Optimization of Artificial Central Pattern Generators with Evolutionary Algorithms

Christian Bauer, Sebastian Braun, Yang Chen, Wilfried Jakob, Ralf Mikut

Institute for Applied Computer Science, Forschungszentrum Karlsruhe
76021 Karlsruhe, Postfach 3640
Tel.: +49 7247 82 6672
Fax: +49 7247 82 5786
E-Mail: christian.bauer@iai.fzk.de

Abstract

In contrast to classical engineering approaches for the generation of movements in robots or prostheses, approaches to this subject inspired by neurophysiological circuits are in advance. One of the key structures of interest in this area is the Central Pattern Generator (CPG) which has been identified to be the source of movement generation in mammals. This neural circuit is capable of generating cyclic muscle activation patterns completely independent from the brain as shown by Brown in the early 20th century [1, 2]. In the past years this knowledge was adapted to the challenges in movement generation and control in robotics, for example for passive dynamic walkers by P. Manoonpong [3].

In this paper the results of a CPG implementation and its optimization to model the human movement generation are presented, which were aiming at a design to be as biologically realistic as possible. Due to the fact, that biological models are the fundamentals of the simulated CPG, a large number of parameters needs to be set, which results in a very complex configuration process to get an optimized CPG behavior. Based on previous works [4–7] an Evolutionary Algorithm was employed to optimize the rhythm-generation of the CPG. For this optimization the tool GLEAMKIT was used [8]. Electromyographic (EMG) data recorded from a human male during walking on a treadmill have been utilized for the fitness function as well as for evaluation purposes.

1 Introduction

In computer sciences, robotics research and development of prostheses, artificial neural networks are nowadays common tools to increase the possibilities and power of all different kinds of systems, e.g. object classification, learning, speech recognition or movement generation. To build these systems a lot of different approaches are possible which bare several advantages and disadvantages depending on specific tasks for which they are developed. These approaches can be split into two main strategies, on the one hand side there is the 'classic technical' way of building an artificial neural network which evolved out of computer science. Few neurons are organized in layers and connected with each other by weighted data streams. On the other hand there is the biological inspired approach which relies on the findings in neural sciences and uses differential equations for simulating behavior of biological neurons during the data transmission in animals or humans. One of the most common models for such a biological realistic simulation is the Hodgkin-Huxley Model, published in 1952 [9].

One of the most popular concepts in humanoid robotics to generate coordinated leg movement is the one of Central Pattern Generators (CPGs). These are neural structures which have been discovered in the 20th century [15] by testing the locomotive capacities of decerebrated cats. The ability of cats, or mammals in general, to be able to walk or trot in an almost normal walking pattern, even with their cerebellum or spine surgically separated from the motor cortex, indicate, that basic gait patterns are encoded in the neural networks of the spine. These neural structures are called Central Pattern Generators and have been found to be responsible for generation of cyclic muscle activation patterns such as respiration, chewing or leg movement during walking. These CPGs are one part of the neural system which controls posture and movement, which is hierarchically organized and of a highly complex, nonlinear structure (see Figure 1).

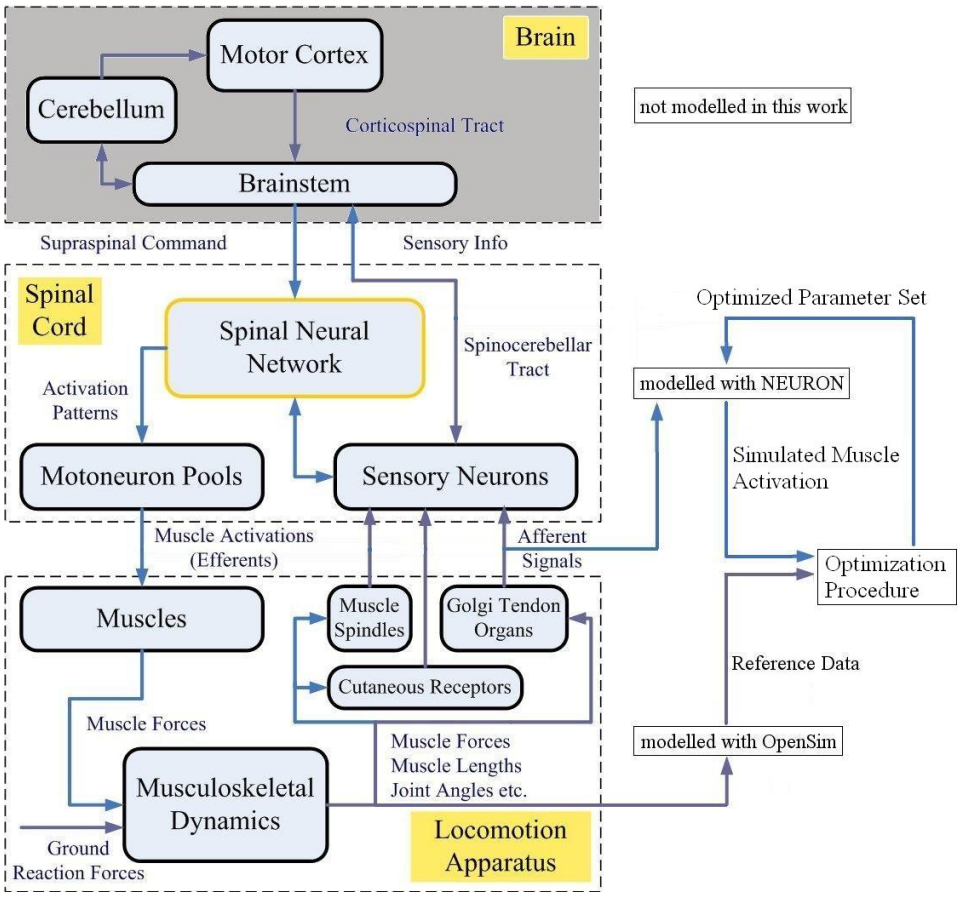

Figure 1: Locomotion control system in humans

This concept has been adapted by many scientists for movement generation in robotics. Matsuoka published his first concepts of artificial neural oscillators in 1985 and 1987 [10, 11]. The concept of CPG has been applied to passive dynamic walkers developed by F. Wörgötter and P. Manoonpong [3, 12] (RUNBOT) or Mori et al. [13]. These approaches have in common that the CPGs used are to a certain degree simplified artificial neural networks which just adapt the idea of CPGs for the task of bipedal locomotion.

In the research group at the Institute for Applied Computer Science (IAI) at the Forschungszentrum Karlsruhe the subject of interest is movement generation for prostheses and robots. The approach used here is of a different kind as the afore-

mentioned and uses biological realistic simulations of neurons and neural networks to simulate CPGs and to generate muscle activation patterns in a way which is biologically as realistic as possible. The first results have been presented in [4] containing simulation results of a CPG with a heuristically found parameter set, without the necessary coupling to a biomechanical simulation environment.

In this paper a more sophisticated simulation of a CPG is presented as well as the application of the Evolutionary Algorithm (EA) GLEAM to optimize the parameters of the CPG network so that the muscle activation pattern becomes as human like as possible. For this several different types of software tools had to be combined to operate in an integrated simulation and optimization platform. This platform has been evaluated regarding different sizes of the parameter sets (four or 14 parameters) and the scalability of the optimization process regarding distributed computing.

This paper is structured as follows. In Section 2 the used tools and principles are explained as well as how these are combined to form the system. In addition to that the methods used for optimizing the system are introduced in this section. Following this, the Evaluation section (Section 3) addresses the evaluation and test scenarios with which the system has been confronted. The results of these tests are presented and explained. Following this section, a discussion of the results can be found in Section 4 and future work is motivated. Section 5 concludes this paper with a short summary of the contents of this paper.

2 System and Methods

In this section the software tools used to build the simulation system are introduced as well as the proceedings and methods of simulating a CPG. Following that, it is explained how the CPG-generated activation patterns are compared to the reference data as well as the procedure to optimize these activation patterns.

2.1 Architecture for Simulation and Optimization

The platform developed consists of two main parts, the simulation and the optimization, and can be seen in Figure 2.

For the simulation of a CPG the neural simulation software NEURON published by a group of researchers at the University of Yale[1] is used. This tool is a programming environment specialized on generating and designing neural structures consisting of several neurons. The strength of NEURON lies in its ability to calculate the differential equations, which describe neural behavior. In principle, these neurons are laying at rest and the transmission of information along these neurons is done by action potentials, which means that specified currents of potassium and calcium ions are diffusing through the cell membranes sending a current along the axons. The duration and amplitude of these are identical to the according currents in biological neuron. The same is valid for the membrane and action potentials and the settling rate of the neurons. Using NEURON an artificial neural network was implemented which contains all parts that are responsible for movement generation in mammals. In Figure 3 the structure of this CPG network, the involved neurons and their interconnection is illustrated.

Former research results [14–16] prove that neural structures like this one exist in mammals and are the crucial part in the generation of cyclic motion patterns. Coupling these CPGs the generation of more sophisticated locomotion patterns is possible. To achieve this, each joint has to be controlled by one CPG, which means

[1] `http://www.neuron.yale.edu/neuron/`

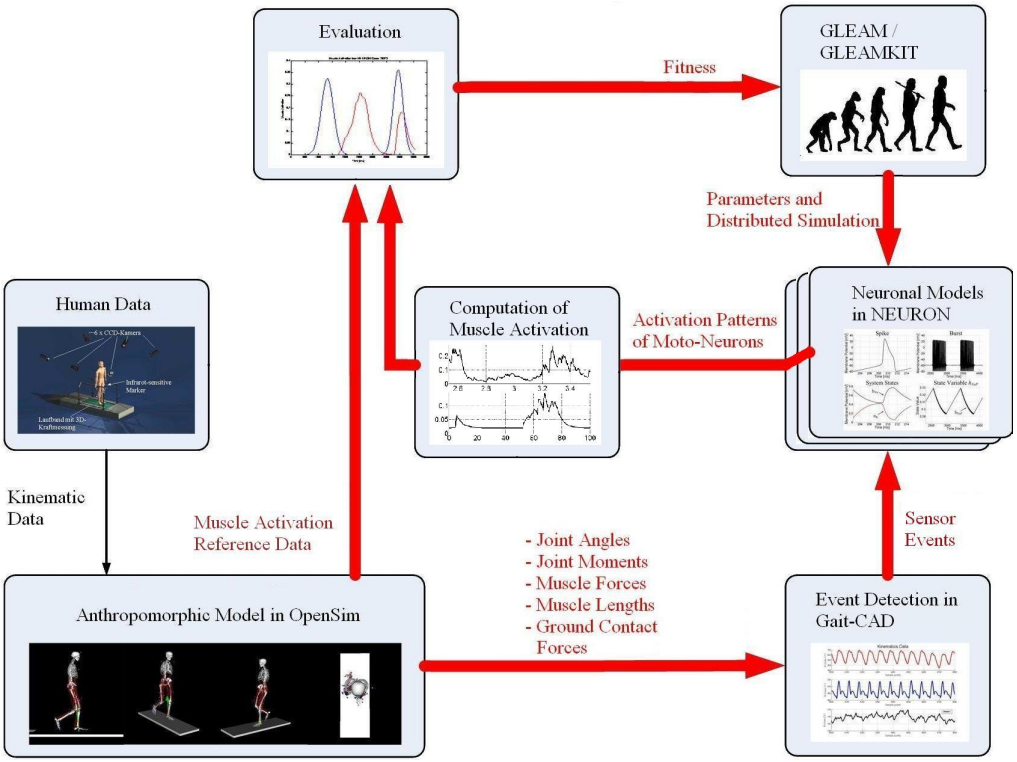

Figure 2: Overview of the software tools building the system

that several muscle activations for the flexors and extensors of this joint have to be generated by this CPG.

The biomechanics simulation Software OpenSim is a freeware tool for the analysis of muscle forces in motion capturing data recorded from patients or test persons to calculate direct and inverse kinematics and dynamics. It uses free available models from the commercial software SIMM and is available as open source from the projects homepage[2]. Using OpenSim the motion data which has been recorded at the Orthopädische Klinik in Heidelberg, Germany, was used to calculate the muscle activation signals from the test person. The motion data for this calculation was recorded from a 23 years old human male via a commercially available 3D motion analysis system. For the marker placement the Helen Hayes marker set was used and the motion was performed on a treadmill running with different speeds. In addition to motion data, EMGs of eight prominent muscles in both thighs and shanks were recorded.

To use the benefit of biological realistic simulation the activation patterns generated by the CPG programmed in NEURON have to be as close to real human muscle activation patterns as possible. The pattern generation in NEURON can be adjusted by a large number of parameters which influence the output in a very complex way. To match this artificial activation to the recorded human muscle activation an Evolutionary Algorithm is employed to find a best matching result for the desired purpose.

EAs are able to deal with a large number of parameters and control figures like it is the case of the presented simulation. The simulation framework is extended by the GLEAM engine (General Learning Evolutionary Algorithm and Method [7, 17]) and GLEAMKIT which have been developed at the IAI [18, 19]. The former

[2] https://simtk.org/project/xml/downloads.xml?group_id=91

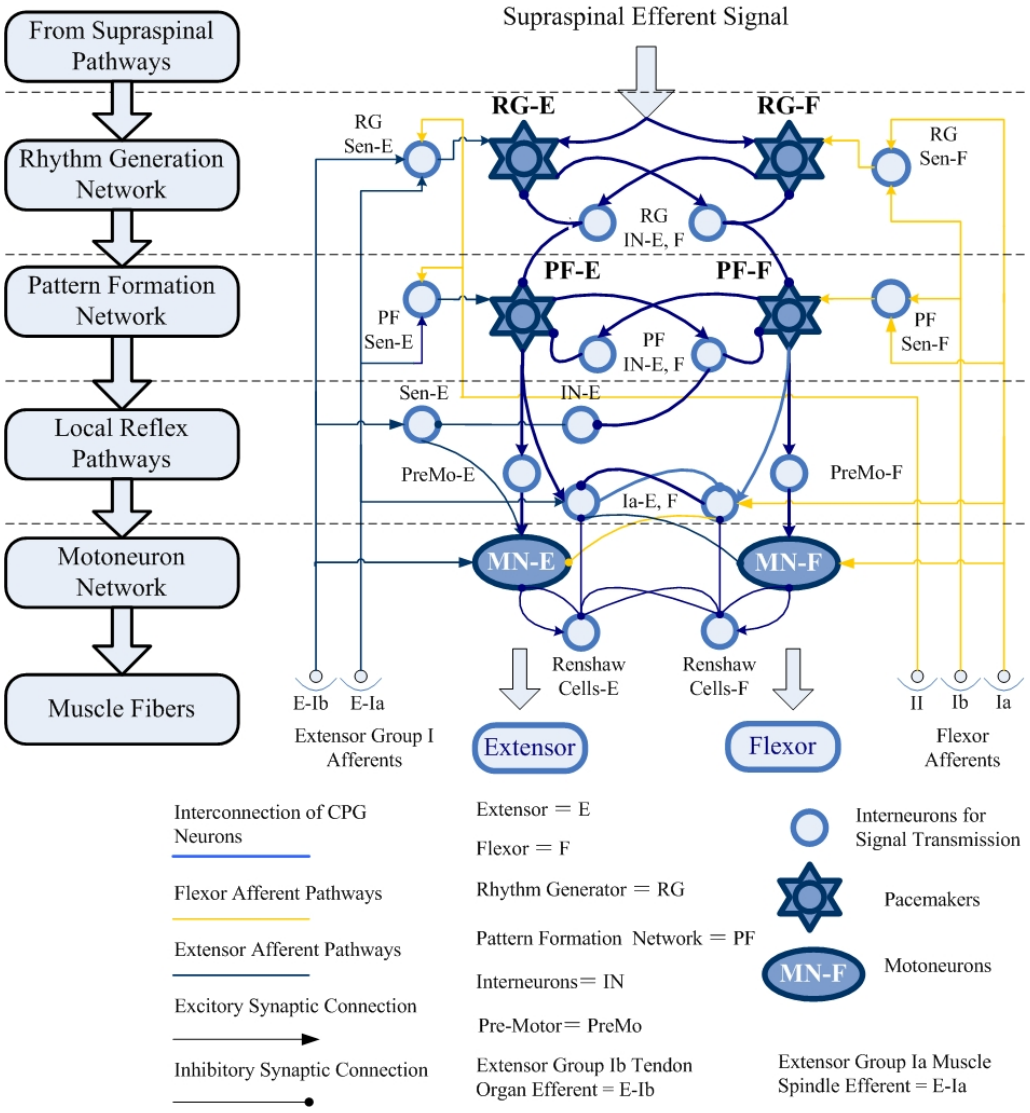

Figure 3: Structure of the neural connection of a CPG

one is an open source implementation of the evolutionary algorithm GLEAM with interfaces to Windows and Matlab [18], the latter one is a Matlab toolbox forming a GUI for GLEAM for configuring the EA and the optimization runs. In addition to that an online visualization frontend is provided. These two toolboxes are used to apply the EA GLEAM to optimize the parameter sets of the simulation to match the generated artificial muscle activation patterns as close as possible to the muscle activation patterns of human locomotion.

The described simulation and optimization system is completed by the Matlab toolbox Gait-CAD for filtering and evaluation tasks. This toolbox has been developed at the IAI and is specialized for filtering, visualization and data mining in large datasets and time series [20].

Considering the fact that each simulation to evaluate one genome model of the EA takes a certain amount of time to be completed (8 to 50 seconds, depending on the computer), the concept of distributed computing was used to enhance the performance and time effectiveness of the EA application. To achieve this the IAI-DataShare software, which is a Client-Server-Architecture for assessing data based

jobs for distributed computing was coupled with the system. IAIDataShare consists of one server instance which distributes the jobs to registered clients which calculate the simulation depending on the individuals of this EA-population. Each of these contains a NEURON parameter set and computes the resulting activation patterns of moto-neurons. If finished the results are written to a file which then is copied to the results directory by the server.

2.2 Methods - Generation of Muscle Activation

The problem of crosstalk between the recorded EMG signals made it impossible to differentiate exactly between the muscle activation of the targeted muscle and the activation of surrounding muscles involved in the locomotion. Therefore in our application, OpenSim has to be called to extract muscle activations and events such as ground contact from the kinematic data. This has to be done only once to get a reference dataset for the optimization process.

The CPG is built up of different neurons, implemented in NEURON. Former research has shown that a combination of neurons modeled with the Hodgkin-Huxley (HH) model [9] and Integrate-and-Fire neurons [21] is the way of choice in respect of computation efficiency and closeness to reality [5]. The HH-Model was used to build up the pacemakers and moto-neurons whereas the Integrate-and-Fire model was applied for building the interneurons. The advantage coming with the Integrate-and-Fire neuron model is that its computation time is proportional to the number of events and independent of the number of neurons involved, because no numerical integration is required but an analytical solution is used instead. This type of model is especially useful for discrete event simulation, which are characterized by only time relevant events such as spikes [21].

This model works in a very simple way. Mathematically it can be described by

$$\text{if spike comes, then } m_{IntF} = m_{IntF} + w \tag{1}$$

$$\text{if } m_{IntF} > 1, \text{ then } \begin{cases} \text{deliver a spike event} \\ \tau \frac{dm_{IntF}}{dt} + m_{IntF} = 0 \end{cases} \tag{2}$$

where the system dynamics is determined by a single membrane state variable m_{IntF}. If the neuron receives a spike event from the connected neuron, the membrane state value is increased by w, defined as the synaptic weight. If m_{IntF} exceeds 1, the neuron fires and a spike event is delivered. The membrane state variable decays exponentially with the time constant τ after firing. Refractory period can be easily added to the model by defining for how long the neuron can not receive new spike events after firing.

In contrary the HH neuron model is mathematically more complex and describes the membrane potentials of single neurons very realistic. Its general form is as follows:

$$C_m \cdot \frac{dV_m(t)}{dt} = -\sum_{i}^{n_{Ion}} I_{Ion,i}(t) - \sum_{i}^{n_{Syn}} I_{Syn,i}(t) - \sum_{i}^{n_{Ext}} I_{Ext,i}(t) \tag{3}$$

in which $C_m = 1\mu F/cm^2$ is the membrane capacity, and V_m the membrane potential. $I_{Ext,i}$ represents external influences like afferent inputs or flexor/extensor phases, $I_{Syn,i}$ is the synaptic current built as the product of the maximum conductance $\overline{g}_{Syn,i} = 0.05mS/cm^2$ of the synapse with the connection weight $w_{Syn,i}$ and time relevant variables ($t_{ls,i}$ for the time of the last spike and $\tau_{Syn,i} = 5ms$ as time constant for the synaptic current, n is the number of the connected source neurons).

$$I_{Syn,i}(t) = \overline{g}_{Syn,i} \cdot w_{Syn,i} \cdot \exp\left(\frac{-t - t_{ls,i}(t)}{\tau_{Syn,i}}\right) \cdot (V_m(t) - E_{Syn,i}) \quad (4)$$

The differentiation of inhibitory and excitatory synaptic currents is achieved by different values of reversal potentials $E_{Syn,i}$ which are about -40mV and -10mV respectively.

The ion currents $I_{Ion,i}$ are used to model the characteristic behaviors of a neuron by a combination of the three different types of involved ion channels, Sodium, Potassium and leakage. To model the CPG typical behavior the persistent sodium current I_{NaP} has been incorporated. Together with the three former currents of the HH model (I_{Na}, I_K and I_L for the sodium, potassium and leakage, respectively) the ion currents of a pacemaker neuron, key part of a CPG, can be modeled:

$$I_{Ion,Burster}(t) = I_{Na}(t) + I_K(t) + I_L(t) + I_{NaP}(t) \quad (5)$$
$$I_{Na}(t) = g_{Na,max} \cdot m_{Na}^3(t) \cdot h_{Na}(t) \cdot (V_m(t) - E_{Na}) \quad (6)$$
$$I_K(t) = g_{K,max} \cdot n_K^4(t) \cdot (V_m(t) - E_K) \quad (7)$$
$$I_L(t) = g_L(V_m(t) - E_L) \quad (8)$$
$$I_{NaP}(t) = g_{NaP,max} \cdot m_{NaP}(t) \cdot h_{NaP}(t) \cdot (V_m(t) - E_{Na}) \quad (9)$$

The constants $E_x\,(E_{Na} = 55mV; E_K = -85mV; E_L = -55mV)$ are the reversal potentials of the ion currents, and $g_x\,(g_{Na,max} = 120mS/cm^2; g_{K,max} = 48mS/cm^2; g_{NaP,max} = 13mS/cm^2; g_L = 3mS/cm^2)$ the maximum ionic conductances. m and n are the activation variables, h the inactivation variables of the neurons, representing the permeability of the ion channels in values ranging from 0 to 1.

To complete the CPG network the moto-neurons have to be modeled by using the following currents. The ion currents in soma include

$$I_{Ion,Moto,Soma}(t) = I_{Na}(t) + I_K(t) + I_L(t) + I_{CaN}(t) + I_{K(Ca)}(t) \quad (10)$$
$$I_{CaN}(t) = g_{CaN,max} \cdot m_{CaN}^2(t) \cdot h_{CaN}(t) \cdot (V_m(t) - E_{Ca}) \quad (11)$$
$$I_{K(Ca)}(t) = g_{K(Ca),max} \cdot m_{K(Ca)}(t) \cdot (V_m(t) - E_K) \quad (12)$$
$$g_{K(Ca),max} = \frac{Ca}{Ca + K_d} \quad (13)$$
$$\frac{dCa}{dt} = f_{Ca}(-\alpha_{Ca} \cdot I_{Ca}(t) - k_{Ca} \cdot Ca) \quad (14)$$

and the dendrite current is composed of

$$I_{Ion,Moto,Dend}(t) = I_{CaN}(t) + I_{K(Ca)}(t) + I_{CaL}(t) \quad (15)$$
$$I_{CaL}(t) = g_{CaL,max} \cdot m_{CaL}(t) \cdot (V_m(t) - E_{Ca}) \quad (16)$$

where the constants are set by the expert to:

- $g_{CaL,max}$ $0.33mS/cm^2$
- $g_{CaN,max}$ $14mS/cm^2$ in soma, $0.3mS/cm^2$ in dendrite
- $g_{K(Ca),max}$ $1.1mS/cm^2$ in soma, $5mS/cm^2$ in dendrite

These are the newly added ion conductances of the L-like calcium current I_{CaL}, N-like calcium current I_{CaN} and the calcium-dependent potassium current $I_{K(Ca)}$.

To make the implemented CPG as biologically correct as possible the parameters to influence its behavior have to be chosen very carefully. Considering the fact that a very large number of parameters exists and in addition to that each change to one parameter affects the system and therefore the other parameters, an optimization of the system by hand comes not into account.

2.3 Methods - Computation of Muscle Activation

In NEURON the equations of Section 2.2 have been used to implement pacemaker and moto-neurons. Once implemented, neural behavior and muscle activation can be simulated with NEURON and frequency coded muscle activation patterns, comparable to those measurable by EMG, are produced. Because of the crosstalk effect observed when measuring EMGs in a clinical environment, reference data has to be computed using OpenSim. The relevant information is coded in the amplitude in the reference data and therefore the NEURON results have to be converted into an amplitude coded signal to make it comparable. From former clinical research it is known that the muscles, target of our investigation, are activated by two neural bursts during walking. This is equivalent to two spikes in the amplitude coded signal.

This leads to a calculation that has to be applied to get a more feasible representation of the signal to make it comparable with the recorded human data. For this a Gauss-representation algorithm is applied to the signal (see Figure 4) which encodes the frequency based information (upper part of Figure 4) into a data point consisting of three variables A, B and C (middle part of Figure 4). In the frequency encoded data the important values for calculation of the moto-neuron action potentials are the following:

- I: Average value of spike intervals of a burst
- D: Burst duration
- T_{MB}: Middle time of a burst

The applied Gaussian filter is defined as:

$$Y = \sum_{i}^{N} Y_i = \sum_{i}^{N} A_i \cdot Exp(-\frac{(t - B_i)^2}{C_i}) \tag{17}$$

The Gaussian variables A, B and C are $A = I/S_A$, $B = T_{MB}$ and $C = D/S_D$ with S_A and S_D as scaling factors for the amplitude and duration respectively and N as the number of bursts and i as the bursts index.

With this representation it is possible to compare the simulated CPG activation pattern to the human muscle activation from the clinical data of OpenSim (lower part of Figure 4).

2.4 Methods - Evaluation Function

For the optimization process parameter sets with different numbers of parameters are considered to be the individuals of one population of the EA. It was decided to use one small parameter set consisting of the four most important parameters and one large parameter set consisting of 14 parameters. The most remarkable influence to the behavior of moto-neurons is considered to be the time constant $\tau_{Syn,i}$ and ionic conductance of the persistent sodium current g_{NaP}. By changing these two values for the flexor and extensor muscles in one joint the muscle activation and therefore the movement itself is changed. In the large parameter set values for potassium,

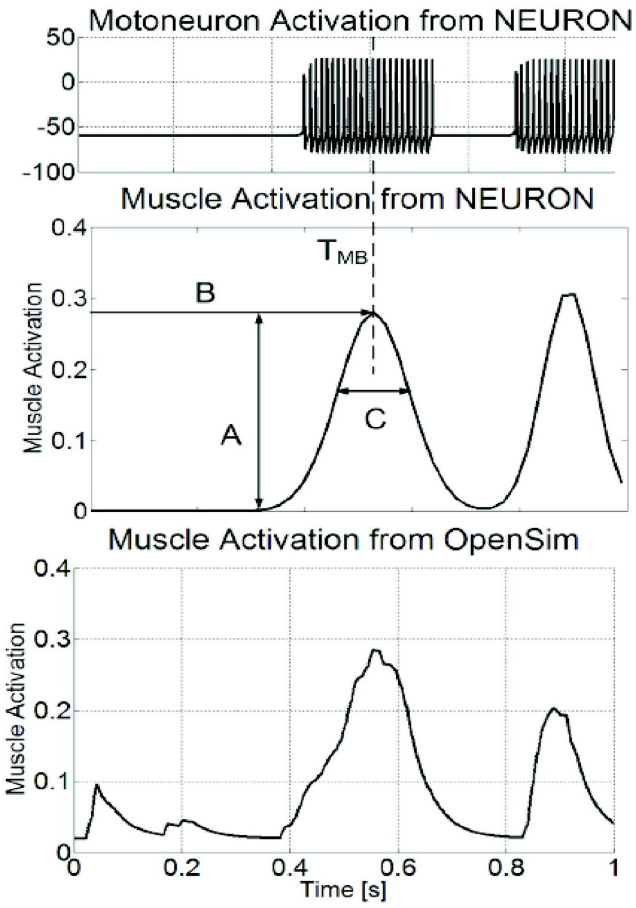

Figure 4: Making NEURON signals (upper third) comparable by applying a Gauss transformation (middle part) to the reference graph (lower third)

sodium, chloride, N-like calcium and potassium controlled calcium conductances (g_{Na}, g_K, g_L, g_{CaN} and $g_{K(Ca)}$) are added for flexor and extensor muscles. These constants have been introduced in the equations of Section 2.2 and their values, set by an expert are given. Using distributed computing by IAIDataShare, a simulation of the CPG in NEURON is run with each parameter set.

Once the simulation of the CPG muscle activation is finished (left area of Figure 5) the results are written to a result file.

Comparing this result to the reference data, a fitness value is calculated and assigned to the according parameter set individuum.

The fitness calculation is first done separately for flexor and extensor muscle activation. The results are then combined to get an overall fitness value for the actual simulated parameter set. First the sum of deviations of the values of each point of the simulated graph regarding the corresponding point in the reference graph is calculated:

$$Q_{sum} = \sum_{k=0}^{m} |x_N[k] - x_R[k]| \tag{18}$$

The location (X-axis: x_1 and x_2) and amplitude (Y-axis: y_1 and y_2) of the two maxima for the extensor and flexor muscles are the other four criteria which are

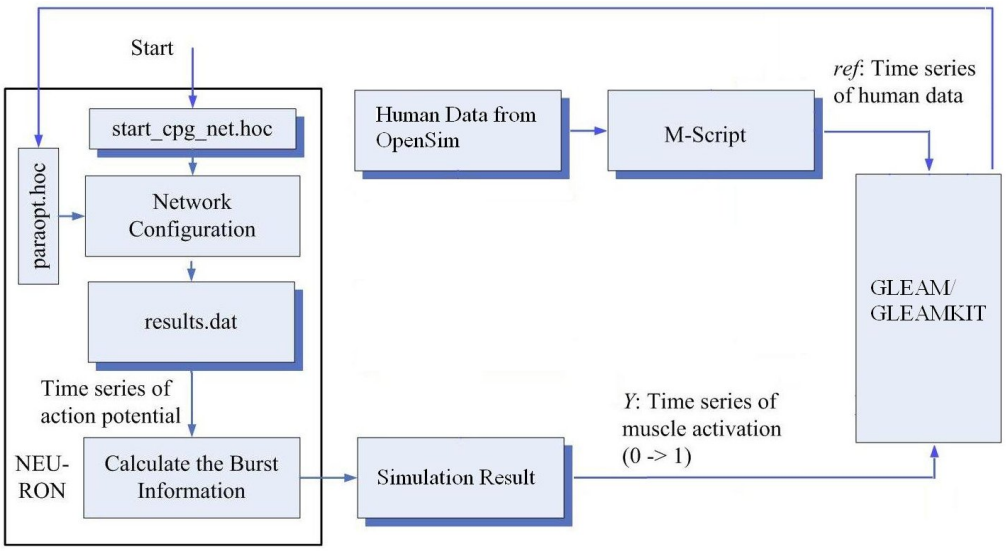

Figure 5: Connection between NEURON simulation and the EA

used as a measurement for the matching of simulation result and reference graph. N and R in the indices assign the values to the simulation graph and reference graph, respectively. The developed fitness function takes into account that the importance of each of these four criteria influences the matching of the two graphs in different ways.

$$Q_{ges} = \alpha_1 \sum_{n=1}^{2} \sum_{k=0}^{m} (|x_{n,N}[k] - x_{n,R}[k]|) + \alpha_2 |x_{1,n,N} - x_{1,n,R}| + \alpha_3 |x_{2,n,N} - x_{2,n,R}| + \alpha_4 |y_{1,n,N} - y_{1,n,R}| + \alpha_5 |y_{2,n,N} - y_{2,n,R}| \quad (19)$$

These criteria are weighted by weight factors α_i which leads to the fitness function for the whole optimization.

3 Evaluation and Results

For the evaluation of the complete system two different types of parameter sets have been used by the EA platform to generate activation patterns by the CPG. The first parameter set consists of only four parameters and therefore the influence to the generated activation pattern is more limited than with the second parameter set which uses 14 parameters leading to a larger search space.

Each optimization run with the EA lasted nine to ten hours which corresponds to 160 generations and several different runs have been performed. Three examples shall be depicted in detail in this section:

- Run 1: small parameter set with weighting $\alpha_1 = 10, \alpha_2$ to $\alpha_5 = 1$
- Run 2: small parameter set with weighting α_1 to $\alpha_5 = 1$
- Run 3: large parameter set with weighting α_1 to $\alpha_5 = 1$

In addition to these three runs which were designed to test the quality of the optimization, the scalability of the EA was tested doing several optimization runs on different workstations and processors.

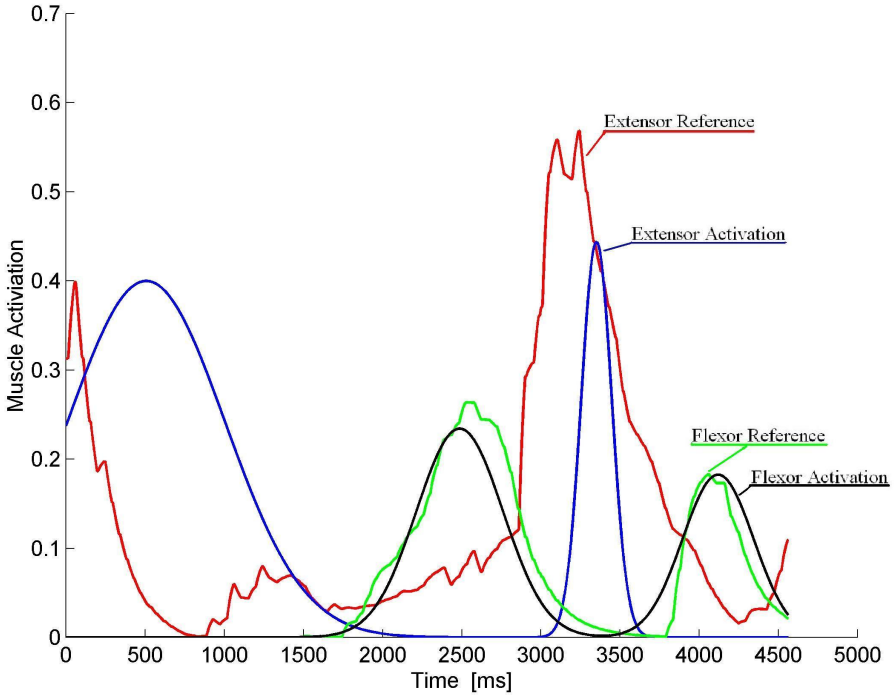

Figure 6: Result of optimizing 14 parameters

Run	**Weights** $\alpha_1 = 10, \alpha_2$ to $\alpha_5 = 1$	**Weights** α_1 to $\alpha_5 = 1$
Start by Expert	22323	40262
1. Run	**43886**	78937
2. Run	43886	**78937**
3. Run	54166	**83907**

Table 1: Runs with different weights

Table 1 shows the fitness values of the three mentioned runs. Two fitness values are shown, differing by the weights used for calculating the fitness function. The bold written ones are the ones of the weights used during the optimization process, the others are the results of the fitness function with the different weights applied to the result.

Comparing the first and second run the resulting fitness values are identical. Comparing these two to the third run, it is obvious that switching to the greater number of parameters improves the result significantly. In conclusion, no matter what parameter set is used for optimization an improvement to the original parameter setting done by an expert is obvious.

For the test how the process is scaling three different configurations were used:

- A two processor machine with four cores each. Only one core used (Scenario A and C)
- A two processor machine with four cores each. All cores used (Scenario B, D and E)

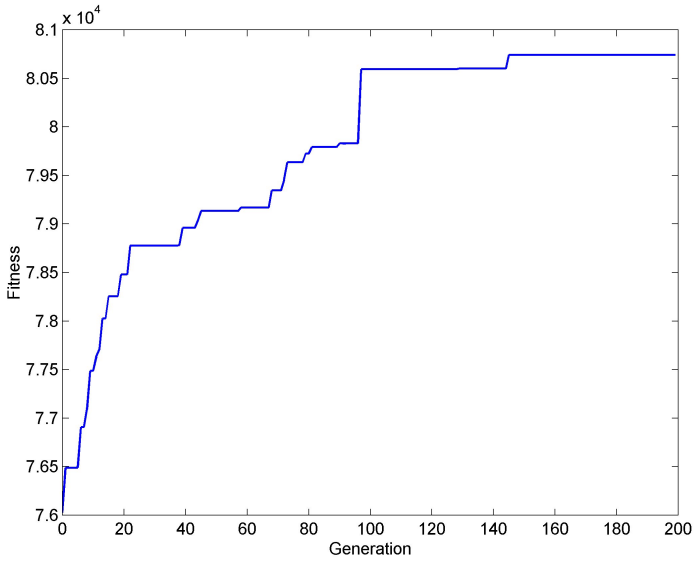

Figure 7: Fitness over Generation

- A heterogenous mixture of different office workstations (Scenario F)

The results are displayed in Table 2.

Scenario	**Function**	**Number of machines/cores**	**Individuals per minute**	**Scaling factor**
A	CPG	1/1	7.39	—
B	CPG	1/8	34.72	4.7
C	CPG	5/16	45.13	6.1

Table 2: Scaling

Comparing case A with B of Table 2, the scaling factor of 4.7, is not very good regarding the fact that eight times more cores have been used. The cause for this lies in two different circumstances. The first one is, that these two scaling tests have been performed on one machine with several cores, but other resources like the working memory had to be shared which led to waiting cycles during the computation. The other cause, which comes into account in case C as well, leads to waiting cycles after finishing a job and conflicts when two processes want to write results to the result folder at the same time. This is rooted in the fact that IAIDataShare was designed for graphical applications and real time capacities have not been a design criteria in the first place. Taking this into consideration scalability can be improved dramatically by adding real time capabilities to IAIDataShare.

In Figure 7 the development of the fitness value over the generations during the optimization process is shown. This optimization run lasted several days and showed that at about 160 generations a stagnation occurs and lasts (this is why the last 200 generations have not been displayed in this graph). Further investigation of this fact and the influence of the amount of parameters involved as well as the weights in the fitness function to it is necessary.

The resulting graph of scenario C of Table 2 is shown in Figure 6. The matching of the flexor activation to the flexor reference is very good but the matching of the

extensor activation curves bares potential for optimization. The cause for this can be found in oversimplification of the model which was part of the compromise between accuracy and computing effort.

4 Discussion

By employing NEURON for simulating a CPG the possibility is given to generate cyclic activation patterns in a very sophisticated and biological realistic way. Combining this simulation with the EA engine GLEAM and its GUI GLEAMKIT the optimization of the simulation despite its huge number of parameters has been made possible. Tests and evaluations have shown that there is still a lot of space for improvements which will be targeted in further research. Possible extensions could be tests targeting the robustness of the system against input from external sensors or multiple parallel execution.

The presented system is capable of generating biological correct and realistic actor activation patterns which, in the future, should be used for the motion generation or actor activation in robotics. To make this possible two challenges to get from simulation to application have to be targeted. First an interface has to be developed which is capable of translating the neural activation patterns into actuator signals for controlling several devices which function as 'muscles' in a robot. Then a new iteration of optimization processes is necessary. Apart from that, stepwise simplifications have to be performed to make the system suitable for real time applications in robots.

Apart from that, further potential for research lies in the setting of the weights in the calculation of the fitness. As depicted in Section 3 possibility exists, that despite different weights in the fitness function the resulting parameter set leads to the same activation pattern generated. This fact motivates further investigation how this fact may be useful for designing more advanced optimization procedures.

In addition to that, as was shown by the scaling tests, there is still space for improvement, especially in the design of the IAIDataShare tool which takes responsibility for the distributed computing.

5 Conclusion

In this paper, an extended simulation environment for biological Central Pattern Generators has been presented and an optimization has been applied to match the generated muscle activation as close as possible to human locomotion data. For this optimization an Evolutionary Algorithm has been employed and several optimization runs have been performed which showed the influence of different prioritization of the parameters. Now a system exists which is capable of generating activation patterns for joint movements. This will be expanded and enhanced for applications in robotics and the development of prostheses in our future research.

6 Acknowledgements

Software tools used were the NEURON of N.T. Carnevale and M.L. Hines and their group at the University of Yale and the OpenSim toolbox for biological mechanics simulation. The project is part of the Collaborative Research Center (SFB) 588 of the German Research Foundation (DFG). Thanks go to the Orthopädische Klinik Heidelberg, especially Rüdiger Rupp, Christian Schuld and Joachim Schweidler, in Heidelberg Germany for assisting in recording human motion data and EMG signals.

References

[1] Brown, T.: The Intrinsic Factors in the Act of Progression in the Mammal. *Proceedings of the Royal Society: London* 84 (1911), S. 308–319.

[2] Brown, T.: The Factors in Rhythmic Activity of the Nervous System. *Proceedings of the Royal Society: London* 85 (1912), S. 278–289.

[3] Manoonpong, P.; Geng, T.; Porr, B.; Woergoetter, F.: The RunBot Architecture for Adaptive, Fast, Dynamic Walking. In: *Proc., IEEE International Symposium on Circuits and Systems (ISCAS)*, S. 1181–1184. 2007.

[4] Chen, Y.; Bauer, C.; Burmeister, O.; Rupp, R.; Mikut, R.: First Steps to Future Applications of Spinal Neural Circuit Models in Neuroprostheses and Humanoid Robots. In: *Proc., 17. Workshop Computational Intelligence*, S. 186–199. Universitätsverlag Karlsruhe. 2007.

[5] Chen, Y.: *A Concept for the Application of Neural Oscillators and Spinal Reflexes to Humanoid Robots and Neuroprostheses*. Master thesis, Universität Karlsruhe (TH), Institut für Regelungs- und Steuerungstechnik, Forschungszentrum Karlsruhe GmbH. 2008.

[6] Jakob, W.: Auf dem Weg zum industrietauglichen Evolutionären Algorithmus. In: *Proc., 15. Workshop Computational Intelligence*, S. 212–226. Universitätsverlag Karlsruhe. 2005.

[7] Blume, C.: GLEAM - Ein EA für Prozessabläufe am Beispiel von Steuerungen für Industrieroboter. In: *Proc., 16. Workshop Computational Intelligence*, S. 11–24. Universitätsverlag Karlsruhe. 2006.

[8] Blume, C.; Jakob, W.: GLEAM - an Evolutionary Algorithm for Planning and Control Based on Evolution Strategy. In: *Proc., GECCO*, Bd. Late-Breaking Papers, S. 31–38. L. Livermore National Laboratory. 2002.

[9] Hodgkin, A.; Huxley, A.: A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. *The Journal of Physiology* 117(4) (1952), S. 500–544.

[10] Matsuoka, K.: Sustained Oscillations Generated by Mutually Inhibiting Neurons with Adaptation. *Biological Cybernetics* 52 (1985), S. 367–376.

[11] Matsuoka, K.: Mechanisms of Frequency and Pattern Control in the Neural Rhythm Generators. *Biological Cybernetics* 56 (1987), S. 345–353.

[12] Manoonpong, P.; Geng, T.; Wörgötter, F.: Exploring the dynamic walking range of the biped robot RunBot with an active upper-body component. In: *Proc., IEEE-RAS International Conference on Humanoid Robots*, S. 418–424. 2006.

[13] Mori, T.; Nakamura, Y.; Sato, M.; Ishii, S.: Reinforcement Learning for CPG-Driven Biped Robot. In: *Proc., National Conference on Artificial Intelligence*, S. 623–630. 2004.

[14] Duysens, J.; Tax, A.; Murrer, L.; Dietz, V.: Backward and Forward Walking Use Different Patterns of Phase-Dependent Modulation of Cutaneous Reflexes in Humans. *Journal of Neurophysiology* 76 (1996) 1, S. 301–310.

[15] Duysens, J.; Van de Crommert, H. W. A. A.: Neural Control of Locomotion; Part 1: The Central Pattern Generator from Cats to Humans. *Gait & Posture* 7(2) (1998), S. 131–141.

[16] Dietz, V.; Zijlstra, W.; Duysens, J.: Human Neuronal Interlimb Coordination during Split-Belt Locomotion. *Experimental Brain Research* 101 (1994), S. 513–20.

[17] Blume, C.: GLEAM - A System for Simulated 'Intuitive Learning'. In: *Proc. PPSN I*, Nr. 496 in LNCS, S. 48–54. Springer. 1990.

[18] Braun, S.: *Softwareentwurf und Teilimplementierung einer Toolbox GLEAMKIT unter MATLAB*. Projektarbeit, Berufsakademie Karlsruhe, Forschungszentrum Karlsruhe. 2007.

[19] Braun, S.: *Entwicklung einer Plattform für Evolutionäre Algorithmen mit verteilter Simulation am Beispiel der Parameteroptimierung von zentralen Mustergeneratoren für Roboterbewegungen*. Diplomarbeit, Berufsakademie Karlsruhe. 2008.

[20] Mikut, R.; Burmeister, O.; Reischl, M.; Loose, T.: Die MATLAB-Toolbox Gait-CAD. In: *Proc., 16. Workshop Computational Intelligence*, S. 114–124. Universitätsverlag Karlsruhe. 2006.

[21] Carnevale, N.; Hines, M.: *The NEURON Book*. Cambridge University Press. 2006.