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

Master Thesis Computer Science

Title of thesis

Vor- und Nachname

Datum

Reviewers

Name Erstgutachter
(Bioinformatik)
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

Name Zweitgutachter
(Biologie/Medizin)
Medizinische Fakultät
Universität Tübingen

Nachname, Vorname:
Title of thesis
Master Thesis Bioinformatics
Eberhard Karls Universität Tübingen
Thesis period: von-bis

Abstract

Write here your abstract.

Zusammenfassung

Bei einer englischen Masterarbeit muss zusätzlich eine deutsche Zusammenfassung verfasst werden.

Acknowledgements

Write here your acknowledgements.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Methodology

1.1 motor learning model

Spiking artificial neural networks (SNN) a subclass of ANNs work with time as second dimension. That means not only the activation of a neuron matters but also the point of time when the activation happened. In the model used in this paper every millisecond of time is simulated and not as usually for ANNs only time window steps that include all responses. Further on a continuous cell activation function is used. That is for each time an activation reaches a following cell the activation function gets altered and than returns over time to its initial state. it is important to note, that a biological neurons follows the "all or nothing principle". That means if a specific activation threshold is reached it fires. And returns than back to a initial activation level.

This work optimizes a motor spicing model introduced by Spüler et al. [SNR15][SNR15] which is an extension of the work of Chadderdon et al. [CNKL12]. The model presents a biologically realistic model of the motor cortex, which uses reinforcement learning to reach a target angel with a simulated forearm. As a standard reinforcement model [KLM96][CNKL12] it consists of an actor, that maps perceptions to actions a environment that reacts and a critic providing reward or punishment feedback to the actor. Spüler et al. and Chadderton et al. realized the environment as a forearm model with a one degree of freedom joint. Therefore the forearm could be moved either up or down. The joint angle limited by 0°and 135°. Whereby 0°means fully straightened and 135°means fully flexed. The actor got implemented with a spiking neural network. A spiking neural network is a weighted connected graph with some input and output nodes. Each edge is called a connection and each vertex called a neuron. The network calculates a time depending mapping from some input to output values. Figure 1.4 gives an overview over all components of the model as also of the reinforcement learning process. Each neuron in this model represents an either excitatory or inhibitory dynamic unit. Excitatory

neurons excite connected neurons, while inhibitory neurons inhibit connected neurons. The dynamics or in other words the time depending state of each neuron is represented by the membrane potential $V_m(t)$.$V_m(t)$ is modelled by an differential equation introduced by Izhikevich [Izh03], which is a good and efficient approximation of the Hodgkin-Huxley model [HH52]. Izhikevich's model slightly adapted by Spüler et al. with noise is described as follows (variable names changed):

$$
\begin{aligned}
V_m(t)' &= 0.04V_m(t)^2 + 5V_m(t) + 140 - u + I(t) + V_n(t) \\
u(t)' &= a(b \cdot V_m(t) - u(t)) \\
&\text{reset after spike:} \\
&\text{if } v \geq V_t, \text{ then } \begin{cases} 1 & \leftarrow V_r \\ 0 & \leftarrow u + d \end{cases}
\end{aligned}
$$

where u represents the membrane recovery variable. I defines the synaptic input currents. That is the sum of weighted inputs to a neuron.(TODO ref to ANN fundamentals) The firing pattern of a neuron depend on the choice of parameters a, b, d, V_r and V_t. Different resulting spiking patterns can be seen in figure 1.5. V_r is the resting potential of the neuron. V_t the spiking threshold. V_n is a 300Hz noise input that leads to motor babbling. That is, network send output signals although no input signal is given. This behaviour is important for reinforcement learning since if the actor doesn't produce any actions the critic isn't able to give feedback [CNKL12].The parameter a,b,c,d are chosen as one can shown in table 1.1.

Tabelle 1.1: Parameters used in the dynamic model

Parameter	Excitatory	Inhibitory
a	0.02	$0.02 + 0.08r_i$
b	0.2	$0.25 - 0.05r_i$
d	$8 - 6 \cdot r_i^2$	2
V_r	-65	-63

r_i is uniform distributed in $[0, 1]$. For excitatory neurons moving r_i from 0 to 1 leads to a transition from a regular to a chattering spiking pattern. For inhibitory neurons from a fast to a low-threshold spiking pattern. Examples of those patterns can be seen in (TODO ref appendix.). The neurons are further divided into 5 logical groups: proprioceptive (P), excitatory sensory (ES), inhibitory sensory(IS), excitatory motory (EM) and excitatory sensory (IS) neurons. Spüler's et al. model contains 48 P, 96 ES, 32 IS, 48 EM and 32 IM cells. Which leads to a total amount of 256 neurons. Each logical group is connected with a specific probability as shown in 1.2:

Based on these connections the resulting network is shown in figure 1.3.

Tabelle 1.2: Connection probabilities of neuron types in the model of Spüler's et al.

	P	EM	IM	ES	IS
P	0	0	0	0,1	0
EM	0	0	0,43	0	0
IM	0	0,44	0,62	0	0
ES	0	0,08	0	0	0,43
IS	0	0	0	0,44	0,62

The motor control cycle works as follows: A new movement angle is encoded by EM neurons, after a time gap of 50 ms the forearm moves. At the same time reward or punishment is send. After a second time step of 25 ms the new position angle is encoded by the proprioceptive cells. [SNR15] grounds the time gaps on peripheral and sub cortical processing delays. Why there is an additional delay for the P cells to fire but not for the reward keeps ungrounded. The whole system works with a frequents of 0.02kHz. That means the EM neurons encode a new angle every 50ms. An overview of this process can be seen in figure 1.6. The time needed by EM cells to encode a new angle is again 50ms. EM cells are parted into two equal sized groups of 24 neurons. The first 24 are in the first group the next 24 in the second. Each firing of a cell in the first group leads to an 1°downward motion of the arm; in the lower part it leads to an 1°upward motion. Over the time window of 50ms all spikes in each group are summed up. The resulting movement angle the difference of those sums. Whether reward or punishment is send by the critic is decided whether the distance $\Delta\theta_t$of the current angle θ_t to the target angle θ_{target} got smaller or higher compared to the mean of the last two differences $\Delta\theta_{prev}$. This can be seen in the following formula:

$$\begin{aligned} \Delta\theta_t &= |\theta_t - \theta_{target}| \\ \Delta\theta_{prev} &= |\theta_{t-1} - \theta_{target}| \\ \text{critic response} &= \begin{cases} \text{reward} & \text{if } \Delta\theta_t < \Delta\theta_{prev} \\ \text{punishment} & \text{if } \Delta\theta_t > \Delta\theta_{prev} \\ \text{no response} & \text{if } \Delta\theta_t = \Delta\theta_{prev} \end{cases} \end{aligned}$$

Note that reward or punishment doesn't influence every connection. Connections according to the Hebbian theory [Heb49] are only able to learn if a post-synaptic spike followed a pre-synaptic spike in the 50 ms where the movement to the current angle got encoded. Further on only connections from ES to EM neurons are able to learn. (TODO Why find prove). Reward or punishment increases or decreases a weight scale factor of each connection that is able to learn. For more informations look Appendix. TODO just copy picture. A weight lies in the interval [0,5].

Depending on the current joint angle All P cells are assumed to be located 0.5 space units apart. The position p_theta of the joint angle is calculated by:

$$p_\theta = \frac{1}{(\theta_{max} - \theta_{min}) * 2} = \frac{1}{135 * 2} = \frac{1}{270}.$$

. p_θ is a scaled position on interval $[0, 67.5]$. Now a normal distribution with mean p_theta and standard deviation 0.8 ($\mathcal{N}(p_theta, 0.8)$) is considered and neuron positions are applied to all. Note that this is identical to placing a normal distribution to every neuron position and reading out the value at p_theta for each (see [Mal13]). The result multiplied by 2 is the probability that the neuron fires or not. Multiplying by 2 is necessary to provide a maximal probability near to 1 (0.997356). In average 5 neighbouring cells will fire to encode an angle. Nagel [Nag] refers to this kind of coding as population coding. Note that this description doesn't correspond with the exemplar formula from Nagel [Nag, p 18]. It is reversed engineered form Nagels source code. An example of this encoding is shown in 1.7. The model described so far is the öngoing learning model"where learning is always necessary to reach a target angle. Spüler and Nagel extended this model to be able to reach every target angle without learning after a learning phase. This extended model is called ßtatic learning model". Only slight changes have been done. Instead of 96 P neurons describing the current angle, 96 D Neurons describing the distance from the current position to the target position are used. Population coding is implemented nearly identical. The only difference is, that p_θ describes now a scaled distance from the former in the interval $[-135, 135]$. Since the direction of the distance has also to be encoded. Detailed information how p_θ is calculated are provided in [Nag, p. 41-43] and [SNR15]. Instead of learning on ES-M synapses. Learning is now applied on D-ES synapses since this lead to better results. In addition a mechanism called Structural Synaptic Plasticity (SSP) is added to D-ES synapses. If a connection is rarely used between two neurons it connects to a new output neuron. This is implemented the way, that if a weight scale factor drops below 0.2 the connection will be randomly reconnected to a not yet connected ES neuron. The learning phase consists of the task to start at maximum angle and than consecutively reaching the minimal and maximal angle again. Since most of the angles of the whole distance space from -135 to 135 will occur, Spüler and Nagel argue that this procedure is sufficient to learn every other angle. After the learning phase a simulation phase is undergone to check how the model behaves with different target angles.An exemplar learning and simulation behaviour can be found in figure 1.1 and figure 1.2.

TODO Appendix picture weight formula todo. TODO Appendix picture sebastian al

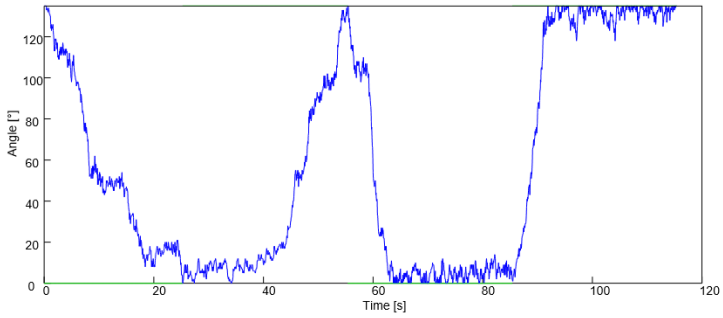

Abbildung 1.1: Exemplar Learning Phase: Starting from angle 135°, the angles 0°and 135°have to be reached. This example needs 50 seconds to achieve this. During this 50 seconds learning is enabled. Afterwards it is checked whether the model is able to reach the minimal an maximal angle without learning. Both angles are applied as target angles for 30 seconds. Not that time is referred to as simulation time not as actual time. [Nag, p. 47]

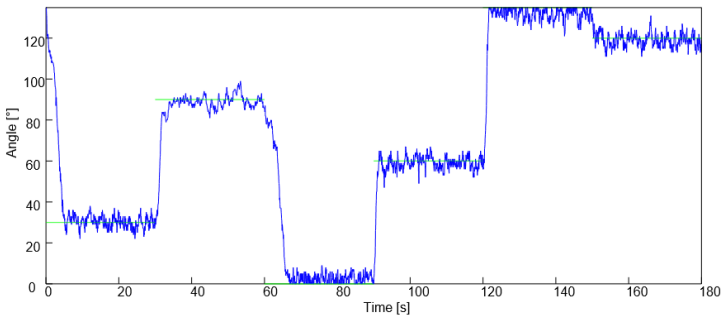

Abbildung 1.2: Spüler's simulation with lowest Overall RMSD: After learning the model gets tested in a simulation where several angles have to be reached. Starting with 135°consecutively 30°,90°,0°,60°,135°and 120°have to be reached. Each target angle is presented for 30 seconds. This model is the best performing model found by Spüler et al. with an RMSD of 3.3°. [Nag, p. 52]

1.2 general approach implementation

The HyperNeat experiments where build upon the HyperSharpNEAT HyperNEAT framework implemented by David D'Ambrosio in C# (source code [D'A]). For ES-HyperNeat experiments the ES-HyperNEAT framework by Sebastian Risi which is an extension of HyperSharpNEAT is used (source code [SR]). Note that a probable faster C++ Hyperneat implementations (source code [Che]) is available but not for ES-HyperNEAT. The motor control model code is given in MATLAB Programming Language. It was extended with the use of MATLAB 9.2.0.556344 (R2017a). The evolution process is done in the extended HyperNEAT framework. To determine the fitness of an individual the C# process communicates with a motor control matlab instance. The

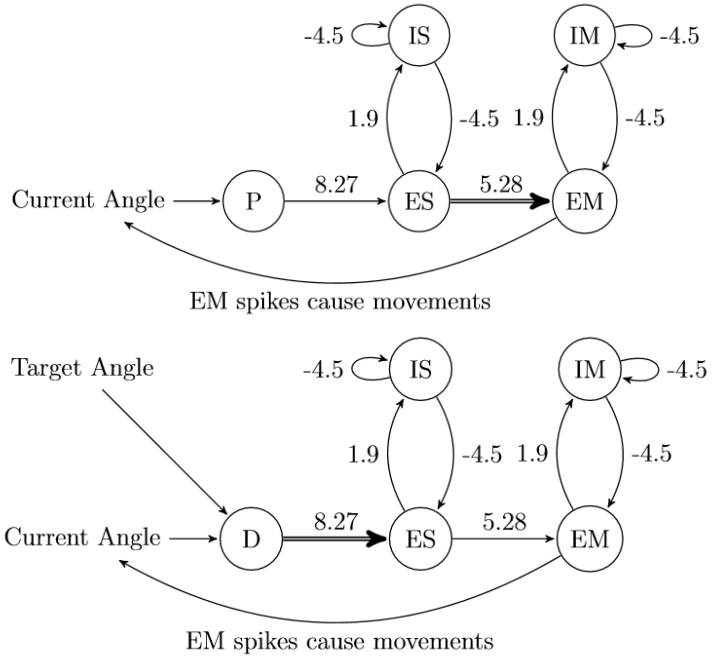

Abbildung 1.3: Spüler et al. actor model structures: The upper figure shows the structure of the Ongoing Learning Model. The lower figure the structure of the Static Learning Models. Each circle represents a cell type. Each arrow represents a connection probability greater 0. The double lined arrow represents inter cell connections where learning is applied. [Nag, p. 25,41]

communication is based on a communication file a result file and a matlab file describing the evolved parameters for the motor control model: e.g. the weight matrix. The C# process evaluates several genomes concurrently by communicating with several matlab instances. Each motor control model implementation waits until a evaluation command arrives and waits again after successful evaluation. In that way matlab instances can be reused and long starting times of them are prevented. The architecture can also been seen in figure 1.8; To write matlab files with C# the CSMatIO library from David Zier (web link: [Zie]) was used. Two attempts to provide a direct not file based communication between C# and matlab have been tried. The first attempt was based on communication over Microsoft's Component Object Model (COM) interface. Although less information about the usage of this interface with matlab are existing, this way of communication got successfully implemented. Unfortunately after finishing this solution, it became known that although promised no windows server will be access able. The COM interface only works on Windows. The second approach was to compile the matlab code directly to a C# library using matlab's Application Compiler. For single process application this approach would work fine but unfortunately only one thread is allowed

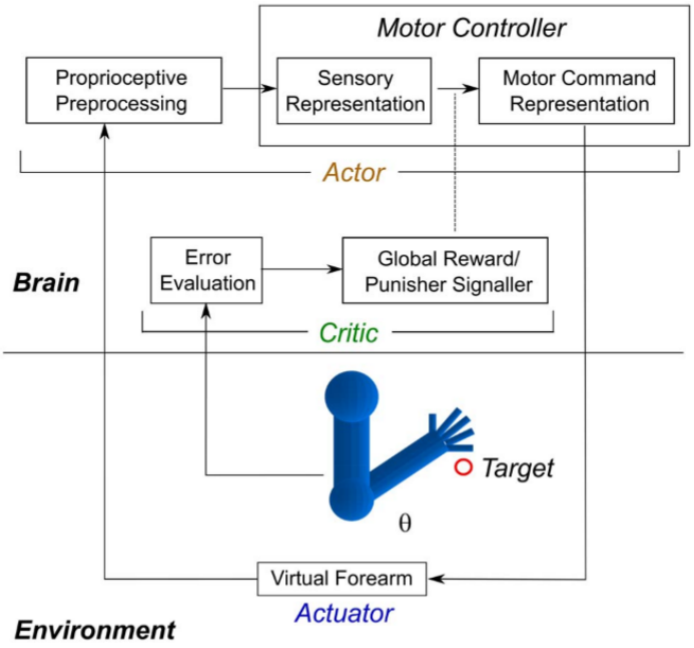

Abbildung 1.4: Spüler et al. whole model design: This figure gives an whole overview over Spüler et al.'s model components and its reinforcement learning circle. The critic and actor ar parts of the brain (or more general the learning system). The forearm is part of the environment. A detailed view of the actor's structure can be found in **??**. [CNKL12, p. 3].

.

to access the library, which makes parallel genome processing impossible. An further problem occurred was, that matlab crashed due to segmentation faults in the matlab core libraries at unpredictable point of times. This was no error of the code of the motor control model, since the crashes occur only when running multiply independent matlab instances at the same time. After contacting Matalab support they stated, that this should not happen, but no solution has been found. Thankfully the problem was not severe since a crashed Matlab instance is simply restarted again. Further on only between 2 to 10 crashes happened during a runtime of 60 hours. So no new compilation is necessary to change parameters of an experiment. The parameters for HyperNEAT in general and the implemented experiments can be set by specific configuration text files. All experiments run on a linux server with 64 cores with a 64 bit architecture and clocked with 2.3GHz. The available RAM was 251.9GB. Since some precessing researches where reserved for other researcher only 50 single core matlab instances have been used. To run the C# code on a linux server the mono project environment (web link: [Fou]) was used. It took approximately 60 hours to run an typical experiment with 450 generations and a population of 90 individuals. Thereby 40500 individuals were evaluated.

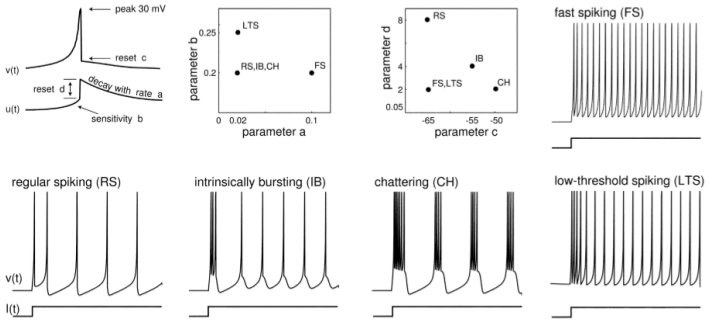

Abbildung 1.5: Izhikevich simple model [Izh03] spiking patterns. Different resulting spiking patterns by variation of $a, b, c, d = V_r$ patterns. The here depicted patterns are correspond to known biological neuron types. RS, IB and CH are cortical excitatory neurons. FS and LTS are cortical inhibitory neurons. Time resolution is 0.1ms. Each pattern is initialized by a dc-current (I) step from 0 to 10. [Izh03]

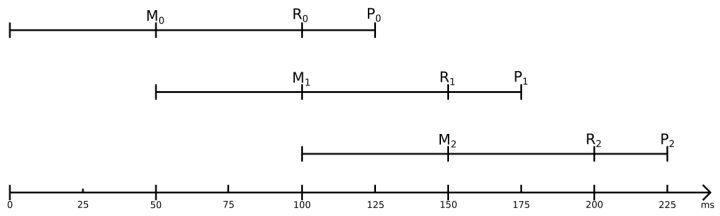

Abbildung 1.6: 3 exemplar consecutive motor control cycles: Until M a new motor signal gets encoded by EM cells. Than until R there is a 50ms time gap. At R the motor arm is moved and the critic responses. After 25 more milliseconds the new position is encoded by the P cells. Note that P cells get activated during the encoding of the next motor command. The time gaps are biological founded by Spüler et al as peripheral and subcortical delay times.

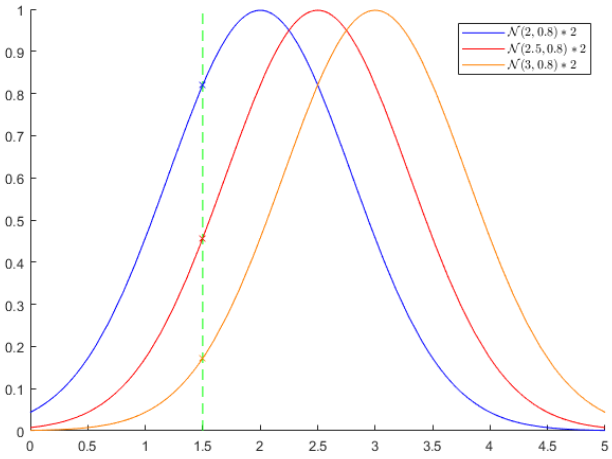

Abbildung 1.7: Exemplar Population coding by D cells. Here the normal distributions of activation probabilities for neurons at 2 (blue), 2.5 (red) and 3(orange) are shown. The dashed green line marks the position of an angle to encode. Note that every cell will fire with a different probability. ($P(blue) = 0.8204$, $P(red) = 0.4566$, $P(orange) = 0.1720$)

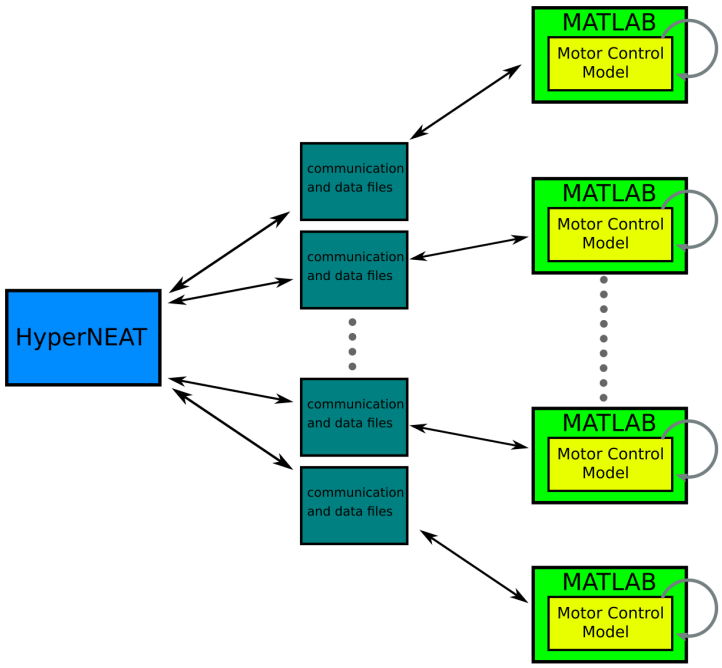

Abbildung 1.8: experiment architecture: One hyperNEAT process communicates over communication and data files with several motor control model to concurrently evaluate genomes. Each motor control model runs in it's own matlab instance. The gray dots point to the existence of multiply more Matlab instances

Kapitel 2

Experiments

2.1 HyperNEAT applied to the Ongoing Model

Methodology vllt: Behavioud of ongoing and static model. TODO

2.1.1 ES-EM learning

description

To ensure that HyperNEAT works on this filed of application, a random connection sample was chosen. ES-EM connections were evolved by hyperNEAT all other connections were kept as randomly initiated. The aim of this experiment is to show that hyperNEAT can improve the moving behaviour of the arm by searching for an optimal ES-EM connectivity. That learning only the ES-EM connectivity is sufficient for to get an good behaviour was empirically proven by [SNR15]. (This experiment is parametrically equal to Nagel's ongoing experiment with $ppd = 1, pa = 1, w_e = \tau_{remaining/20}$. Note that those parameters haven't been explained here directly but they are identical with the way the Ongoing Model was explained in this work todo ref). Direct comparison between Nagel's and this experiment is not possible since here one pattern for all connections except of ES-EM were used while Nagel chose 100 different ones. The amount of neurons was set to 256 with the same type distribution explained in the methodology (TODO ref). Since the evaluation of the motor control model is computation expensive hyperNEAT searched only for an optimal solution to reach angles of 15°and 115°starting with an angle of 65°. Although training with more angles would be more promising, the basic concept of reacting to the critics input should be learned anyway. Further on the time to reach an angle was decreased from 120 to 40 seconds compared to Nagel to decrease the calculation time. This is justified because derived from

the low rmsds of Nagels models couldn't needed much more than 20 seconds to reach their target angle. The fitness function $f_{ongoing}$ is given by:

$$f_{ongoing} = 135 - \frac{\sqrt{\frac{1}{40}\sum_{t=7}^{40}(\theta_t - 15)^2} + \sqrt{\frac{1}{40}\sum_{t=7}^{40}(\theta_t - 115)^2}}{2} \tag{2.1}$$

where θ_t is the forearms angel at time t. Both square root terms calculate the RMSD from a run from 7 to 40 seconds.(Note that this is the same scale than 20 to 120 seconds).((Note that the time gap of 7 seconds is not necessary but makes the fitness more clear since the maximum value could theoretically be reached. 450 generations were simulated and a population size of 90 were chosen. Although 50 concurrent matlab instances were available a population size of 90 showed a better mean time performance for a individual than a population size of 100 or 80. Assuming an average calculation time of 8 minutes per generation (TODO real numbers) results in a calculation time of 60 hours. Whereby 40500 models have been evaluated. The connection threshold θ_c is set to 0.8. The activation function of output neurons is set to a bipolar sigmoid function ($\frac{2}{1+\exp(-4.9x)} - 1$) with a value domain of $[-1, 1]$ next to $f(x) = 2x$ on the x interval $[-0.4, 0.4]$. If the output value of the sigmoid function is higher than 0.8 a connection is realized width the initial weight for this connection defined by the ongoing model. Note that weight scaling as usually done in HyperNEAT is not applied. Thus assuming the output of the sigmoid function is equal distributed between -1 and 1 the threshold of 0.8 leads to a connection realisation rate of 0.9. (ncessary?). There are 4 types of activation function which a new node in the CPPN can get with equal probability. The available activation functions where bipolar sigmoid, Gaussian ($2\exp(-(2.5x)^2)-1$), absolute value ($|x|$) and sine ($sin(2x)$). (TODO plot?) Which have been empirically proven to be successful for hyperNEAT experiments by [DS08]. (at least for one todo) Those are the base functions for the patterns created by CPPNS. Based on the fact that cells of the brains sensory and motory cortex are aligned to each other along the approximately straight sulcus centralis, the cppn substrate configuration is chosen as two parallel lines of ES and EM neurons. The exact configuration can be seen in figure 2.1.

The used NEAT parameters are as follows: A add connection mutation appears with a probability of 0.15, a add node mutation with 0.05 and weight mutations with 0.96 for each individual per generation. If a individual is selected to have a weight mutation depending on 5 equal probable metrics a varying amount of connections gets either assigned a randomly chosen new weight value or a even distributed value is added to the weight. The elitism proportion is 0.1 which complies with 9 individuals. All CPPN connection weights were set to be in the interval $[-1.5, 1.5]$.

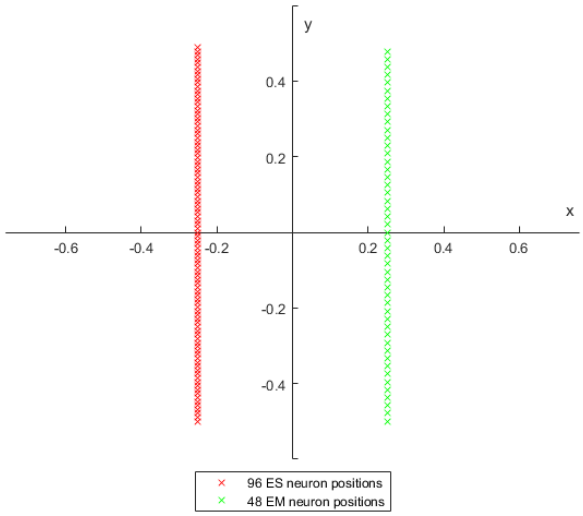

Abbildung 2.1: Ongoing experiment substrate configuration: 96 Es neurons are placed with equal y distances between $(-0.25, -0.5)$ and $(-0.25, 0.5)$. 48 EM Neurons are placed accordingly at $x = 0.25$

TODO put neat params in general descriprion. TODO further on just give table with parameters.

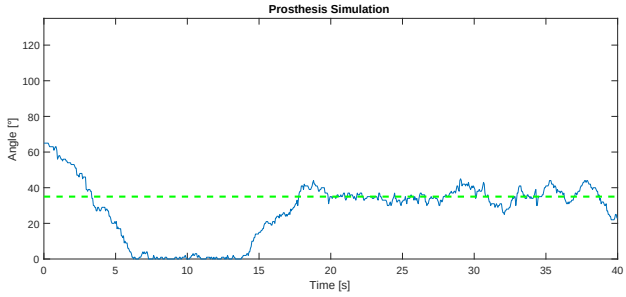

Abbildung 2.2: Training behaviour of the initial ongoing model: Sarting from 65°a 35°angle is to be reached. The RMSD is 17.7°and calculated from 7 seconds onwards.

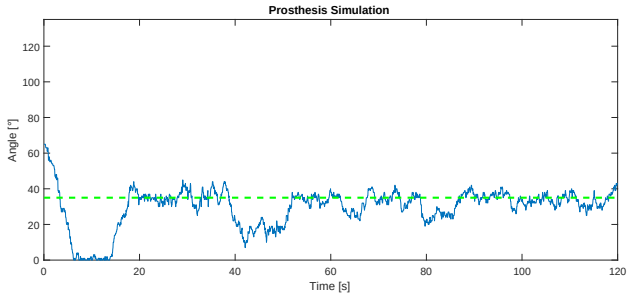

Abbildung 2.3: Simulation behaviour of the initial ongoing model: Sarting from 65°a 35°angle is to be reached. The RMSD is 7.5°and calculated from 20 seconds onwards.

result

discussion

2.2 HyperNEAT applied to the Static Model

2.2.1 D-ES learning

description

result

discussion

2.2.2 All Connections 2D neuron space

description

result

discussion

2.2.3 All Connections 1D neuron space

description

result

discussion

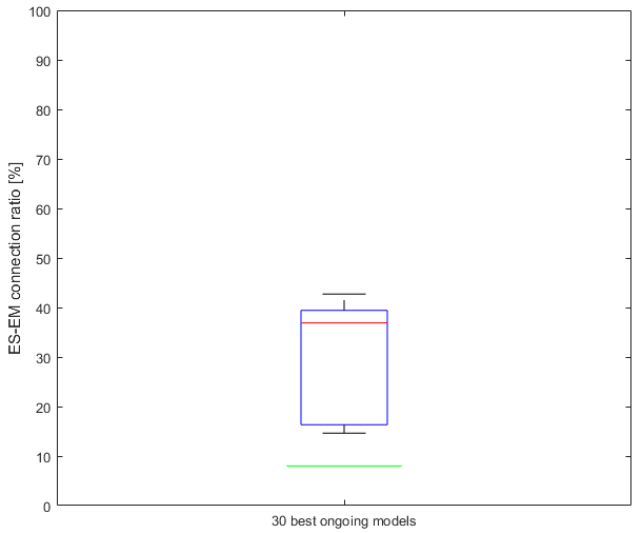

Abbildung 2.4: Ongoing Experiment ES-EM connection ratios: The connection ratio of the initial model is indicated by the green line.The mean of the 30 best ongoing model is represented by the red line. The box represents the interquartile range (IQR) from the first to the third quartile . The whiskers extend the quartiles by $\pm 1.5 \cdot IQR$. Red crosses show outlier.

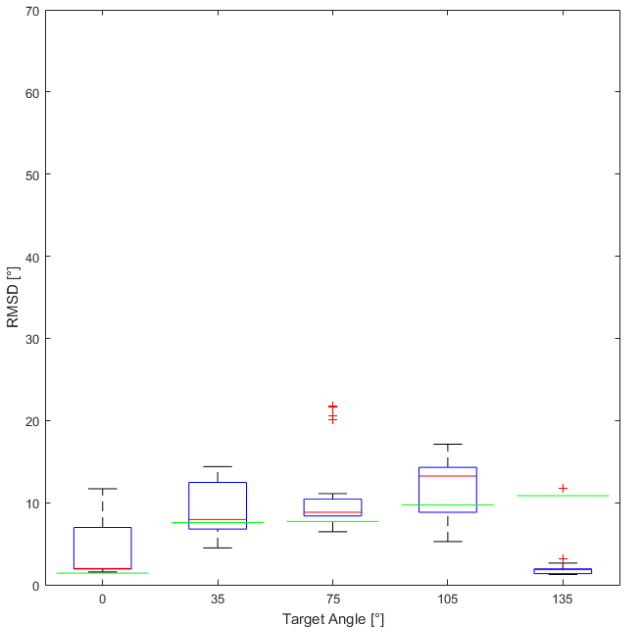

Abbildung 2.5: Ongoing Experiment RMSDS: The RMSD distribution of the 30 best ongoing models of HyperNEAT's last generation. Each Boxplot corresponds to one target angle: 0°, 35°, 75°, 105°and 135°. The initial angle was 65°. The RMSD of the initial model is indicated by the green line. The mean RMSD of the simulated models is represented by the red line. The box represents the interquartile range (IQR) from the first to the third quartile. The whiskers extend the quartiles by $\pm 1.5 \cdot IQR$. Red crosses show outlier.

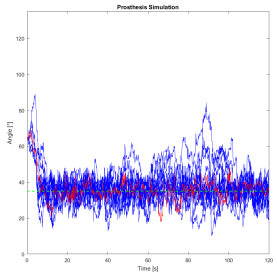

Abbildung 2.6: Ongoing Experiment mean movements: The movements of the 30 best models of HyperNEAT's last generation are represented in blue. The red line is the mean movement. Form a start angle of 65°the target angle of 35°was to be reached. The RMSD is ca 8 (todo calculate boxplot matlab mean)°and calculated from 20 seconds onwards.

Literaturverzeichnis

[Che] Peter Chervenski. Multineat c++ with python bindings. `http://multineat.com/`. [Online; accessed 29/08/2017].

[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.

[D'A] David B. D'Ambrosio. Hypersharpneat 2.1. `http://eplex.cs.ucf.edu/software-list#hypersharpneat`. [Online; accessed 29/08/2017].

[DS08] David B D'Ambrosio and Kenneth O Stanley. Generative encoding for multiagent learning. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 819–826. ACM, 2008.

[Fou] .NET Foundation. Mono cross platform, open source .net framework. `http://www.mono-project.com/`. [Online; accessed 29/08/2017].

[Heb49] Donald Olding Hebb. *The organization of behavior: A neuropsychological approach*. John Wiley & Sons, 1949.

[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):500–544, 1952.

[Izh03] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[Mal13] Hanspeter A Mallot. *Computational Neuroscience*. Springer, 2013.

[Nag] Sebastian Nagel. Using spiking neural networks to simulate human motor learning.

[SNR15] Martin Spüler, Sebastian Nagel, and Wolfgang Rosenstiel. A spiking neuronal model learning a motor control task by reinforcement learning and structural synaptic plasticity. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[SR] David B. D'Ambrosio Sebastian Risi. Es-hyperneat. `http://eplex.cs.ucf.edu/neat_software/#ES-HyperNEAT`. [Online; accessed 29/08/2017].

[Zie] David Zier. Csmatio: Mat-file i/o api for .net 2.0. `https://de.mathworks.com/matlabcentral/fileexchange/16319-csmatio--mat-file-i-o-api-for-net-2-0`. [Online; accessed 29/08/2017].