organization despite that they only interact locally. According to Bonabeau et al. (1999) [p. 9-11] the self-organized systems exhibit three basic properties.First,is the existence of dynamical non-linearity, so a stimuli (input from the environment) results to a non-static output which, also, is not proportional to the input. Second, there is a balance between exploration and exploitation, the agents of the system do not only exploit their state but they also exploring other potential states that could result to a better state of the entire system. Finally, a large number of interactions between the agents occurs.

The phenomenon of self-organization takes place in many systems such as physical, chemical and biological systems. An example of self-organization in biology is the formation of a swarm of animals like birds, fishes, wolves. Figure 1.1 illustrates the formation of a swarm of birds, this formation is a result of self-organization.

Figure 1.1: Birds' swarm as result of self-organization

This formation is not controlled by a single individual but occurs based on negatives and positives feedbacks from the agents. Those feedbacks depend on a high-level goal. At this particular example the formation is a result of two opposing goals. The need of birds for immigration and dealing with threats such as predators. Thus if they attempt to fly away from the swarm there is a high risk because they are more vulnerable to predators and the coordination arises because all the individual want to immigrate. In many examples the occurrence of self-organization is the result of two opposing forces. The swarm will exhibit a variety of patterns according to the environmental stimulus such as the wind, the existence of a specific threat, need for food e.t.c. This results to the emergence of new capabilities of the system.

Based on the definition and the provided example of self-organization, could be concluded that using this principal in a single robot, considering as system the robot and its environment as agents the robot's motors and sensors that would sense the environment, should occur a coordination of motors in order to achieve a defined high-level goal. The robot does not know *a priori* how to achieve this goal, so it has to generate sensorimotor coordination, through this coordination, a wide range of behaviours can occur.

Thus, in contrast with evolutionary strategies the computational complexity can be kept low because in self-organization process the change of an agent's state does not affect the whole system and also the limitations that arising from the configuration of the robot and from the environment are not introduced with explicitly because a self-organized controller produce motor commands that fit to the embodiment. The concept of embodiment is crucial, the robot's brain, body and environment have to be treated as unity (Brooks, 1991) because the self-organization process relies on a continuous interaction between the agents and the environment.

Thus, the control scheme of the robot should include two characteristics because of the continuous interaction with the environment and the need for feedback from the agents. Namely a closed loop, through which the controller receives the effect of the motor commands and an internal model, which encodes the belief of the agents about the environment.

1.3 Internal model

The use of internal model in robots is biologically inspired and exist in a large number of biological organisms, it is believed that the central neural system performs trans-

formations between the actions and the sensory states. According to Kawato (1999) *"Internal models are neural mechanisms that can mimic the input/output characteristics, or their inverses, of the motor apparatus"* thus internal model represents the belief of an organism about its environment.There are numerous researches and experiments that prove the existence of internal model in humans (Wolpert et al., 1995; Flanagan and Wing, 1997; Thoroughman and Shadmehr, 2000).

For instance, in Thoroughman and Shadmehr (2000) the existence of an internal model was proved using a simple experiment in which subjects where ask to perform reaching movements at a plane towards a desired plane holding a robotic manipulandum, which produced viscous forces. After a number of trials where the viscous force is applied the subjects reduced the error between desired and actual position, when a catch trial occurs (viscous force is not applied) the error becomes large and has opposite direction compared to the field trials, which denotes the existence of internal model.

Thus, the use of an internal model in the control scheme of a robot could be interpreted as adding consciousness in the robot. An interesting use of internal models is presented in Bongard et al. (2006) where, the authors, motivated by the ability of animals to be functional after the occurrence of an injury, they developed an algorithm that applied in a four-legged starfish robot. It uses relationships between actions and sensations in order to infer its configuration and then it uses a self-model for achieving a desired locomotion. The self-model is based on multiple competitive internal models, thus it is capable to adjust its self-model when a leg-part is removed and create an alternative gait for moving.

A recent survey on internal models that are used for robot control was made by Nguyen-Tuong and Peters (2011). The type of used internal models depends on the prediction problem, so there are three main types of internal models the forward, the inverse and the mixed problem. Figure 1.2 illustrates a general example of a plants' control scheme using a forward internal model as a representation of the plant. Forward models perform a mapping from the control signal to the predicted state of the plant, thus it is a prediction of the state of the process if the control signal emitted by the controller will applied at the plant. The inverse model performs the opposing function, given the state of the plant it reconstructs the control signal that was applied to the plant and had as result the observed state, thus they do not represent a cause-effect relationship but an effect-cause. A problem of those models is that the effect-cause relationship is, in some cases, ill-defined. This happens when the output space is not

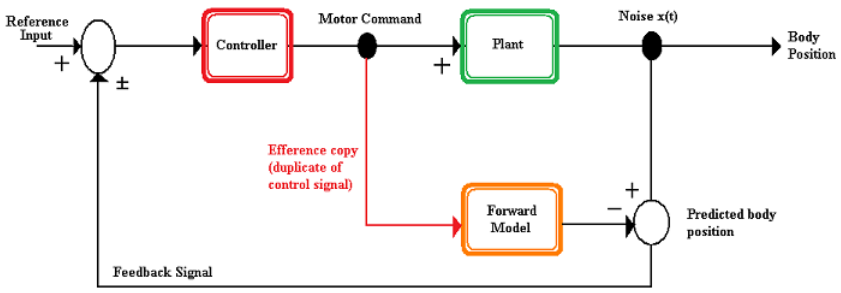

Figure 1.2: Control scheme with forward internal model

convex (Jordan and Rumelhart, 1992). A mixed model is a combination of a forward and an inverse model and were introduced in order to overcome the problems that arising from the use of inverse models.

The forward models are generally used in robots in order to achieve fast movements with high compliance and adaptation to changes. This is achieved by using the closed loop setup. The error between the true and the predicted state of the plant is fedback to the controller and is taken into account for the generation of the next motor command.Thus, the controller smoothly corrects the state of the plant and makes the control system capable to cope with unexpected changes of the plant state.

An interesting attribute of internal models is the efference copy.In biology, it is believed that the signals emitted from the central neural system are copied locally and makes the organisms capable to distinguish between the sensory inputs that were a result of the organism action and those that were a result of environmental stimuli. The effect of the efference copy at the internal models' predictability in robotics was investigated by Schröder-Schetelig et al. (2010) and Manoonpong and Wörgötter (2009)

1.4 Background

In this section will be presented the previous work made on the research field of the self-organized control on autonomous robotics. The introduce of the self-organization principle in machines was made by Neumann (1966) where a machine that has knowledge about its own configuration can assemble itself. This work is based on theoretical concepts and is not concerned about the practical implementation but it was the basis for research in the field of self-organization of physical systems (Murata and Kurokawa, 2012, p .38).

The research is mainly focused on applying the self-organization principle in sys-

tems that consist of several robots (multi-robots systems and swarm robotics). An influencing research on this field was made by Reynolds (1987), who proved that the emergence of a flocking behaviour can be achieved by using simple rules that describe the local interactions between the agents. Also the patterns of the flock change according to attributes such as the noise of the system, the density of the flock and the parameters of the local interactions, also the agents of the systems tend to keep a distance between each other while they perform complex tasks such as obstacle avoidance. Thus, a very complex high-dimensional system can be controlled through the adaptation of a small number of parameters. One drawback can be considered that those findings are result of simulations, the difficulty of applying those principal in real robots rise significantly in real world applications due to the environmental complexity and the robots' configuration.

A research about the emergence of self-organization in a group of physical robots was made by Trianni and Dorigo (2006) the study was focused on the generation of self-organized behaviour for a group of robots. Their movement is coordinated through communication strategies, a variety of such strategies were tested and the findings denote that evolutionary controllers produce a robust self-organized system. Similar researches on swarm robotics were made by Svennebring and Koenig (2004) and Nouyan et al. (2008). The same characteristic at the application of self-organization on robot swarm is that they investigate the behaviour of a group of robots and not of an individual robot which would result the generation of capabilities, instead the individuals are controlled through a decentralized control technique.

Different robot behaviours can emerge by using the homeokinetic controller which was introduced by Der et al. (1999). Based on the fact that the artificial evolution of an individual can be significantly fast if the individual can learn fast its embodiment, the authors used the principle of homeokinesis which, despite the fact that is completely unspecific gives rise to specific behaviours. Also, the importance of an adaptive internal model is underlined. The general goal of this principle is to keep the agent in a kinetic state which can be predicted by the internal model, thus this principle can be considered as the opposite of homeostasis, which is the ability of a system to maintain a stable state. Despite that, in this article, the proposed method is tested on linear and non-linear control tasks only by simulations there is a number of subsequent researches that this principal was tasted in both simulated and real robots.

Particularly in Der (2001) the homeokinetic controller was tested in a real Khepera robot. The aim of the research was to investigate the emergence of situated behaviours

on a robot controlled by the principal of homekinesis. A situated behaviour is defined as the behaviour of an agent who controls its actions by using the maximum sensory information. The experimental results proved the assumption that different behaviours can emerge from the homeokinetic principle using the same internal model and learning parameters, namely the robot was able to avoid obstacle, navigate through corridors and complex mazes and to follow walls.

Der et al. (2004) examined the ability of the homeokinetic controller for sensory integration, to use the sensor inputs that origin from both the agents' body and environmental stimuli in order to use its body effectively within the environment. This is achieved by using simple learning rules for the controller that result to an automatic integration of the sensors after they emit a signal to the controller. The interesting findings of this research was that the sensory integration make the robot capable "survive" in a simple world also the system is not affected by problems that occur when the sensor properties change rapidly. Also the robot adopts an explorative behaviour which, however depends from the environment.

It is clear that the principle of homeokinesis does not specify a goal for the agent. This issue is solved by guiding the behaviours that emerge from the self-organization process towards a desired goal (Martius et al., 2007) according to this approach reinforcement signals are used to shape the behaviour of the robot (negative signals for punishment and positive for reinforcement). The experiments applied this approach for achieving desired spin and speed in different robots. Also the effect of different signal strategies were examined (Martius and Herrmann, 2012). Finally, a summary of the research on goal-free and guided self-organization was made by Der and Martius (2012)

1.5 Motivation

The previous sections illustrated the significance of self-organization to the evolution of intelligent robots. The main characteristic that makes this research field interesting is that machines can produce their own capabilities without being given a specific goal, such capabilities could be too complex to be preprogrammed with explicitly. Through the self-organization process we can mimic the sensorimotor coordination that occurs in real life this results to the emergence of behaviours that depend on the body and the stimuli of the environment. For example, humans are not taught with explicitly how to walk and also they are not given a direct goal to achieve it. The walking behaviour is

result of exploiting the intrinsic capabilities and exploring the environmental stimuli.

The homeokinetic principle that is used in this dissertation is based on the self-organization process that takes place in nature. The research that has been made on its application on robotics exhibits interesting results about the behaviours that emerge in robotics since relatively complex behaviours emerge despite the simple environment and internal model that was used. Furthermore, since this principle is goalless, it could be safe to say that robots are acting based on an intrinsic motivation.

Also, the research does not investigates the impact of different types of internal models on the generated robots' behaviours. As it will be presented in the next chapter, the homeokinetic controller depends on both the inner dynamics of the internal model and its prediction accuracy. Thus, both the complexity and the predictability of the model should have an impact on the behaviours.

Finally, the dependence between the emergence of explorative behaviour and the internal model, will be examined since robots that have an explorative behaviour are used in many real world applications.

1.6 Hypothesis

- The quality of the model affects the generated behaviours

As indicators of model's quality will be used, the mean squared error between the actual and the predicted state and the area of the environment that was covered by the robot. Through this will be examined if better predictors result to inactivity because the robot falls into an uninteresting regime.

- There is a dependence between model and environment complexity

Since more complex predictors perform better at the approximation of complex functions compared to simple predictors, it would be interesting to investigate if complex internal model perform better in complex environments and as result are driving the robot towards an exploratory behaviour.

- Complex models result to complex behaviour

Based on this hypothesis the models that have complex inner relationships can adapt faster to a change of the target function that they predict, contrary to simple models that need much time to adapt in changes. This could result in complex behaviour because the fringe states that may occur would be learned faster from the agent. Despite

that the term of behavioural complexity is fuzzy, a method has been proposed by Ay et al. (2008). N. Ay and colleagues, measured complexity in terms of the predictive information in sensor space.

Dissertation outline

In Chapter 2 will be presented the structure of closed loop control scheme and the main reasons that make this scheme important. Also, will be explained the sensori-motor loop and its significance in self-organized control. Finally, the derivation of the homeokinetic principal will be introduced.

Chapter 3 includes a presentation of the internal models that will be used, the basic reason for selecting those, their possible advantages and disadvantages, the derivation of the learning rules and references to research areas that were successfully applied.

The evaluation procedure that will be used for testing the hypothesis will be presented in Chapter 4. Also, will be mentioned, the setup of the experiments, including the different environments and description of the used robot. Finally, the results of the evaluation process will be illustrated.

In Chapter 4 is contained the outcome of the results, a discussion about how the findings validate the hypothesis and possible drawbacks of this approach. Finally, will presented directions for future work on this research field.

Chapter 2

The Homeokinetic control scheme

The homeokenetic controller is applied on a closed loop scheme. The advantages of the closed loop control, such as the feedback and the representation as a dynamical system enables the derivation of simple learning rules for the controller. Based on those learning rules the robot can exhibit a variety of behaviours that fit its body and environment (embodiment). In this approach there is not any external teacher who will favour a specific kind of behaviour, thus there is not a desired goal. There are two intrinsic high-level goals that have to be achieved. Namely, the robot should be able to active and be able to predict the results of its actions in the environment.

2.1 Open and closed loop control

Figure 2.1 illustrates a general open loop control system. The reference or desired value is fed to the controller, in most control systems there is a process unit which converts the input into a form that can be used from the controller, the controller emits a noise contaminated command signal which is applied to the controlled plant and produces an output.

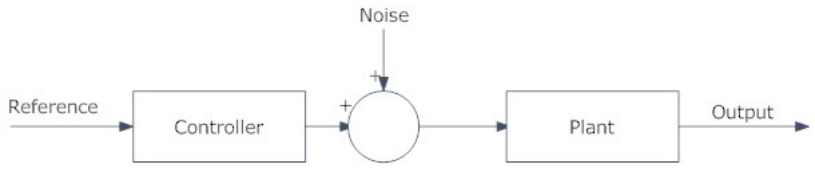

Figure 2.1: Open loop control scheme

The main characteristic of this control scheme is that it cannot take care of any disturbances that adds to the emitted command signal from the controller and any noise

that could affect the output signal. Hence, open loop systems are unable to detect and correct any disturbances that occur and they are only commanded by the input. As result open loop control schemes are not only sensitive to noise but also are unable to detect and correct the disturbances that occur. Despite those drawbacks they are applied in simple processes where the output is controlled by external factors because of their simplicity and low cost.

An example of open loop control is presented by Der and Martius (2012) [p. 39]. This experiment consists of a barrel that is moved by two internal heavy weights that slide on two perpendicular axis. The weights are controlled by a linear motor, thus the by sifting the weights the barrel changes its center of gravity, the weights' movement result to the system's movement. The experiment starts by using a periodic control signal which changes the weights with a phase sift of $\pi/2$ then the frequency is doubled over the time in order to achieve a higher rolling speed. This results to a sacadic motion of the barrel and makes clear that the open loop control is insufficient for generating a smooth fast motion of the barrel.

The drawbacks of the open loop controlled systems can be overcome in closed loop systems. A general architecture of that kind of systems is illustrated in figure 2.2. The reference signal, desired plant state, is often translated into a form that is interpretable from the controller. For instance if the controller uses electrical signals to operate the motors of a robotic arm and the output of the system is the pose of the arm, then the pose have to be converted to electrical signals.

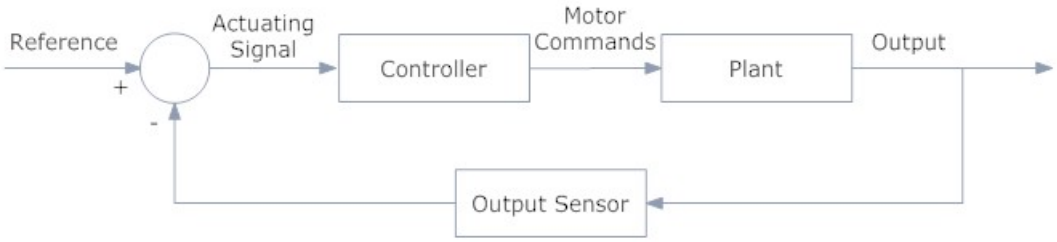

Figure 2.2: Closed loop control scheme

After the translation of the output the feedback signal is subtracted from the reference creating the actuating signal or error. This actuating signal is fed to the controller which generates the necessary motor commands in order to regulate the plant towards the desired output. Thus the closed loop control systems can deal with disturbances by measuring the output of the plant, feeding back the output measurements through a feedback plant and comparing the output with the desired input. If the desired input is

different from the output the controller uses that difference in order to adjust the output, in case that there is no difference between the desired and the current state there is not an actuating signal and the controller does not drive the plant because it is already operates in the desired state.

It is obvious that closed loop systems are more accurate than the open loop systems because they have a limited sensitivity to noise, disturbances and external (environmental changes). On the contrary those systems are more complicated since they need an output sensor and a converter for translating the output signal as described above. Thus, it is also more expensive.

Despite the disadvantages, for taking into account the interaction between the agent and the environment,which is crucial for developing self-organized machines, the use of a closed loop control scheme seems to obligatory. Thus, in this way, the agent is able to adjust its behaviour based on the environmental stimuli. But also, as it is already mentioned in the previous chapter, there is not a specific desired output that is set to the agent extrinsically, instead the goals are result of an intrinsic process. Thus, in the homeokinetic control scheme is used the biologically inspired concept of the sensorimotor loop.

2.2 The sensorimotor loop

The concept of sensorimotor loop signifies the strong dependence between sensory input and motor output in a closed loop cycle. Its importance in biological agents is highlighted in a number of researches (Wiener, 1965; Ferezou et al., 2007; Nguyen and Kleinfeld, 2005) this inspired it application in robotics. The figure 2.3 illustrates the process of a sensorimotor loop in humans.

The approach of homekinesis is based on applying closed loop control in a sensorimotor loop, this makes possible an implementation of closed loop system without an externally defined reference value (desired output of the system). Thus, any change on the controller (motor commands) is driven by the state of the sensors, this fact makes the robot capable to act according to intrinsic parameters instead of acting based on an external goal and the result can be interpreted as emergence of self-organized behaviour.

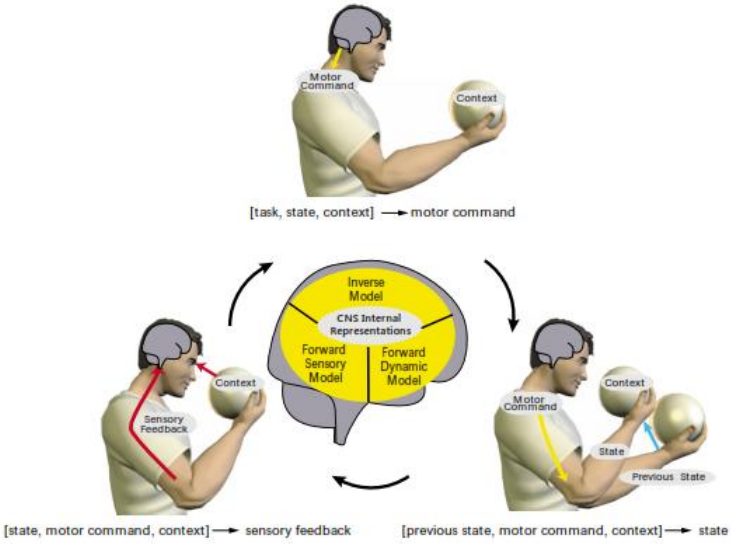

Figure 2.3: The sensorimotor loop in humans (Wolpert and Ghahramani, 2000)

An implementation of the sensorimotor loop is presented in figure 2.4 in a simple case with one input and one output. At every time step $t = 0, 1, 2, 3, \ldots$ the controller K receives the sensor values x_t and generated motor commands y_t the motor commands are applied to the robot R which interacts with the environment and, by this way, the new sensor values x_{t+1} are generated and are fed to the controller as input at time step $t + 1$. Thus $x_t \in I$ and $y_t \in O$ where I and O are the sensors' and motors' space respectively.

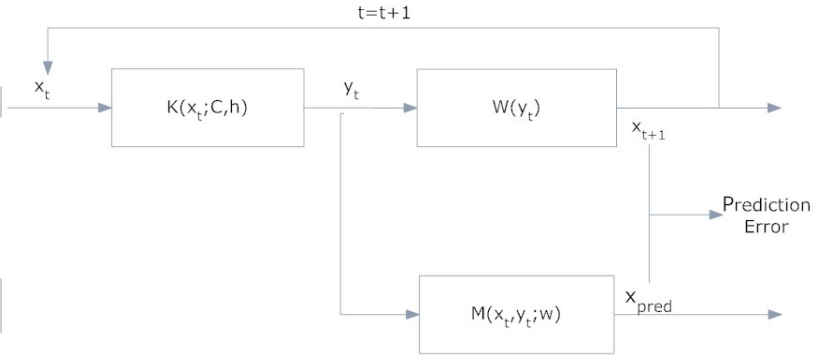

Figure 2.4: Application of sensorimotor loop in robotics

Thus, the controller in the sensorimotor loop performs a mapping $K : I \to O$ according to the following equation

$$y_t = K(x_t; C, h) \tag{2.1}$$

where x_t is the current sensory inputs and C, h are adjustable parameters of the controller. The controller of the homeokinetic control scheme represents a single neuron, thus equation 2.1 can be rewritten as:

$$y_t = K(x_t; C, h) = \tanh(Cx_t + h) \tag{2.2}$$

So the parameters C and h can be interpreted as the the synaptic weights and the bias term respectively. The function tanh is the hyperbolic tangent which belongs to the class of sigmoid functions and is used for keeping the motor commands within the desired interval $y_t \in [-1, 1]$.

Since the motor commands are applied to the robot which is interacting with the environment the sensor's values are changing according to the configuration of the robot. Thus the transformation of the motor commands y_t to the new sensor values x_{t+1} can be represented by a function $W(y_t)$ which performs a mapping from motor's space to sensor's space $W : O \to I$ and the sensor value is defined as

$$x_{t+1} = W(y_t) + \xi_t \tag{2.3}$$

Where ξ_t are disturbances such as noise. The function W represents the world, and assuming that the processes taking place in the world are unknown we need to approximate them using an internal model which is implemented by the function $M(x_t, y_t; w)$. Thus $M(x_t, y_t; w)$ performs, also, a mapping from the motor commands to the sensor's space. By adjusting the model's parameters w based on the prediction error, the difference between the predicted sensor values and the the true sensor values (output of function W), we can obtain a better approximation of the world. The significance of the internal model on the homeokinetic control principle will presented with explicitly in the next sections of this chapter.

Assuming a simplified world defined as

$$W(y_t) = ay_t \tag{2.4}$$

The equation 2.3 can be written as

$$x_{t+1} = ay_t + \xi_t \tag{2.5}$$

Where a is an unknown constant. From the equations 2.5 and 2.2 is derived the complete equation of the control system.

$$x_{t+1} = \psi(x_t) = a\tanh(Cx_t + h) + \xi_t \tag{2.6}$$

Equation 2.6 can be interpreted as a non-linear neuron model with self-connection, the neuron is activated from the stimuli that are generated from itself. The dynamics of such types of neurons were the research topic of numerous researchers (Renals and Rohwer, 1990; Pasemann, 1993). Thus the complete equation of the system can be written in terms of membrane potential of the neuron.

$$z_{t+1} = r\tanh(z_t) + h \tag{2.7}$$

Where $z_t = Cx_t + h$ and $r = Ca$. The study of the dynamics of the controller in terms of the membrane potential is crucial for anticipating the way that the controller changes the generated motor commands.

2.3 Sensorimotor loop dynamics

Before the illustration of the homeokinetic controller's dynamics, it is useful to present a general case of a controller's dynamics in order to anticipate how the movement policies can be represented as dynamical systems and the benefits of such representation. Given that there is a desired state of the system, the goal is the derivation of a differential equation that will have a stable attractor landscape the attractors can be modified through a learning process.

A differential equation can have fixed points, if $f(x)$ is differentiable and x is a fixed point, then it is true that $f(x) = x$. Thus the fixed point do not change from the application of mapping f. The fixed points could be attractors, repellers or saddle points. Attractors are points towards which a variable evolves over time according to a dynamical system. Reppelers have exactly the opposite properties from the attractors and saddle points are also stationary points but they are not extrema (reppelers or attractors). Thus a dynamical system changes states according to the positions of attractors and reppelers and by representing a control scheme as dynamical system we can obtain a better understanding about the rules that define its operation.

For interpreting the properties of the membrane potential as a dynamical system, we should examine the fixed points of equation 2.7 The fixed points of the membrane potential can be found either graphically or analytically using the series expansion theorem (Martius, 2010). Figure 2.5 illustrates the change of the fixed points depending on the value of the feedback weight r and for $h = 0$. If the value of r is less than 1 then there is a single fixed point at $z = 0$, for values of r greater than 1 the previous fixed

point becomes unstable and new stable points appear. The interesting fact is that the fixed points which appear if $r > 1$ has opposing signs.

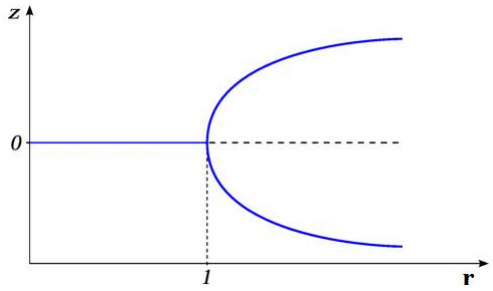

Figure 2.5: Fixed points of membrane potential as a function of r (Hesse, 2009)

Based on this fact it is clear that a bifurcation appears. According to Guckenheimer (2007) the bifurcation of a dynamical system, is defined as: *"The qualitative change in system's dynamics produced by varying parameters."* . Thus at the systems with this property a small change of a parameter results to a sudden change in its behaviour. Furthermore it was proved that the bifurcation created from the derivative of membrane potential function is a special case of bifurcation, namely a cusp bifurcation (Martius, 2010, p. 37)

This property, in the example of one-dimensional controller, will result to a robot that would not move when $r < 1$ and it will move forward or backward if $r > 1$. The direction that it will take is not a result of an intrinsic choice but is the result of the symmetry breaking phenomenon which occurs in bifurcations and the system follows a branch of bifurcation according to small fluctuations that occur in the system (noise).Thus, in order to generate motion is crucial to maintain the value of r above the critical point and since $r = Ca$ this is achieved by changing the parameter C because a is constant.

So far the impact of the bias h is not examined since we assumed that $h = 0$, but it also has impact on the occurrence of fixed points. A detailed investigation on the effects of the bias term is presented by Hesse (2009). It was proved that, assuming a critical value of the bias h_c where a symmetry breaking occurs, and if it is true that $h > h_c \vee h < -h_c$ three fixed points are created. Figure 2.6 illustrates the case $h < -h_c$ if $h > h_c$ the only change is that the fixed points change sign. From this illustration can be concluded that a merge of two branches occurs resulting to a catastrophic bifurcation.

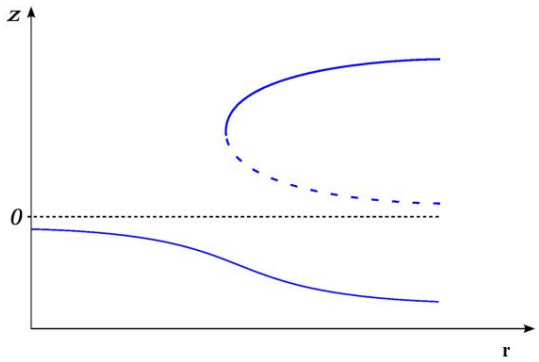

Figure 2.6: Fixed points of membrane potential as a function of r and $h < 0$ (Hesse, 2009)

There are three fixed points, two stable (continuous lines) and one unstable (dashed line) when the value of bias slowly changes (increases or decreases) the state of the system can be found in one of the two stable fixed points and remains at this stable point until its fixed point disappears due to the change of the bias. Then it jumps to the fixed point that has the opposite sign (Martius, 2010).Thus, the parameter C controls the two possible states of the system,a change of this parameter produces two new possible states. The parameter h controls the transition between those two states. Therefore, is suitable to maintain the parameter r above the bifurcation point and change the bias in order to achieve a transition between the two states/

The analysis of the sensorimotor loop dynamics shows that, using a closed loop system with a controller which can be realized as a single neuron with self feedback weight, fixed points are found that can be adapted in order to achieve a self-organized behaviour based on the principle of homeokinesis.

2.4 Principle of honeokinesis

In the previous section 2.3 was illustrated that exploiting the dynamics of the sensorimotor loop, different states of the systems can be emerge. A small change of the parameter has as result a change of the system which is common phenomenon in the self-organized processes. In order to have a self-organized system we need to derive a method for adapting the parameters of the sensrimotor loop (r,h) that would have common characteristics with the rules that cause self-organization in systems.

If the minimization of the prediction error is used (as illustrated in figure 2.4) will result in a static behaviour because the system will prefer states that can be easily predicted from the model. Therefore the robot would not move because it could predict

with bigger accuracy a stable state. Thus, we need a principal which will drive the system in changing its state resulting into an active and innovative robot. This issue is solved using a control scheme based on the principle of homeokinesis which could be characterised as a supervised learning algorithm but using a learning rule that is internal to the agent. The goal of this rule is to adapt the parameters r and h in order to generate a class of behaviours that will be explorative and fit the agents' embodiment.

For achieving the described operation of the system Der and Liebscher (2002) proposed the Time Loop Error (TLE) as objective function used for the adaptation of parameters r and h. This adaptation requires knowledge of the unknown hardware constant a therefore an internal model is used in order to approximate this factor. The control scheme that applies the principal of hoemokinesis for robotic control is illustrated in figure 2.7. This control scheme will be used in this thesis.

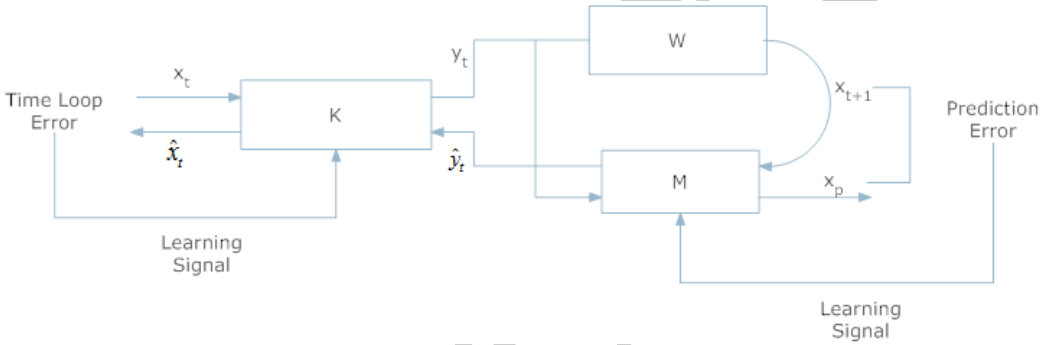

Figure 2.7: Sketch of the homeokinetic control scheme

The time loop error is derived by back-propagating the sensor values x_{t+1} throw the internal model, this results to the derivation of predicted motor commands $\hat{y}_t$ which can be interpreted as an estimation of the actual motor commands, then the predicted motor commands are also propagated back through the controller and the reconstructed sensor value are derived.

The robot's intrinsic anticipation of the sensorimotor loop can be formulated as

$$x_{t+1} = \psi(x_t) + \xi_t \tag{2.8}$$

Where $\psi(x_t) = M(K(x_t))$ and ξ_t the prediction error at time step t. As result, in each time step, the sensorimotor loop transforms the input x_t to the output x_{t+1}. The reconstruction of the input based on the observed output is equivalent to the calculation of an input shift u such that

$$x_{t+1} = \psi(x_t) + \xi_t = \psi(x_t + u_t) \tag{2.9}$$

Thus the parameter of the input shift can be realized as the change that should occur at the sensor input in order to compensate for the prediction error ξ (Martius, 2010, p. 42). Assuming that the inverse of function ξ exists, the reconstructed sensor values can be written as

$$\hat{x}_t = x_t + u_t = \psi^{-1}(x_{t+1}) \tag{2.10}$$

Thus, a quantity u that represents the input shift can be written as the difference between the true and the reconstructed sensor values.

$$u_t = \hat{x}_t + x_t \tag{2.11}$$

The goal of the homekinetic control scheme is to minimize this value which can be calculated assuming that it is small and thus we can use a Taylor expansion, so the last part of equation 2.9 is written as

$$\psi(x_t + u_t) = \psi(x_t) + L(x_t)u_t \tag{2.12}$$

Where L is the Jacobian of the sensorimotor loop and is defined as

$$L = \frac{\partial \psi(x_t)}{\partial x_t}$$

Thus the input shift can be derived using the equations 2.9 and 2.12 as follows

$$u_t = \frac{\xi_t}{L_t} \tag{2.13}$$

Finally the time loop error is defined as

$$TLE = |u_t|^2 = \left|\frac{\xi_t}{L_t}\right|^2 \tag{2.14}$$

Minimizing the time loop error can be realized as finding the best trade-off between the change of the sensor value and the prediction error. It consists of two parts, the prediction error and the Jacobian, for minimizing the time loop error we need to minimize the prediction error and maximize the Jacobian. The Jacobian can be interpreted as a measurement of the system's sensitivity because it represents the rate of the output's change with respect to the input, the maximization of the sensitivity drives the system to activity on the contrary the minimization of the prediction error drives the robot to "prefer" highly predictable states.

It is clear that two opposing forces are acting to the system, the need for high sensitivity and the predictability. The two opposing forces that act to a system is common

phenomenon in real life, for instance, at the example of the birds' swarm presented in section 1.2, two opossing force are acting, the need for immigration and the threat from predators. If the birds do not want to immigrate then they remain hidden in their nests because of the predators, on the other hand, if there was not a threat then the birds will not formate a swarm.

2.5 Learning rule

The learning rule of the controller involves the calculation of the gradient of the time loop error. For keeping the simplicity of the formulas, the derivation of the learning rule in one-dimensional case will be presented and a simple linear model will be defined as

$$M(y) = ay + b \tag{2.15}$$

Recalling the formulas that describe the system, denoting as $g(z)$ the activation function of the controller such that $g(z) = \tanh(z)$ and neglecting the time-step indicator t it is true that

$$K(x) = g(Cx+h)$$

$$\psi(x) = M(K(x)) = a(g(cx+h)) + b$$

$$L = \frac{\partial \psi(x_t)}{\partial x_t} = acg'$$

Where g' and g'' are the first and second order derivative of the sigmoid function $g(cx+h)$ Then, the gradient of the time loop error (equation 2.14) with respect to the parameter C can be calculated as follows

$$\Delta C = -\varepsilon \frac{\partial \mathrm{E}}{\partial C} = -\varepsilon 2 \frac{\xi}{L} \left(-\frac{\xi}{L^2} \frac{\partial L}{\partial C} + \frac{1}{L} \frac{\partial \xi}{\partial c} \right) \tag{2.16}$$

Using in equation 2.16 that

$$\frac{\partial L}{\partial C} = ag' + acg''x$$

$$\frac{\partial \xi}{\partial C} = 0$$

It can be simplified to the form

$$\Delta C = \varepsilon \frac{\xi^2}{L^3}(ag' + acg''x)$$

Which can be further simplified using a time-dependent learning rate $\varepsilon_t = \varepsilon\frac{\alpha\xi^2}{L^3}g'$ and $g'' = -2gg'$ to the final form

$$\Delta C = \varepsilon_t(1 - 2cyx) \tag{2.17}$$

Using the similar method the adaptation rule of the bias term is derived

$$\Delta h = \varepsilon_t(2cyx) \tag{2.18}$$

Thus the updating rule for the parameter C(equation 2.17) consists of a driving term and an anti-hebbian rule and the updating rule ofthe bias drives h to the opposite direction than y (Martius, 2010, p.45). The derivation of the learning rules in high-dimensional cases can be found in (Der and Martius, 2012, Ch. 15)

Chapter 3

The Internal Models

From Chapter 2 it is clear that the adaptation of the controller depends on the estimation of an unknown hardware constant and from the prediction error between the true and the estimated output of the sensorimotor loop in each time-step. Thus the need for investigation of the models' impact on the homeokinetic controller arises. In this chapter will be presented the internal models that were tested for evaluating their impact on the homeokinetic control scheme.

The internal model has to perform predictions based on training samples $[y_t, x_t]$ that denote a desired output for a given input and occur in real time. Thus it can be realized as a machine learning problem which belongs to the class of supervised learning. The term supervised denotes the existence of a training signal that is used for the adaptation of the model's parameters. In the framework of the homeokinetic controller, this fact does not imply an external teacher because the desired output is result of the agent's interaction with the environment. For investigating the hypothesis, two classes of machine learning algorithms were tested, namely recurrent neural networks and regression analysis.

3.1 Recurrent Neural Networks

Recurrent Networks (RNN) are a special case of Artificial Neural Networks and are biological inspired from the neural networks of organisms. They consist of nodes, that can be interpreted as an analogy of the brain's neurons, when a node is activated it drives its signal to the connected nodes those connections between them are called weights or synaptic connections. According to the type of their inner connections they are classified as feed-forward and recurrent. ANN performs a mapping from input

space to output space based on a learning signal (supervised learning) which is used for adapting the weights. ANN has been used as predictor in many applications.

In Feed-forward Neural Networks (FNN) the activation flows forward from input to output. In many occasions are used nodes that are neither input or output therefore they are called hidden-nodes, using this nodes the network can approximate static non-linear mappings. There is huge literature with applications and weights' adaptation rules because they combine both simplicity and very good performance.

The RNN, whose performance will be investigated in this thesis, have at least one loop in their structure, i.e. the activation is also fed backwards or and parallel (between units of the same layer) this makes them capable to preform approximation of dynamical systems. The property of recurrency has as result the appearance of long short-term memory Hochreiter and Schmidhuber (1997) and they are a better approximation of biological neural networks since all of them are recurrent. Due to those advantages the class of RNNs was selected as model of the world.

In the following subsections will be presented the RNNs that were studied. RNNs with different types of structure, topologies and learning rules were implemented, the main reason for selecting the learning algorithms was their ability to train the network in online mode.

3.1.1 Real time recurrent learning

Real time recurrent learning (RTRL) was introduced by Williams and Zipser (1989) as a way to efficiently train recurrent networks that are updated continuously with new data without suffering from the high computational cost of algorithms like back-propagation through time which involves unfolding the network back in time, this results to a feed-forward network in which the back-propagation algorithm is used for the adaptation of weights (Werbos, 1990). This procedure becomes more computationally demanding as time rises. RTRL has been successfully applied in numerous applications including stream-flow forecasting (Chang et al., 2002) and recognition of finite-state grammars (Smith and Zipser, 1989)

The RNN is fully connected, all the nodes are connected to each other except the input nodes which do not receive a synaptic connection. Thus the weight matrix W will have as many rows as the sum of hidden and output nodes and as many columns as the number of all network's nodes. Denoting by w_{mn} the weight from node m to node n and by $x_k(t)$ the value of node k at time step t, where $k \in I$ if the node is input

node, $k \in H$ if the node is hidden and $k \in O$ if it is an output node. The activation of the node $x_k : k \in H \cup O$ at time step $t+1$ is calculated as

$$x_k(t+1) = \sum_{i \in I \cup H \cup O} f(w_{ki} x_i(t)) \tag{3.1}$$

Where f is a sigmoid function. The weights are updated according to the gradient of the error function, which is calculated form the difference between the values of the output nodes and their desired value as follows

$$E(t+1) = \frac{1}{2} \sum_{i \in O} (d_i - x_i)^2 \tag{3.2}$$

$$\begin{aligned} \Delta w_{mn}(t+1) = -\mu \frac{\partial \mathrm{E}(t+1)}{\partial w_{mn}} = -\mu \sum_{i \in O} \left(\frac{\partial \mathrm{E}(t+1)}{\partial x_i} \frac{\partial x_i}{\partial w_{ml}} \right) = \\ -\mu \sum_{i \in O} \left(e_i(t+1) \frac{\partial x_i}{\partial w_{ml}} \right) \end{aligned} \tag{3.3}$$

Where μ is the learning rate. Thus, according to equation 3.3, the term $\frac{\partial x_i}{\partial w_{ml}}$ have to be calculated in order to find the change of the weights. This term describes the sensitivity of the outputs' node value x_i to a small change of the weights over all the time steps. An other interesting fact is that the weight w_{mn} has not to be connected with output x_i, thus this learning rule is non-local because the sensitivity of a output node depends on changes that could have occurred in a different place of the network's topology.The sensitivity is computed recurrently at each time step from the following formula assuming that $\frac{\partial x_i(0)}{\partial w_{ml}} = 0$.

$$\frac{\partial x_i(t+1)}{\partial w_{mn}} = \dot{x}_i(t+1) \left[\sum_{j \in O \cup H} w_{mj} \frac{\partial x_i(t)}{\partial w_{mn}} + \delta_{ji} x_n \right] \tag{3.4}$$

Where $i, m :\in I \cup H$, $n \in I \cup H \cup O$ and δ_{ji} is the Kronecker delta.

Summarizing, the complete algorithm was implemented as follows

1. Calculate nodes activation using equation 3.1
2. Calculate error (equation 3.2)
3. Recurrently compute the nodes' sensitivity according to equation 3.4
4. Update weights based on equation 3.3

Despite that RTRL requires less computational resources than the back-propagation through time algorithm which has $O(n^4)$ complexity, it is computational expensive

with $O(n^3)$ complexity and also has large storage demands. Thus researches focused on the derivation of less complex learning algorithms. The most fruitful approaches are the Atiya-Parlos recurrent learning (APRL) introduced by Atiya and Parlos (2000) which has $O(n^2)$ and Back-Propagation Decorrelation introduced by Steil (2004) and is based on APRL and has $O(n)$ complexity.

3.1.2 Back-propagation decorrelation

BPDC combines two main methods for training RNN, the first is the Echo State Network (ESN) introduced by Jaeger (2002a) which uses a dynamic reservoir in the network's topology in order to store information about the temporal behaviour of inputs (Steil, 2004). The second method is APRL which can be applied in both continuous and discrete time. The similarity of those two approaches is that during the training procedure the dynamic reservoir exhibits a slow changing rate contrary to the output layer which is adapted quickly.

Based on those facts Steil introduced BPDC which uses decorrelation for information processing and also propagates the errors derived from a teaching signal back in time. This learning rules is based on three main principles, namely:

1. The errors are one-step back-propagated using the concept of virtual teacher forcing, similarly with the APRL.

2. Temporal memory is introduced in the reservoir and adapted based on the decorrelation of the activations

3. Complexity is reduced using a reservoir with fixed weights

Thus the structure that is illustrated in figure 3.1 is used for applying the BPDC learning rules. The dynamic reservoir has fixed weights which remain unchanged and is fully connected (illustrated as continuous lines), only the weights from the reservoir to the output and the weights connecting the output nodes with each other are updated (dashed lines). The activations of a node $x_k : k \in H \cup O$ are calculated based on the following equation.

$$x_k(t+1) = \sum_{i \in I \cup H \cup O} w_{ki} f(x_i(t)) \tag{3.5}$$

This equation is different than the equation used for the calculation of activation in the RTRL approach, since only the connected nodes are squashed through a sigmoid

function. This modification allows the outputs to take values out from the interval $[-1,1]$, which is useful in applications such as time series prediction. According to Steil (2004) the network achieves its maximum processing capacity if the reservoir weights are maximally decorrelated with respect to the given input.

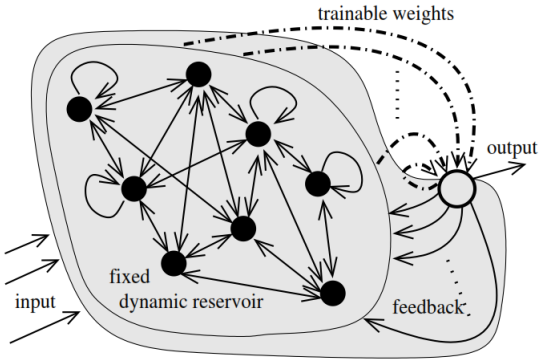

Figure 3.1: Structure of RNN trained with BPDC (Steil, 2004)

Equation 3.6 illustrates the proposed simple learning rule.

$$\Delta w_{mn}(t+1) = \eta \frac{f(x_n(t))}{\sum_{s \in O} f(x_s(t))^2 + \varepsilon} \gamma_m(t+1) \tag{3.6}$$

Where $m \in O$, $n \in O \cup H$ and η the learning rate. The division term performs an approximation of the decorrelation rule. The term γ_i preforms a weighted back-propagation of the errors as presented by the following equation.

$$\gamma_m(t+1) = \sum_{s \in O} w_{ms} \dot{f}(x_s(t)) e_s(t) - e_m(t+1) \tag{3.7}$$

This terms includes the usage of the errors in the current $e(t+1)$ and the previous time step $e(t)$, the errors are calculated by the difference between the desired and the true values of the output neurons $e_s = d_s - x_s$ where $s \in O$.

The BPDC learning rule is based on the constrained minimization of an error function E w.r.t. $g \equiv 0$. If x_s is an output unit with a corresponding desired value d_s and T the total time steps then E is computed as

$$E = \frac{1}{2} \sum_{t=1}^{T} \sum_{s \in O} [x_s(t) - d_s(t)]^2$$

The constrain function g is derived from the networks dynamics as

$$g(t+1) = -x(t+1) + W f(x(t)) = 0$$

Where W is the weights' matrix of the network. The minimization of the error function is performed according to the method introduced by (Atiya and Parlos, 2000), the constrain equation is used for obtaining the desired weight changes in order to approach a state defined from a virtual target, the virtual target is obtained by calculating the partial derivative of the error with respect to the state.

This is one significant difference with conventional approaches such as RTRL and back-propagation through time which differentiate an error function with respect to the weights using the gradient descent method for adapting the weights. This difference makes the algorithm faster but have two serious drawbacks. First it is characterised by low memory since the algorithm itself does not stores values corresponding to memory so it depends only on the reservoir dynamics for keeping a type of memory. Second BPDC is extremely sensitive to the training sequence because the weights are adapted based on the errors of the previous time step this cause low generalization capability and thus an overfitting behaviour (Steil, 2005).

3.1.3 Echo state network

As mentioned in the previous subsection, the Echo State Network was introduced by Jaeger (2002a) who proposed a network topology and weights' initialization for RNNs. The notion "echo" is based on that the nodes' activation $x(t)$ is a function of the input history presented to the network (Jaeger, 2001) thus the activations echo the input history. A RNN is an ESN if its weights have the Echo State Property (ESP) which according to Jaeger (2002b) is defined as:

> *"Assuming an untrained network with weights matrix W is driven by teacher input $u(n)$ and teacher-forced by teacher output $d(n)$ from compact intervals U and D. The network has echo states w.r.t. U and D, if for every left-infinite input/output sequence $(u(n), d(n-1))$, where $n = ..., -2, -1, 0$ and for all state sequences $x(n)$, $x(n)$ compatible with the teacher sequence."*

This can be interpreted as that if an ESN is trained for sufficient large time (a large input history has been memorized in the dynamic reservoir) then the state of the network depends only on the history of the input-output pair. ESN, also have the state contracting property which ensures that the initialization of the weights do not affect the performance of the network for a long time. A RNN has those properties by adjusting the weights based on the following steps:

1. Chose of an initial random weight matrix of the reservoir W_0
2. Normalization of W_0 using its spectral radius $|\lambda_{max}|$ defined as the largest eigenvalue of W_0, thus $W_1 = \frac{1}{|\lambda_{max}|} W_0$
3. Conversion of W_1 to a scaled matrix $W = \alpha W_1$ where $\alpha < 1$

The topology of the used ESN is illustrated in figure 3.2 where solid arrows represent fixed weights and dashed arrows trainable weights. This topology has similarities with the topology used for BPDC learning (figure 3.1) but in this case there are also weights from the input to the output units, furthermore the weights are updated based on a different learning rule and also the networks activation is calculated differently.

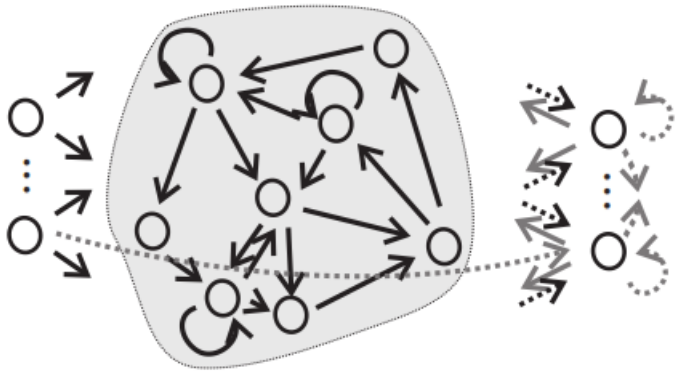

Figure 3.2: Structure of the ESN (Jaeger, 2001)

For instance the activations of the network's units are calculated based on the following equation

$$x_k(t+1) = \sum_{i \in I \cup H \cup O} f(w_{ki} x_i(t)) \tag{3.8}$$

The weights are adapted throw a gradient descent learning rule, namely the delta rule which minimizes an error function of the form

$$E = \sum_{j \in O} \frac{1}{2} (d_j - x_j)^2$$

Computing the partial derivatives of this error with respect to the weights, the following equation of weights' adaptation is derived.

$$\Delta w_{mn} = \eta (d_m - y_m) \dot{f}(\sum_{i \in H \cup I \cup O} w_{mi} x_i) x_n \tag{3.9}$$

The code used for the implementation of the described algorithm was created by Martius et al. (2010). The ESNs have exhibit good performance as predictors in many applications including grammar structure learning (Tong et al., 2007), stock price prediction (Lin et al., 2009) and speech recognition (Skowronski and Harris, 2007)

3.1.4 Jacobian matrix of RNN

In this subsection will be presented the derivation of the RNN Jacobian matrix, this feature is crucial in the homeokinetic control scheme because, recalling the general equation of the internal model (equation 2.15) and the controller's learning rules (equations 2.17, 2.18)

$$M(y) = ay + b$$

$$\Delta C = \varepsilon_t(1 - 2cyx)$$

$$\Delta h = \varepsilon_t(2cyx)$$

$$\varepsilon_t = \varepsilon\frac{\alpha\xi^2}{L^3}\dot{g}$$

It is clear that the hardware constant α has a significant impact on the learning procedure of the controller, this constant describes the effect that has a small change of the motor commands y to the predicted sensory value x thus can be interpreted as the Jacobian matrix of the internal model, the partial derivative of the outputs with respect to the inputs.

For the derivation of the Jacobian matrix, will be used a simplified case of a network that has one input x and one output y and N hidden neurons h. Also the weights' matrix of the network $\mathbf{W}$ is divided in the following six parts.

- $w_{in\to out}$ the direct weight from the input to the output
- $\mathbf{w}_{in\to res}$ column vector containing the weights from the input to reservoir
- $\mathbf{W}_{res}$ NxN matrix of the reservoir internal connections
- $\mathbf{w}_{res\to out}$ row vector of the weights from the reservoir to the output node
- $\mathbf{w}_{out\to res}$ column vector of the weights from output node to the reservoir nodes
- $w_{out\to out}$ is the self-connection of the output node

Then the activation of the network at time step $t+1$ is computed from the equations:

$$y(t+1) = f(\mathbf{w}_{res\to out}\mathbf{h}(t+1) + w_{in\to out}x(t+1) + w_{out\to out}y(t)) \tag{3.10}$$

$$\mathbf{h}(t+1) = f(\mathbf{w}_{in\to res}x(t+1) + \mathbf{W}_{res}\mathbf{h}(t) + \mathbf{w}_{out\to res}y(t)) \tag{3.11}$$

Setting for clarity $k(t+1) = \mathbf{w}_{res\to out}\mathbf{h}(t+1) + w_{in\to out}x(t+1) + w_{out\to out}y(t)$ and differentiating with respect to the input we get

$$\frac{\partial y(t+1)}{\partial x(t+1)} = \dot{f}(k(t+1))\frac{\partial k(t+1)}{\partial x(t+1)} =$$

$$\dot{f}(k(t+1))\left[\mathbf{w}_{res\to out}\frac{\partial \mathbf{h}(t+1)}{\partial x(t+1)} + w_{in\to out}\frac{\partial x(t+1)}{\partial x(t+1)} + w_{out\to out}\frac{\partial y(t)}{\partial x(t+1)}\right]$$

Given that $\frac{dx(t+1)}{dx(t+1)} = 1$ and assuming that the output at time step t does not depend on the input at time step $t+1$, $\frac{dy(t)}{dx(t+1)} = 0$ the formula can be simplified as

$$\frac{\partial y(t+1)}{\partial x(t+1)} = \dot{f}(k(t+1))\left[\mathbf{w}_{res\to out}\frac{\partial \mathbf{h}(t+1)}{\partial x(t+1)} + w_{in\to out}\right]$$

Similarly the term $\frac{\partial \mathbf{h}(t+1)}{dx(t+1)}$ is calculated setting $l(t+1) = \mathbf{w}_{in\to res}x(t+1) + \mathbf{W}_{res}\mathbf{h}(t) + \mathbf{w}_{out\to res}y(t)$ and assuming that $\frac{\partial \mathbf{h}(t)}{dx(t+1)} = 0$ and $\frac{dy(t)}{dx(t+1)} = 0$, as result the final equation is

$$\frac{\partial y(t+1)}{\partial x(t+1)} = \dot{f}\left(k\left(t+1\right)\right)\left[\mathbf{w}_{res\to out}\left(\mathbf{w}_{in\to res}\otimes \dot{f}\left(l\left(t+1\right)\right)\right) + w_{in\to out}\right] \tag{3.12}$$

Where $\otimes$ is the row-wise multiplication operator. The similar process can be generalized for multidimensional cases in order to be calculated the partial derivative of each output with respect to each input.

3.2 Regression Analysis

Regression methods are popular in statistics for the derivation of rules that describe a set of investigated data. The general approach involves the estimation of a plane in the training data space that describe the data best. The estimation of future outputs is derived based on the fitted plane, thus regression methods are widely used as prediction methods because through those methods dependencies between training inputs and

outputs are derived which can be used for the prediction of future outputs given a new input.

The main reason for selecting regression methods is that the mapping performed from the internal model is linear or nearly-linear in the most cases, also regression methods are characterized by simplicity. The regression algorithms that were tested are the linear regression, which is the most common method for the approximation of linear mappings, the locally weighted regression which is based on the linear regression with main difference that the inputs are weighted according to the training data and a simple linear predictor which is used as internal model of the homeokinetic control scheme in the majority of researches.

3.2.1 Linear regression

Linear regression uses a linear combination of the inputs value in order to predict the outputs value according to the following equation

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_n x_n + \beta_0 \tag{3.13}$$

Given a dataset $\{\mathbf{Y}, \mathbf{X}\}$ with N number of inputs and the corresponding desired outputs, the matrix $\mathbf{Y}$ contains the outputs and has N rows and O columns where O the dimension of outputs and similarly, matrix $\mathbf{X}$ is N by I where I the dimension of inputs. The predicted outputs are calculated using:

$$\hat{\mathbf{Y}} = \mathbf{XB} \quad \text{where} \quad \hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_n^T \end{bmatrix} \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \mathbf{B} = \begin{bmatrix} \beta_{01} & \beta_{02} & \cdots & \beta_{0o} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1o} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{i1} & \beta_{i2} & \cdots & \beta_{io} \end{bmatrix} \tag{3.14}$$

And the bold lower-case letters denote column vectors. Thus $\mathbf{y_n}$ denotes the n^{th} output sample which is a column vector of O rows and $\mathbf{B}$ is the regression matrix which contains the bias coefficient β_0 so the input vectors have the form $\mathbf{x}^T = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_i \end{bmatrix}^T$

The goal is to estimate the matrix of the regression coefficients that result to the minimum error between predicted and true outputs. There are two major classes of estimators the least square estimators (LSE) and the maximum likelihood estimators (MLE). The MLE assumes that the distribution of errors is known that belongs to a certain family of parametric distributions, thus it was not implemented because it adds unnecessary computational complexity to a real-time algorithm.

On the other hand, the LSE is simple commonly used method and according to Gauss-Markov theorem it is a Best Linear Unbiased Estimator this means that there is not any other unbiased estimator that results to less prediction error than the LSE. Thus, the LSE is the regression matrix that minimizes the least square error between the predicted outputs $\hat{\mathbf{Y}}$ and the desired $\mathbf{Y}$ defined as

$$\mathrm{E} = \frac{1}{2}(\mathbf{Y} - \hat{\mathbf{Y}})^T(\mathbf{Y} - \hat{\mathbf{Y}}) = (\mathbf{Y} - \mathbf{W}\mathbf{X}^T)^T(\mathbf{Y} - \mathbf{W}\mathbf{X}^T) \tag{3.15}$$

The LSE estimation of the regression matrix is calculated from the following equation

$$\hat{\mathbf{B}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y} \tag{3.16}$$

It is clear that the use of equation 3.16 requires the storage of every input which make its application infeasible due to the need of high computational resources, this is solved by recursively calculating the weights as soon an input appears based on the recursive least square update as illustrated in the following equation

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mathbf{P}_{t+1}\mathbf{x}(\mathbf{y} - \hat{\mathbf{y}})^T \tag{3.17}$$

Where $\mathbf{P}$ is identical to the inverse of the inputs' covariance matrix $(\mathbf{X}^T\mathbf{X})^{-1}$ and is calculated recursively as

$$\mathbf{P}_{t+1} = \left(\mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}\mathbf{x}^T\mathbf{P}_t}{1 + \mathbf{x}^T\mathbf{P}_t\mathbf{x}}\right) \tag{3.18}$$

The derivation of this recursive rule is based on the Woodbury matrix identity and the matrix $\mathbf{P}$ is initialized as diagonal matrix consisting of diagonal elements with big values. The algorithm consists of the following steps

1. At $t = 0$ initialize $\mathbf{P} = \mathbf{I}\frac{1}{\gamma}$ where $\gamma \ll 1$
2. For every new input $\mathbf{x}$
3. Calculate predicted outputs
4. Calculate $\mathbf{P}_{t+1}$
5. Update weights' change

A significant drawback of this algorithm is an under-fitted behaviour that may occur for two reasons. First it performs a linear mapping which makes hard the prediction of non-common input-output pairs also it does not have any module in its structure or

in its learning rule in order to be adaptive in a change of the environment's or robot's state, e.g. when the robot halts.

3.2.2 Locally weighted regression

Locally weighted regression (LWR) was introduced by Cleveland and Devlin (1988),in their article the authors are more concerned about the statistical properties of this approach, later Atkeson et al. (1997) proposed a complete learning method based on the LWR. This learning method belongs to the class of non-parametric methods. The characteristic of those methods is that they do not assume a certain structure for the data such as linear regression which assume that the input output mapping is linear. Thus LWR has the advantage that there is not the need of defining a global model that fit the data.

This is achieved by weighting the values of a new incoming input $\mathbf{q}$ (query point) according to a memorized history of inputs and their corresponding outputs. The weighting procedure consists of three steps. First, the euclidean distance between the query point and all the memorized inputs is calculated using the following equation.

$$\mathbf{d}(\mathbf{x},\mathbf{q}) = \sqrt{(\mathbf{x}-\mathbf{q})^{\mathbf{T}}(\mathbf{x}-\mathbf{q})} \tag{3.19}$$

The result of the above equation is a column vector $\mathbf{d}$ that has as many elements as the number of memorized inputs. At the second step a kernel function is applied at the distance vector, the kernel used in this implementation was the Gaussian which produces the characteristic bell-shaped distribution.

$$K(\mathbf{d}) = e^{\frac{-d^2}{2\sigma^2}} \tag{3.20}$$

The term σ controls the width of the distribution's shape, thus it determines the degree of impact that each memorized input has to the new query point and through this parameter is controlled the fitting of the model Finally the weights are calculated from the square root of the kernel

$$\mathbf{w} = \sqrt{K(d)} \tag{3.21}$$

Each memorized input and output is multiplied with its corresponding weight and

the following matrices are created

$$\mathbf{V} = \begin{bmatrix} w_1\mathbf{y}_1^T \\ w_2\mathbf{y}_2^T \\ \vdots \\ w_n\mathbf{y}_n^T \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} w_1\mathbf{x}_1^T \\ w_2\mathbf{x}_2^T \\ \vdots \\ w_n\mathbf{x}_n^T \end{bmatrix}$$

Using the weighted matrices of inputs and outputs, the regression coefficients are derived by the minimization of least squared error

$$\hat{\mathbf{B}} = \left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{V} \tag{3.22}$$

The predicted output of the query point is calculated using the simple regression equation

$$\mathbf{y}_{pred} = \mathbf{q}^T\mathbf{B} \tag{3.23}$$

The implemented algorithm consists of the following steps:

1. For every new input (query point)
2. Calculate the weights (equation 3.21) from the distance between the query point and all the memorized inputs applying the square root of the Gaussian kernel.
3. Weight the memorized inputs and outputs
4. Calculate the estimation of regression matrix using the memorized inputs-outputs (equation 3.22)
5. Predict the output of query point (equation 3.23)

Summarizing, it is true that despite the advantages, this algorithm has two serious drawbacks. First it requires to memorize the input-output history and second could have an under-fitting behaviour because of the memorized history. In order to deal with those drawbacks a slight modification was applied. The algorithm does not keeps in memory all the previous history of inputs-outputs but has a fixed-size memory of the last 2000.

This modification reduces the needed computational resources and makes the model more adaptable to changes, thus this modification can be interpreted as a soft forgetting module.

3.2.3 Linear predictor

The linear predictor is the simplest examined model and is used in the majority of the researches about the homeokinetic control scheme, as an internal model. It will be examined for obtaining an interpretation about the impact of the models' complexity to the control scheme.

The predicted outputs are calculated by the equation

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{3.24}$$

The parameters of the model, $\mathbf{A}$ and $\mathbf{b}$ are updated using the gradient descent learning rule on the prediction error (delta rule) and are calculated by the following equations

$$\Delta\mathbf{A} = 2\eta\mathbf{e}_{pred}\mathbf{y}^T \tag{3.25}$$

$$\Delta\mathbf{b} = 2\eta\mathbf{e}_{pred} \tag{3.26}$$

Where $\mathbf{e}_{pred}$ is the difference between the actual and the predicted outputs and η is a learning rate. The code that was used for the implementation of this model is made by Martius et al. (2010)

Chapter 4

Evaluation

Chapter 5

Conclusion and Discussion

Bibliography

Atiya, A. F. and Parlos, A. G. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence. *Trans. Neur. Netw.*, 11(3):697–709.

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning for control. *Artificial intelligence review*, 11(1-5):75–113.

Ay, N., Bertschinger, N., Der, R., Gttler, F., and Olbrich, E. (2008). Predictive information and explorative behavior of autonomous robots. *The European Physical Journal B*, 63(3):329–339.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA.

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–21.

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

Chang, F.-J., Chang, L.-C., and Huang, H.-L. (2002). Real-time recurrent learning neural network for stream-flow forecasting. *Hydrological Processes*, 16(13):2577–2588.

Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610.

Der, R. (2001). Self-organized acquisition of situated behaviors. *Theory Biosci.*, 120:179–187.

Der, R., Hesse, F., and Liebscher, R. (2004). Self-organized exploration and automatic sensor integration from the homeokinetic principle. In *Proc. 3rd Workshop on Self-Organization of AdaptiVE Behavior (SOAVE'04)*, Fortschritt-Berichte VDI, Reihe 10, Nr. 743, pages 220–230. VDI-Verlag.

Der, R. and Liebscher, R. (2002). True autonomy from self-organized adaptivity. In *Proc. Workshop Biologically Inspired Robotics*, Bristol.

Der, R. and Martius, G. (2012). *The Playful Machine - Theoretical Foundation and Practical Realization of Self-Organizing Robots*. Springer.

Der, R., Steinmetz, U., and Pasemann, F. (1999). Homeokinesis - a new principle to back up evolution with learning. In *Proc. Intl. Conf. on Computational Intelligence for Modelling, Control and Automation (CIMCA 99)*, volume 55 of *Concurrent Systems Engineering Series*, pages 43–47, Amsterdam. IOS Press.

Ferezou, I., Haiss, F., Gentet, L. J., Aronoff, R., Weber, B., and Petersen, C. C. (2007). Spatiotemporal dynamics of cortical sensorimotor integration in behaving mice. *Neuron*, 56(5):907 – 923.

Flanagan, A. and Wing, A. M. (1997). The role of internal models in motion planning and control: Evidence from grip force adjustments during movements of hand-held loads. *J. Neurosci*, 17:1519–1528.

Guckenheimer, J. (2007). Bifurcation. *Scholarpedia*, 2(6):1517.

Haken, H. (2008). Self-organization. *Scholarpedia*, 3(8):1401.

Hesse, F. (2009). *Self-Organizing Control for Autonomous Robots*. PhD thesis, University of Göttingen, Institute for Nonlinear Dynamics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jaeger, H. (2001). The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148.

Jaeger, H. (2002a). Adaptive nonlinear system identification with echo state networks. In *Advances in neural information processing systems*, pages 593–600.

Jaeger, H. (2002b). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik.

Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718 – 727.

Lin, X., Yang, Z., and Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, 36(3, Part 2):7313 – 7317.

Manoonpong, P. and Wörgötter, F. (2009). Efference copies in neural control of dynamic biped walking. *Robot. Auton. Syst.*, 57(11):1140–1153.

Martius, G. (2010). *Goal-Oriented Control of Self-Organizing Behavior in Autonomous Robots*. PhD thesis, Georg-August-Universität Göttingen.

Martius, G. and Herrmann, J. M. (2012). Variants of guided self-organization for robot control. *Theory in Biosci.*, 131(3):129–137.

Martius, G., Herrmann, J. M., and Der, R. (2007). Guided self-organisation for autonomous robot development. In Almeida e Costa, F., Rocha, L., Costa, E., Harvey, I., and Coutinho, A., editors, *Advances in Artificial Life 9th European Conference, ECAL 2007*, volume 4648 of *LNCS*, pages 766–775. Springer.

Martius, G., Hesse, F., Güttler, F., and Der, R. (2010). LPZROBOTS: A free and powerful robot simulator. `http://robot.informatik.uni-leipzig.de/software`.

Murata, S. and Kurokawa, H. (2012). *Self-organizing robots*. Springer.

Neumann, J. V. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA.

Nguyen, Q.-T. and Kleinfeld, D. (2005). Positive feedback in a brainstem tactile sensorimotor loop. *Neuron*, 45(3):447 – 457.

Nguyen-Tuong, D. and Peters, J. (2011). Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–40.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. A Bradford book. A BRADFORD BOOK/THE MIT PRESS.

Nolfi, S. and Floreano, D. (2002). Synthesis of autonomous robots through evolution. *Trends in Cognitive Sciences*, 6(1):31 – 37.

Nouyan, S., Campo, A., and Dorigo, M. (2008). Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23.

Pasemann, F. (1993). Dynamics of a Single Model Neuron. *International Journal of Bifurcation and Chaos*, 03(02):271–278.

Renals, S. and Rohwer, R. (1990). A study of network dynamics. *Journal of Statistical Physics*, 58(5-6):825–848.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34.

Schröder-Schetelig, J., Manoonpong, P., and Wörgötter, F. (2010). Using efference copy and a forward internal model for adaptive biped walking. *Auton. Robots*, 29(3-4):357–366.

Skowronski, M. D. and Harris, J. G. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, 20(3):414 – 423. Echo State Networks and Liquid State Machines.

Smith, A. W. and Zipser, D. (1989). Learning sequential structure with the real-time recurrent learning algorithm. *International Journal of Neural Systems*, 01(02):125–131.

Steil, J. (2004). Backpropagation-decorrelation: online recurrent learning with o(n) complexity. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 843–848 vol.2.

Steil, J. (2005). Memory in backpropagation-decorrelation o(n) efficient online recurrent learning. In Duch, W., Kacprzyk, J., Oja, E., and Zadrony, S., editors, *Artificial Neural Networks: Formal Models and Their Applications ICANN 2005*, volume 3697 of *Lecture Notes in Computer Science*, pages 649–654. Springer Berlin Heidelberg.

Svennebring, J. and Koenig, S. (2004). Building terrain-covering ant robots: A feasibility study. *Auton. Robots*, 16(3):313–332.

Thoroughman, K. a. and Shadmehr, R. (2000). Learning of action through adaptive combination of motor primitives. *Nature*, 407(6805):742–7.

Tong, M. H., Bickett, A. D., Christiansen, E. M., and Cottrell, G. W. (2007). Learning grammatical structure with echo state networks. *Neural Networks*, 20(3):424 – 432. Echo State Networks and Liquid State Machines.

Trianni, V. and Dorigo, M. (2006). Self-organisation and communication in groups of simulated and physical robots. *Biol. Cybern.*, 95(3):213–231.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wiener, N. (1965). *Cybernetics: Or, Control and Communication in the Animal and the Machine*. The @MIT paperback series: Massachusetts Institute of Technology. Mit Press.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280.

Wolpert, D. and Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nature Neuroscience*, 3 Suppl:1212–1217.

Wolpert, D., Ghahramani, Z., and Jordan, M. (1995). An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882.