Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Bioinformatics

Using spiking neural networks to simulate human motor learning

Sebastian Nagel

January 31st, 2015

Reviewers

Prof. Dr. Wolfgang Rosenstiel
Wilhelm-Schickard-Institute
Department of Computer Engineering
University of Tübingen

Prof. Dr. Hanspeter Mallot
Institute of Neurobiology
Department of Cognitive Neuroscience
University of Tübingen

Supervisor

Dr. Martin Spüler
Wilhelm-Schickard-Institute
Department of Computer Engineering
University of Tübingen

Nagel, Sebastian:
Using spiking neural networks to simulate human motor learning
Master Thesis Bioinformatics
Eberhard Karls University of Tübingen
Thesis period: August 1st, 2014 - January 31st, 2015

Abstract

The control of machines using thoughts opens up many possibilities. Brain-Computer-Interfaces (BCIs) allow to record brain activities, which in turn must be interpreted correctly. As brain activities adapt over time, where (i.e. visual) feedback plays an important role, even the BCI has to adapt. However, this leads to further adaption of brain activities, because of the changing feedback. A biologically realistic neuron model of the brain's sensorimotor area is required to study the mutual influence as well as to evaluate different classifiers. Recent sensorimotor models already have a realistic structure, but the learning behaviors seem to be unrealistic: the networks learn a specific mapping for a single target, therefore, they have to relearn the mapping for each target and the mapping for the previous target gets lost. In this work, a biologically realistic spiking neural network of the sensorimotor area is proposed, which is able to learn a static mapping for all targets without the need for further learning, using a biologically plausible learning approach. Using the distance to the target as population encoded synaptic input, the resultant models are able to simulate simple movement tasks in realtime, with low deviations to targets and fast movements from one target to another. Furthermore, a new neural coding was developed that allows a population of cells to calculate approximately the difference between two values. This neural coding allows the network to calculate independently the distance to target, using the current position and the target position as synaptic inputs.

Zusammenfassung

Die Steuerung von Maschinen mit Hilfe von Gedanken eröffnet viele Möglichkeiten. Mit Brain-Computer-Interfaces (BCIs) können Hirnaktivitäten gemessen und interpretiert werden. Da sich Hirnaktivitäten über die Zeit anpassen können, wobei (z.B. visuelles) Feedback eine wichtige Rolle spielt, muss sich auch das BCI anpassen. Dies führt allerdings durch ein sich veränderndes Feedback dazu, dass sich die Hirnsignale erneut anpassen. Um diesen gegenseitigen Einfluss zu erforschen und um unterschiedliche Vorhersagealgorithmen beurteilen zu können, ist ein biologisch realistisches Model der sensomotorischen Hirnareale wichtig. Neueste Modelle zeigen zwar eine realistische Struktur, aber die Learnverhalten sind biologisch betrachtet nicht plausibel: die Modelle lernen für jedes Ziel ein spezifisches Mapping, wenn sich das Ziel ändert muss ein neues Mapping erlernt werden, wodurch das vorherige Mapping verloren geht. In dieser Arbeit wird ein realistisches sensomotorisches Neuronenmodel vorgestellt, welches mit Hilfe eines biologisch plausiblen Lernverfahrens, ein statisches Mapping für alle Zielpositionen erlernen kann. Die resultierenden Modelle können einfache Bewegungsaufgaben in Echtzeit simulieren, wozu nur die Distanz zur Zielposition als synaptischen Input dient. Die Abweichungen zur Zielposition sind dabei sehr gering und die Bewegung von einer Zielposition zu einer anderen verlaufen schnell. Des Weiteren wurde eine neurale Codierung entwickelt, die es einer Neuronen-Population erlaubt die Differenz zwischen zwei Werten annäherungsweise zu berechnen. Diese Codierung erlaubt dem Netzwerk die Distanz zwischen der aktuellen Position und der Zielposition eigenständig zu berechnen, damit ist es möglich nur die aktuelle Position und die Zielposition als synaptische Inputs zu verwenden.

Acknowledgements

First of all, I would like to thank my family for all the support over the last years, in order that I was able to concentrate fully on my study.

This thesis would not have been possible without the help of some people. Special thanks goes to my supervisor Martin Spüler who helped me with all my questions, offered constructive criticism and advised me on how it can best be implemented. I would also like to thank Wolfgang Rosenstiel and Hanspeter Mallot for being the reviewers as well as for the opportunity to write my master thesis at the department of computer engineering. Furthermore, I would like thank Alexander Peltzer, who took the time to made a linguistic proofreading.

Finally, I would particularly like to thank my girlfriend, Kristina Heller, for the support even in stressful times and for all the motivation.

Contents

List of Figures

List of Tables

List of Abbreviations

ACh	Acetylcholine
BCI	Brain-Computer Interface
CNS	Central Nervous System
DOF	Degree of Freedom
ECoG	Electrocorticography
EEG	Electroencephalography
EM	Excitatory Motor
ES	Excitatory Sensory
IM	Inhibitory Motor
IS	Excitatory Sensory
LMN	Lower Motor Neuron
M1	Primary Motor Cortex
nAChR	Nicotinic Acetylcholine Receptor
NMJ	Neuromuscular Junction
PARC	Perception-Action-Reward-Cycle
PMC	Premoter Cortex
PSC	Primary Somatonensory Cortex
RL	Reinforcement Learning
RMSD	Root Mean Square Deviation
SAC	Somatosensory Association Cortex
SMA	Supplementary Motor Area
SSP	Structural Synaptic Plasticity
STDP	Spike Time Depended Plasticity
UMN	Upper Motor Neuron

Chapter 1

Introduction

The brain is the most important organ in humans. It is not only indispensable as a control center for the human body but also a bearer of human personality. Brain research tries to understand not only the functioning of the brain as an organ, but also the linkage of brain activities with our perceptions, feelings, thought processes, and body movements.

Today, brain research is in the early stages of development, but many research groups deal with it. If we could understand how the brain works, we might be able to heal or at least help people with several diseases like Parkinson, epilepsy, dementia, multiple sclerosis, locked-in syndrome, and many more.

This work focus on muscle contraction, the associated brain signals and proprioceptive feedback. To this day, many researchers make an effort to develop prosthesis with fine motor skills and sensory feedback. While sensory feedback is not yet well understood, both motor and control capabilities are already well advanced, allowing persons to perform activities of daily living, such as grasping a bottle and take a sip of it [HBJ^{+}12] or to eat independent of other people [CWD^{+}12].

Brain activity can be recorded with several methods, for example with electroencephalography (EEG) where electrodes are placed along the scalp, or electrocorticography (ECoG) where electrodes are placed directly on the surface of the brain. The interface between the brain and a computer is called Brain-Computer-Interface (BCI). To estimate the movement intention, a classifier uses the current brain activity. However, the recorded signals can undergo considerable changes between training and feedback mode as well as during feedback itself, for example: (1) task differences between training and feedback (e.g. real vs. imagined actions), (2) variability of the recording caused by drying gel or micro movements of the electrodes, (3) plasticity of the brain, due to experience with the task, and (4) modulation of cognitive states like attention, motivation and vigilance [BRW^{+}07]. Because of the adaptivity of the brain, neural activities change over time, therefore, a classifier is required which adapts to the changing brain signals. This means we have two adaptive systems, on the one hand the learning individual and on the other hand the adaptive classifier.

We need to combine experimental studies of animal and human nervous systems with numerical simulation of large-scale brain models, in order to answer open questions: How does the brain signals stimulate the muscles to move? How does the brain know about the correct position? How do neurons interact, and what is the role of the mutual adaptivity? [I^+03]

Izhikevich [I^+03,Izh07] developed a spiking neural network to simulate synaptic activity of different types of neurons. Based on this model, Chadderdon *et al.* [CNKL12] developed a method which simulates cortical activity and uses proprioception and spike time depended plasticity (STDP) to train the neural network to reach a given target angle. STDP is a biological process that re-weights connections between neurons in the brain based on spike-times. On the one hand, the network has a biologically realistic structure, but on the other hand the network only learns one movement (one target angle), not a static mapping. So the mapping needs to be relearned for every target angle and the mapping for the previous angle gets lost. This behavior does not appear to be realistic.

The aim of this thesis was the extension of the method of Chadderdon *et al.* [CNKL12] with a more biologically realistic learning-behavior, in order to lay the foundation for studying the mutual influence of both the adaptive brain and the adaptive classifier, in a biologically realistic environment. Both STDP and structural synaptic plasticity [CDM12] are used to train spiking neuronal network models of sensorimotor areas, in order that neuronal outputs navigate a simple prosthesis simulation to given target angles. For each model a static mapping will be learned that works for all target angles without the need for further learning, using only information about both proprioception and target angle as synaptic inputs, which should be more realistic than ongoing learning.

The following Chapter 2 covers all required biological, technical, and computational fundamentals, including the recording of brain signals, different types of neuron models, neural codings, and related works. In Chapter 3 an improved reimplementation of the method of Chadderdon *et al.* [CNKL12], is described. Chapter 4 covers all modifications in order to learn a static mapping for all target angles. Finally, a discussion and outlook is given in Chapter 5.

Chapter 2

Fundamentals

2.1 Biological Background

Both, extensors and flexors of a muscular system are mainly controlled by the brain's motor cortex. Flexion refers to a movement that decreases the angle between two bones at a joint, whereas extensions is a movement that increases the angle. In the following subsections, the cortex areas, the corresponding types of neurons and their signaling will be described in detail.

2.1.1 Motor Cortex

Among proprioceptive muscle reflexes (i.e. the patellar reflex), where a specific muscle movement is initiated by its own receptors (proprioceptors), muscles are mainly controlled by the brain. The motor cortex is a region of the cerebral cortex and can be further divided into several parts (Figure 2.1), which are involved in the planning, control, and execution of conscious movements. The most important ones will be explained in the following.

2.1.1.1 Primary Motor Cortex (M1)

The M1 contains large neurons, called Betz cells, which communicate through axons of the spinal cord with α-motor neurons and they are responsible for the actual perceived movement of the body. As for most brain activities, initiated signals of the left region controls the right side of the body and vice versa. An interesting characteristic of the M1 is the somatotopic arrangement, which means that neighbored regions of the body are also neighboured in this cortex (Figure 2.2). However, proportions are different, because some body areas are controlled more sensitively than others. Especially hands and mouth

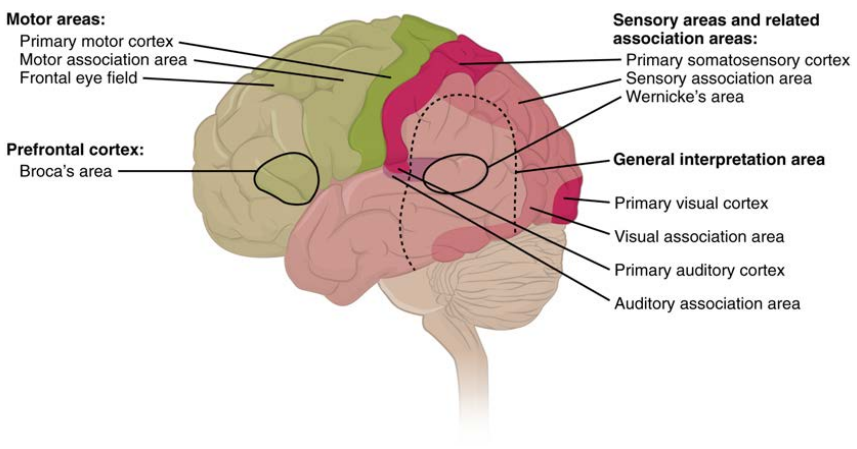


Figure 2.1: The cerebral cortex is containing three types of processing regions: primary, association, and integration areas. The primary cortical areas are where sensory information is initially processed, or where motor commands emerge to go to the brain stem or spinal cord. Association areas are adjacent to primary areas and further process the modality-specific input. Multimodal integration areas are found where the modality-specific regions meet; they can process multiple modalities together or different modalities on the basis of similar functions, such as spatial processing in vision or somatosensation. [Colst]

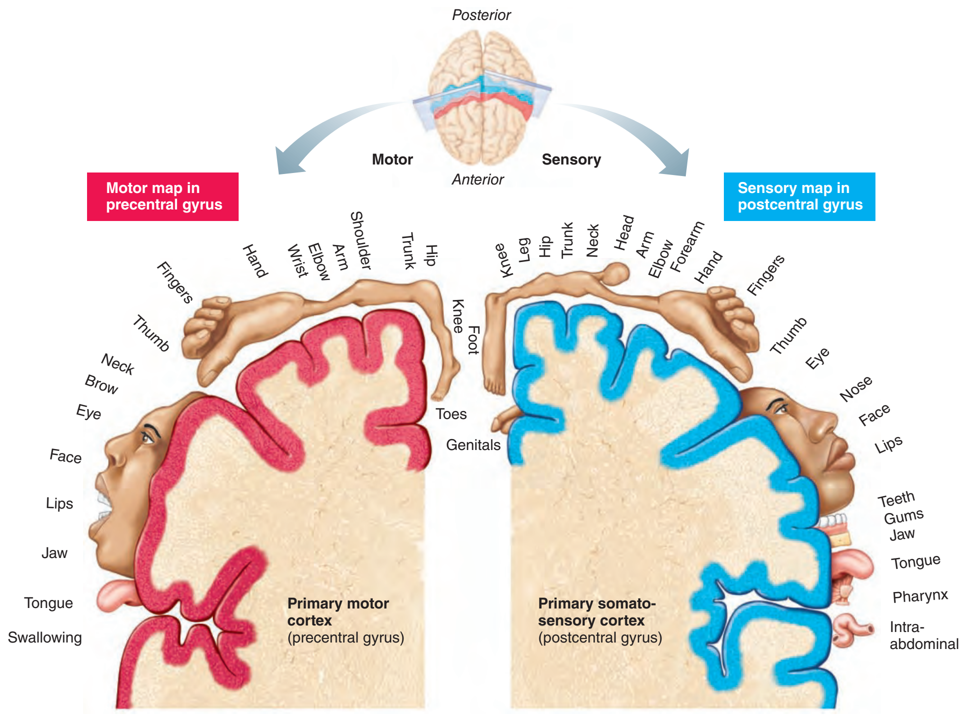


Figure 2.2: The motor/sensory homunculus. A cartoon representation of the motor/sensory homunculus arranged adjacent to the cortical region in which the processing takes place. [MH13]

muscles have fine motor skills, whereas the back has only gross sensory motor skills. [PAF+04]

2.1.1.2 Motor Association Cortex

The motor association cortex lies anterior to the M1 and is further divided in subareas.

The Premotor Cortex (PMC) is responsible for more complex tasks than M1. The exact functions of the PMC are diverse and not fully understood. Its upper motor neurons influence motor behavior both through extensive reciprocal connections with the M1, and directly via axons that project through the corticobulbar and corticospinal pathways to influence local circuit and lower motor neurons of the brainstem and spinal cord [PAF+04]. As in the primary motor area, many of the neurons in the lateral PMC fire most strongly in association with movements made in a specific direction. However, the neurons of the lateral PMC are especially important in conditional motor tasks and they seem to be particularly involved in the selection of movements based on external events (i.e. visual cues). The medial PMC also mediates the selection of movements, however, this region appears to be specialized for initiating movements specified by internal rather than external cues. [PAF+04]

The Supplementary Motor Area (SMA) is involved in programming complex sequences of movements and coordinating bilateral movements. Whereas the PMC appears to be involved in selecting motor programs based on visual stimuli and other external cues, the SMA appears to be involved in selecting movements based on remembered sequences of movements. [RLLS80]

2.1.2 Sensory Cortex

Sensory areas of the brain, occur in the parietal, insular, temporal, and occipital lobes. These areas are concerned with conscious awareness of sensation. In the following, two sensory cortices, which are involved in movement, will be explained.

2.1.2.1 Primary Somatosensory Cortex (PSC)

The PSC resides posterior to the M1. The cortex's neurons get their information from the sensory receptors in the skin and from proprioceptors in skeletal muscles, joints, and tendons. Therewith, the brain is able to detect differences in spatial location, this ability is called spatial discrimination [MH13]. Just as the M1, the PSC has a somatotopic arrangement, which is slightly different to the arrangement of the M1, as shown in figure 2.2. The amount of sensory cortex devoted to a particular body region is related to that region's sensitivity (amount of receptors), not its size [MH13].

2.1.2.2 Somatosensory Association Cortex (SAC)

The SAC lies posterior to the PSC and is well connected with it. The PSC relays sensory information to the SAC, in example information about temperature and pressure. With this, the SAC is able to determine the size, texture, and other properties of an object. [MH13]

The SAC also uses stored memories of past sensory experiences to determine known objects (for example to distinguish different coins). People with damage to this area could not recognize these objects without looking at them. [MH13]

2.1.3 Multimodal Association Areas

The above parts of the cortex have there own well defined function. But, many activities of the human body depend on a mixture of different information, in order to this, it is required that the different parts are able to communicate with each other [MH13]. Therefore it is not surprising, that the main part of the cortex consists of complexly connected multimodal association areas which receive inputs from multiple senses and send outputs to multiple areas.

2.1.4 Types of Neurons

The above areas of the cortex contains several types of neurons, which are classified depending on the function. In the context of body movement, the neurons are classified as motor neurons (efferent), sensory neurons (afferent), and interneurons.

2.1.4.1 Motor Neurons

Motor neurons are further divided into upper and lower motor neurons. Upper motor neurons (UMNs) originate either in the motor region of the cerebral cortex or in the brain stem and send their signals down to the lower motor neurons (LMNs) (figure 2.3, step ⑥-⑦).

An example for UMNs are Betz cells. As already mentioned, they are huge pyramidal cells, lie within the M1, and are the main effector for voluntary

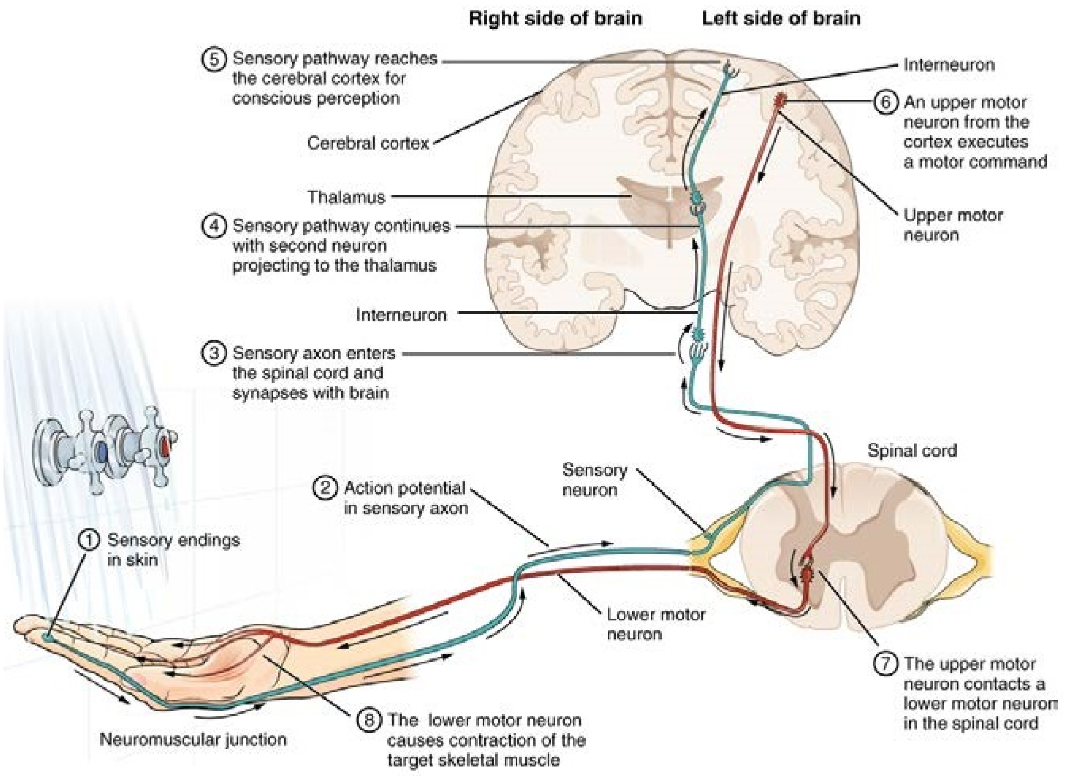

Right side of brain
Left side of brain
⑤ Sensory pathway reaches the cerebral cortex for conscious perception
Cerebral cortex
Thalamus
④ Sensory pathway continues with second neuron projecting to the thalamus
Interneuron
③ Sensory axon enters the spinal cord and synapses with brain
Interneuron
⑥ An upper motor neuron from the cortex executes a motor command
Upper motor neuron
Spinal cord
Sensory neuron
② Action potential in sensory axon
① Sensory endings in skin
Lower motor neuron
⑦ The upper motor neuron contacts a lower motor neuron in the spinal cord
⑧ The lower motor neuron causes contraction of the target skeletal muscle
Neuromuscular junction

Figure 2.3: Motor and sensory Pathways. ①-⑤ In case of a stimulus, the sensory receptors (i.e. thermoreceptors) send an action potential through a sensory neuron to the spinal cord, where the pathway continuous through other sensory neurons to the cerebral cortex. ⑥-⑧ An upper motor neuron, originating either in the motor region of the cerebral cortex or in the brain stem, send a signals down to a lower motor neuron which in turn causes contraction of a skeletal muscle. [Colst]

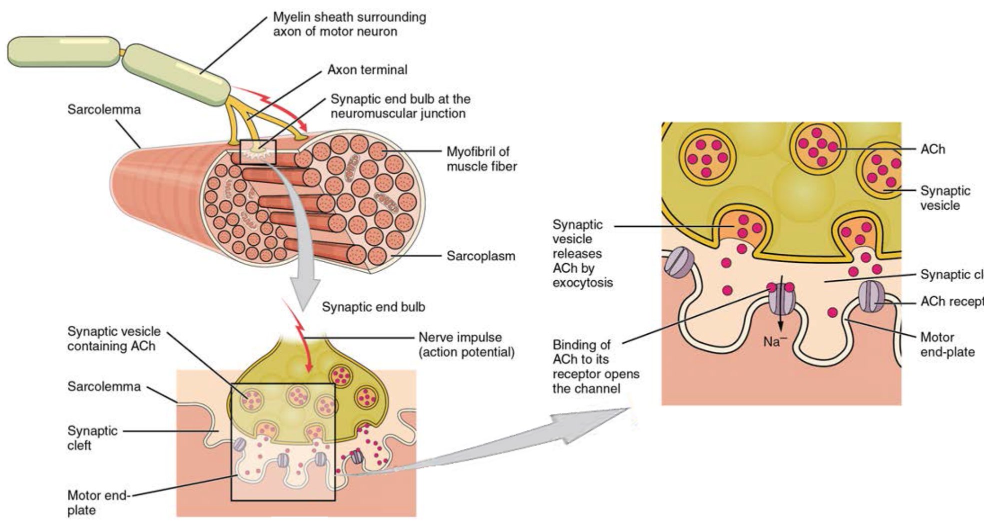

Myelin sheath surrounding axon of motor neuron
Axon terminal
Synaptic end bulb at the neuromuscular junction
Sarcolemma
Myofibril of muscle fiber
Sarcoplasm
Synaptic end bulb
Synaptic vesicle containing ACh
Nerve impulse (action potential)
Sarcolemma
Synaptic cleft
Motor end-plate
ACh
Synaptic vesicle
Synaptic vesicle releases ACh by exocytosis
Synaptic cleft
ACh receptor
Binding of ACh to its receptor opens the channel
Na⁻
Motor end-plate

Figure 2.4: When a nerve impulse reaches a neuromuscular junction, acetylcholine (ACh) is released. Upon binding to sarcolemma receptors, ACh causes a change in sarcolemma permeability leading to a change in membrane potential. [Colst]

movement. They send their axons down through the spinal cord (corticospinal tract) to the LMNs (somatic motor neurons) [PAF$^+$04]. LMNs are classified by the type of muscle fiber they innervate. There are alpha (α), beta (β), and gamma (γ) motor neurons. The α motor neurons innervate skeletal muscle fibers, therefore, they are responsible for skeletal movement (figure 2.3, step ⑧. Each muscle fiber and the innervating α motor neuron is called motor unit. [PAF$^+$04]

The connection within a motor unit is a neuromuscular junction (NMJ), its function is to transmit signals quickly and reliably from the neuron to the corresponding muscle fiber (see figure 2.4). The synaptic vesicles contains the neurotransmitter acetylcholine (ACh), which will be released into the synaptic cleft where it binds to the nicotinic acetylcholine receptors (nAChRs) on the plasma membrane of the muscle fiber, this leads to a depolarization of the motor end plate and spreads across the surface of the muscle fiber, which in turn results in a muscle contraction [HE11]. The effect is always excitatory, and if stimulation reaches threshold, the muscle fibers contract, whereas muscle relaxation is obtained only by inhibition of the motor neuron itself (see 2.1.4.3) [MH13].

2.1.4.2 Sensory Neurons

Sensory neurons are responsible for the sensation, they transmit sensory information like sound, sight, and feelings (light as well as painful stimuli).

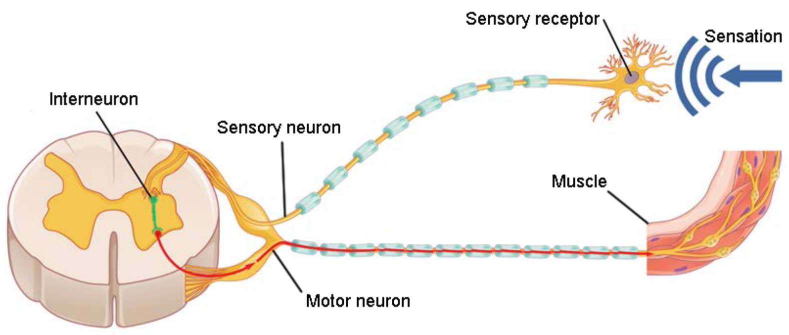


Figure 2.5: A simple reflex arc. Sensory receptors detect a change in the internal or external environment and project to the CNS stimulating the motor neuron via an interneuron. [Colst]

The somatosensory system is a subclass of the sensory system and receives inputs from different sensory receptors, which are classified by the source of the stimulus: exteroceptors (provide information about external environment), interoceptors (provide information about the events in the viscera), and proprioceptors (provide information about the position of our body in space) [MH13]. In addition, sensory receptors are also classified by the type of stimulus: mechanoreceptors (touch, pressure, vibration, and stretch), thermoreceptors (temperature changes), photoreceptors (light), chemoreceptors (smelling/tasting molecules), and nociceptors (potentially damaging stimuli that result in pain: extreme heat, etc.).

Proprioceptors perceive changes in joint angle, muscle length, and muscle tension. The changes in muscle length are detected by the proprioceptor/mechanoreceptor, called muscle spindle. The Golgi tendon organ is another proprioceptor/mechanoreceptor that detects changes in muscle tension. The information about muscle or tendon stretch is transmitted to the cerebellum. The information is used to determine the position of the limb in space and to coordinate skeletal muscle activity. [MH13]

In figure 2.3, steps ①-⑤ illustrate a sensory pathway using the example of testing water temperature with the hand.

2.1.4.3 Interneurons

Interneurons form connections between other neurons. For example, they connect motor and sensory neurons in neural pathways and transmit signals through the central nervous system (CNS) pathways. They make up over 99% of the neurons of the body, including most of those in the CNS [MH13]. Figure 2.5 shows the schematic of the linkage of sensory and motor neurons via interneurons in the case of an simple reflex arc.

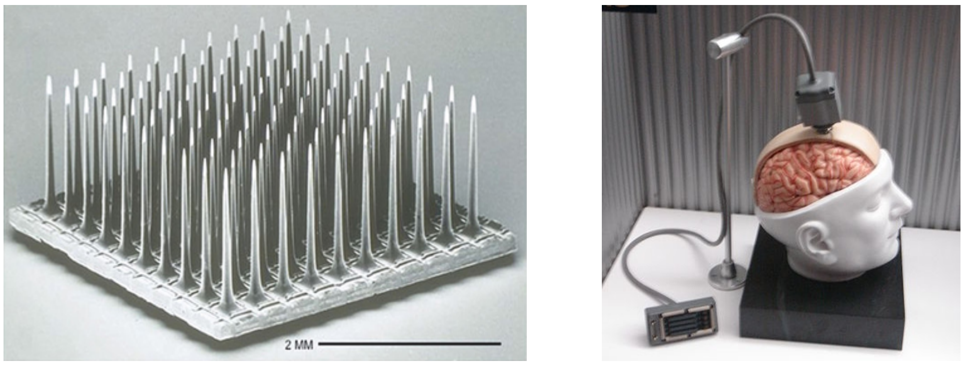


Figure 2.6: Intracortical sensor and placement. (a) The electrodes of a tiny electrode array [FGH+14] record the brain signals from the cortex's neurons and forwards them via (b) a percutaneous pedestal to an amplifier [Wic14].

Interneurons in the CNS are typically inhibitory, and use the neurotransmitter GABA or glycine. An example for an inhibitory interneuron is the Renshaw cell [Ren46], they are located in the gray matter of the spinal cord and are associated with α motor neurons. They perform a negative feedback mechanism: the stronger the innervation of a muscle, the stronger is the inhibitory negative feedback, which in turn lead to inhibition of the innervation. [AF07]

2.2 Technical Background

2.2.1 Recording Brain Activity

A brain-computer interface (BCI) is used as a communication pathway between the brain and an external device. One part of a BCI is recording of the brain activity. There are invasive as well as non-invasive recording methods, where the electrodes of an invasive BCI will be implanted directly in the brain and those of non-invasive BCIs will be placed on the surface of the scalp. These two methods will be described in the following subsections.

2.2.1.1 Invasive

Invasive BCIs make the use of electrocorticography (ECoG), where a tiny microelectrode array (see figure 2.6(a)) containing hundreds of electrodes is placed in or on the surface of the desired cortex. The electrodes record the electrical signals from the cortex's neurons and forward them via a percutaneous pedestal to an amplifier (figure 2.6(b)). The amplifier is connected to a

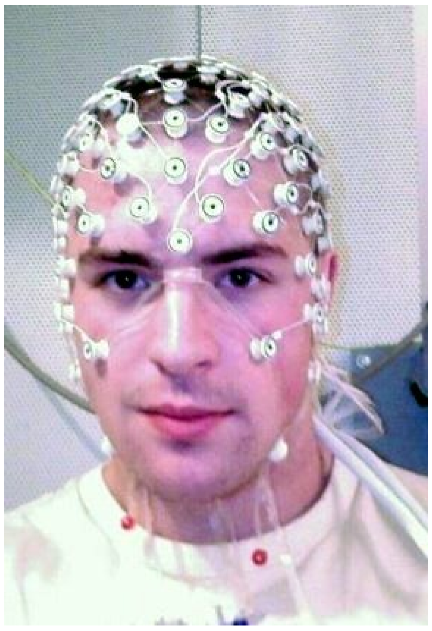


Figure 2.7: An portable non-invasive EEG cap. [Mye14]

computer that translates the signals and induces specific actions. The number of cortical implants can vary, the more brain areas are covered the finer the control could be, but also the more signals have to be decoded [LN06].

Recent BCIs are output-only, meaning that they can be used to control a prosthesis, but they do not receive feedback like proprioception, touch or pain. For this, signals has to be send to the brain's sensory cortex, which is yet quite challenging, because so little is known about how it works. Theoretically, the signals could be recorded from the nerves running from the limb into the spinal cord, from the spinal cord itself, or from the brain's thalamus, where incoming sensory signals are integrated. [Abb06]

The advantage of invasive BCIs is that they have the largest signal-to-noise ratio, but it is risky to implant the electrode into the brain, because this may lead to infection or permanent tissue damage.

2.2.1.2 Non-Invasive

Non-invasive brain-computer interfaces make primarily use of electroencephalography (EEGs). A cap with electrodes is placed on the head of the patient (figure 2.7), whereby the number of electrodes can vary. It must be observed that all electrodes are placed at the correct position to get the desired signals. Despite having the advantage of not expose the patient to the risks of brain surgery, EEG-based techniques only provide a poor signal resolution because the skull dampens signals. Because of this limited capacity, the non-invasive BCI might not be sufficient to control the movements of an prosthesis that has multiple degrees of freedom, however, research in this field indicated that it offer some practical benefit, such as cursor control, communication, computer operation and wheelchair control [LN06].

2.2.2 Brain Waves

Usually, multiple electrodes are used and the recorded information (voltage) is represented for each single electrode in a time-voltage-plot. The recorded neuronal activities are unique for each individual and change with age, sensory stimuli, brain disease, and the chemical state of the body, however, they can be grouped into different frequency domains [MH13]:

Alpha waves have low-amplitude, synchronous waves and are relatively regular and rhythmic. In most cases, they indicate that the brain is in an ”idling”-state. They fire with a frequency of about 8-13 Hz. An alpha-wave-like variant called mu (μ) can be found over the motor cortex that is reduced with movement, or the intention to move [Ste05]. Beta waves have a higher frequency (14-30 Hz) and are also rhythmic, but less regular than alpha waves. They occur in the case of mental alertness as well as when focusing on something. Theta waves are common in children, but uncommon in awake adults. They are still more irregular and fire with a frequency of 4-7 Hz. Delta waves are high-amplitude waves and occur in the state of deep sleep with a frequency of 4 Hz or less. The occurrence in awake adults indicates a brain damage.

2.2.3 Mutual Adaptivity

The development of classifiers for recorded neural activities must consider the problem that neural activity changes over time, because of the plasticity (adaption) of the brain, hence, classifiers are required to adapt to changing brain signals.

But that means we have two adaptive systems, on the one hand the learning individual and on the other hand the adaptive classifier. To study the mutual influence on each other, we have to use biologically realistic simulations. This area is not yet well studied, there are just a few papers [CNG^{+}11, SRB12].

2.2.4 Motor Prosthesis

Motor prosthesis can generally be categorized into two types: body-powered and externally-powered prostheses. Body-powered prostheses use physical movements, such as moving the shoulder, to control the artificial limb. While

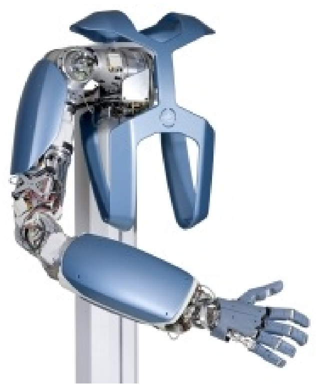


Figure 2.8: Anthropomorphic DLR hand arm system with 26 degrees of freedom. [Deu14]

they are independent of external power supplies, they demand muscle power from the patient and merely have a moderate functionality. Externally-powered artificial limbs are an attempt to solve this physical exertion through using electricity and BCIs to control movements, as described above.

There are many design constraints for externally-powered prosthesis, which have to be minded. At first, the appearance is a crucial factor, the artificial limb should look optimally like the natural limb, including weight and size, because if the prosthesis is too heavy in order that the patient is not able to wear it, then it is quite useless for everyday life. Figure 2.8 shows the anthropomorphic hand arm system of DLR with 26 degrees of freedom.

2.3 Computational Background

2.3.1 Biological Neuron Models

As we know, the brain is quite complex and not fully explored yet. The human brain contains billions of neurons, latest findings talk about that a adult male human brain contains on average 86.1 ± 8.1 billion neurons [ACG+09]. It is obvious, that current computers are unable to simulate such huge and complex models in real-time, especially if the model should be as realistic as possible. So we have to find a trade-off between computational simplicity and biological correctness. At least, the model must be able to produce firing patterns of real biological neurons. Until today several models have been developed to simulate neuronal behavior.

One of the first models is the integrate-and-fire model by Louis Lapicque: for each neuron an input current can be applied, resulting in an increasing membrane voltage until it reaches a constant threshold and point out a spike, followed by a reset to the resting potential [Abb99]. This model is computationally effective, but the model is unrealistically simple and incapable of producing different firing patterns of cortical neurons [I+03].

In contrast, the Hodgkin–Huxley model by Alan Lloyd Hodgkin and Andrew Huxley [HH52] consists of a set of nonlinear differential equations to simulate cell membranes and the different ion channels to produce action potentials. Even though the model is biologically accurate, it is not the model of choice, because it is only possible to simulate a handful of neurons in realtime.

Izhikevich [I+03] developed a simple spiking model that is as computationally efficient as the integrate-and-fire model and at the same time as biologically plausible as the Hodgkin–Huxley model. The model reproduces spiking and bursting behaviors of known types of cortical neurons, using only four parameters and a small number of neurons. This model will be explained in detail, because it is used as the underlying model of this work.

2.3.1.1 Simple Spiking Model by Izhikevich

Izhikevich [I+03] uses bifurcation methods [IM08], which allow to reduce the complexity of Hodgkin–Huxley neuronal models to two-dimensional systems of time-depending ordinary differential equations:

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{2.1}$$

$$u' = a(bv - u) \tag{2.2}$$

$$\text{after-spike reset: if } v \geq 30\text{mV, then } \begin{cases} v \longleftarrow c \\ u \longleftarrow u + d \end{cases} \tag{2.3}$$

where v represents the membrane potential and u represents the membrane recovery variable. The latter provides negative feedback to v and stands for both the activation of K^+ and the inactivation of Na^+ ionic currents. The variable I defines the synaptic input currents. The choice of dimensionless parameters a, b, c, and d leads to various intrinsic firing patterns.

The parameter a is the scaling factor for u, where greater values result in faster recovery. The parameter b is the sensitivity of u to the sub-threshold neuronal voltage fluctuations of v. The greater b, the stronger v and u are coupled resulting in possible sub-threshold membrane potential oscillations and low-threshold spiking. The after-spike reset value of v and u can be defined by the parameters c and d, respectively. Typical values are $a = 0.02$, $b = 0.2$, $c = -65$, and $d = 2$.

The first three terms of equation 2.1 resulted from fitting the spike initiation dynamics of a cortical neuron, in a way that the time has *ms* scale and v has mV scale.

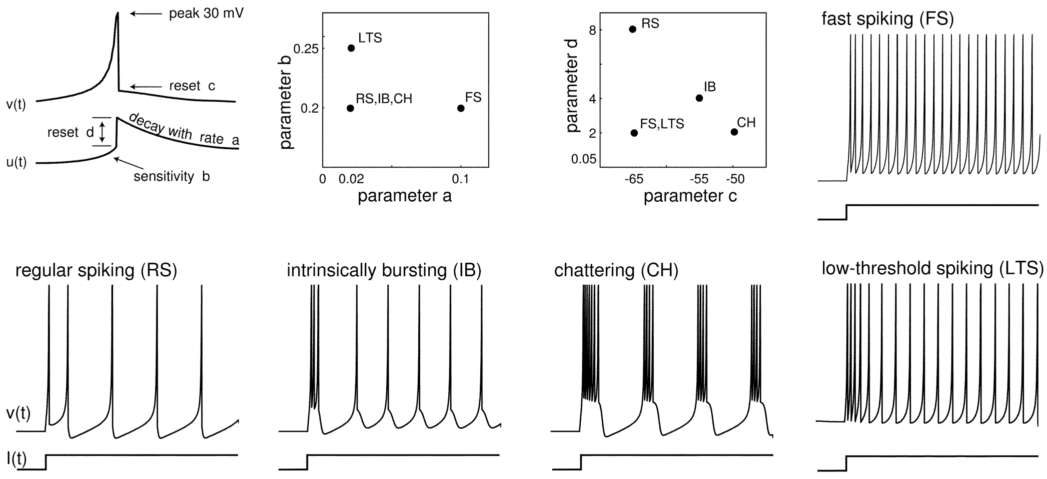


Figure 2.9: Known types of neurons correspond to different values of the parameters a, b, c, d in the model described by the equations (2.1)-(2.3). RS, IB, and CH are cortical excitatory neurons. FS and LTS are cortical inhibitory interneurons. Each inset shows a voltage response of the model neuron to a step of dc-current $I = 10$ (bottom). Time resolution is 0.1 ms. [I$^+$03]

Depending on the firing pattern, mammalian cortical cells are divided into several neuronal classes [CG90, GM96, GBC99], these can be simulated by varying above values for the parameter a, b, c, and d. Excitatory cortical cells are classified in regular spiking ($(c, d) = (-65, 8)$), intrinsically bursting ($(c, d) = (-55, 4)$), and chattering neurons ($(c, d) = (-50, 2)$). Inhibitory cortical cells are divided into fast spiking ($a = 0.1$) and low-threshold spiking ($b = 0.25$) neurons. Figure 2.9 illustrates the resulting patterns.

2.3.2 Spike-Timing-Dependent Plasticity

At this point, we got a model to produce the different cortical firing patterns. In order to train a model, we could use the biological process, called spike-timing-dependent plasticity (STDP), that adjusts the connection strength between neurons using the relative timing of output and input spikes of a specific neuron. If an input spike occur immediately before that neuron's output spike, then that corresponding input is strengthened.

Assumed, a specific body movement has a specific firing pattern of recorded neurons. Now we want to strengthen (reward) all connections involved in the correct pattern. The problem, known as distal reward problem, is that reward comes delayed to the input spiking event. Consequently, the neuronal model has to know what firing patterns of what neurons are responsible for the reward if on the one hand the patterns are no longer there when the reward arrives and on the other hand all neurons and synapses are active during the waiting period to the reward [Izh07].

Izhikevich has solved this problem using a linkage of STDP and dopamine signaling. Dopamine modulation of STDP can reinforce firing patterns occurring on a millisecond timescale even when they are followed by reward that is delayed by seconds. This relies on the existence of slow synaptic processes that act as "synaptic eligibility traces". [Izh07]

2.3.3 Neural Coding

Neural codings describe the relationship between the stimulus and the individual or ensemble neuronal responses [BKM04]. Today, there are several coding schemes based one different theories.

2.3.3.1 Rate Coding

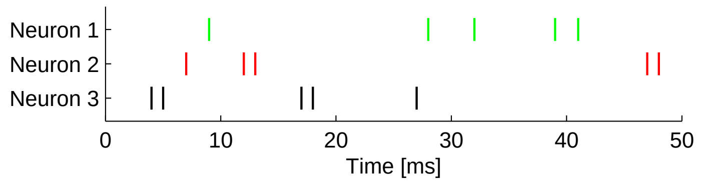
Rate coding is based on the assumption that information about a stimulus is contained in the neuron's firing rate. In example, if we want to encode an integer variable $x = 1, 2, 3, \ldots$, for this we let the neuron fire $1, 2, 3, \ldots$ times in a well-defined time-window (figure 2.10(a)).

Rate coding can be found in the muscle spindle, there are differences between spindle firing rate in imposed and actively reproduced movements, where the firing rate was usually higher during periods of reproduced movements when the muscle was relatively short whereas it was identical when the muscle was relatively long [VaF90].

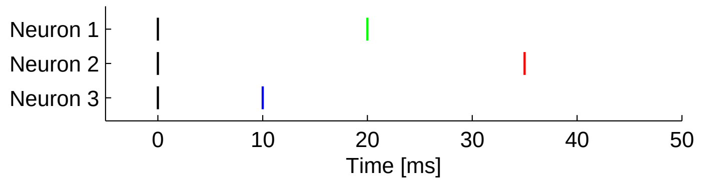
2.3.3.2 Temporal Coding

Temporal coding is based on the assumption that information about a stimulus is contained in the precise spike timing of a neuron, within a well-defined time-window or after an initial stimulus. In example, if we want to encode an integer variable $x \in [1, 50]$ in a 50 ms time window, we let the neuron fire at time $t = x$, after an initializing spike at time t_0 (figure 2.10(b)).

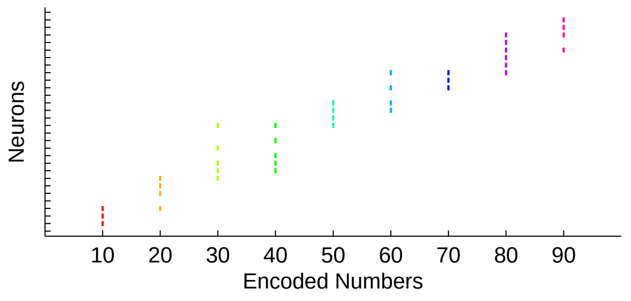
Temporal coding can be found in the visual cortex [EKK$^+$92] as well as for other sensory information [LCdGLC99]. It also seems to appear among populations of motor cortex neurons as they become engaged in the production of various motor behaviors [DSHG98].



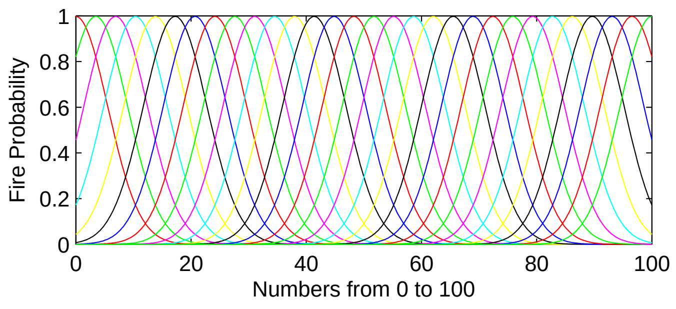
(a) Rate Coding



(b) Temporal Coding



(c) Population Coding



(d) Fire Probability of Population Coding

Figure 2.10: Examples of neural codings. (a) Rate coding: each neuron has a different firing pattern, but all of them fire 5 times within the 50 ms time-window, means they encode the same information. (b) Temporal Coding: if we want to encode an integer variable $x \in [1, 50]$ in a 50 ms time window, we could let the neuron fire at time $t = x$, after an initializing spike $t_0 = 0ms$ (black bars). Shown are examples for $x \in \{10, 20, 35\}$. (c) Population coding: if we want to encode an integer variable $x \in [0, 100]$, we have to use a probability functions, that define fire probabilities for each single neuron. Shown are nine numbers encoded with 30 neurons according to the probability functions shown in (d), where each single curve represents the probabilities for a single neuron.

2.3.3.3 Population Coding

Population coding is based on the assumption that information about a stimulus is contained in the joint activities of numerous neurons. For this, a probability function is used for each neuron, defining the fire probability according to the encoded number. In example, if we want to encode an integer variable $x \in [0, 100]$ using n neurons, we could use the normal probability density function

$$f(x|\sigma, \mu) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma}} \ . \tag{2.4}$$

For each neuron i we define the probability

$$p_i(x) = 4 \cdot f\left(\frac{x}{100}(n-1)|1.6, i-1\right) \ , \tag{2.5}$$

resulting in distributions shown in figure 2.10(d). Different values for x result in different fire probabilities for each neuron. Examples are shown in figure 2.10(c).

Georgopoulos *et al.* found when individual cells are represented as vectors that make weighted contributions along the axis of their preferred direction (according to activity changes during the movement) the resulting vector sum of all cell vectors (population vector) was in a direction congruent with the direction of movement [GSK86]. This implies that the direction of a movement can be uniquely predicted by the action of a population of motor cortical neurons.

2.3.4 Reinforcement Learning

Reinforcement learning (RL) methods do not, contrary to teacher-supervised learning methods, require a known desired output representation to match against the model's current output, however, they do offer some feedback regarding fitness of the behavior [CNKL12].

The Perception-Action-Reward Cycle (PARC) [STC$^+$11] is a method for motor RL. This system is based on an actor-critic method, whereas the actor maps the signals (perceptions) to actions, and the critic evaluates those actions and sends reward/punishment feedback to the actor. Figure 2.11 illustrates the PARC as used by Chadderdon *et al.* [CNKL12].

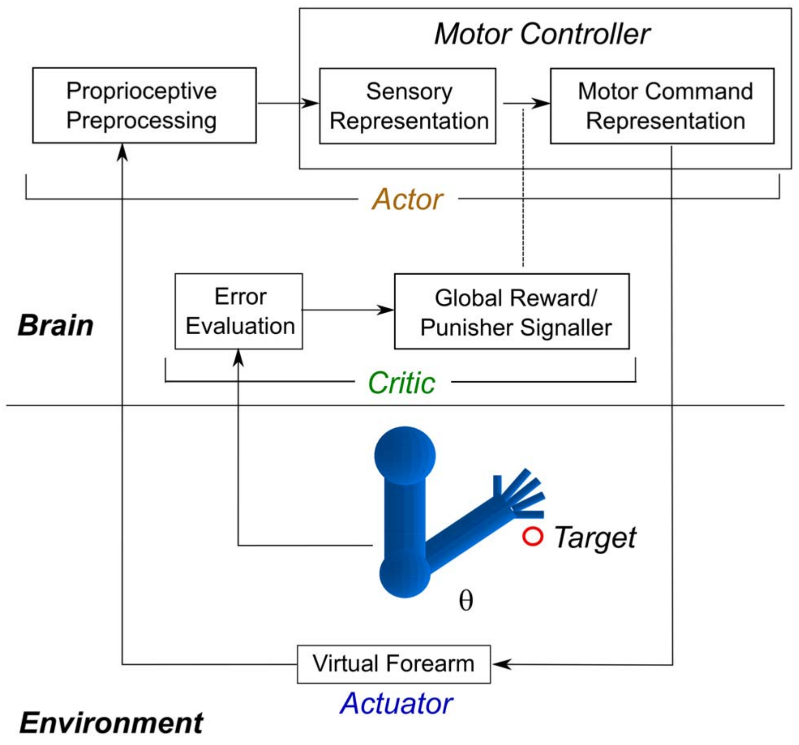


Figure 2.11: Perception-action-reward cycle. The actor maps proprioceptive input (muscle lengths) into an arm configuration representation and is trained by the critic which evaluates error and sends reward/punishment feedback to the actor [CNKL12].

2.3.5 Motor Babbling

Brain activity in M1 can be noticed, even is somebody do not intend to perform a movement. These signals, called neuronal noise, originate from several sources [Koc04].

In machine learning, the noise is called *motor babbling* and is processed by repeatedly performing random signals. Motor babbling plays an important role for reinforcement learning, the naive actor must produce some actions in order that the critic is able to feedback reward/punishment [CNKL12].

2.3.6 State of the Art: Motor Learning

Bouganis and Shanahan [BS10] introduced a feed-forward spiking neural network (based on Izhikevich's simple spiking model) that autonomously learns to control a 4 DOFs robotic arm based on STDP. The network is organized into seven input layers (1200 neurons each) and four output layers (800 neurons each), whereas each neuron within an input layer is connected to each neuron

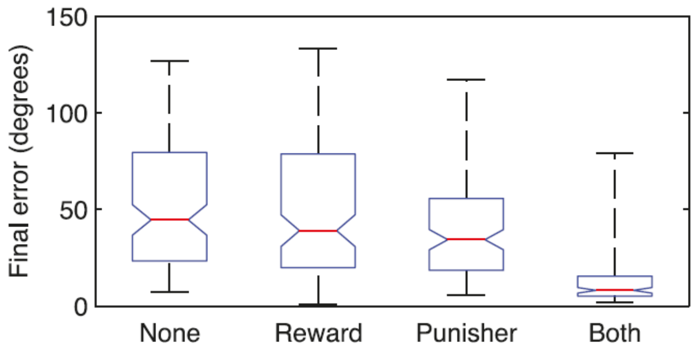


Figure 2.12: Performance across learning conditions and random wiring seeds. End-of-trial errors under different learning conditions: no learning, reward-only, punisher-only, reward-and-punisher; $N = 125$ for each: 5 target angles $(0°, 35°, 75°, 105°, 135°) \times 5$ random wirings $\times 5$ random babble noise inputs [CNKL12].

within each output layer. The input layers encode proprioceptive information (4 joint angles of the robotic arm, discretized into bins of 5°) and the direction in space (target) where the robotic arm should move at the next time step. Given a set of initial arm poses, the network was able to learn 26 movements of the end-effector from its original position.

Chadderdon *et al.* [CNKL12] developed a closed-loop method by simulating the motor cortex also based on the simple spiking model of Izhikevich [I+03]. An STDP reinforcement algorithm was used to train the model to move a simulated forearm with one degree of freedom, based on proprioceptive information which is sent to simulated sensory neurons. The neuron model has a biological realistic structure, but the learning process has to be enabled during each movement, therefore, it is not possible to statically train the neuron model, which does not appear to be realistic. They found that using both reward and punishment together outperforms the individual application of reward or punishment, respectively (Figure. 2.12). This method was extended to two degrees of freedom by Neymotin *et al.* [NCK+13].

2.3.7 Related Work: Mutual Adaption

As mentioned, the mutual influence of two adaptive systems on each other, is not yet well studied. Recent methods will be shortly described in the following.

Spüler *et al.* [SRB12] developed a method that uses genetic algorithms to study the mutual interaction. They simulated an adapting BCI user (simulated EEG) and allow to test and compare different adaptive algorithms considering the co-adaptivity between BCI and user.

Cunningham *et al.* developed a closed-loop simulator to study feedback effects in simulation-based validation methods for testing of decoding algorithms. For this, a healthy human subject or a monkey makes real arm reaches in a 3-D reaching environment, whereas the kinematics were recorded and used to generate synthetic neural activity which in turn is decoded into physical reaching behavior. The reaching behavior is shown to the subject in a 3-D visual environment, which allows the subject to bring to bear all of his/her online, closed-loop control strategies to drive a desired reach [CNG$^+$11]. Furthermore, they also used real neural activity of the monkey's brain.

Chapter 3

Ongoing Learning Approach

This chapter describes a simplified reimplementation of the spiking neuronal model of Chadderdon *et al.* [CNKL12], with some improvements. The neuron model, its equations and parameters, the system design including the learning process will be described in detail, followed by the results.

3.1 Neuron Model

Each individual neuron is modeled as a dynamic unit and is labeled as excitatory or inhibitory. Each cell has a membrane potential variable (V_m) describing the current voltage state, a resting membrane potential (V_r) describing the baseline voltage, a spiking threshold (V_t) describing the voltage at which the cell fires an action potential, and a noise input variable (V_n) describing the motor babble (see 2.3.5) input.

The cells are classified as excitatory motor (EM), inhibitory motor (IM), excitatory sensory (ES), or inhibitory sensory (IS). At initiation, V_m is set to V_r for each cell (see 3.3). The underlying step-wise processing is done by the simple spiking model equations described in section 2.3.1.1 whereas $v = V_m$. The parameter c is equivalent to V_r (inhibitory: -63mV, excitatory: -65mV), and the values for parameters a, b, and d are shown in Table 3.1 and are recommended by Izhikevich [I^{+}03]. To achieve heterogeneity, in order that different neurons have different dynamics, a uniformly distributed random variable r_i is introduced for each neuron i. If a neuron i is excitatory than $r_i = 0$ corresponds to regular spiking and $r_i = 1$ leads to a chattering behavior. Otherwise, if a neuron i is inhibitory than different values for r_i shifts the neuron's dynamic between fast spiking and low-threshold spiking.

The noise input V_n is represented by random stimulation of AMPA synapses for all EM, IM, and IS cells. The noise stimulation occurs with a rate of

Table 3.1: Used parameters for the simple spiking model as recommended by Izhikevich [I+03]. A description of their meaning can be found in section 2.3.1.1. Values for excitatory and inhibitory neurons differ from each other. A uniformly distributed random variable $r_i \in [0, 1]$ is used for each single neuron i.

Parameter	Excitatory Neurons	Inhibitory Neurons
a	0.02	$0.02 + 0.08 \cdot r_i$
b	0.2	$0.25 - 0.05 \cdot r_i$
d	$8 - 6 \ \cdot r_i^2$	2

300 Hz in order to conform with the AMPA stimulation of Chadderdon *et al.*. The calculation of synaptic noise inputs V_n depends on the current membrane potential V_m

$$dV_n = w_n \left(1 - \frac{V_m}{E_n}\right) \tag{3.1}$$

where w_n is the synaptic noise weight and E_n the reversal potential relative to V_r: $E_n(AMPA) = 65mV$.

3.2 System Design

The system consists of an virtual brain (network) with the underlying neuron model as descried previously and an virtual prosthesis with one degree of freedom. The interaction between the prosthesis and the neuron model is illustrated in figure 3.1.

The sensory part of the network is composed of 96 ES cells and 32 IS cells. The motor part is composed of 48 EM cells, and 32 IM cells, whereas the EM population is split equally into two groups, one for extensor muscle and one for flexor muscle. Additionally, there are 48 proprioceptive (P) cells, representing the muscle length (muscle spindle) and therefore the joint angle of the virtual prosthesis.

The connections between the cells are generated probabilistically and their initial weights w_0 vary depending on the involved cell types (Table 3.2). Each P cell got its own normal distributed stimuli (population coding) depending on the current joint angle, they have no further input connections from other cells.

The noise weights w_n and initial cell's connection weights w_0 were adjusted, such that the average spiking rates f_a (motor babble) are $f_a(ES) = 0.4$,

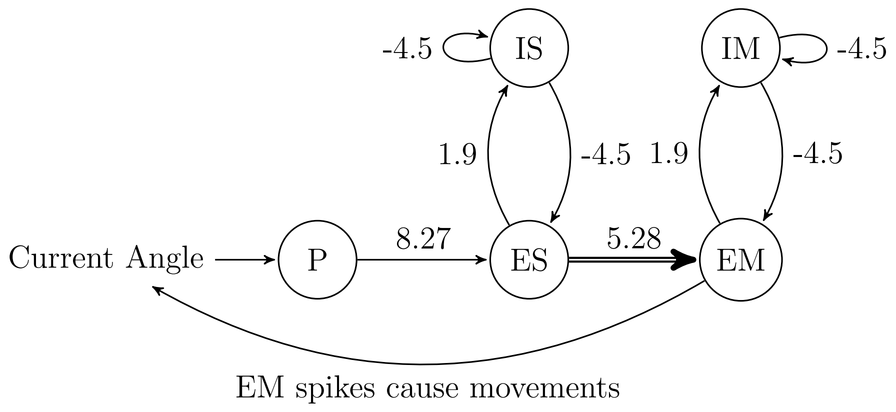


Figure 3.1: Weighted connection graph of ongoing learning approach. Shown are different cell types and the connections between them, including connection weights. P cells are stimulated by encoded current angle of the prosthesis (population coding). The thick edge means that ES→EM synapses will be rewarded/punished.

Table 3.2: Connection Probabilities and Connection Weights. The rows represent the presynaptic cells and the columns represent the postsynaptic cells. A negative weight value means a negative contribution (inhibitory). The cell types are: excitatory motor (EM), inhibitory motor (IM), excitatory sensory (ES), inhibitory sensory (IS), and proprioceptive (P). Because P cells get external stimuli, they have no input connections from other cells. A dash (-) means that no connections exist.

	EM		IM		ES		IS	
	prob.	weight	prob.	weight	prob.	weight	prob.	weight
EM	-	-	0.43	1.9	-	-	-	-
IM	0.44	-4.5	0.62	-4.5	-	-	-	-
ES	0.08	5.28	-	-	-	-	0.43	1.9
IS	-	-	-	-	0.44	-4.5	0.62	-4.5
P	-	-	-	-	0.10	8.27	-	-

$f_a(IS) = 4.4$, $f_a(EM) = 0.5$, $f_a(IM) = 4.3$ in the absence of learning: for EM cells $w_n(AMPA) = 2.2$ and for both IM and IS cells $w_n(AMPA) = 1.5$.

The virtual prosthesis' current joint angle $\theta \in [0, 135]$ varies on its interval, whereas $\theta = 0°$ stands for a straight arm position and $\theta = 135°$ stands for fully flexed, respectively. EM spikes cause a change of the current θ, plus-or-minus depending on extension or flexion (see 3.3).

For the purpose of reinforcement learning, each synapse will be eligibility tagged if a post-synaptic spike followed a pre-synaptic spike for the corresponding synapse within a time window τ. Eligibility ceased after a same sized time window τ. The difference between both the latest distance $\Delta\theta_t$ to the target angle and the previous distance $\Delta\theta_{prev}$ is used as error evaluation

$$\Delta\theta_t = |\theta_{\mathrm{t}} - \theta_{\mathrm{target}}| \tag{3.2}$$

$$\Delta\theta_{\mathrm{prev}} = \left|\sum_{i=1}^{pa} (\theta_{t-i})\,/pa - \theta_{\mathrm{target}}\right| \tag{3.3}$$

$$\text{reward/punishment?} : \begin{cases} \text{send reward} & \Delta\theta_t < \Delta\theta_{\mathrm{prev}} \\ \text{send punishment} & \Delta\theta_t > \Delta\theta_{\mathrm{prev}} \end{cases} \tag{3.4}$$

where pa is a parameter that allows to define the number of previous angles, whose median should be used for distance calculation. Reward/punishment is sent to all eligibility tagged synapses. Not the connection weight w will be adjusted directly, but weight scale factors w_s are used for each connection

$$w(t) = w_0 w_s(t) \tag{3.5}$$

$$w_s(t) = w_s(t-1) + \Delta w_s \tag{3.6}$$

$$\Delta w_s = \begin{cases} 1 - \dfrac{w_s(t-1) \cdot w_e}{w_s^{\max}} & \text{in case of reward} \\ -\dfrac{w_s(t-1) \cdot w_e}{w_s^{\max}} & \text{in case of punishment} \end{cases} \tag{3.7}$$

$$w_e = \begin{cases} 1 & \text{static} \\ \dfrac{\tau_{\mathrm{remaining}}}{20} & \text{dynamic} \end{cases} \tag{3.8}$$

where $w_s^{\max}$ is the maximum weight scale factor and w_e a scaling factor that can be static or dynamic relative to the remaining time $\tau_{remaining}$ until eligibility is ceased. A dynamic behavior of w_e means that w_s has more influence if the pre/post-synaptic spikes are closer together. w_s is initialized to 1.0 for all synapses and is forced to lie within the interval $[0, w_s^{\max}]$, whereas $w_s^{\max} = 5$ for all models.

3.3 Implementation

The model was completely implemented in MATLAB 8.3.0.532 (R2014a) [MAT14]. Pseudocode in Algorithm 1 should help to understand the step-wise processing.

At initialization one has to define the total *runtime* in ms (Line 8). The following three lines define, for each neuron, the simple spiking model parameters, the initial membrane voltage (resting potential), and the initial recovery variable. After the initialization, the step-wise processing starts in a millisecond scale. At the beginning of each iteration, the latest $fired$ synapses are stored (Line 13). Next, we memorize each pre-synaptic spike for τ ms, in order to detect a following post-synaptic spike. If this is the case, the synapse will be eligibility *tagged* for τ ms.

As mentioned, EM spikes are responsible for changes of the joint angle, therefore, we have to inform the virtual prosthesis about it, separated by both extension and flexion (Lines 16-18). Angle updates are provided at 50 ms intervals (Line 27), based on the differences of the extension and flexion EM spikes, which will be summed up in a 50 ms time window that began 50 ms prior to the movement event in order to have a 50 ms network-to-muscle propagation (Lines 26 and 36). The parameter *ppd* defines the needed number of spikes that causes 1° of angle change (Line 28). The larger the value, the slower the movement, which should result in a smaller deviation if the target angle has been reached, but it also should lead to an increase of time needed to reach the target angle.

Reinforcement also occurs each 50 ms, if the new distance $\Delta\theta_t$ to the target angle θ_{target} is shrunk, compared to the previous distance $\Delta\theta_{\text{prev}}$ (Line 30), reward will be sent to all eligibility tagged ES→EM synapses (Line 32), otherwise, if the distance is enlarged then punishment will be sent (Line 34). The idea behind the *pa* parameter is, that reward/punishment doesn't occur if $\Delta\theta_t$ equals $\Delta\theta_{\text{prev}}$ which in turn is a problem if $\theta_{t-1} = 0°$ (minimum angle) or $\theta_{t-1} = 135°$ (maximum angle), because if θ_t theoretically exceed these limits, it will be fixed at the same angle, and therefore no punishment occurs if $pa = 1$ (under the condition that $\theta_{\text{target}} \neq \theta_{t-1}$). If in the same case, $pa > 1$, than multiple previous angles will be included in the distance calculation of $\Delta\theta_{\text{prev}}$, increasing the chance that punishment will occur. Furthermore, it could help to avoid "excessive" reward, therefore, it could lead to smaller overall deviation. Assume the state, $\theta_{target} = 60°$, $\theta_{t-2} = 60°$, $\theta_{t-1} = 50°$, $\theta_t = 70°$, here, if $pa = 1$ reward will be sent, although, it's a total overshoot. In order not to move further away from the target, punishment could be the better decision, and this would happen if set $pa = 2$.

As mentioned, a 50 ms network-to-muscle propagation delay is used, therefore,

Algorithm 1 Simulation Processing, Prosthesis Movement, and Learning-Behavior

```
$\theta_t \leftarrow$ set initial joint angle    ▷ start position
$\theta_{target} \leftarrow$ set target angle    ▷ target position
$spikes \leftarrow 0$    ▷ initialize spike counter
$spikes_{\text{delay}} \leftarrow 0$    ▷ memorize spike counter
$ppd \leftarrow 1$    ▷ number of EM spikes causing 1° angle change
$pa \leftarrow 1$    ▷ number of previous angles that were used for $\Delta\theta$ calculation

procedure RUN_SIMULATION($runtime$)
    $a, b, d \leftarrow$ simple spiking model parameters    ▷ see 2.3.1.1
    $V_m \leftarrow V_r$    ▷ initial membrane potential
    $u \leftarrow b \cdot V_m$    ▷ initial recovery variable
    for $t \leftarrow 1, runtime$ do    ▷ run simulation for $runtime$ ms
        $fired \leftarrow$ currently fired synapses    ▷ $(V_m >= V_t)$
        $pre \leftarrow$ occurred pre-synaptic spikes
        $tagged \leftarrow$ post-synaptic spike after $pre$ within $\tau$
        $exSpikes \leftarrow$ sum up extensor EM spikes
        $flSpikes \leftarrow$ sum up flexor EM spikes
        MOVE_PROSTHESIS($t$, $exSpikes$, $flSpikes$)
        $V_m(fired) \leftarrow V_r$    ▷ reset membrane potential
        $I \leftarrow$ summed input of presynapses + noise
        $V_m \leftarrow V_m + 0.04v^2 + 5v + 140 - u + I$
        $u \leftarrow u + a(b \cdot V_m - u)$
    end for
end procedure
procedure MOVE_PROSTHESIS($t$, $exSpikes$, $flSpikes$)
    $spikes \leftarrow spikes + (exSpikes - flSpikes)$
    if $\mathrm{mod}(t, 50) = 0$ then
        $\theta_t = \theta_{t-1} + spikes_{\text{delay}}/ppd$    ▷ move to new position
        $\Delta\theta_t = abs(\theta_t - \theta_{target})$
        $\Delta\theta_{\text{prev}} = abs\left(\sum_{i=1}^{pa}(\theta_{t-i})/pa - \theta_{target}\right)$
        if $\Delta\theta_t < \Delta\theta_{\text{prev}}$ then
            send reward to eligibility $tagged$ ES$\rightarrow$EM synapses
        else if $\Delta\theta_t > \Delta\theta_{\text{prev}}$ then
            send punishment to eligibility $tagged$ ES$\rightarrow$EM synapses
        end if
        $spikes_{\text{delay}} \leftarrow spikes$    ▷ memorize spikes for next 50ms
        $spikes \leftarrow 0$    ▷ reset spike counter
    end if
    if $\mathrm{mod}(t, 50) = 25$ then    ▷ 25ms delayed to change of angle
        let P cells fire according to $\theta_{current}$
    end if
end procedure
```

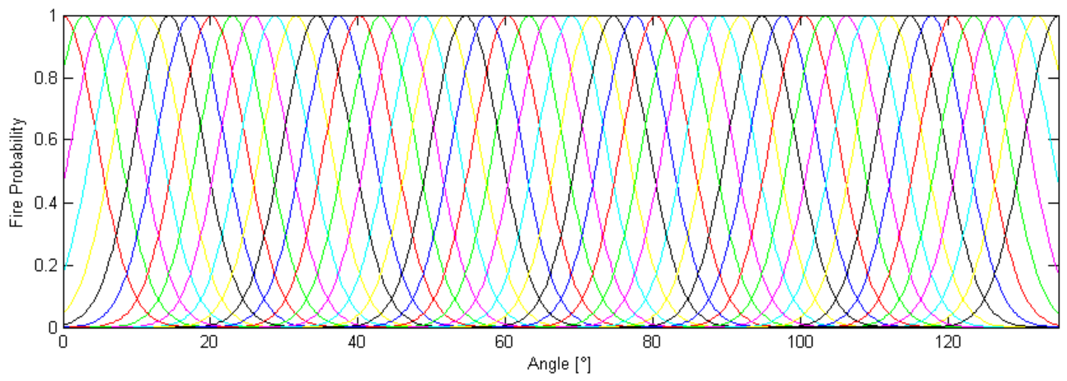


Figure 3.2: Population coding of proprioceptive neurons. The joint angle θ is encoded with population coding, whereas each proprioceptive cell has its own normal distributed fire probability relative to the angle interval $[0, 135]$. Each distribution belongs to one single P cell.

the latest counted spikes have to be memorized for the next 50 ms (Line 36). The proprioceptive cells fire each 50 ms with an delay of 25 ms to the angle change, which represents the peripheral and sub-cortical processing delay. The joint angle θ is encoded with a population coding, whereas each cell has its own normal distributed fire probability relative to the angle interval $[0, 135]$ (see figure 3.2). On average, a population of five P cells will fire approximately for each angle. A linear shift of θ causes a linear shift of the population. At the end of each iteration, the potential of fired synapses will be reset to the resting membrane potential (Line 19) and the synaptic interactions are calculated as described in section 2.3.1.1 (Line 20-22).

3.4 Results

As mentioned before, weights were tuned to get similar average spiking rates as Chadderdon *et al.* [CNKL12]. The simulation of 100 models (120 seconds each, learning disabled) reveals average spiking rates of EM: 0.54 Hz, ES: 0.43 Hz, IM: 4.51 Hz, IS: 4.37 Hz, and P: 1.6 Hz. Obviously, inhibitory cells (IM, IS) fired faster than excitatory cells (EM, ES). Figure 3.3 shows a 5000 ms sample of the baseline spiking of each individual cell grouped by different neuron types.

In total, 600 randomly wired models were generated, 100 for each of 6 different sets of values for (1) the needed number of spikes per 1° of angle change (*ppd*), (2) the number of previous angles that were used for angle difference $\Delta\theta$ calculation (*pa*), and (3) the type of eligibility weight (w_e), whereas the latter could be static or dynamic. Every single model has been simulated with

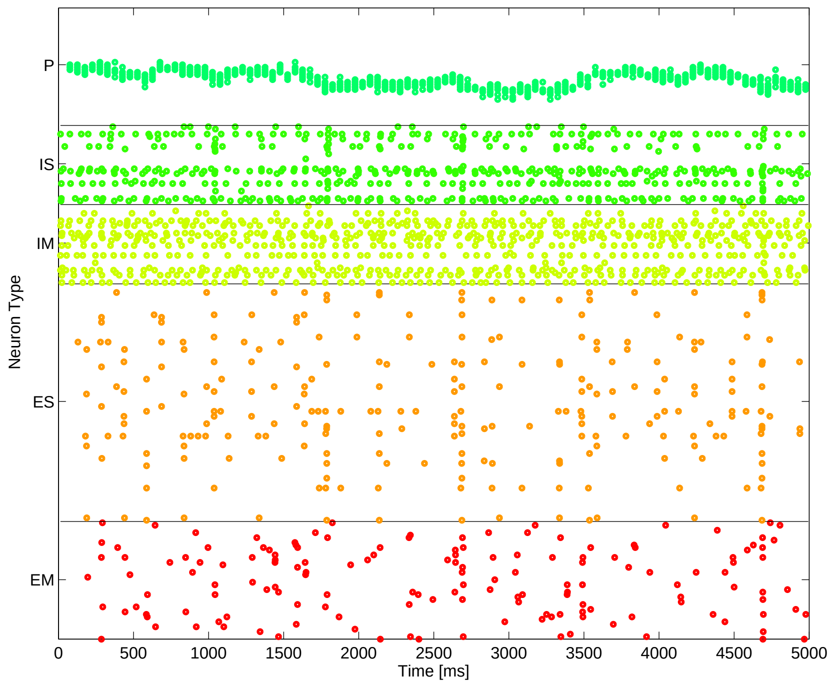


Figure 3.3: Baseline spiking in the absence of learning. A 5000 ms sample of the spiking behavior of each individual cell, grouped by the different neuron types: proprioceptive input (P), inhibitory sensory (IS), inhibitory motor (IM), excitatory sensory (ES), and excitatory motor (EM). Each colored dot represents an action potential at the corresponding time (x-axis).

5 different target angles $\theta_{\text{target}} \in \{0°, 35°, 75°, 105°, 135°\}$ with 5 simulations per angle. Altogether, $5 \cdot 5 \cdot 600 = 15000$ runs has been simulated for 120000 ms each.

For each angle (5 runs), the average root mean square deviation (RMSD) of the current angle to the target angle was calculated within $20s < t \leq 120s$. The results using different values were statistically analyzed by multiple one-sided two-sample t-tests. The p-values are reported in Table 3.3. In the following subsections, the results will be annotated in detail.

3.4.1 Different Values for the pa Parameter

The comparison of different values for the pa parameter reveals that RMSDs are lower for $pa = 3$ compared to $pa = 1$ for all simulated target angles, except 75°. This confirms the assumption, that the pa parameter could lead to smaller overall deviation. But, on the other side, there is no further improvement of $pa = 5$ compared to $pa = 3$, which could be explained by the fact that triggering events lie far apart in time.

Table 3.3: Statistical analysis of ongoing learning behavior. Shown are the p-values of multiple one-sided two-sample t-tests comparing the RMSDs obtained by using different values for the parameters: (1) the needed number of spikes per 1° of angle change (*ppd*), (2) the number of previous angles that were used for angle difference $\Delta\theta$ calculation (*pa*), and (3) the type of eligibility weight (w_e), whereas the latter can be static (S) or dynamic (D). In the first row each $X < Y$ defines the corresponding H_1 : average RMSDs of X are smaller than them of Y.

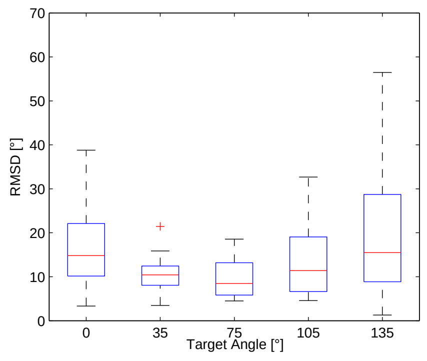
	pa=3 < *pa*=1	*pa*=5 < *pa*=3	*ppd*=4 < *ppd*=1	w_e=S < w_e=D
0°	$p < 0.0005$	$p > 0.99$	$p > 0.999$	$p < 10^{-23}$
35°	$p < 0.0004$	$p > 0.99$	$p > 0.875$	$p < 10^{-22}$
75°	$p > 0.1000$	$p > 0.77$	$p < 10^{-15}$	$p < 10^{-16}$
105°	$p < 0.0009$	$p > 0.99$	$p > 0.999$	$p < 10^{-14}$
135°	$p < 0.0001$	$p > 0.99$	$p > 0.999$	$p < 10^{-17}$

3.4.2 Different Values for the *ppd* Parameter

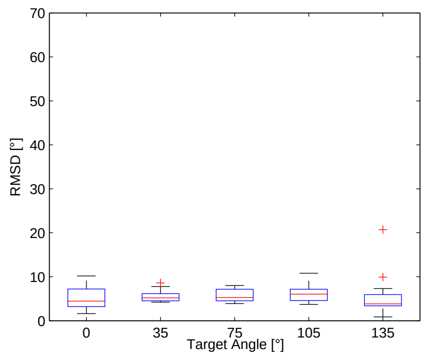
The consumption can be confirmed that larger values for the number of spikes causing 1° of angle change (*ppd*), lead to an increase of accuracy, but also to an increase of time to reach the target. In Table 3.3 we can see that in case of $\theta_{\text{target}} = 75°$ the RMSDs are significant smaller for *ppd* = 4 then for *ppd* = 1 ($p < 10^{-15}$), but not for all other target angles (figure 3.4(a) and 3.4(e)). The latter can be explained by the slowed movement, for example, it takes more than 120 seconds to reach a 70° distant target (figure 3.5(d)), compared to approximately 20 seconds for *ppd* = 1 (figure 3.5(b)). Contrary, if the target is near the current position, we got a higher accuracy (figure 3.5(a) vs. 3.5(c)).

3.4.3 Static vs. Dynamic w_e Parameter

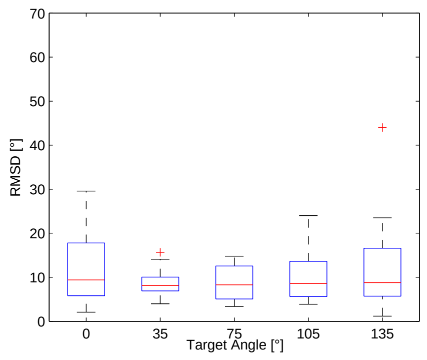
As expected, a dynamic value (relative to the time) for the eligibility weight scale factor $w_e = \tau_{\text{remaining}}/20$ outperforms a constant value $w_e = 1$. Last column of Table 3.3 shows the p-values of the one-sided two-sample t-test, validating the highly significant decrease of RMSDs for all simulated target angles. Even the worst run has an average RMSD of 20.7°, which is not bad at all (figure 3.4(b), $\theta_{\text{target}} = 135°$), especially compared to $w_e = 1$, here the worst average RMSD is 56.46°.



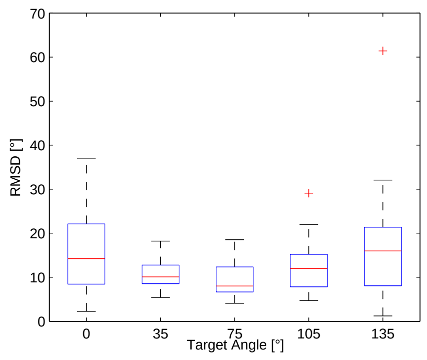
(a) $ppd = 1$, $pa = 1$, $w_e = 1$



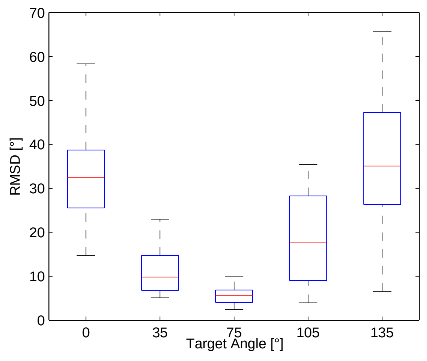
(b) $ppd = 1$, $pa = 1$, $w_e = \tau_{remaining}/20$



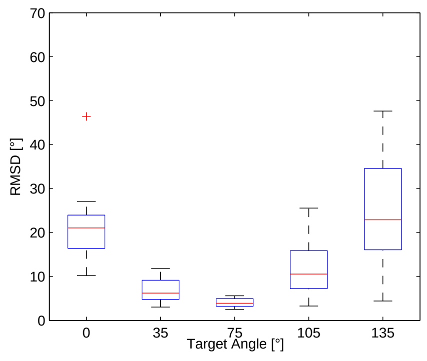
(c) $ppd = 1$, $pa = 3$, $w_e = 1$



(d) $ppd = 1$, $pa = 5$, $w_e = 1$

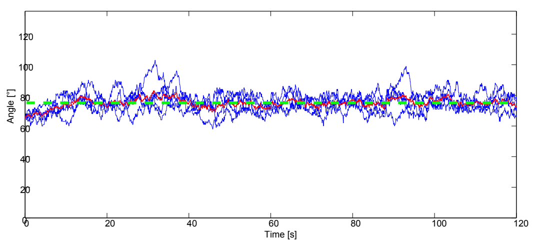


(e) $ppd = 4$, $pa = 1$, $w_e = 1$

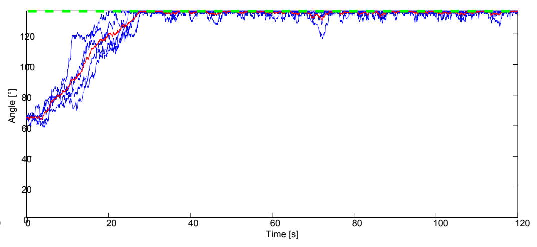


(f) $ppd = 4$, $pa = 1$, $w_e = \tau_{remaining}/20$

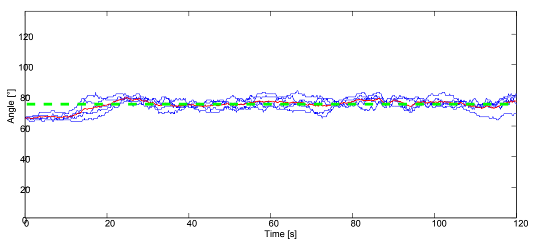
Figure 3.4: Performance of ongoing learning behavior. Different settings were used for (1) the needed number of spikes per 1° of angle change (*ppd*), (2) the number of previous angles that were used for distance calculation (*pa*), and (3) the type of eligibility weight (w_e, static or dynamic, see Equation 3.8). 100 models were generated for each setting combination, and for each model 5 different target angles θ_{target} were simulated in 5 runs per angle: 0°, 35°, 75°, 105°, and 135°. The initial angle is 65° for all simulations. In total, 500 simulations for each single boxplot. Each boxplot represents the RMSDs relating to the distance of the current angle to the target angle, where the RMSD calculation runs from $t = 20s$ until $t = 120s$. The red line represents the mean and the box represents the interquartile range (IQR) limited by the first (q_1) and third quartile q_3. The whiskers extend to $q_3 + 1.5 \cdot \text{IQR}$ and $q_1 - 1.5 \cdot \text{IQR}$, respectively. The red crosses show the outliers.



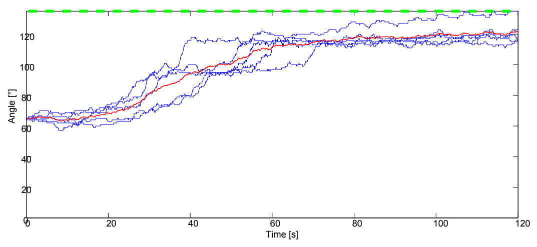
(a) $\theta_{\text{start}} = 65°$, $\theta_{\text{target}} = 75°$, $ppd = 1$



(b) $\theta_{\text{start}} = 65°$, $\theta_{\text{target}} = 135°$, $ppd = 1$



(c) $\theta_{\text{start}} = 65°$, $\theta_{\text{target}} = 75°$, $ppd = 4$



(d) $\theta_{\text{start}} = 65°$, $\theta_{\text{target}} = 135°$, $ppd = 4$

Figure 3.5: Comparison of the *ppd* parameter. Shown is a simulation of the models with the settings: $ppd = 1$, $pa = 1$, dynamic eligibility (see figure 3.4(b)), $\theta_{\text{target}} \in \{0, 75, 105, 135\}$ and both $ppd = 1$ and $ppd = 4$, respectively. Target angles are shown as green dashed lines, the start position is always the same ($\theta_{\text{start}} = 65°$). Blue lines represent the joint angles θ over 120 seconds for each single run (5 runs in total). The red line represents the mean of these 5 runs.

3.4.4 Comparison with Results of Chadderdon *et al.*

Figure 3.6 shows a comparison to Chadderdon *et al.*. As expected, the average RMSDs using static eligibility ($ppd = 1$, $pa = 1$, $w_e = 1$) are pretty similar to results of Chadderdon *et al.* using both reward and punishment. But, using dynamic eligibility weights ($w_e = \tau_{remaining}/20$) outperforms theirs results. Comparing the worst runs, we get 20.7° vs. ≈ 80°, indicating that all models were seemingly able to learn all of the target angles.

3.4.5 Turning-off Learning

As mentioned, this is an ongoing learning approach, if learning will be turned off, no further target angle will be reached. This is because as soon as the target angle is reached, proprioceptive information remains the same, which in turn causes that synaptic weights are trained in way that both extension and flexion EM spikes are more or less equal. Turning off the learning means that synaptic weights remains the same, which in turn means that the angle remains the same. An example of this behavior is shown in figure 3.7.

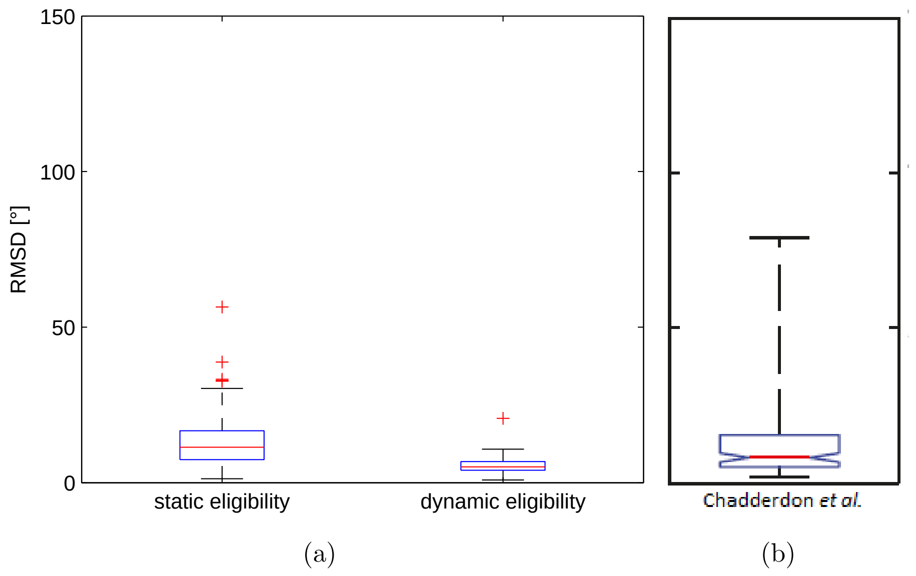


Figure 3.6: Accuracy comparison with Chadderdon *et al.*. For comparison, the plots are scaled to the same range ($[0°, 150°]$). The red line represents the mean and the box represents the interquartile range (IQR) limited by the first (q_1) and third quartile q_3. The whiskers extend to $q_3 + 1.5 \cdot \mathrm{IQR}$ and $q_1 - 1.5 \cdot \mathrm{IQR}$, respectively. The red crosses show the outliers. (a) Boxplots of average RMSDs of all simulated angles of all generated models with the following parameter for static eligibility: $ppd = 1$, $pa = 1$, $w_e = 1$ and for dynamic eligibility: $ppd = 1$, $pa = 1$, $w_e = \tau_{remaining}/20$. (b) Result of Chadderdon *et al.* [CNKL12]. Shown is the boxplot of final error (in degrees) of all simulated angles of all models, using both reward and punishment.

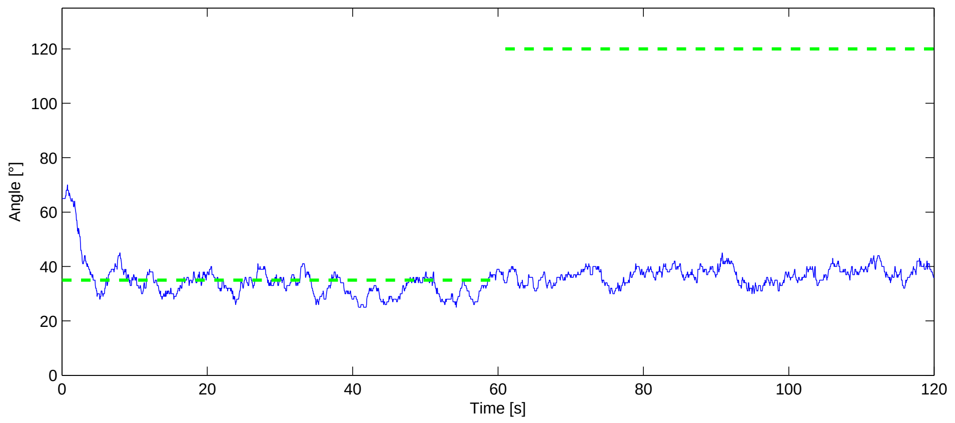


Figure 3.7: Behavior of turned-off learning. In the first 60 seconds, learning has been enabled to reach the target angle $\theta_{\text{target}} = 35°$. After that, learning has been disabled and target angle was changed to $\theta_{\text{target}} = 120°$. The current angle (blue line) does not adjust to the new target angle, but remains the same. This is because the model depends on ongoing learning.

Chapter 4

Static Learning Approach

Using an ongoing learning method as described in the previous chapter operates well, however, it seems not to be the real biological behavior: each EM cell could be responsible for each movement, because ES$\rightarrow$EM synaptic weights will be relearned for each angle. It should be more realistic that once learned synaptic plasticity, no further adaption is required to perform a volitional movement. Based on this assumption, a method was developed to train a neural network in a way that a static mapping will be learned that works for all target angles, using two synaptic inputs: current angle and target angle. In the following, a model that learned such a static mapping will be named "statically trained". In this chapter, all modifications that must be taken to the ongoing learning model will be described.

4.1 System Design

While the neuron model is the same as described in section 3.1 (using Izhikevich neurons), there are some changes in the system design. The system also consists of an neural network and an virtual prosthesis with one degree of freedom. Both, the sensory part and motor part of the network consist of the same number and same types of cells: 96 ES cells, 32 IS cells, 48 EM cells, and 32 IM cells. The EM population is also split equally into two groups, one for extension and one for flexion. However, a statically trained network needs the distance (plus or minus) between the current position and the target position to reach the latter. Assume, the current angle $\theta_{\text{current}} = 60°$, now, if $\theta_{\text{target},1} = 120°$ the virtual prosthesis has to move $+60°$, contrary, if $\theta_{\text{target},2} = 0°$ it hast to move $-60°$. If only proprioceptive input is used, we got the same input ($\theta_{\text{current}} = 60°$) for both cases, therefore, the network is unable to make a correct decision for both cases. If we use distances to the target angles ($\Delta\theta_{\text{target},1} = +60°$ and $\Delta\theta_{\text{target},2} = -60°$) as synaptic input, the network should be able to learn both cases, because of the different input.

It's not yet fully known how the brain stimulates the motor neurons and which information is used to perform volitional body movements to desired positions, anyhow, using the distance as direct synaptic input can't be biologically correct. Therefore, we have to find a way to stimulate the network only with proprioceptive information (θ_{current}) and target angle (θ_{target}). In the following subsections, all approaches will be described which paved the way and finally led to a working solution.

The concept of reinforcement learning is retained, but with some modifications, including structural synaptic plasticity, whereby synaptic connections between neurons can both emerge and disappear over time compared to STDP (see 4.1.2).

4.1.1 Construct a Static Model

As we know, proprioceptive input does not contain enough information to statically train the network. The most obvious idea is to add a new population of cells encoding the target angle. At this point, theoretically, the network has all of the requisite information. The new cell population for the target angle consists of 48 cells (T) which are connected to ES cells, in the same manner as P cells. In total, 300 statically trained models were generated, 100 for each of three different sets of synapses that were rewarded/punished: (1) only ES→EM, (2) T→ES and P→ES, and (3) T→ES, P→ES, and ES→EM. For each model, the learning phase was to move from $\theta_{\text{start}} = 135°$ to $\theta_{\text{target},1} = 0°$ and after that back to $\theta_{\text{target},2} = \theta_{\text{start}}$. Once a model was trained, a simulation starts for different target angles $\theta_{\text{target}} \in \{0, 30, 60, 80, 135\}$. The result was that none of the models was able to reach all target angles. Figure 4.1 shows the movement of one of the best trained models, it is of type (3) and learning took approximately 200 seconds, none of the target angles was reached, not even close, except $\theta_{\text{target}} = 135°$. Huge efforts were made, including slightly changes to connection weights, connection probabilities, population sizes, et cetera. However, none of the models was able to provide a satisfying result. At this point, this approach has been discarded.

Instead of using populations for both proprioception and target angle, the next approach was to use a population of cells, called distance (D) cells, encoding directly the distance to the target. Like P cells, D cells were connected to ES cells with an initial connection weight of $w_0(\text{D} \rightarrow \text{ES}) = 8.27$, but with a doubled connection probability of 0.2. The current distance to the target $\Delta\theta_{\text{target}}$ is encoded with population coding, in a similar manner like the P cells (figure 3.2), but in a range of $[-135°, 135°]$. The doubled range is the reason why 96 D cells were used, against 48 P cells. Figure 4.2 shows an example of

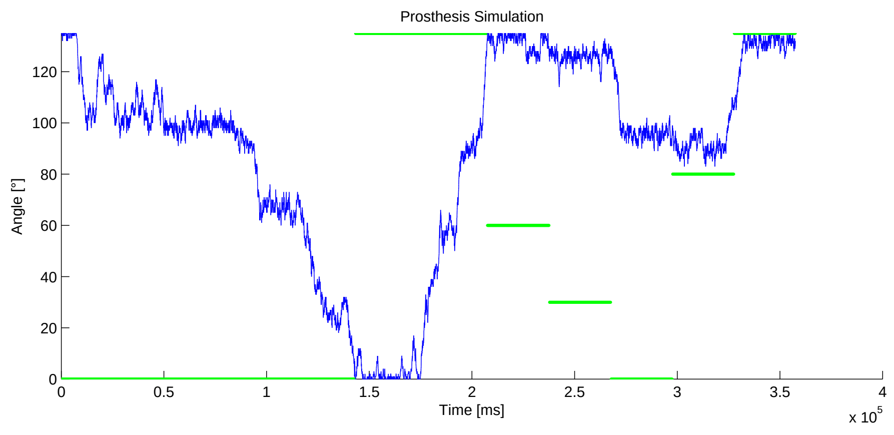


Figure 4.1: Learning and movement using target (T) and proprioceptive (P) cells. Shown is the the joint angle of the simulated prosthesis (blue line) while learning and simulation. The RL algorithm rewarded/punished T$\rightarrow$ES, P$\rightarrow$ES, and ES$\rightarrow$EM synapses. Learning took approximately 200 seconds, while the prosthesis hast to move first to $0°$ and next to $135°$. Once the model was trained a simulation starts for different target angles $\theta_{\text{target}} \in \{0, 30, 60, 80, 135\}$ (green lines, 30 seconds per angle). Although, this is the best of all models, it was not able to reach all target angles, not even close, except $\theta_{\text{target}} = 135°$.

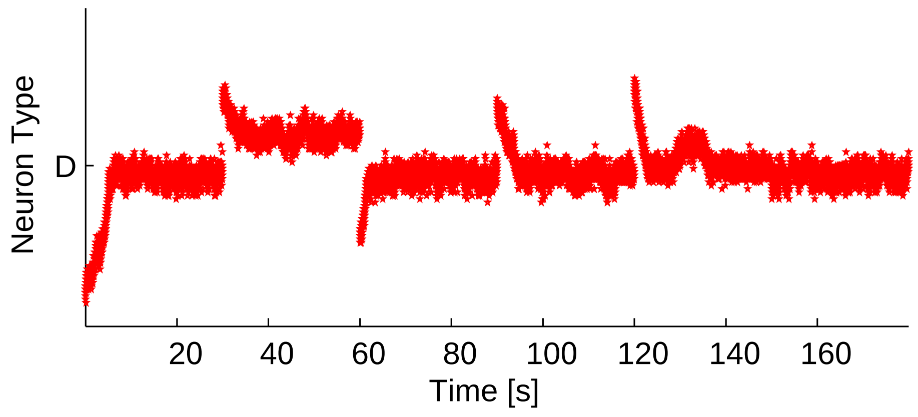


Figure 4.2: Example of the firing pattern of distance (D) cells. Each point represents one spike and each point in a row belongs to a single neuron. To get this pattern, the current distance to the target $\Delta\theta_{\text{target}}$ is encoded with population coding, in a similar manner like the P cells (figure 3.2), but in a range of $[-135°, 135°]$. Therefore, if spikes occur in the middle of the population means that θ_{target} equals θ_{current}, or at least that the distance $\Delta\theta_{\text{target}}$ is quite small.

the firing pattern of a population of D cells, were each point represents one spike and each point in a row belongs to a single neuron.

Because of the randomly generated connections between the neurons, different D cells might be connected with (exactly) the same ES cells and therefore stimulating the same EM cells. In the case of ES→EM reward/punishment the network will not be able to learn, even with structural synaptic plasticity (see 4.1.2.2). Therefore, reward/punishment will be sent to D→ES synapses.

The learning phase was split into two parts, one to learn extension and one to learn flexion. While learning, the joint angle was changed in steps of 1° per second: staring at $\theta_{\max}$ and moving to $\theta_{\min}$ and again back to $\theta_{\max}$. This means that each D cell will be stimulated approximately for the same time, which in turn results in similar learning durations for each D→ES synapse. But the angle movement is not caused by EM spikes anymore, therefore, joint angles are the same for consecutive time-points, which means that reward/punishment can't work based on distances to the target. Instead, the difference Δn_{spikes} between both extension EM spikes and flexion EM spikes is used as learning rule

$$\text{if } \theta_{\text{target}} < \theta_{\text{current}} \quad : \quad \begin{cases} \text{reward} & \Delta n_{spikes} < 0 \\ \text{punishment} & \Delta n_{spikes} > 0 \end{cases} \tag{4.1}$$

$$\text{if } \theta_{\text{target}} > \theta_{\text{current}} \quad : \quad \begin{cases} \text{reward} & \Delta n_{spikes} > 0 \\ \text{punishment} & \Delta n_{spikes} < 0 \end{cases} \tag{4.2}$$

where Δn_{spikes} is exactly the same difference that was used to change the angle (see Algorithm 1 Line 28).

While the previously described usage of cells for proprioception and target angle did not even result in one functioning model, the usage of D cells has led to a few good models, which were able to perform a full flexion followed by a full extension. But the amount of good models is not satisfying: only 16 out of 500 models were able to fully flex/extend in the absence of learning (detailed results are shown below). Hence, this learning approach must be improved.

4.1.2 Improve the Learning-Behavior

Because of the fact that the neural network is generated randomly, it is quite hard to statically train a working model. Therefore, attempts were made to improve the learning-behavior such that more working models can be generated.

4.1.2.1 Static vs. Dynamic Learning Durations

As mentioned above, the first approach was to use predefined learning angles in order that the joint angle will be changed in steps of 1° per 1 second. Therefore, each D cell will be stimulated (and trained) for 1 second according to each possible distance $\Delta\theta \in [-135^\circ, 135^\circ]$. It is assumed that longer learning durations could lead to better results, therefore, a parameter was introduced to adjust the learning duration τ_{learning} per 1° angle change.

Another approach was to use arbitrary learning angles, where the movement is caused by EM spikes of the neural network, similar to the ongoing learning approach. It is expected that network-controlled changes could led to more good models, because synapses that already correspond to the correct movement will be learned shorter, whereas synapses that correspond to the wrong movement will be learned until they correspond to the correct movement, because the current joint angle will only be changed in the direction of the target if synaptic weights were trained successfully. But, this also means that the learning duration is dynamic and varies from model to model.

4.1.2.2 Structural Synaptic Plasticity

Structural synaptic plasticity (SSP) is the ability of synaptic connections to emerge and disappear over time between different neurons. Recent experimental work provides compelling evidence that SSP is a crucial component of learning [CDM12]. Therefore, the neural network was extended by this feature in the following manner.

The criteria for determining whether a reconnection has to occur or not is made by the weight scale factor w_s. If w_s of a single synapse drops below 0.2 then the synapses will be disconnect and randomly reconnected to a new not yet connected cell of the same type as the previous cell. This ensures that the number of connections between two cell types remains the same. The weight scale factor for a freshly connected synapse is set to $w_s = 1.0$. Theoretically, this should increase the probability that a D cell is connected to an "optimal" ES cell, and therefore, this should help to increase the number of working models and speed up the learning process.

4.1.3 Synaptic Input for Distance Cells

As stated before, a way must be found to learn or reconstruct the firing pattern of D cells based on both the current angle (proprioception) and the target angle. For this, two approaches were tested, which will be described in the following subsections.

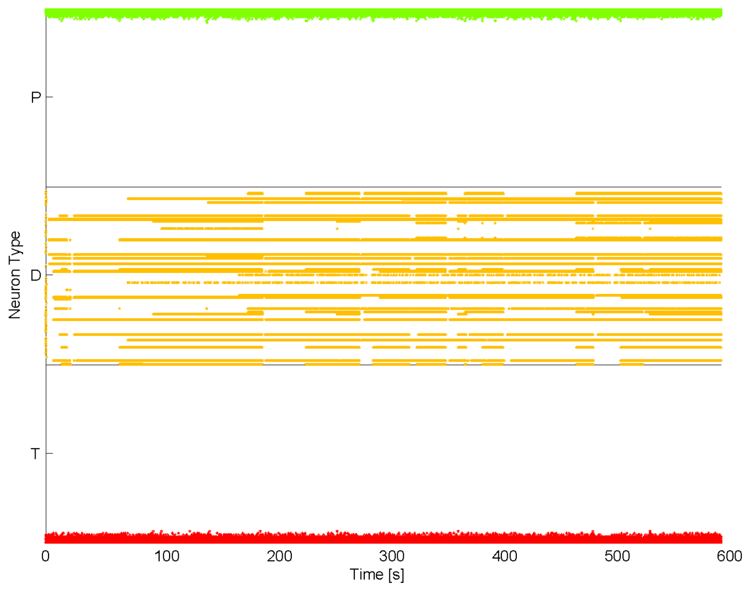


Figure 4.3: Example of the learned firing pattern of distance cells (D, yellow), stimulated by proprioceptive cells (P, green) and cells encoding the target angle (T, red). Each point represents one spike and each point in a row belongs to a single neuron. T cells were encoded with the target angle $\theta_{\text{target}} = 135°$, while P cells were stimulated with the current angle $\theta_{\text{current}} = 0°$. Shown is a learning process for 600 seconds, whereas P→D and T→D synapses were rewarded/punished. The optimal pattern would be that only the upper half of the D cells fire, or at least significantly more than the lower half.

4.1.3.1 Using Cells for Proprioception and Target Angle

The first approach was to use cells for proprioception (P) and for target angles (T), similar to the approach above. Both, P and T cells were connected to the D cells. For this purpose, the remaining cells were not taken into account. Learning exactly the same pattern of D cells for all distances as described above, could be very difficult, not to say impossible. Theoretically, it should be sufficient if the network would be able to learn extension/flexion. Therefore, the learning condition was quite simple: if the distance to the target angle is negative ($\theta_{\text{target}} < \theta_{\text{current}}$), then reward occurs if the lower half of the D cells fire more spikes as the upper half otherwise punishment occurs. Contrary, if $\theta_{\text{target}} > \theta_{\text{current}}$, then reward occurs if the upper half of the D cells fire more spikes as the lower half otherwise punishment occurs. Reward/punishment was send to T→D as well as to P→D synapses.

Several models were generated with different connection probabilities and connection weights. Contrary to expectations, none of the models was able to learn extension/flexion very well, even not after 600 seconds of training (see figure 4.3). However, this does not mean that this approach is impossible at

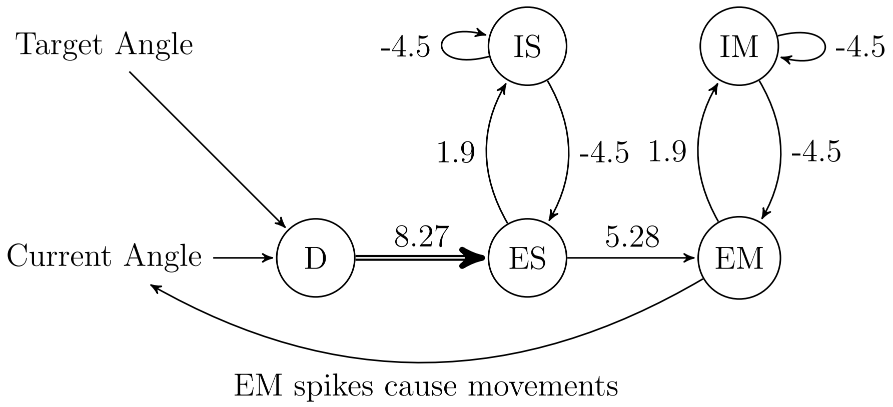


Figure 4.4: Weighted connection graph of static learning approach. Shown are different cell types of the neuronal network, including connection weights. The thick edge means that D→ES synapses will be rewarded/punished. D cells are stimulated by both encoded current angle and encoded target angle. EM spikes cause changes of the current angle.

all, maybe longer learning durations could led to satisfying results at sometime. But this approach is too time consuming for the scope of this master thesis, therefore, no further effort was spent to it. A more elegant solution was developed that does not require learning will be presented in the next section.

4.1.3.2 Stimulate Distance Cells using Current and Target Angle

Because of the poor performance of the previous approach, the idea evolved that it must be possible to find an encoding scheme, allowing the D cells to "calculate", and therefore to produce the desired pattern.

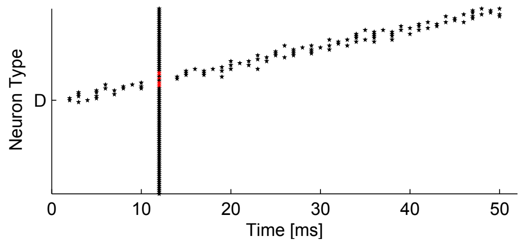
The concept was to stimulate the D cells with both proprioceptive information and target angle at the same time or rather in the same time window (figure 4.4). The idea was to use different neural codings for both proprioception and target angle. Each of them should slightly stimulate the corresponding D cells, in order that no action potential occurs, only those cells which were stimulated by both encodings fire an action potential.

A temporal coding scheme is used for target angles and a mixture of both population coding and temporal coding is used for proprioception. A well-defined time-window $\tau_{\text{coding}} = 50ms$ is used for both codings. The temporal coding rule for the target angles is

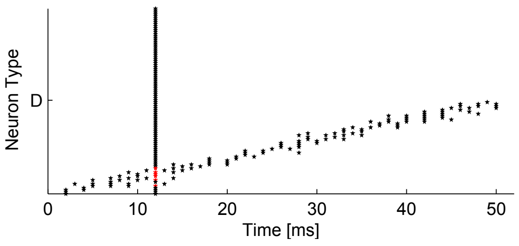
$$\theta_{\text{norm}} = \frac{\theta_{\text{target}}}{\theta_{\max} - \theta_{\min}} \tag{4.3}$$

$$t_{\text{fire}} = \text{round}\left((\tau_{\text{coding}} - 1) \cdot \theta_{\text{norm}}\right) + 1 \tag{4.4}$$

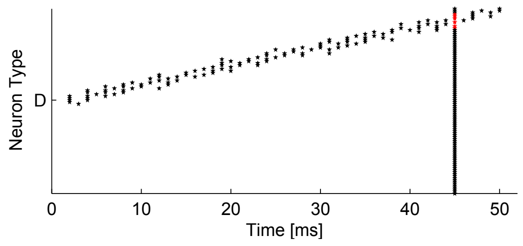
where θ_{norm} is the normalized target angle and t_{fire} is the exact time (in ms) at



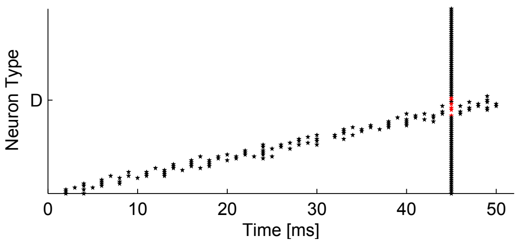
(a) $\theta_{\text{target}} = 30°, \theta_{\text{current}} = 0°$



(b) $\theta_{\text{target}} = 30°, \theta_{\text{current}} = 135°$



(c) $\theta_{\text{target}} = 120°, \theta_{\text{current}} = 0°$



(d) $\theta_{\text{target}} = 120°, \theta_{\text{current}} = 135°$

Figure 4.5: Example of the coding scheme of distance (D) cells. Vertical black bars were obtained by the corresponding temporal encoded target angle θ_{target} and the black "diagonals" represents the current angle θ_{current} encoded with a mixture of both population and temporal coding. Each black dot represents a slightly stimulation (no action potential) and each red dot represents an action potential caused by stimulation of both codings. The temporal coding parts use a well-defined time-window $\tau_{coding} = 50ms$. The resulting firing pattern of D cells (red dots) represents the distance $\Delta\theta_{\text{target}} = \theta_{\text{target}} - \theta_{\text{current}}$, whereas the lowest cell in the plot is linked to $\Delta\theta_{\text{target}} = -135°$ and the uppermost cell is linked to $\Delta\theta_{\text{target}} = +135°$. Shown are examples for different current angles and different target angles.

which *all* D cells were stimulated within each time-window τ_{coding}. Because of the ms time-scale, t_{fire} lies within $[1, 50]$, whereas $t_{\text{fire}} = 1$ means a target angle of $\theta_{\text{target}} = 0°$ and $t_{\text{fire}} = 50ms$ means $\theta_{\text{target}} = 135°$. Assume, if $\theta_{\text{target}} = 30°$ then using the equations above $t_{\text{fire}} = 12ms$ (vertical bars in Figures 4.5(a) and 4.5(b)), otherwise if $\theta_{\text{target}} = 120°$ then $t_{\text{fire}} = 45ms$ (vertical bars in Figures 4.5(c) and 4.5(d)).

The coding of the proprioceptive information is a bit more complex, because it is a mixture of population coding and temporal coding. First of all, at each 1 ms step within τ_{coding} some of the D cell were stimulated. The coding depends on both the current angle θ_{current} and the current time t_{current} within τ_{coding}

$$t_{\text{norm}} = \frac{\tau_{\text{coding}} - t_{\text{current}}}{\tau_{\text{coding}} - 1} \tag{4.5}$$

$$\theta_{\text{shift}} = \text{round}\left((\theta_{\max} - \theta_{\text{current}}) - t_{\text{norm}} \cdot (\theta_{\max} - \theta_{\min})\right) \tag{4.6}$$

where t_{norm} is the reversely normalized time relative to the time-window and θ_{shift} is an angle in the interval $[-135°, 135°]$, whereas this angle will be increased by increasing t_{current} and decreased by increasing θ_{current}, and vice versa. Therefore, if $\theta_{\text{current}} = 135°$ and $t_{\text{current}} = 1ms$ then $\theta_{\text{shift}} = -135°$, otherwise if $\theta_{\text{current}} = 0°$ and $t_{\text{current}} = 50ms$ then $\theta_{\text{shift}} = 135°$.

At each time-point $t_{\text{current}} \in [1, \tau_{\text{coding}}]$ within each time-windows τ_{coding} D cells were slightly stimulated with the same probabilistic model as used for P cells, but using θ_{shift} and the interval $[-135°, 135°]$, instead of θ_{current} and the interval $[0°, 135°]$ (figure 3.2). Patterns for $\theta_{\text{current}} = 0°$ and $\theta_{\text{current}} = 135°$ are shown as "diagonals' in figure 4.5, the patterns for each other θ_{current} shifts linearly between these.

Each cell that is stimulated by both codings, sends an action potential. Examples are illustrated in figure 4.5, whereas each black dot represents a stimulation by one of both codings and each red dot represents a stimulation by both, indicating that an action potential was sent. The resulting pattern of fired D cells is (exactly) the same as the pattern obtained by directly stimulating D cells with pre-calculated distances. Except the firing time differs within the time-window τ_{coding} according to the target angle.

4.2 Implementation

The general process is pretty much the same as described in section 3.3. There are just a few modification to the `MOVE_PROSTHESIS` procedure, which are summarized in this section.

All changes to the procedure are colored in red in Algorithm 2. At initialization, the start angle is set to $\theta_t = \theta_{\max}$ and the target angle is set to $\theta_{\text{target}} = \theta_{\min}$ (Lines 1 and 2). This has the effect, that the first learning phase corresponds to flexion learning, during which time each D cell is stimulated which belongs to flexion. As soon as the target is reached, the target angle will be set to $\theta_{\text{target}} = \theta_{\max}$, which in turn leads to extension learning, during which time each D cell is stimulated which belongs to extension (Line 18).

As already mentioned, reward/punishment will be sent to D→ES synapses (Lines 10 and 12). Additionally, after punishment occurs low weighted D→ES synapses will be reconnected, under the condition that SSP is enabled (Line 13). The stimulation of D cells will be done as described in section 4.1.3.2 (Line 22). In the case of predefined learning angles (see 4.1.2), we have to predefine a path of angles and change the learning rule as shown in equations 4.1 and 4.2.

Algorithm 2 Simulation Processing, Prosthesis Movement, and Learning-Behavior

```
1: $\theta_t \leftarrow \theta_{\max}$                                  ▷ start angle, most extended angle
2: $\theta_{target} \leftarrow \theta_{\min}$                           ▷ initial target angle to learn flexion
3: procedure MOVE_PROSTHESIS($t$, $exSpikes$, $flSpikes$)
4:     $spikes \leftarrow spikes + (exSpikes - flSpikes)$
5:     if $\mathrm{mod}(t, 50) = 0$ then
6:         $\theta_t = \theta_{t-1} + spikes_{\mathrm{delay}}/ppd$        ▷ move to new position
7:         $\Delta\theta_t = abs(\theta_t - \theta_{target})$
8:         $\Delta\theta_{\mathrm{prev}} = abs\left(\sum_{i=1}^{pa}(\theta_{t-i})/pa - \theta_{target}\right)$
9:         if $\Delta\theta_t < \Delta\theta_{\mathrm{prev}}$ then
10:            send reward to eligibility tagged D→ES synapses
11:        else if $\Delta\theta_t > \Delta\theta_{\mathrm{prev}}$ then
12:            send punishment to eligibility tagged D→ES synapses
13:            reconnect low weighted D→ES synapses
14:        end if
15:        $spikes_{\mathrm{delay}} \leftarrow spikes$                    ▷ memorize spikes for next 50ms
16:        $spikes \leftarrow 0$                                          ▷ reset spike counter
17:        if $\theta_t = \theta_{target} = \theta_{\min}$ then
18:            $\theta_{target} = \theta_{\max}$                          ▷ learn extension
19:        end if
20:    end if
21:    if $\mathrm{mod}(t, 50) = 25$ then                                ▷ 25ms delayed to change of angle
22:        stimulate D cells according to $\theta_{current}$ and $\theta_{target}$
23:    end if
24: end procedure
```

4.3 Results

Based on the finding of chapter 3, the following parameters were used for each model: $ppd = 1$, $pa = 1$, and $w_e = \tau_{\mathrm{remaining}}/20$. It is assumed that each successfully trained model should have similar RMSDs, therefore, the results are split into two parts: (1) Finding parameters such that the number of successfully trained models is maximum and (2) analysis of successfully trained models.

4.3.1 Parameters for static Learning

Several models were generated for different values of parameters: (1) learning duration τ_{learning} (static and dynamic), (2) eligibility time window $\tau_{\mathrm{eligibility}}$, (3)

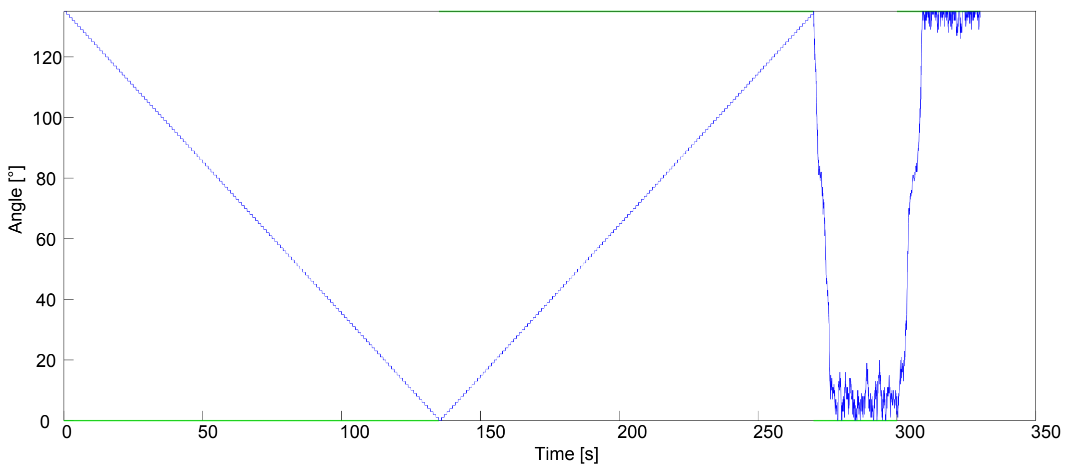
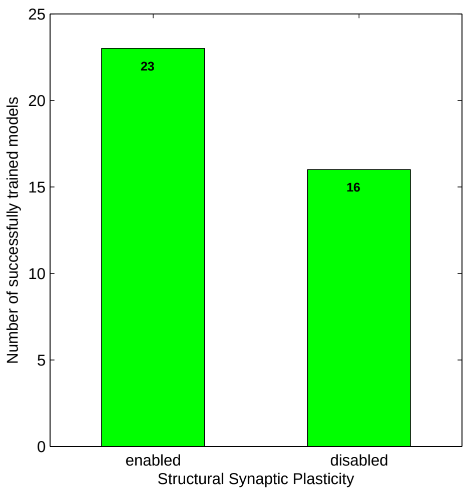


Figure 4.6: Shown is an example of the learning phase using predefined learning angles. The blue line represents the angle of the simulated prosthesis. The first 265 seconds represent the learning phase using predefined learning angles, whereas the following 60 seconds show the simulation of both full flexion and full extension in the absence of learning. The green lines visualize the corresponding target angle.

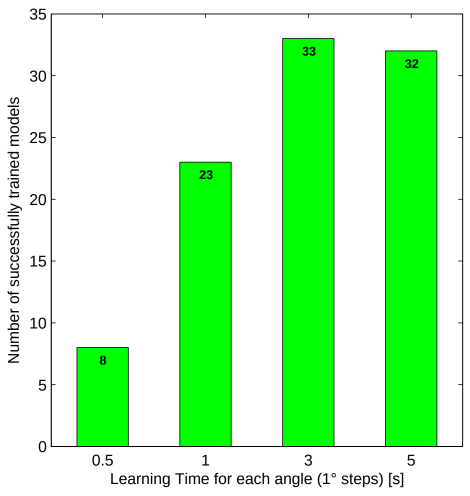
total number of cells n_{cells}, (4) SSP enabled/disabled, (5) average number of firing D cells per angle $n_{\text{D,firing}}$ (within 50 ms), and (6) the type of synapses that will be rewarded/punished.

For each model, the initial angle is set to $\theta_{\max}$ and network has to learn two target angles $\theta_{\text{target},1} = \theta_{\min}$ followed by $\theta_{\text{target},2} = \theta_{\max}$ as soon as $\theta_{\text{target},1}$ has been reached (see 4.2). Once $\theta_{\text{target},2}$ is reached, the learning phase is completed. In order to determine whether or not a model was trained successfully, the training phase is followed directly by a simulation phase, within which the angles $\theta_{\text{target},1}$ and $\theta_{\text{target},2}$ must be reached again, but in the absence of learning (figure 4.6). A model is determined as successfully trained, if it was able to reach both angles within 30 seconds each. Theoretically, a model should now be able to reach all other target angles, because it learned both full flexion and full extension. In the case of dynamic learning durations (arbitrary learning angles), a model is also determined as not successfully learned, if learning phase is not completed within 30 minutes of simulated time.

Unless otherwise mentioned, the settings were: $\tau_{\text{eligibility}} = 100ms$, $n_{\text{cells}} = 304$ (96 D, 96 ES, 48 EM, 32 IS, 32 IM), SSP enabled, $n_{\text{D,firing}} \approx 5$, and reward/punish D→ES synapses. The following subsections compare the results of different settings.



(a)



(b)

Figure 4.7: Performance using predefined learning angles. The bars show the numbers of successfully trained models relating to structural synaptic plasticity as well as to different learning durations used for each 1° of angle change, respectively.

4.3.1.1 SSP enabled vs. disabled

Initially, it has been tested if there is a difference between SSP enabled versus disabled, using static learning duration ($\tau_{\text{learning}} = 1s$). 500 models were generated for both SPP enabled and disabled, respectively. Using SPP shows an increase of 44% in the number of successfully trained models (23 vs. 16), but the overall performance is still quite bad.

4.3.1.2 Different learning durations

Different learning durations per angle were tested using SSP: for each $\tau_{\text{learning}} \in \{0.5s, 1s, 3s, 5s\}$ 500 models were trained. As seen in figure 4.7(b), we still got a poor overall performance. For $\tau_{\text{learning}} = 0.5s$ just 8 out of 500 models were successfully trained. Although increasing τ_{learning} results in more successfully trained models, the performance remains bad, while increasing drastically the learning duration. In example, there are 135+135+1=271 angles to learn, which means that the learning duration is 135.5 seconds in the case of $\tau_{\text{learning}} = 0.5s$, compared to 1355 seconds in the case of $\tau_{\text{learning}} = 5s$.

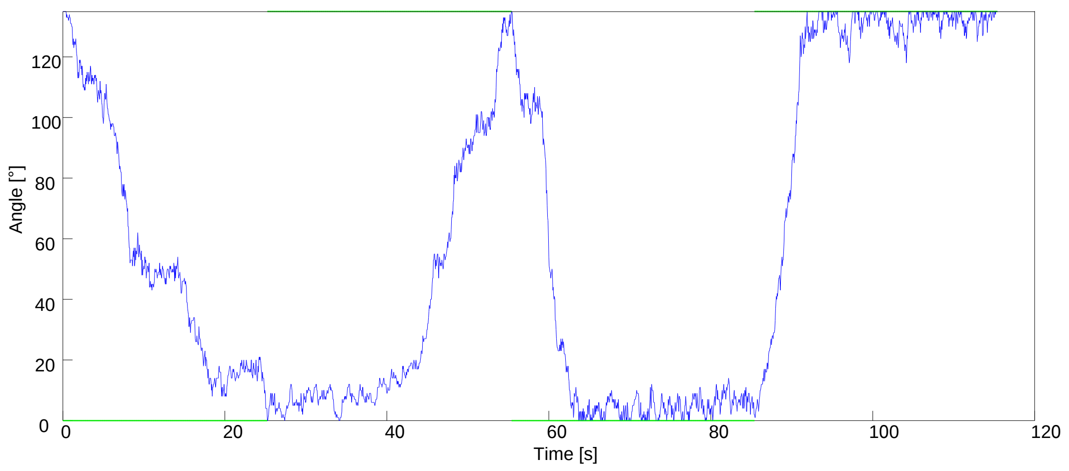


Figure 4.8: Shown is an example of the learning phase using network-controlled learning angles. The blue line represents the angle of the simulated prosthesis. The first 55 seconds show the learning phase, whereas the following 60 seconds show the simulation of both full flexion and full extension in the absence of learning. The green lines visualize the corresponding target angle.

4.3.1.3 Static vs. dynamic learning durations

Dynamic learning durations were tested (500 models each), meaning that the current joint angle will be changed relative to EM spikes (figure 4.8). It is expected that using network-controlled angle changes should cause both skipping ”good” initial synapses and longer learning durations for ”bad” synapses, because the current joint angle will only be changed in the direction of the target if synaptic weights were trained successfully. Whereas ”good” means that corresponding initial weights already result in the correct movement and ”bad” means the opposite. Figure 4.9 shows that using dynamic learning durations totally outperforms the results of static learning durations. In total, 257 models were successfully trained, this is an increase of 779% compared to the 33 models using $\tau_{\text{learning}} = 5s$, confirming recent expectations.

In order to reconfirm that SSP increases the performance, 500 additional models were generated without the usage of SSP. Indeed, the result shows that using SSP produces 30% more successfully trained models (Figure. 4.9).

4.3.1.4 Different eligibility time windows

It was also tested to increase the eligibility time window to $\tau_{\text{eligibility}} = 150ms$, however, this results in a strong decrease of performance of about -40% (figure 4.9). This implies that reward/punishment of temporally distant events (spiking$\rightarrow$movement) is only possible up to a certain level.

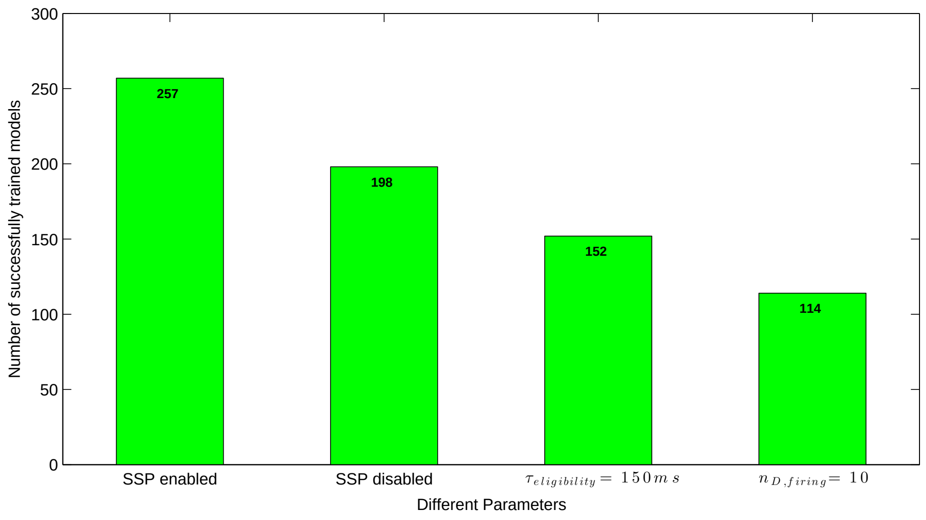


Figure 4.9: Performance using network-controlled learning angles (τ_{learning} = dynamic). The bars show the numbers of successfully trained models obtained by different parameters. The used parameters are: $\tau_{\text{eligibility}} = 100ms$, $n_{\text{cells}} = 304$, SSP enabled, $n_{\text{D,firing}} \approx 5$. The varying parameters for the bars are below of them.

4.3.1.5 Vary the average number of firing D cells

Another assumption concerns the average number of firing D cells per angle $n_{\text{D,firing}}$. An increasing number results in an increased stimulation of ES cells, therefore, more D→ES synapses could be trained at the same time, which in turn could result in an increased performance. However, it could also cause a global shift relative to the target angle, because of the increased overlap of firing D cells of adjacent angles. The result shows that doubling $n_{\text{D,firing}}$ to 10 decreases the number of successfully trained models by -55% to 114.

4.3.1.6 Double the total number of cells

To counteract the increased overlap by increasing $n_{\text{D,firing}}$, the total number of cells n_{cells} has also been doubled to 608, maintaining both the ratio of cell types and the baseline spiking rate. Figure 4.10 shows the results for all $n_{\text{D,firing}} \in \{5, 10, 20\}$. It is worth to mention, that $n_{\text{D,firing}} = 20$ in case of $n_{\text{cells}} = 608$ causes the same overlap per angle as $n_{\text{D,firing}} = 10$ in case of $n_{\text{cells}} = 304$, because of the doubled number of D cells. For $n_{\text{D,firing}} \in \{5, 10\}$ the results show decreased performances by -79% and -28%, respectively. But for $n_{\text{D,firing}} = 20$ we see a slight increase by 8% resulting in 278 successfully trained models. But, because of the highly increased computational time, the increased performance is put into perspective.

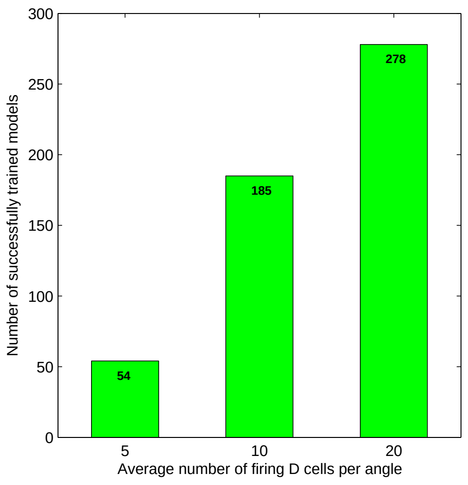


Figure 4.10: Performance using doubled number of cells. The bars show the numbers of successfully trained models obtained by different numbers of firing D cells per angle.

4.3.1.7 Reward ES→EM synapses

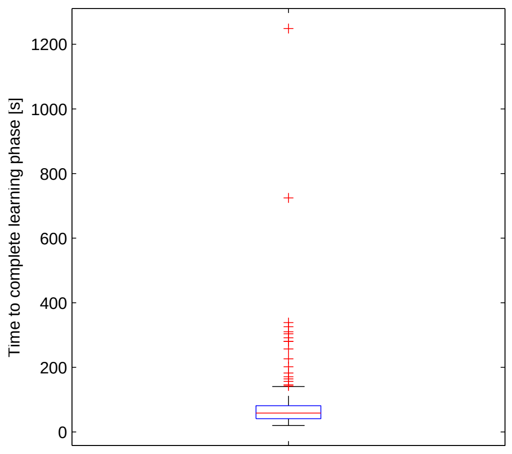
Finally, in order to prove the statement that a network could not be able to learn using reward/punishment of ES→EM synapses, additional models were generated with both SSP enabled and disabled as well as using static ($\tau_{\text{learning}} = 1s$) and dynamic learning durations. Only one single model out of these 2000 models has been trained successfully, confirming the statement.

4.3.2 Analysis of Successfully trained Models

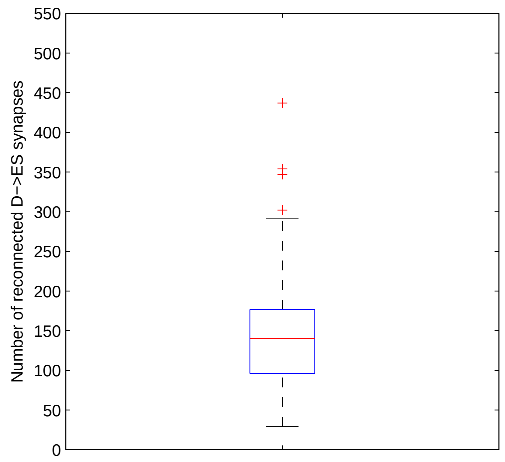
After improving the learning behavior, the performance of the successfully trained models will be evaluated. For this, the 257 models were analyzed, which were generated by using the following parameters: $\tau_{\text{eligibility}} = 100ms$, $n_{\text{cells}} = 304$, SSP enabled, $n_{\text{D,firing}} \approx 5$, $\tau_{\text{learning}} = \text{dynamic}$.

4.3.2.1 Analysis of the learning phase

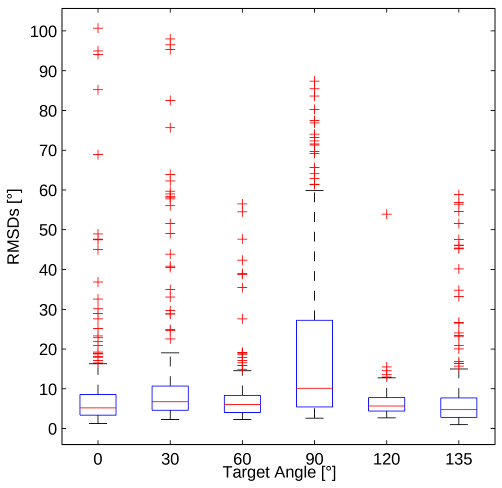
Figure 4.11(a) shows required simulation times to train the models. On average it took 79 seconds until the learning phase has been completed. Interestingly the shortest learning duration amounts only 19.6 seconds, whereas the longest learning phase required 1249 seconds to complete. While learning, the SSP rule reconnected on average 142 D→ES synapses (figure 4.11(b)), wheres the average number of D→ES synapses amounts 1847. This implies that approximately 7% of the initial synapses were connected incorrectly, or more formally they contributed to the wrong (opposite) movement.



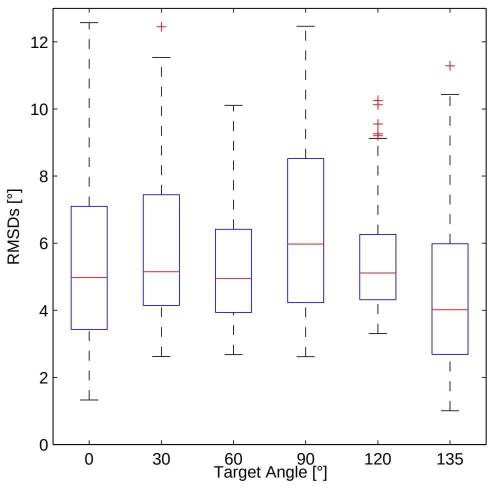
(a)



(b)



(c)



(d)

Figure 4.11: Performance of 257 statically trained models using the parameters: $\tau_{\text{eligibility}} = 100ms$, $n_{\text{cells}} = 304$, SSP enabled, $n_{\text{D,firing}} \approx 5$, $\tau_{\text{learning}} =$ dynamic. The red line represents the mean and the box represents the interquartile range (IQR) limited by the first (q_1) and third quartile q_3. The whiskers extend to $q_3 + 1.5 \cdot \text{IQR}$ and $q_1 - 1.5 \cdot \text{IQR}$, respectively. The red crosses show the outliers. (a) Variations of the needed learning duration. (b) Number of reconnected D→ES synapses caused by SSP. (c) RMSDs of the 257 models: starting at $\theta_{\max}$, the task was to reach different angles in this sequence: 30°, 90°, 0°, 60°, 135°, and 120°, whereas targets were changed each 30 seconds (in total 180 seconds). (d) RMSDs of top 100 models.

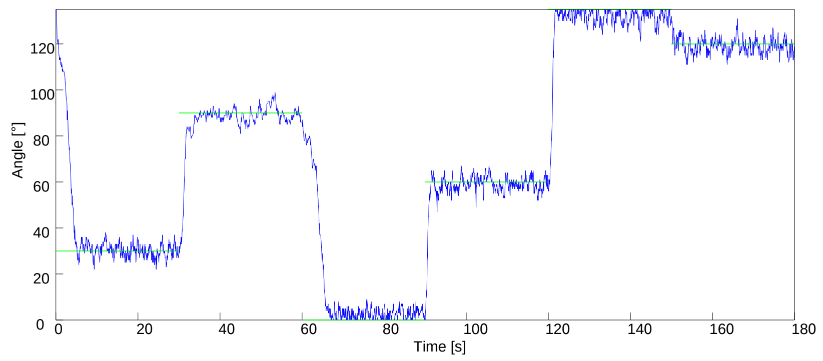
4.3.2.2 Analysis of the simulation phase

For each of the 257 trained models, additional target-reach tasks have been simulated. Starting at θ_{max}, the task was to reach different angles in this sequence: 30°, 90°, 0°, 60°, 135°, and 120°, whereby targets were changed each 30 seconds (in total 180 seconds). For each target angle, the RMSDs were calculated within the last 20 seconds. Contrary to the assumption that each successfully trained model should have similar RMSDs, figure 4.11(c) reveals that most of the trained models have a similar performance, but not all of them. Interestingly, 90% of all models were able to reach all angles with RMSDs lower than 20°, except for $\theta_{\text{target}} = 90°$, here only 68% of the models were able to perform with RMSDs lower than 20°. This might be caused by the temporal coding of the target angle or more precisely by the corresponding firing time $t_{\text{fire}}(90°) = 34ms$ (Equation 4.4), possibly the reinforcement does not work very well because of the fixed time-steps (50 ms) at which reinforcement occurs. But it is worth to mention that the top 100 of the models perform with RMSDs lower than 13° for all simulated target angles (mean = 5.5°, figure 4.11(d)), therefore, one out of five models can be trained statically with a very well performance.

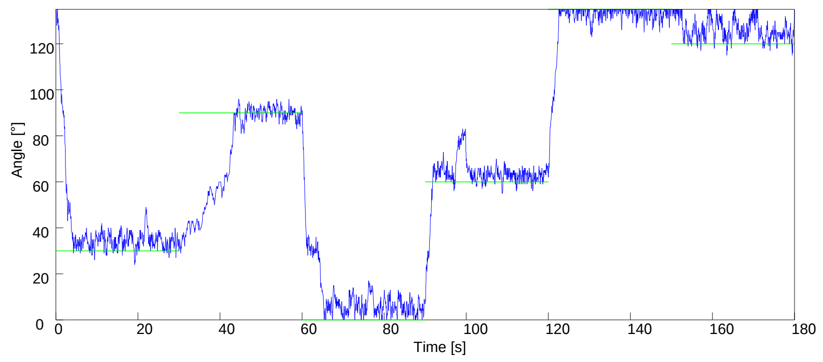
The best out of all successfully trained models has an average RMSD of 3.3° for all simulated angles and it took on average 3.5 seconds to move from one to another target angle. The latter is an improvement of about 85% compared to the examples of Chadderdon *et al.* (see Figure 5 in [CNKL12]). Obviously, this is because the here presented method does not has to re-weight synapses again and again for each change of the target angle. This model has been trained for ≈230 seconds (simulation time), but the duration of learning is not necessarily an indicator for a good performance. The model which was trained successfully after 19.6 seconds, has an average RMSD of 6.4° (figure 4.12(b)), whereas the model which needed the longest time (1249 seconds) until it was trained has an average RMSD of 4.99° (figure 4.12(c)). Furthermore, a few models show a conspicuous shift to all target angles (figure 4.12(d)), which may be due to the fact that the resting position (as soon as target angle has been reached) is not trained but only the movement (figure 4.8).

4.3.2.3 Runtime analysis

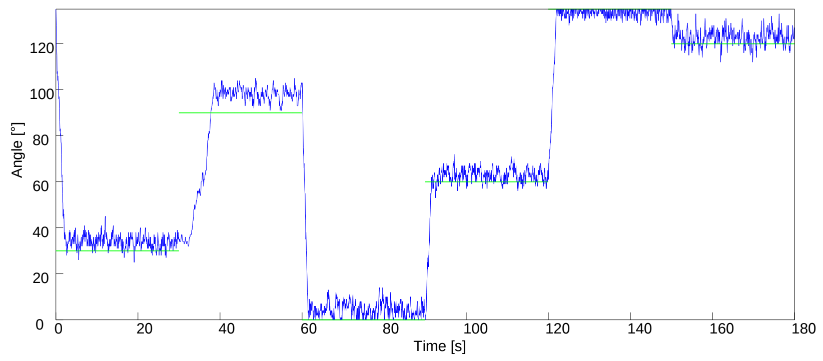
A runtime analysis reveals the computational efficiency: the simulation of 1000 ms took on average 1027 ms of calculation time, while learning is enabled. After a model was trained, the simulation of 1000 ms took on average 987 ms, meaning that the simulation can be done in real-time (Intel® Core™ i5-3210M, 2×2.5 GHz, 8 GB RAM).



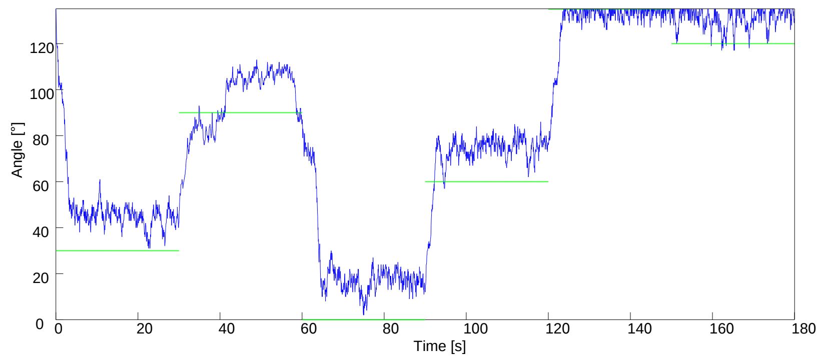
(a) Model with lowest overall RMSD



(b) fastest-trained model



(c) slowest-trained model



(d) model with global shift to target

Figure 4.12: Simulation of statically trained models. The blue line represents the angle of the simulated prosthesis, during simulation of several target angles (green lines). Target angles were changed each 30 seconds.

Chapter 5

Discussion and Outlook

5.1 Discussion

The findings of chapter 3 reveal, that a simple biologically realistic random model of both motor (M) and sensory (S) neurons – excitatory (E) as well as inhibitory (I) – can be trained to reach a given target angle, using proprioceptive information and spike time dependent plasticity (STDP). Several approaches have been tested to improve the performance of Chadderdon's model. For this, three additional tuning parameters were introduced.

Neurophysiological studies reveal that neurons in certain brain structures carry specific signals about past and future rewards, whereof dopamine neurons display a short-latency, phasic reward signal indicating the difference between actual and predicted rewards [Sch02]. Details of the kinetics of intracellular processes triggered by STDP and dopamine are unknown, therefore, it is reasonable to suggest that different concentrations of dopamine could have a different influence on the contribution to STDP. Izhikevich [Izh07] suggested a method that captures the essence of dopamine modulation of STDP, where the contribution depends on the eligibility trace and the extracellular dopamine concentration.

In this work a simplified version has been used, where the remaining time of a eligibility trace scales the contribution to the synaptic weight in the case of a reinforcement learning event. The results show a highly significant decrease of the average final error, compared to a static contribution. Additionally, even though the randomly generated connections, the dynamic behavior led to the fact, that all models were able to learn all simulated target angles, whereas Chadderdon's method produced some "unlearnable" models.

The remaining two parameters allow to define (1) the number of previous angles that were used for angle difference calculation, used for the reward rule, and (2) the needed number of spikes causing $1°$ of angle change. These parameters can not be justified in the biological sense, but the results show that the former could contribute to an overall decreased final error, whereas the latter allows to define how fast the simulated prosthesis should move.

The issue of this model is, that learning is required permanently until the target has been reached. Considering this from a biological perspective, it seems unrealistic that each synapse could be responsible for each movement and has to relearn again and again. It should be more realistic that once synaptic plasticity has been learned, volitional movements does not require further adaption or at least only slightly.

Indeed, the findings in chapter 4 unveil that using both STDP and structural synaptic plasticity (SSP) allow to statically train random models, in order that different target angles will be reached with no further learning. The results prove that proprioception alone contains not enough information, instead, it is required to use information about the target angle as an additional synaptic input.

First attempts show that using an additional population, encoding the target angle, does not lead to the objective, but using the distance to the target as exclusive input. However, first simulations, using predefined angle changes, result in a poor density of successfully trained models. Only tuning of parameters and the usage of SSP resulted in a satisfying amount of successfully trained models. Especially, using arbitrary angle changes while learning resulted an an increase of 600%. This can be explained by the fact that in the case of predefined angle changes, on the one hand "good" initial synapses might be worsen (because of the noise input during resting phase) and on the other hand "bad" initial synapses might require more time to learn, whereby "good" means that corresponding initial weights already result in the correct movement and "bad" means the opposite.

Additionally, using SSP results in a further increase in the number of successfully trained models by approximately 30%. Contrary, both increasing the time-window of the eligibility traces and increasing the population size of firing cells according to each distance (D), result in a decrease of about 40% and 55%, respectively. The former might be caused by the fact that reward/punishment of temporally distant events (spiking$\rightarrow$movement) is only possible up to a certain level, whereas the latter could be caused by the increased overlap of firing D cells of adjacent angles. Also doubling the total number of cells does not necessarily lead to a better performance, even using the same population size of firing D cells results in a decrease of 79%. Only by quadrupling the firing D cells, an increase of 8% has been found. Possibly, a further increase in the total number of cells could lead to better results, but at the same time the computational time will be increased drastically.

But, using distances as synaptic input can't be the realistic behavior, therefore an encoding scheme has been developed, which results in some kind of calculation behavior of the D cells by stimulating them with information about both proprioception and target angle. The resulting firing pattern is (exactly) the same as the pattern obtained by directly stimulating them with pre-calculated distances, except the firing time differs within the time-window.

Furthermore, it has been proven that reinforcement of ES$\rightarrow$EM synapses is unsuitable to learn a static model, at least for the here presented method. This is caused by the randomly generated connections between the neurons: different D cells might be connected with (exactly) the same ES cells and therefore stimulating the same EM cells.

The analysis of successfully trained models confirms, that most of the successfully trained models were able to reach any given target angle within an average duration of 3.5 seconds, which is a slow but realistic and an improvement of about 85% compared to Chadderdon *et al.*.

5.2 Outlook

This work is the basis that the mutual adaptivity can be studied using spiking neurons, but further investigation is required. Most importantly, we need an interface for different classifiers. For this the neurons must be arranged in biologically realistic 3-dimensional space in order to allow the positioning of virtual electrodes, which in turn record the neural activities. A further issue is the fact that cramping of muscles is not taken into account, meaning that EM spikes causing extension neutralizes spikes causing flexion, and vice versa. In addition to that, the movement of the virtual prosthesis never rests, because of the ongoing EM spikes. The parameter which defines the number of EM spikes that causes 1° of angle change is maybe a beginning, but it also causes an overall slower movement and it would be more realistic if it is controlled by the neural network itself.

It would also be desirable to increase the number of successfully trained models, maybe by tuning the used parameter or even by using new parameters. For example, an additional training phase for the resting position (as soon as target angle has been reached) could improve the overall performance.

Another interesting issue, is to find a successful way to use populations for proprioception and target angles, which in turn stimulate D cells (or directly ES cells), as tried in this work. Even if we do not know how the brain transforms thoughts to voluntary movements, but we know about the existence of proprioceptors, and therefore, it seems to be more realistic than using one population which is stimulated directly by both information.

5.3 Conclusion

Finally, the here presented method uses biologically plausible algorithms and is at the same time computational efficient. The usage of spike time depended plasticity, structural synaptic plasticity, and the introduced neural coding offers the ability to learn a static mapping for all possible targets using proprioception and encoded information about the target position as synaptic inputs.

Simulations of trained models perform with low deviations to targets and computations run in realtime. Also the movements from one target to another are relatively fast, compared to previous methods. Furthermore, the introduced neural coding offers a population of cells the possibility to "calculate" the difference between two values.

Bibliography

[Abb99] Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5):303–304, 1999.

[Abb06] Alison Abbott. Neuroprosthetics: In search of the sixth sense. *Nature*, 442(7099):125–127, 2006.

[ACG$^+$09] Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Roberto Lent, Suzana Herculano-Houzel, et al. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

[AF07] Francisco J Alvarez and Robert EW Fyffe. The continuing case for the renshaw cell. *The Journal of physiology*, 584(1):31–45, 2007.

[BKM04] Emery N Brown, Robert E Kass, and Partha P Mitra. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature neuroscience*, 7(5):456–461, 2004.

[BRW$^+$07] Julie Blumberg, Jörn Rickert, Stephan Waldert, Andreas Schulze-Bonhage, Ad Aertsen, and Carsten Mehring. Adaptive classification for brain computer interfaces. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 2536–2539. IEEE, 2007.

[BS10] Alexandros Bouganis and Murray Shanahan. Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[CDM12] Pico Caroni, Flavio Donato, and Dominique Muller. Structural plasticity upon learning: regulation and functions. *Nature Reviews Neuroscience*, 13(7):478–490, 2012.

[CG90] Barry W Connors and Michael J Gutnick. Intrinsic firing patterns of diverse neocortical neurons. *Trends in neurosciences*, 13(3):99–104, 1990.

[CNG$^+$11] John P Cunningham, Paul Nuyujukian, Vikash Gilja, Cindy A Chestek, Stephen I Ryu, and Krishna V Shenoy. A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces. *Journal of neurophysiology*, 105(4):1932–1949, 2011.

[CNKL12] George L Chadderdon, Samuel A Neymotin, Cliff C Kerr, and William W Lytton. Reinforcement learning of targeted movement in a spiking neuronal model of motor cortex. *PloS one*, 7(10):e47251, 2012.

[Colst] OpenStax College. *Anatomy & Physiology.* OpenStax College, 2013. `http://cnx.org/content/col11496/latest/`.

[CWD$^+$12] Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus JC McMorland, Meel Velliste, Michael L Boninger, and Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381:557–564, 2012.

[Deu14] DLR - Deutsches Zentrum für Luft- und Raumfahrt. Anthropomorphic DLR hand arm system for future service robotics, 2014. [Online; accessed 20-January-2014]. `http://www.dlr.de/rmc/sr/en/desktopdefault.aspx/tabid-5486`.

[DSHG98] John P Donoghue, Jerome N Sanes, Nicholas G Hatsopoulos, and Gyöngyi Gaál. Neural discharge and local field potential oscillations in primate motor cortex during voluntary movements. *Journal of Neurophysiology*, 79(1):159–173, 1998.

[EKK$^+$92] Andreas K Engel, Peter König, Andreas K Kreiter, Thomas B Schillen, and Wolf Singer. Temporal coding in the visual cortex: new vistas on integration in the nervous system. *Trends in neurosciences*, 15(6):218–226, 1992.

[FGH$^+$14] Eduardo Fernández, Bradley Greger, Paul A House, Ignacio Aranda, Carlos Botella, Julio Albisua, Cristina Soto-Sánchez, Arantxa Alfaro, and Richard A Normann. Acute human brain responses to intracortical microelectrode arrays: challenges and future prospects. *Frontiers in neuroengineering*, 7, 2014.

[GBC99] Jay R Gibson, Michael Beierlein, and Barry W Connors. Two networks of electrically coupled inhibitory neurons in neocortex. *Nature*, 402(6757):75–79, 1999.

[GM96] Charles M Gray and David A McCormick. Chattering cells: superficial pyramidal neurons contributing to the generation of synchronous oscillations in the visual cortex. *Science*, 274(5284):109–113, 1996.

[GSK86] AP Georgopoulos, AB Schwartz, and RE Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[HBJ$^+$12] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick van der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

[HE11] Ivan HK Hong and Sarah J Etherington. Neuromuscular junction. *eLS*, 2011.

[HH52] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[I$^+$03] Eugene M Izhikevich et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[IM08] Eugene M Izhikevich and Jeff Moehlis. Dynamical systems in neuroscience: The geometry of excitability and bursting. *SIAM review*, 50(2):397, 2008.

[Izh07] Eugene M Izhikevich. Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452, 2007.

[Koc04] Christof Koch. *Biophysics of computation: information processing in single neurons*. Oxford university press, 2004.

[LCdGLC99] Catherine Liégeois-Chauvel, Jozina B. de Graaf, Virginie Laguitton, and Patrick Chauvel. Specialization of left auditory cortex for speech perception in man depends on temporal coding. *Cerebral Cortex*, 9(5):484–496, 1999.

[LN06] Mikhail A Lebedev and Miguel AL Nicolelis. Brain-machine interfaces: past, present and future. *TRENDS in Neurosciences*, 29(9):536–546, 2006.

[MAT14] MATLAB. *version 8.3.0.532 (R2014a)*. The MathWorks Inc., Natick, Massachusetts, 2014.

[MH13] Elaine Nicpon Marieb and Katja Hoehn. *Human anatomy and physiology*. Pearson Education, 9th edition, 2013.

[Mye14] Douglas Myers. Eeg cap — Wikipedia, the free encyclopedia, 2014. [Online; accessed 15-November-2014]. `http://commons.wikimedia.org/wiki/File:EEG_cap.jpg`.

[NCK$^+$13] Samuel A Neymotin, George L Chadderdon, Cliff C Kerr, Joseph T Francis, and William W Lytton. Reinforcement learning of two-joint virtual arm reaching in a computer model of sensorimotor cortex. *Neural computation*, 25(12):3263–3293, 2013.

[PAF$^+$04] Dale Purves, George J Augustine, David Fitzpatrick, William C Hall, Anthony-Samuel LaMantia, James O McNamara, and Leonard E White. *Neuroscience*. Sinauer Associates, 3rd edition, 2004.

[Ren46] Birdsey Renshaw. Central effects of centripetal impulses in axons of spinal ventral roots. *J Neurophysiol*, 9(3):191–204, 1946.

[RLLS80] Per E Roland, Bo Larsen, NA Lassen, and Eric Skinhoj. Supplementary motor area and other cortical areas in organization of voluntary movements in man. *Journal of Neurophysiology*, 43(1):118–136, 1980.

[Sch02] Wolfram Schultz. Getting formal with dopamine and reward. *Neuron*, 36(2):241–263, 2002.

[SRB12] Martin Spüler, Wolfgang Rosenstiel, and Martin Bogdan. Co-adaptivity in unsupervised adaptive brain-computer interfacing: a simulation approach. In *COGNITIVE 2012, The Fourth International Conference on Advanced Cognitive Technologies and Applications*, pages 115–121, 2012.

[STC$^+$11] J.C. Sanchez, A. Tarigoppula, J.S. Choi, B.T. Marsh, P.Y. Chhatbar, B. Mahmoudi, and J.T. Francis. Control of a center-out reaching task using a reinforcement learning brain-machine interface. In *Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on*, pages 525–528, April 2011.

[Ste05] Mircea Steriade. *Cellular Substrates of Brain Rhythms*, pages 33–63. Lippincott Williams & Wilkins, 2005.

[VaF90] A B Vallbo and N A al Falahe. Human muscle spindle response in a motor learning task. *The Journal of Physiology*, 421(1):553–568, 1990.

[Wic14] Paul Wicks. Brain-computer interface — Wikipedia, the free encyclopedia, 2014. [Online; accessed 22-December-2014]. `http://en.wikipedia.org/wiki/Braincomputer_interface`.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Tübingen, 31. Januar 2015

Ort, Datum   Unterschrift