CLOSED-LOOP VERIFICATION APPROACH FOR THE IDENTIFICATION OF MOTION-BASED ACTION POTENTIALS IN NEURAL BUNDLES USING A CONTINUOUS SYMBIOTIC SYSTEM

Volkhard Klinger[(a)], Sebastian Bohlmann[(b)]

[(a)]Department of Embedded Systems

University of Applied Science Hannover (FHDW)

D.30173 Hannover, LS, Germany

[(b)]Institute of Applied Mathematics

Gottfried Wilhelm Leibniz University Hannover

D-30167 Hannover, Germany

[(a)] volkhard.klinger@fhdw.de, [(b)]bohlmann@ifam.uni-hannover.de

ABSTRACT

The identification and the corresponding verification of motion- and sensory feedback-based action potentials in peripheral nerves are prerequisites for applications like prosthesis control or limb stimulation in medical technology. After a description of a prototype of a biosignal acquisition and identification system we introduce the symbiotic cycle, based on the well-known term symbiotic simulation (Aydt, Turner, Cai, and Low 2008), (Aydt, Turner, Cai, and Low 2009). As an example we present a data driven method to create a human readable model without using presampled data. This method consists of symbiotic continuous system combining a physical system, a simulation system and an agent based machine learning system. All components in the system interact in a symbiotic way. The result of each component is used as an input by the others and vice versa. Finally bootstrapping as a direct consequence of the symbiotic interaction is trivial for all components. The closed-loop identification method is integrated into this symbiotic cycle. This verification approach is the main focus of the paper.

Keywords: symbiotic simulation, symbiotic cycle, system identification, agent-based evolutionary computation

1. INTRODUCTION

The identification of motion- and sensory feedback-based action potentials in peripheral nerves is a great challenge in medical technology. It is the prerequisite for applications like prosthesis control or limb stimulation. Based on the acquisition of action potentials, the identification process correlates physiological and motion-based parameters to match movement trajectories and the corresponding action potentials. The identification method is based on the continuous mode symbiotic cycle, combining a physical system, a simulation system and an agent based machine learning system. As an example a data driven method to create a human readable model without using presampled data is presented. All components in the system interact in a symbiotic way. The result of each component is used as an input by the others and vice versa. At first the prototype of a smart modular biosignal acquisition and identification system is presented, acting as the physical target system in the symbiotic cycle, presented subsequently. In this paper the focus is on the identification method based on a data driven approach and its verification. We present the closed-loop identification method, implemented using a symbiotic continuous system (Aydt, Turner, Cai, and Low 2008), (Aydt, Turner, Cai, and Low 2009), consisting of a robotic based trajectory generation, the nerve simulation and an agent-based machine learning system. We introduce the model generation process and show the closed-loop verification approach of the identification method.

1.1. The Prototype of a Smart Modular Biosignal Acquisition and Identification System

The key challenge is the human machine interface of prosthesis and its movement control. The objective is to use biosignals for the information transfer between human being and prosthesis. So an interface is needed to interfere between the command-level and the actuator- and sensor- level. The approach discussed in this paper is based on the direct use of the action potentials of peripheral neural bundles via an electroneurogram (ENG) (Gold, Henze, and Koch 2007), (Neymotin, Lytton, Olypher, and Fenton 2011). So, the employment of invasive intra-neural sensors (Micera, Carpaneto, and Raspopovic 2010), (Micera, Citi, Rigosa, Carpaneto, Raspopovic, Pino, Rossini, Yoshida, Denaro, Dario, and Rossini 2010), (Raspopovic, Capogrosso, Petrini, Bonizzato, Rigosa, Di Pino, Carpaneto, Controzzi, Boretius, Fernandez, Granata, Oddo, Citi, Ciancio, Cipriani, Carrozza, Jensen, Guglielmelli, Stieglitz,

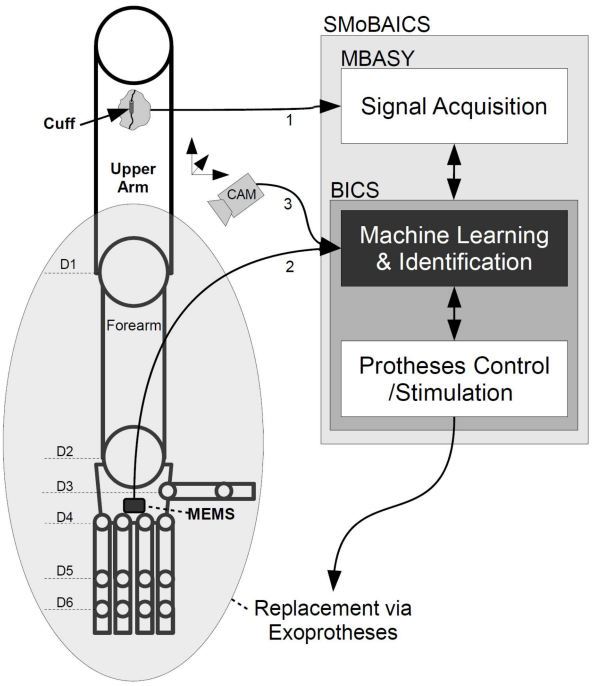

Figure 1: System Overview

Rossini, and Micera 2014) is in this project not in the focus, but the identification (Cesqui, Tropea, Micera, and Krebs 2013) of motion-based action potentials is the proposal to realize a smart minimal-invasive solution. To record ENG-signals with a very low amplitude, which are only of the order of a few microvolts, a special frontend-hardware/software system has been designed, realized in two different prototypes, introduced in (Klinger 2015) and (Klinger and Klauke 2013). In this paper the focus is on a new combined identification and verification method, taking advantage of a continuous symbiotic system. This work continues the former work about system identification presented in (Bohlmann, Klauke, Klinger, and Szczerbicka 2011), (Bohlmann, Klinger, and Szczerbicka 2009) and (Bohlmann, Klinger, and Szczerbicka 2010a).

The prototype of the Smart Modular Biosignal Acqusition, Identification and Control System (SMoBAICS), shown in Figure 1, integrates all necessary tasks (Hazan, Zugaro, and Buzsáki 2006). The biosignal acquisition is done by the Modular Biosignal Acquisition System (MBASY)-subsystem, the next generation of our own frontend-hardware/software-system. The MBASY is redesigned to get a better functionality and to optimize the modular concept (Klinger 2015). The central part of the identification process is integrated in the Biosignal Identification and Control System (BICS). It consists of two parts: the machine learning & identification and the control/stimulation from a prosthesis. While the second part is designed by state-of-the-art technology, the machine learning & identification is composed of multi-agent-based optimization algorithm and an evolutionary correlation of different types of nerve signals and of additional information like camera positioning or micro-electro-mechanical systems (MEMS). We introduce this part in more detail in section 2.

Before describing the overall function one aspect regarding the learning procedure has been taken into consideration. The objective of SMoBAICS is the action potential based control or stimulation of upper or lower limbs of handicapped human beings. SMoBAICS provides not only a base identification step (learning phase) but an ongoing supervision (operating phase). Obviously, the operating phase has to be executed on a small body mounted system but in this paper the focus is not on this system detail.

In Figure 1 a typical application is shown, the cuff electrode is implanted in the upper arm enclosing a neural bundle. The action potentials are recorded by the MBASY and passed to the BICS (1). Two additional information streams are used by the BICS for the action potential based identification: a camera-based motion capturing, used during the learning phase (3), and a motion tracking device at the end effector (e.g. the hand), used during the operational phase (2). These data streams are important for the identification.

1.2. The Symbiotic Cycle

To understand or control any type of complex process a process model is essential, an empirical process description does not provide a detailed functional and time related process specification. The process model, based on the combination of physical equations and a graph structure, allows the reconstruction of process behavior, the optimization of the entire process and a forecast of process behavior. This paper describes a method to generate a process model from scratch, without using any type of initial model description. The method is based on machine learning and symbiotic simulation and describes a type of symbiotic cycle. The model generation is a continuous process and provides therefore an adaptation to changing process parameters, like friction or bearing clearance, boundary conditions and constraints.

Based on the classification of symbiotic simulation in (Aydt, Turner, Cai, and Low 2008), (Aydt, Turner, Cai, and Low 2009), the symbiotic cycle combines a symbiotic simulation control system (SSCS), a symbiotic simulation forecasting system (SSFS), a symbiotic simulation model validation system (SSMVS) and a data driven agent based online machine learning system with a real world process (Trianni 2014).

The central challenge in this paper is on the one hand to create a system which can produce a human readable model without the existence of prior data, model or knowledge (Yang, Koziel, and Leifsson 2013). And on the other hand to be able to simulate and/or predict the behavior of the system without a known model. Both challenges presuppose each other. First it is basically a kind of chicken or the egg dilemma. The simulation system needs a model, this is produced by the machine learning system, which needs some sort of input data,

but this data is produced by the motion of the robot, finally controlled by the simulator. The basic method proposed and demonstrated in this paper is the symbiotic solution. If all steps are running in parallel the central dilemma disappears. This is what is called the symbiotic cycle. All components of the self learning system are connected by using a streaming event-driven approach. If an event which could cause an action in a different component is happening, it is immediately streamed to the corresponding component. In fact there is no macroscopic sequence or stepping between components. Everything is processed simultaneously in parallel. The machine learning module for example continuously outputs model candidates at a relatively unknown rate. The simulator then reacts by profiling the solution proposal. Clearly this could lead to a short-time overload of modules, if some burst input is generated. To minimize this effect buffering and load balancing technology are used.

2. IDENTIFICATION & MACHINE LEARNING

The machine learning and identification is the most complex module within SMoBAICS. In this paper the focus is on different verification approaches, providing a new continuous symbiotic method including machine learning.

2.1. Data Preprocessing

All real data, recorded from the cuff electrode have to be preprocessed to improve the data conditioning. The verification data generated from the NEURON-simulator can be used directly.

- Filtering: The recorded action potentials are disturbed by intrinsic noise. In addition these are overlaid by a substantial extrinsic noise, originated for example by electromyogram (EMG) from surrounding muscles. Therefore the recorded data has to be filtered with integrated analogue filter and additional digital filter.
- Re-sampling: The recorded data has two main weaknesses: The samples are asynchronous and aperiodic. In order to get a time series of data samples the following steps are performed:
    - Interpolation and FIR Filter (finite impulse response)

      For each sequence the given values are interpolated and smooth the result with a convolution.
    - Error Correction

      The interpolated data is equalized with the original samples gained from the Data Factory.
    - Down-sampling

      We pick Euclidean equidistant samples from each sequence and combine them to data samples with a time-stamp.

      During the machine process the data samples will not stay in their chronological ordering. To be able to perform time derivation, it is necessary to save the chronological neighbors for each sample. The resulting time series of equidistant data samples p consists of a time-stamp p_{time}, a vector $p_{data} = [p_{out}, p_{in}]$, with $p_{out} \in \Re$ and $p_{in} \in \Re^m$, containing the output and input data and its chronological neighbors p^{pre} and p^{post}. With P we denote the set of all such data samples. Furthermore we define the delta value $p_{in}^{\Delta} \in \Re$ with

$$(p_{\text{in}}^{\Delta}) := \frac{1}{2}\left(\frac{(p_{\text{in}}) - (p_{\text{in}}^{\text{pre}})}{p_{\text{time}} - p_{\text{time}}^{\text{pre}}} + \frac{(p_{\text{in}}) - (p_{\text{in}}^{\text{post}})}{p_{\text{time}} - p_{\text{time}}^{\text{post}}}\right),$$

$$(p_{\text{out}}^{\Delta}) := \frac{1}{2}\left(\frac{p_{\text{in}} - p_{\text{in}}^{\text{pre}}}{p_{\text{time}} - p_{\text{time}}^{\text{pre}}} + \frac{p_{\text{in}} - p_{\text{in}}^{\text{post}}}{p_{\text{time}} - p_{\text{time}}^{\text{post}}}\right).$$

2.2. Identification via Machine Learning based on Continuous Symbiotic System

The machine learning is based on evolutionary algorithms – a generic population-based metaheuristic optimization algorithm inspired by biological evolution (DeJong 2006) – embedded in a multi-stage and multi-agent implementation, shown in Figure 2. The planet structure models the environment for the populations inside the evolutionary algorithm. Every planet provides a data field, the software agents can operate on.

The number of planets is scalable, the current predetermined size is $n=9^4=6561$. Using a multiprocessor system, the number of planets have to be multiplied by the number of cores. Data acquired from the process connection are preprocessed, equal to the filling of the

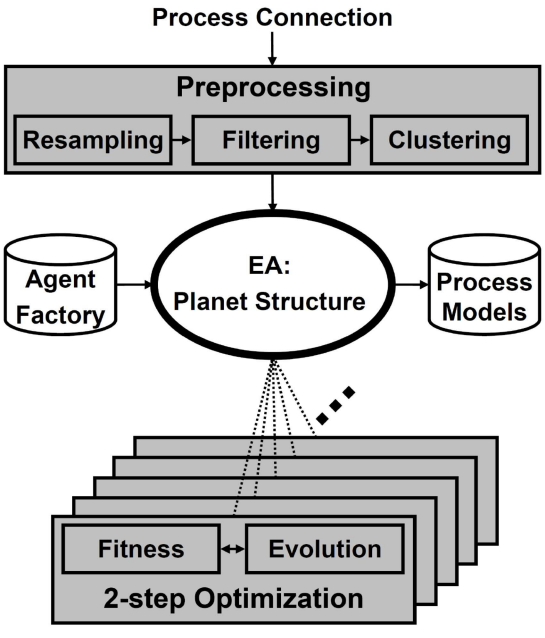

Figure 2: The architecture of the machine learning method

planet structure. The preprocessing consists of several steps to guarantee a high average information content of the data, so called data entropy. One step is an appropriate data preprocessing, described in principle in subsection 2.1. One further step is the data filling. This last step of the data preprocessing arranges the data samples on a 2D surface of a so-called planet. The surface of the planets is built in a recursive pattern of squares containing nine elements, filled meanderlike. This method leads to the planet size 9^4. This arrangement has the advantage, that the data set used for the local optimization consists of data samples, which may be spread more widely across the input sequences.

In the central part of Figure 2 the agent factory, the model library and the 2-step optimization, dedicated to every planet, is shown. The agents have 4 essential features: an age, an energy level, an area and their model function, approximating the corresponding process function. Moreover a replication mechanism is implemented, meaning the agents are able to produce a child and put it on an area. The age and the energy level are increased after each iteration. All operations an agent can perform, have an energy effort, by which the energy level is lowered, if the operation is executed. Furthermore the agents have the ability to learn from their local data and improve their model function by executing different evolutionary operations to change the structure of the model function and a local optimization algorithm to calibrate the parameters. In each iteration the software agents perform the following operations: Calculate Fitness, Move, Local Optimization, Evolutionary Operation and Nomination (The agents elect a few individuals with the highest fitness values and age on each planet to form an *Elite Population*). The Evolutionary Operation consists of

- Mutation: The agents model function gets changed randomly: Either a sub-tree of the model function is exchanged or new operations are inserted.
- Crossover: When an agent moves it may happen that the chosen area is already occupied with another individual. In this case, a sub-tree of the individuals model function is replaced by a randomly chosen, suitable sub-tree of the other agents model function.
- Replication: The agent duplicates himself.
- Global Optimization: The agents, which own enough energy or are not adult yet, optimize the parameters of their model function in the Memetic Coprocessor, explained below.

The global optimization is realized on memetic coprocessors, running on an extra processor core, executing more sophisticated algorithms for a global optimization. In the current configuration a downhill-simplex algorithm (Nelder and Mead 1965) is used. The algorithm chosen for this local parameter optimization is resource-saving, because it is executed for all agents in every iteration.

The machine learning provides the capability of running the evolutionary algorithm described above on several planets at the same time. If this is the case, some of the areas on each planet get marked as so called beam areas. After each iteration copies of all individuals placed on such an area are send to a randomly chosen area on a randomly chosen planet, provided the chosen area is not yet occupied by an agent. In the experiments 100 of the 6561 areas on every planet were marked as beam areas. Our implementation associates each planet to one processor core, on an additional processor core a universe supervisor is executed. This supervisor manages the elite population using the data from all planets and controls the termination condition. The information exchange between the cores is implemented via a non-blocking Message Passing Interface.

3. VERIFICATION APPROACHES

The verification block confirms the quality of the model and is an essential part, it helps to evaluate the model which is improved or build up during identification process. The verification strategy is based on a set of process input sequences $((x_1)_t, \dots, (x_m)_t, t \in \aleph)$ and output sequences $((y_1)_t, \dots, (y_j)_t, t \in \aleph$ and of the simulation output sequences $((z_1)_t, \dots, (z_j)_t, t \in \aleph$. The output sequences of the simulator are related to the input sequences by functional relationships $f: \in \Re^m \to \Re^j$. In principle, the verification method can be executed with synthetic data (generated by a model of the evaluated system) or live data (acquired from the physical process). We first focus on the synthetic data verification.

The Curve Fitting, presented in (Klinger and Klauke 2013), provides an open loop verification, which is not able to start the verification process without an initial model. Furthermore, this open loop verification is not able to control the verification process to increase the quality of the model. Precisely for this reason, a closed-loop verification process has been designed, the symbiotic cycle, shown in Figure 3. The system architecture follows the paradigm of the symbiotic cycle and it is based on the machine learning, introduced in subsection 2.2. Five modules form this symbiotic cycle which is application independent, the additional modules are described in the following.

3.1. Process

The process block covers all physical relations of the considered process. Analog inputs or outputs have to be transformed using analog-to-digital- or digital-to-analog-converters. The interface to the digital in-/out- signals is handled using the process data streaming protocol (PDSP) managing distributed process data flows (Bohlmann, Klinger, Szczerbicka, and Becker 2010b). This protocol is designed to be used in mixed continuous and discrete environments (Zeigler, Praehofer, and Kim 2000) referred to as hybrid. Focusing on symbiotic simulation (Fujimoto, Lunceford, and Uhrmacher 2002), PDSP is primary designed to satisfy four modes

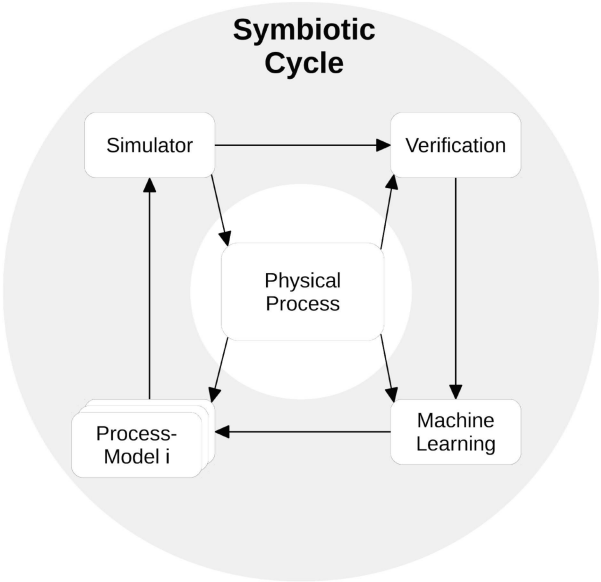

Figure 3: Symbiotic Cycle

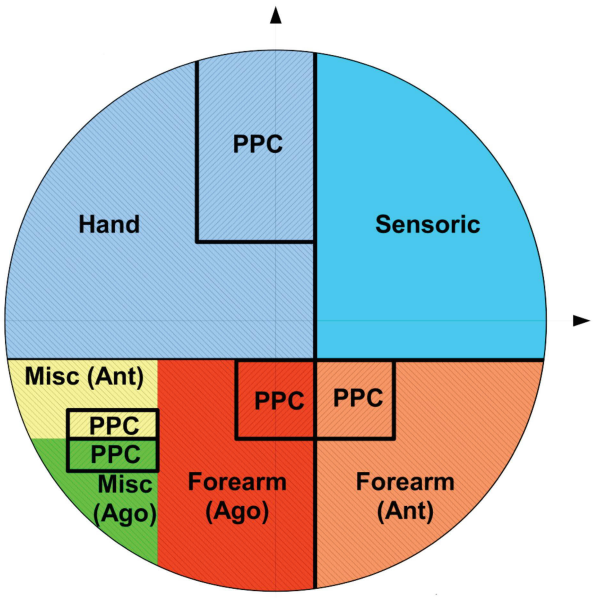

Figure 4: Nerve Bundle: Axon Model

of operation (analytic, transparent, online, prediction); here the focus is on the online mode. In this mode PDSP is used to simulate a process and transmit the results back to the process. The data is directly transmitted between the physical process and the simulator. Therefore latency is minimized although proxy servers may be necessary for large scale simulations.

3.2. Simulator

In the simulator block two different approaches are used, dependent from the application. For general purpose it is a Java based simulation system specifically designed for high speed online and symbiotic simulations. The simulator especially has online compiling capabilities, e.g. models can be compiled during runtime in memory and then dynamically injected into the simulator. It is capable to dynamically load or receive models (basically any kind of java program) and simulate multiple isolated instances in the same memory/thread context. The simulator combines Java class loading mechanism and byte code enhancing to calculate user defined metrics while processing prior structural unknown models on the fly. In combination with an OSGI framework PDSP can be directly embedded to running simulations.

For the specific application of ENG-based identification currently the well established NEURON framework for empirically-based simulations of neurons and networks of neurons is used (Carnevale and Hines 2006), (Coates, Larson-Prior, Wolpert, and Prior 2003), (Law and Kelton 2000). The different constraints, like myelin structures, all-or-none, two directions of information flow, frequency borders of the action potentials, etc. has been taken into account. We have configured the simulator and realized a complex neural bundle including our cuff electrode setup to generate verification data for several information transfer scenarios.

Therefore a cross-sectional area of the nerve bundle is part of the configuration, shown in Figure 4. This model is used for the simulation environment available to build up the axon configuration in NEURON. The action potentials used for the NEURON-simulator are derived by human arm modeling via Matlab Robotics Toolbox (Corke 2011) or using a real robot arm or prosthesis. With this model for verification it is possible to concentrate on specific muscle groups and their reactuatory answer and therefore verification pattern can be generated. The simulation environment uses the Hodgkin-Huxley model (Hodgkin and Huxley 1952) to simulate the axon internal membrane, the ion-channels and the extra-cellular space. So, the propagation of action potentials along the axons is modeled using these equations. Furthermore, the mechanisms concerning the passive membrane channels are included.

3.3. Process Model

The area provides data samples to learn from and calculate the error of the model function ($\Re^m \rightarrow \Re$) is stored in a tree representation. This function is composed of elementary operations (Schmidt and Lipson 2007), like +,−,*,/,*sin*, *sqrt* the variables $x_1, \ldots, x_m$ and a set of parameters within their model function.

According the multi-agent capability, there are several models existing. The number of currently active agents and their inner candidate function is regulated by a software PID control. It is programmed to use the available processing power in nearly optimal conditions. The best models are picked according their ability to survive multiple times longer than the mean agent population. This metric corresponds to the selection of the fittest agents. The complexity of the process models is problem-related dynamic. It depends on the number of input

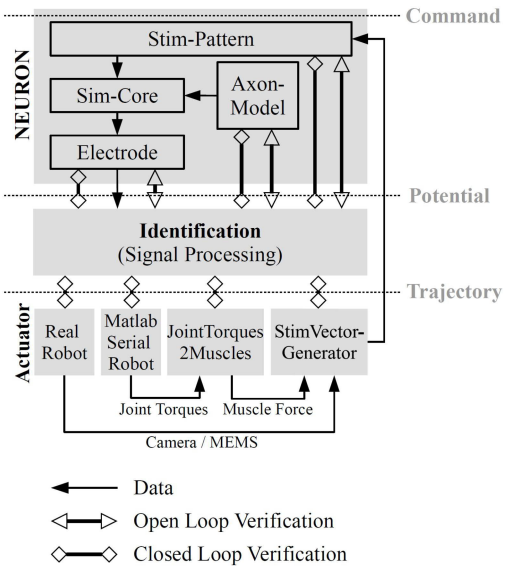

Figure 5: Open- and closed-loop verification

variables, constants and operations. The complexity is defined by counting the number of nodes in the tree representations. The differences between both verification methods are illustrated in Figure 5. The open-loop method allows a verification of the potential-level (electrode-data) against the command-level (Stim-Pattern). It is not able to generate new adequate verification patterns to improve the quality of the model behavior according to specific system states. The new method including the symbiotic cycle integrates the verification in a complex and interactive exchange of data between physical process, process model, machine learning and the simulator. The closed-loop verification provides a verification between the trajectory level and the command-level, including the potential-level. The identification block controls the generation of trajectory-related Stim-Pattern. One major advantage of this method is the fact that no initial model is necessary, the continuous symbiotic system generates it from scratch. Regarding the use of the prosthesis two alternatives are shown in Figure 5 in the actuator level: The real robot, a physical system, or a serial robot, simulated by Matlab. Both alternatives are working as the cause and effect element, closing the gap between command and trajectory level. While the serial robot provides joint torques related to a simulated trajectory, the real robot provides the trajectory information via a camera system or an integrated MEMS device (see Figure 1). To create a symbiotic online learning environment all data from the physical process is streamed via PDSP to and from a central routing software. Machine learning and the massive parallel simulation are running outside the real time domain. The control simulation, which is using the currently best known model, is running inside an embedded system in real time. Communication between the two domains basically consists of a model transmission and a movement direction proposal directed into the real time domain. Data transmission directed to the machine learning, verification and simulation is a set of streamed time series data.

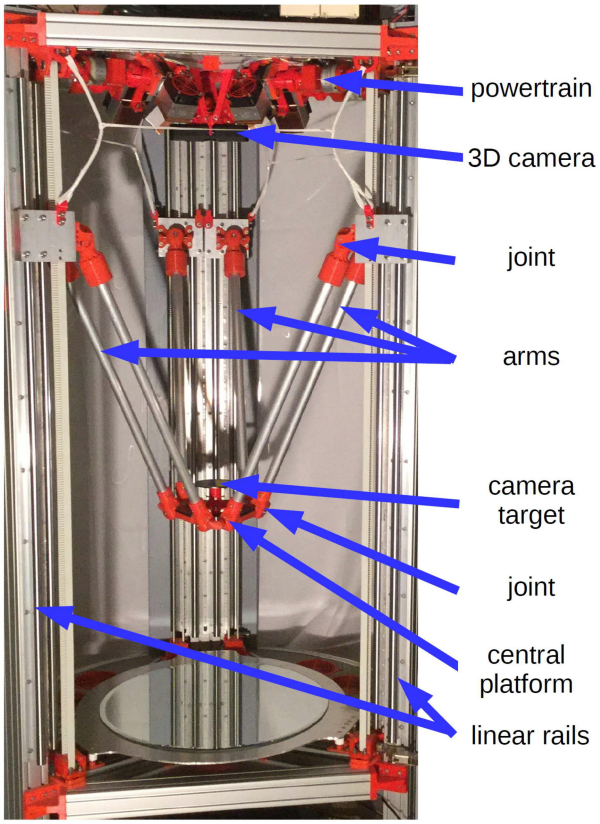

Figure 6: Demonstrator System

4. RESULTS

In this paper one key aspect of the proposed closed-loop identification method is shown. We are using the identification method to generate the model from a real robot using camera information. After the model set-up from the real robot system, this closed-loop verification can be used according Figure 5. The real robot, used in this set-up, is shown in Figure 6. To integrate a robot as a prosthesis prototype this demonstrator system, a parallel delta robot has been used. It basically consists of 6 vertical linear rails grouped into 3 pairs. These are placed around the working area in 120° spacings. Each linear axis is connected to a joint, which is connected to an arm and again to a joint. This joints are all linked to a central platform. As common to all parallel kinematic chains a move in one actuator influences the position of the central platform in all possible degrees of freedom (DOF). This 6 linear actuators are the output stage of the simulation. To be able to detect the concrete position of the system a depth camera combined with a standard 2D camera is mounted above the working area inside the robot. The depth camera used is a infrared structured light sensor. In the experiment the camera follows a round target placed in the center of the platform. Camera output (after some processing) is the position of the platform. This is a 6 channel output: 3 spatial directions and 3 rotation angles. These 6 channels are the sensor output of the robot and the input for the simulation system. All 6 sensors and 6 actuators combined add up to the input for the machine learning

part. This has been referenced in the previous chapter as p_{data}. The position of all actuators influence the position of all axis so the complete information set is required to be able to calculate an accurate prediction model. To create a symbiotic online learning environment the delta robots internal real time network is connected to a external computer. All data produced by the robotic system is streamed via PDSP to and from a central routing software. Machine learning and the massive parallel simulation are running outside the real time domain. The control simulation, which is using the currently best known model, is running inside an embedded system in real time. Communication between the two domains basically consists of a model transmission and a movement direction proposal directed into the real time domain.

To realize the symbiotic machine learning system described in this paper one central question is: Where should the symbiotic robot acquire new data? The basic idea is to collect new experimental data at a location where the possible solutions generated by the machine learning system differ the most.

We define $x_i \in \Re$ as the input variables, $p_j \in \Re$ as the constant parameter and $y_j \in \Re$ as the output variables of a suggested model. The machine learning system produces a list of candidate functions sequences $f_{e,j}(x_0, \dots, x_n, p_0, \dots, p_m) = y_j$. e is defined as the solutions index (typically e<30=e_{max}).

$$D_j = \{f_{g,i} - f_{g,j} | i \neq j \text{ and } g, h = 0, .., e_{max}\} \quad (1)$$

Now the movement direction s for data acquisition can be defined as:

$$s = \underset{d \in D_i, j=[0..6]}{\operatorname{argmax}} \|\nabla d\|_1 \quad (2)$$

This direction s is calculated by the simulator on the fly while processing new input data and transmitted to the robotic system. The simple idea behind this is to acquire new input data where model candidates tend to have different results. This leads to a continuous hypothesis filtering in the elite generation.

The basic challenge of the machine learning system can also interpreted a global non linear optimization problem. The concept behind the above formulated feedback metric is to continuously adapt the target function, while machine learning is running. The adaption process described is on the other hand based on the output of the candidate functions $f_{e,j}(x_0, \dots, x_n, p_0, \dots, p_m) = y_j$ evaluated by the online simulation system. This is again what is defined as the symbiotic cycle. If the global target is the deepest valley on the multidimensional target function, then the symbiotic circle will force secondary minima to increase their value by sampling new contradictory data from the physical system. The only valleys not affected are the global optima formed by different possible formulations of a globally valid and correct process model. Data transmission

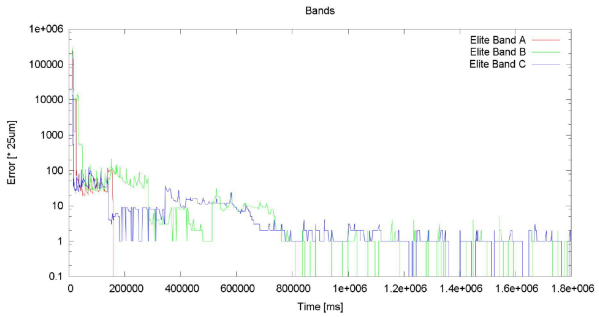

Figure 7: Error band for actuators A, B, C

directed to the machine learning, verification and simulation is a set of streamed time series data. All calculations outside the real time domain are processed on two quad Intel Xeon E7-8860v3 compute systems (128 logical cores) with 1TB of DDR4 RAM each. The real time control System located in the head box of the physical system consists of 14 Arm Cortex M4 Microcontrollers and three 8 Core Arm Cortex-A7 Processors. Local compute power is therefore relatively small compared with the cloud connected machine learning modules. To demonstrate the emergent effects arising from the combination of all modules of the symbiotic circle the same machine learning system has been used as in (Bohlmann, Klinger, Szczerbicka, and Becker 2010b).

It takes some time at the beginning to acquire enough input data to generate a start-up model from scratch. Secondly the error does not progressively decrease as often found in offline machine learning. This is caused by the feedback metric sampling new data at positions with low performance of some model candidate. This is a direct result from new online data injected to the machine learning progress. These spikes for the three actuators are depicted in Figure 7, presenting the decreasing error band for the 3 actuators A, B and C. The interesting aspect here is, that the peaks sometimes but in most cases not correspond to others. We found that this is a direct result from the *argmax* operator in formula 2. It is basically caused by the selection of one pair out of all actuators to determine the robotic movement direction until a different pair has a higher rating.

By far the most amazing aspect is the comparison of the offline performance (Bohlmann, Klauke, Klinger, and Szczerbicka 2011) of the system and the online system interacting with the real world robot. While the offline machine learning system is only capable to identify models with a tree complexity of around 19 nodes with non good success rate, the online system identifies a model with complexity of 33 nodes. Furthermore as described in (Bohlmann, Klinger, Szczerbicka, and Becker 2010b) the machine learning modules offline performance decreases dramatically if additional noise is present in the input data. Obviously, experiments described in this paper additionally and inherently contain measuring noise through the use of a real physical system. The direct and automated online interaction of online simulation, verification, machine learning and physical process produce far better performance than each component alone could produce.

5. SUMMARY AND FURTHER WORK

The SMoBAICS acquisition system of motion-based action potentials in neural bundles for exo-prostheses control or for handicapped limb simulation provides an integrated solution from action potential recording up to the identification procedure. This paper focuses on a new method based on a continuous symbiotic cycle, providing a closed-loop approach. While an open-loop approach helps to identify well-known processes, improving the overall model quality, the closed-loop approach, using the symbiotic cycle, helps to generate a physical process model from scratch and forms an adaptable and scalable identification environment.

As it is continuously present, the robotic controller can react to changes in the system. The most valuable result in fact is that model complexity which could be learned from a real world process is clearly higher using a continuously interacting system. This continuous symbiotic system, where different modules, like machine learning, simulator and process models are interacting with a physical or technical process, improves the identification progress and opens the symbiotic simulation to applications from biotechnology.

The further work has the following key aspects:

- Evaluation of the whole identification process and integrating of the missing elements using the continuous symbiotic cycle.
- Evaluating the identification with regard to quality and performance.
- Adapting the symbiotic cycle to a clinical environment, to replace the Matlab-based part and to integrate the identification into a learning environment and the concerning applications.

REFERENCES

Aydt, H., Turner, S. J., Cai, W., and Low, M. Y. H. (2008a). Symbiotic simulation systems: An extended definition motivated by symbiosis in biology. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation*, PADS '08, pages 109–116, USA. IEEE Computer Society.

Aydt, H., Turner, S. J., Cai, W., and Low, M. Y. H. (2008b). Symbiotic simulation systems: An extended definition motivated by symbiosis in biology. In *Proceedings of the 22Nd Workshop on Principles of Advanced and Distributed Simulation*, PADS '08, pages 109–116, Washington, DC, USA. IEEE Computer Society.

Aydt, H., Turner, S. J., Cai, W., and Low, M. Y. H. (2009). Research issues in symbiotic simulation. In *Proceedings of the 2009 Winter Simulation Conference M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.*

Bohlmann, S., Klauke, A., Klinger, V., and Szczerbicka, H. (2011). Model synthesis using a multi-agent learning strategy. In *The 23rd European Modeling & Simulation Symposium (Simulation in Industry)*, Rome, Italy.

Bohlmann, S., Klinger, V., and Szczerbicka, H. (2009). HPNS - a Hybrid Process Net Simulation Environment Executing Online Dynamic Models of Industrial Manufacturing Systems. In *Proceedings of the 2009 Winter Simulation Conference Rossetti, Hill, Johansson, Dunkin, and Ingalls, eds.*

Bohlmann, S., Klinger, V., and Szczerbicka, H. (2010a). System Identification with Multi-Agent-based Evolutionary Computation Using a Local Optimization Kernel. In *The Ninth International Conference on Machine Learning and Applications*, pages 840–845.

Bohlmann, S., Klinger, V., Szczerbicka, H., and Becker, M. (2010b). A data management framework providing online-connectivity in symbiotic simulation. In *24th EUROPEAN Conference on Modelling and Simulation, Simulation meets Global Challenges*, Kuala Lumpur, Malaysia.

Carnevale, N. T. and Hines, M. L. (2006). *The NEURON Book*. Cambridge Univ. Press, New York, NY, USA.

Cesqui, B., Tropea, P., Micera, S., and Krebs, H. (2013). Emg-based pattern recognition approach in post stroke robot-aided rehabilitation: a feasibility study. *Journal of NeuroEngineering and Rehabilitation*, 10(1).

Coates, T.D., J., Larson-Prior, L., Wolpert, S., and Prior, F. (2003). Classification of simple stimuli based on detected nerve activity. 22(1):64–76.

Corke, P. (2011). *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®*, volume 73 of *Springer Tracts in Advanced Robotics*. Springer.

De Jong, K. A. (2006). *Evolutionary Computation: A Unified Approach*. MIT press.

Fujimoto, R., Lunceford, D., Page, E., and Uhrmacher, A. (2002). Technical report of the dagstuhl-seminar grand challenges for modelling and simulation. Technical Report No. 02351, Schloss Dagstuhl, Leibniz-Zentrum für Informatik, Wadern, Germany.

Gold, C., Henze, D. A., and Koch, C. (2007). Using extracellular action potential recordings to constrain compartmental models. *Journal of Computational Neuroscience*, 23(1):39–58.

Hazan, L., Zugaro, M., and Buzsáki, G. (2006). Klusters, NeuroScope, NDManager: A free software suite for neurophysiological data processing and visualization. *J Neurosci Methods*, (155):207–216.

Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544.

Klinger, V. (2015). Biosignal acquisition system for prosthesis control and rehabilitation monitoring. In Bruzzone, A., Frascio, M., Novak, V., Longo, F.,

Merkuryev, Y., and Novak, V., editors, *4nd International Workshop on Innovative Simulation for Health Care (IWISH 2015)*.

Klinger, V. and Klauke, A. (2013). Identification of motion-based action potentials in neural bundles using an algorithm with multiagent technology. In Backfrieder, W., Frascio, M., Novak, V., Bruzzone, A., and Longo, F., editors, *2nd International Workshop on Innovative Simulation for Health Care (IWISH 2013)*.

Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling and Analysis*. McGraw-Hill.

Micera, S., Carpaneto, J., and Raspopovic, S. (2010a). Control of hand prostheses using peripheral information. *IEEE Reviews in Biomedical Engineering*, 3:48–68.

Micera, S., Citi, L., Rigosa, J., Carpaneto, J., Raspopovic, S., Pino, G. D., Rossini, L., Yoshida, K., Denaro, L., Dario, P., and Rossini, P. M. (2010b). Decoding information from neural signals recorded using intraneural electrodes: Toward the development of a neurocontrolled hand prosthesis. *Proceedings of the IEEE*, 98(3):407–417.

Nelder, R. and Mead, J. (1965). A simplex method for function minimization. *Computer Journal*, 7(4):308–313.

Neymotin, S., Lytton, W., Olypher, A., and Fenton, A. (2011). Measuring the quality of neuronal identification in ensemble recordings. *J Neurosci*, 31(45):16398–409.

Raspopovic, S., Capogrosso, M., Petrini, F. M., Bonizzato, M., Rigosa, J., Di Pino, G., Carpaneto, J., Controzzi, M., Boretius, T., Fernandez, E., Granata, G., Oddo, C. M., Citi, L., Ciancio, A. L., Cipriani, C., Carrozza, M. C., Jensen, W., Guglielmelli, E., Stieglitz, T., Rossini, P. M., and Micera, S. (2014). Restoring Natural Sensory Feedback in Real-Time Bidirectional Hand Prostheses. *Science Translational Medicine*, 6(222).

Schmidt, M. and Lipson, H. (2007). Comparison of tree and graph encodings as function of problem complexity. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1674–1679, New York, NY, USA. ACM.

Trianni, V. (2014). Evolutionary robotics: model or design? *Frontiers in Robotics and AI*, 1:13.

Yang, X.-S., Koziel, S., and Leifsson, L. (2013). Computational optimization, modelling and simulation: Recent trends and challenges. *Procedia Computer Science*, 18:855 – 860.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, San Diego, USA, 2 edition.

AUTHORS BIOGRAPHY

SEBASTIAN BOHLMANN is a Ph.D. candidate at Department of Simulation and Modeling - Institute of Systems Engineering at the Gottfried Wilhelm Leibniz Universität Hannover. He received a Dipl.-Ing. (FH) degree in mechatronics engineering from FHDW university of applied science. His research interests are machine learning and heuristic optimization algorithms, complex dynamic systems, control system synthesis and grid computing. His email address is <bohlmann@ifam.uni-hannover.de>.

VOLKHARD KLINGER is a professor for embedded systems and computer science at the university of applied science FHDW in Hannover and Celle since 2002. After his academic studies at the RWTH Aachen he received his Ph.D. in Electrical Engineering from Technische Universität Hamburg-Harburg. During his 8-year research activity at the Technische Universität Hamburg-Harburg and research laboratories he did research in asic design and automation. He teaches courses in computer science, embedded systems, electrical engineering and ASIC/system design. His email address is <Volkhard.Klinger@fhdw.de>.