Robotics and Autonomous Systems 131 (2020) 103566

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A spiking network classifies human sEMG signals and triggers finger reflexes on a robotic hand ☆

J. Camilo Vasquez Tieck [a,*], Sandro Weber [b], Terrence C. Stewart [c], Jacques Kaiser [a], Arne Roennau [a], Rüdiger Dillmann [d]

[a] FZI Research Center for Information Technology, Karlsruhe 76131, Germany
[b] TUM Technical University of Munich, 80333 München, Germany
[c] Centre for Theoretical Neuroscience, University of Waterloo, Canada N2L 3G1
[d] Karlsruhe Institute of Technology (KIT), Germany

ARTICLE INFO

Article history:
Received 10 April 2019
Received in revised form 18 November 2019
Accepted 15 May 2020
Available online 21 May 2020

Keywords:
Neurorobotics
Human–robot-interaction
Neural control system
Humanoid robot
Motion representation
sEMG classification
Spiking neural networks
Anthropomorphic robot hand

ABSTRACT

The interaction between robots and humans is of great relevance for the field of neurorobotics as it can provide insights on how humans perform motor control and sensor processing and on how it can be applied to robotics. We propose a spiking neural network (SNN) to trigger finger motion reflexes on a robotic hand based on human surface Electromyography (sEMG) data. The first part of the network takes sEMG signals to measure muscle activity, then classify the data to detect which finger is being flexed in the human hand. The second part triggers single finger reflexes on the robot using the classification output. The finger reflexes are modeled with motion primitives activated with an oscillator and mapped to the robot kinematic. We evaluated the SNN by having users wear a non-invasive sEMG sensor, record a training dataset, and then flex different fingers, one at a time. The muscle activity was recorded using a Myo sensor with eight different channels. The sEMG signals were successfully encoded into spikes as input for the SNN. The classification could detect the active finger and trigger the motion generation of finger reflexes. The SNN was able to control a real Schunk SVH 5-finger robotic hand online. Being able to map myo-electric activity to functions of motor control for a task, can provide an interesting interface for robotic applications, and a platform to study brain functioning. SNN provide a challenging but interesting framework to interact with human data. In future work the approach will be extended to control also a robot arm at the same time.

1. Introduction

The interaction of humans and robots is relevant for the field of neurorobotics as it can provide insights on motor control and sensor processing mechanisms in humans that can be applied in robots.

Electromyography (EMG) is a common tool in medicine and biomechanics to monitor and study the electrical activity of the muscles. There are different methods to record EMG signals, and they can be either invasive or non-invasive. Research is being made on processing and classification of surface EMG (sEMG) signals for clinical diagnoses [1,2] or prosthetic applications [3]. For finger motion, [4] analyzed interference in the EMG signals for classification of finger flexion motions using maximum likelihood estimation, and [5] performed a time domain feature extraction to enhance the classification and use it to control a prosthetic hand.

There are methods using a spiking neural network (SNN) [6,7] as a drop-in state representation layer in combination with an artificial neural network (ANN). In [8], a SNN is combined with an ANN for control of musculo-skeletal kinematic structures, and in a similar way, [9] uses an SNN as feature extraction layer to feed an ANN to classify hand gestures based on sEMG. There are also SNN implementations in neuromorphic hardware to process sEMG signals. In [10], a recurrent SNN was used to process sEMG signals from a rock–paper–scissors gesture dataset, and in [11] the performance of a feed-forward SNN to classify the gestures in the same dataset was analyzed. Nevertheless, there are different hypothesis explaining how does the human motor system work. A wide accepted theory states that the central nervous system uses different base motor components in a hierarchy [12] to generate the full repertoire of motions that we can perform [13]. These base components are formed by specific combination of muscle synergies [14] that are active during a motion, and they are commonly called motor primitives [15]. An approach using

☆ This work is an extension of Tieck et al. (2018).

* Corresponding author.
E-mail addresses: tieck@fzi.de (J.C.V. Tieck), webers@in.tum.de (S. Weber), tcstewar@uwaterloo.ca (T.C. Stewart), jkaiser@fzi.de (J. Kaiser), roennau@fzi.de (A. Roennau), ruediger.dillmann@kit.edu (R. Dillmann).

https://doi.org/10.1016/j.robot.2020.103566

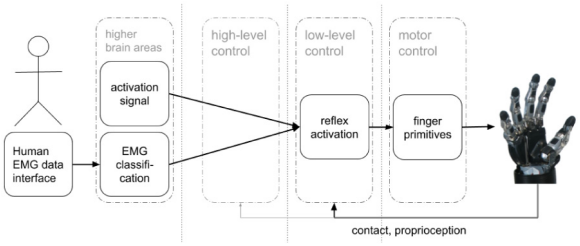


Fig. 1. Concept architecture with main components. Classification of sEMG signals to detect active finger is used to trigger motion on the robot hand. *Source:* Adapted from [19].

these concepts to control a robotic hand was proposed in [16]. Motor primitives can be activated in different modalities [17], for example as a reflex. A reflex is an involuntary response to sensor stimulation and can be either a complete execution or inhibition of a motion. An overview of different spinal reflexes is provided by [18].

In this work, we propose a SNN to control a robot hand with muscle signals from a human recorded with a non-invasive sEMG sensor. In Fig. 1, we present an overview of the main components showing how they interact with each other. Human muscle activity is recorded with a non-invasive sEMG sensor. The sEMG signals are encoded to spikes. The first part of the SNN performs a classification to detect which finger was active and generates an activation signal. This sub network is first trained offline with labeled data from the user, and then used online to classify. The second part of the SNN generates motion and maps it to motor commands according the robot kinematics.

We focus on the activation of single finger motions in response to sensor stimuli as the reflex mechanism presented in [17,19]. A SNN was implemented to classify sEMG data and then trigger the generation of motion as a reflex. An sEMG sensor with eight channels was used to record human muscle activity while moving different fingers. First, sEMG data was encoded to spikes and the signals were classified to identify the active finger. After that, the activation signal was used to trigger a reflex to generate motion using a motor primitive. Then, the primitive output is mapped to the robot kinematics. Finally, the spikes are decoded to motor commands for the robot.

One highly novel aspect of this paper is the fact that the classification and the generation of the motor primitive is implemented using a spiking neural network. There are two reasons for doing this. First, the real biological system must do something similar to this using spiking neurons. Certainly, in real biology, the classification would not be based on sEMG sensor, but would rather be based on neural activity somewhere in the brain, but the classification and generation of movement over time would still need to occur. This means that we can see our system as an initial model of that biological process. Second, there is a pragmatic or engineering reason to implement this system using spiking neurons. Prosthetic applications benefit from low-power hardware implementations, and there is a variety of neuron-inspired low-power computing hardware being developed. For example, the SpiNNaker system [20], Intel's new Loihi chip [21] and IBM's TrueNorth chip [22] all provide extremely energy-efficient spiking neuron hardware. If the algorithms can be mapped onto this hardware, then they may be able to be deployed using significantly less power than traditional implementations.

This work is an extension of the previous conference paper [19]. We extend the modeling of motions with motor primitives and explain in more detail how to map the primitives to different robots with references. We explore related work on other sEMG techniques with machine learning for finger motion classification as well as other SNN approaches for sEMG processing. We present details on the sEMG sensor and the signal characteristics and we add an experiment on another user with the left arm to the evaluation. There are new figures and all captions have been edited, we add extended explanations and discussion on the implications and limitations. We elaborate on how to get the amount of flexion, the advantages of SNN to use neuromorphic hardware, and add a discussion on the EMG dataset.

2. Methods

The main motivation of this work is to control a robotic hand with human muscle signals using a spiking neural network (SNN). As requirement, the sEMG sensor is non-invasive, the finger motions are represented with motor primitives triggered at once to resemble reflexes, and the robot hand has to be controllable with a ROS interface. We divide the problem in two parts that translate to different parts of the same SNN. The first part takes care of the sEMG data interface and classification and the second part takes care of the motion generation and robot control. The first part of the SNN classifies the sEMG signals to detect which finger was active. Human sEMG data is captured in a non-invasive way. For the classification, the sub network is trained offline with labeled data from the user. After that, it is used to classify online the sEMG signals and generate an activation signal that is passed on to the second part of the SNN to trigger single finger reflexes on the real robot. A finger reflex is modeled with an oscillator that activates a motor primitive that is mapped to motor commands according to the robot kinematics. In Fig. 1, we present an overview of the main components showing how they interact with each other. A detailed architecture of the SNN is presented in Fig. 2. Notice that the primitives for the thumb, ring and pinky are mapped to one actuated joint, whereas the index and middle finger primitives are mapped to two actuated joints.

To generate the SNN models, we used the Neural Engineering Framework [23] and the software package Nengo [24]. This software allows for the creation of large-scale spiking neural networks by breaking the networks down into smaller parts. The connection weights for each sub-part are optimized separately, and then they are combined together into one large neural network. Performing this optimization (i.e. finding connection weights) locally means we can generate large systems without using the traditional neural network approach of optimizing over huge amounts of training data. However, the trade-off is that we must make explicit claims about what each sub-part of the model is doing.

In particular, in order to define a spiking neuron model using Nengo and the neural engineering framework (NEF) [25], we must break our algorithm down into vectors, functions, and differential equations. The activity of each group of spiking neurons is considered to be a distributed representation of a vector (i.e. we may have 100 spiking neurons representing a 2-dimensional vector). Connections between groups of neurons compute functions on those vectors. That is, the connection weights ensure that if the first group of neurons represents x, then the second group of neurons will represent $y = f(x)$. By changing the connection weights, we change the function being computed. Finally, recurrent connections can be used to implement differential equations. We make use of this here to implement basic movement primitives.

2.1. Human sEMG data interface and training data

To record sEMG data a single Myo [26] gesture control armband is used. It is made up of eight equally spaced out blocks with non-invasive sEMG sensors that provide a sampling rate of

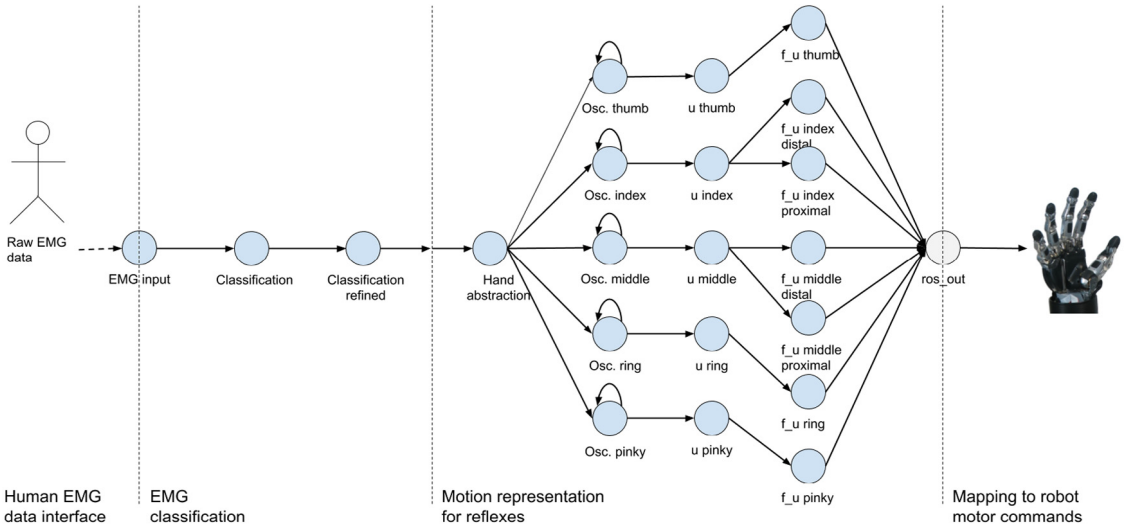


Fig. 2. Detailed architecture with sEMG classification and motion sub networks. Each circle represents a population of spiking neurons. The dotted lines divide the conceptual components, named on the bottom according to Fig. 1.
Source: Adapted from [19].

Fig. 3. The Myo armband sensor placed on the forearm and the activity of the eight channels.
Source: Adapted from [19].

200 Hz. The armband is used around the middle of the forearm as shown in Fig. 3. When a finger is moved the muscle electric activity is recorded with eight different sensors. The sensor has an indicator so that it can be placed always in a similar way. In order to record consistent data with the sensor, the segment with the LED light has to be placed approximately at the same position. We use a marker to mark the position on the underarm skin. Slight variations after re-wearing the myo on and off did not have enough influence on the recordings to make the trained network unusable. Retrieving the raw sEMG signals is done with the help of a Python API provided by [27]. Each channel encodes the individual measurement as *int*8 values.

For each user a training dataset is required with multiple samples. A sample consists of a continuous sequence of finger activation in one hand. Each finger has to be flexed down for a short period of time and then extended again. This procedure is repeated starting from the thumb to the pinky. The training data has to be recorded as a time continuous sEMG stream of all eight channels with appropriate binary labels for the time windows during which a finger was pressed. A sample from the dataset is provided in Fig. 4, and the signals of the eight channels are plotted with different colors in the upper part. The individual channels of the sEMG sensor have similar activation for different fingers, and thus are not enough to identify the motion of a finger. Therefore, the classification network uses a combination of all

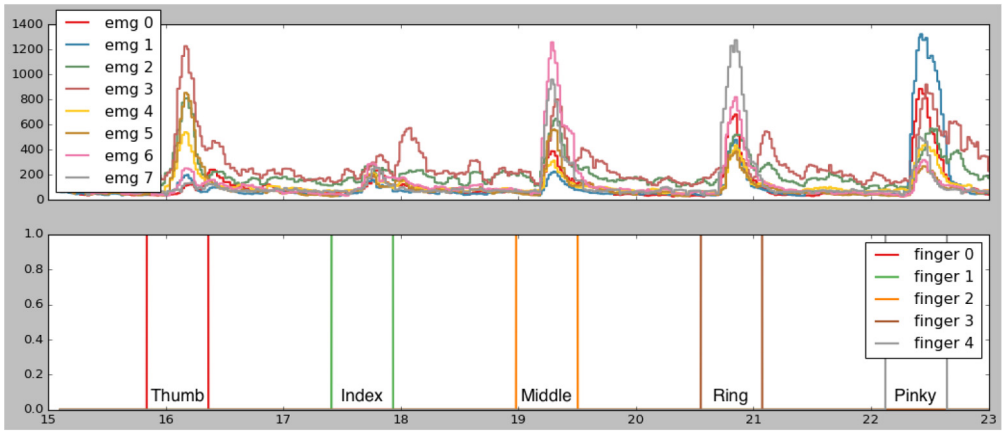


Fig. 4. A sample from the dataset for training with a run of all 5 fingers. From left to right the peaks show sEMG activation of the fingers starting with the thumb to the pinky. Each finger is flexed and then extended.
Source: Adapted from [19].

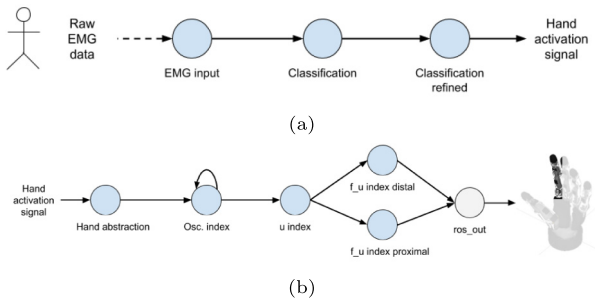


Fig. 5. Detailed architectures for the (a) sEMG classification and (b) motion representation.
Source: Both adapted from [19].

eight channels, which provides a unique representation for each finger.

The Myo uses proprietary EMG sensors and the values in Fig. 4 are the raw output of the eight channels in the armband. We do not really know what each of these raw EMG signal represent, but it is proportional to the muscle contraction intensity. Around 0 you have a relaxed muscle. We cannot say it is the direct muscle contraction measured in millivolts, because the signal is internally processed and converted to 8-bit data.

2.2. Sub network for sEMG classification

After recording training data, the first part of the SNN is trained for classification offline and then used for online classification. The detailed architecture for sEMG classification is presented in Fig. 5(a). Each circle represents a population of spiking neurons. Raw sEMG data is recorded from the user and is encoded into spikes. A population of neurons encodes the signal stream of sEMG input to spikes using stochastic population encoding. The encoded sEMG signals are classified to determine which finger was activated. Then, a second population of neurons is trained offline with a whole training dataset for a user as described above.

The learning rule for offline training the classification population is Prescribed Error Sensitivity (PES) using the labels from the training data serving as error signals E. PES is implemented in [28], and was first presented in [29]. This learning rule is independent of the data to learn, it optimizes the network weight to minimize the error. The weight updates that PES makes to minimize an error signal during learning can be related to Skipped Back-propagation. For the weights w_{ij} from pre-synaptic population i to post-synaptic population j, the update rule is defined as

$$\Delta w_{ij} = \kappa \alpha_j e_j \cdot E a_i, \tag{1}$$

with κ a scalar learning rate, α the gain or scaling factor, e the encoder for the neuron, E the error to minimize, and a the desired activation.

After training, the classification by the second population, the overall signals are low in amplitude and consequently rather close to each other. Therefore, a population is used to refine the classification by amplifying the signals and generating the hand activation signal. An arbitrary defined function scales up activations above a manually set threshold and is used to train this population. The resulting activation signal is passed over to the motion generation part. Examples for classification of all the fingers of the complete procedure are provided in Fig. 11 in the results section. Neurons in consecutive populations are connected all to all.

2.3. Sub network for motion representation of reflexes

A population takes the hand activation signal from the previous classification to trigger the appropriate finger reflex. We model a reflex as the execution of a motor primitive based on a specific stimuli. Accordingly, the motion part of our SNN is divided into reflex activation and motor primitive layers. The whole architecture for the representation of reflexes is presented in Fig. 5(b). Each circle represents a population of spiking neurons. The hand activation signal is processed by the hand abstraction population to extract individual finger activations. Reflexes are modeled as oscillators that oscillate only once. The neural activity is decoded to generate motor commands to the respective robot finger.

The reflex activation for each finger is modeled as an oscillator

$$h(\omega) = a \cdot \sin(b\omega \frac{\pi}{2}), \tag{2}$$

with ω a recurrent connection and a and b the parameters for the amplitude and frequency. The oscillator generates a continuous signal for a finite period that represents the duration of a motion with a start and an end point. By indexing the neurons in the oscillator population, the activity can be mapped to a grid in a $2D$ plane. The total spike activity of the neurons in the population can then be represented with the components x and y. We calculate a continuous and normalized signal $u \in [-1, 1]$ as:

$$u = \sin(\arctan(\frac{y}{x})), \tag{3}$$

where $\arctan(\frac{y}{x})$ represents the angle of a vector with components x and y. To bound and smooth the signal $sin()$ is applied.

The motor primitive is modeled as a mapping of u to a sequence of joint activations during the period of oscillation, and it can be mapped to one or multiple joints. In Fig. 5(b), the distal and proximal joints of the index finger are mapped. First, we define an activation function

$$f(u) = \frac{\sin(u \cdot \pi - \frac{\pi}{2})}{2} + \frac{1}{2}, \tag{4}$$

as a sinusoidal function to have smooth initial an final phases. This characteristic is important when executing the motion in a real robot to prevent unnecessary wear in the motors and mechanical parts. The resulting generic activation function for one joint is depicted in Fig. 6(a). In general terms, there is no difference between voluntary and reflex motions on the muscular activities (synergies), the difference is the activation [17]. For voluntary motions the activation is discrete, and for reflexes it is a complete one time execution.

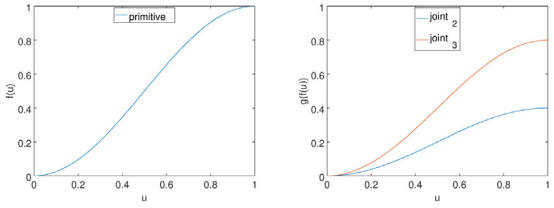
2.4. Mapping to robot motor commands

Finally, in order to actually be able to control the robot, the primitives have to be mapped to the robot kinematics. Which means, scaling to the motion interval $(\theta_{max} - \theta_{min})$ and offset θ_{min} that the joint θ has. For this purpose, we define $g : [0, 1] \rightarrow \mathbb{R}^n$ as a function for each joint as

$$g(f(u)) = f(u) \cdot (\theta_{max} - \theta_{min}) + \theta_{min}, \tag{5}$$

to generate appropriate motor commands. A schema for the mapping g for a robotic hand is illustrated in Fig. 6(b) and is defined in the table in Fig. 6(c). A joint is defined with a name, the associated primitive, and the interval of the joint θ_{min} and θ_{max}. A primitive can be mapped to one or more joints. This parametric representation of motions allows us to further combine and change parameters of the motor primitives.

With this representation of the robot kinematics we also have a flexible and extensible approach that can be adapted to different



(a) (b)

Joint name	Primitive	θ_{min}	θ_{max}
$joint_1$	0	0	0.7
$joint_2$	1	0	0.4
$joint_3$	1	0	0.8

(c)

Fig. 6. Motor primitive representation. The control parameter u is used to evaluate (a) the activation function $f(u)$ and then (b) the mapping $g(f(u))$ to the robot kinematics. (c) Mapping description table.
Source: Adapted from [19].

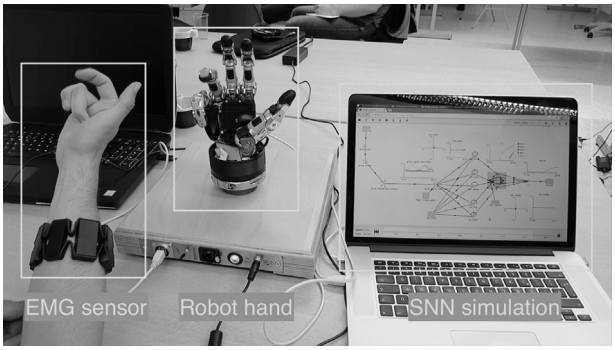


Fig. 7. Experiment setup: user, sEMG sensor, robot hand, and the SNN simulation in Nengo.
Source: Adapted from [19].

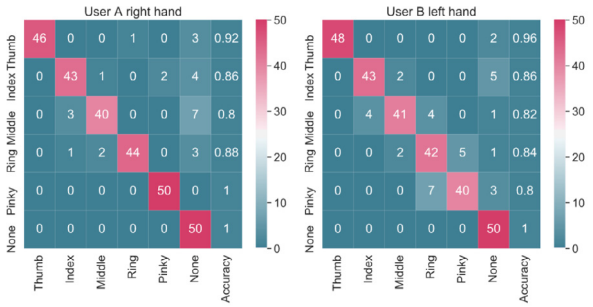
robots. To use a different robot a new mapping has to be defined as in Fig. 10(c). The primitive modeling remains the same and the activation functions too. In other approaches we show how to map the primitives to a robotic arm in simulation [30], a humanoid robot [31], and a simulated six-legged robot [32].

3. Results

The experiment setup is depicted in Fig. 7. The user wears the sEMG sensor (Myo armband) in the forearm, and the signals are sent via bluetooth to the computer. The computer receives the sEMG data and inputs it to the SNN simulation running in Nengo [24]. The computer communicates at the same time with the robot hand (Schunk SVH) via ROS [33]. To control the robot hand the official Schunk ROS driver is used [34].

3.1. Classification performance

In order to evaluate the accuracy of the classification, we selected two random users and one of them using the sensor on the right and the other on the left arm. This experiment is by no means a complete evaluation, nevertheless it provides an idea of how the classification works with different users. The SNN is trained offline from scratch for each arm (see Section 2.2), so one trained model only learns to classify the signals of one particular arm, which would be the use case in prosthetic applications. Each user was asked to perform a sample of 50 trials with each finger (see Section 2.1). The sEMG data was feed to the trained



(a) (b)

Fig. 8. Correlation matrix of classification for two random users – one using the sensor on the right and one on the left arm. Each row represents the results for each finger.

network and we counted the classification output for each trial. We counted which finger class – Thumb, Index, Middle, Ring, Pinky, None – was detected and calculated the ratio of accuracy for each finger $Accuracy = \frac{value}{N\_trials}$. The results are summarized in Fig. 8, where each row is the classification of one finger.

For user A using the right arm see Fig. 8(a). Notice that only the "pinky" and "none" were always classified correctly. This does not mean that it will be always perfect, only that during this 50 samples the classification was correct. For the other fingers there are either false detections or "none" classifications. The middle finger was the hardest to classify, and the network often misclassified it as "none".

For user B using the left arm see Fig. 8(b). Notice that there was misclassification between the ring and pinky fingers. During the test, it was evident that for most people it is hard to move the fingers independently. Also notice that the middle finger was also problematic. For the other fingers there are either false detections or "none" classifications.

In order to improve performance, especially on the problematic fingers, an approach with a time-windowed history of sEMG data and subsequent feature pre-processing could help. In [35] a combination of absolute mean amplitude and cumulative length of the sEMG signal within the analysis window showed the best results. Alternatively, recurrent connections on the group-answer population (see Fig. 9) might also help in processing the time-continuous sEMG data. The main focus going further is to figure out a way to not only detect the active finger being flexed, but to estimate the amount of flexion in that finger.

3.2. SNN implementation

The SNN was implemented in Python with the Nengo simulator using leaky integrate and fire (LIF) neurons. To get an overview of the implementation Fig. 9 presents a view of the whole network running. The structure can be easily mapped to that in Fig. 2. The eight channels human sEMG signals are encoded by a population as stochastic spike rates based on their values. After performing offline training with different datasets from the same user, the sEMG classification takes place. The classification signal is then passed over to trigger motion generation. The reflexes are implemented as oscillators that activate motor primitives. The motor primitives are mapped to the robot kinematics as defined in the methods section. In the following sections the relevant details of each components are described.

3.3. Interface to the robot hand

For this work we used a Schunk SVH 5-finger robot hand (see Fig. 10(a)). The hand has 9 actuated joints and other joints

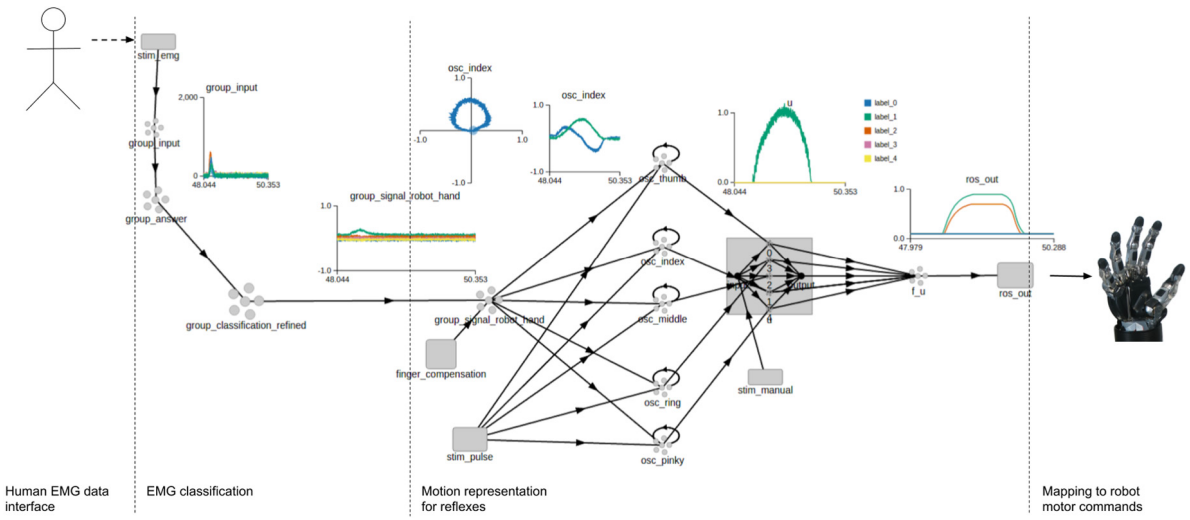
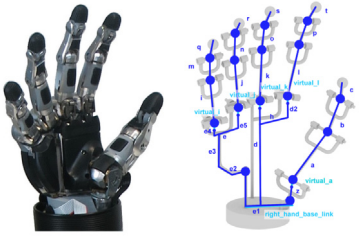


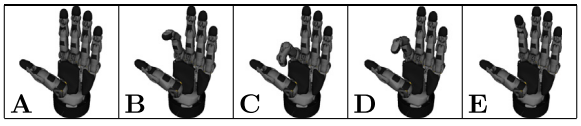
Fig. 9. The SNN pipeline in Nengo. The main components are: human sEMG data capture and encoding, sEMG classification, motion generation, and finally the mapping to the robot. The structure can be mapped to that in Fig. 2.
Source: Adapted from [19].



(a) (b)

Joint name	Primitive	θ_{min}	θ_{max}
Thumb_Flexion	0	0	0.3
Thumb_Opposition	-	-	-
Index_Finger_Distal	1	0	0.9
Index_Finger_Proximal	1	0	0.7
Middle_Finger_Distal	2	0	0.9
Middle_Finger_Proximal	2	0	0.7
Ring_Finger	3	0	0.7
Pinky	4	0	0.7
Finger_Spread	-	-	-

(c)



(d)

Fig. 10. (a) We used a Schunk SVH 5-finger robot hand with (b) kinematic structure (adapted from [34]). (c) Table. Joint mapping schema for the real robot. (d) Frame sequence of the index finger motion generated by the SNN corresponding to the activation shown in Fig. 12 (adapted from [19]).

mirror the activity with a fixed factor (see Fig. 10(b)). Our network controlled 7 of the joints. We describe here the mapping schema that was used with the robot hand. The table in Fig. 10(c) summarizes the data. The "Joint name" column corresponds to the ROS topics described in [34] for the different actuated joints. A different primitive is used for each finger, and the indexing is in column "Primitive". Note all joints of the index and middle finger are mapped to the same primitives respectively. The "min" and "max" values for the joint angle interval for each joint complete the table. The active joints in the robot hand "Thumb_Opposition" and "Finger_Spread" remained constant all the time.

3.4. Training data

For each user a set of training data is required to train offline the SNN as described in Section 2.2. Training data for the classification network was recorded in one session lasting 60 s. During that time individual fingers were periodically pressed against the palm of the hand and subsequently returned to a resting pose. The fingers presses occurred in sequence from thumb to index finger with each press lasting between 300 ms and 500 ms. Together with the resting time one cycle took around 7.5 s and a total of 8 cycles were performed. A sample run of the training data with all 5 fingers is presented in Fig. 4. In order to label sEMG data, with every finger press simultaneously a keyboard button was pressed indicating the respective finger. The data is labeled in time for each finger, and all eight sEMG channels are active. Important note, if the sensor is used in a different position or in a different arm, a new dataset is required and the network needs to be trained again.

3.5. Processing of sEMG data and classification

The first group of 800 neurons (sEMG input in Fig. 5(a)) was activated with the raw sEMG data as *int*8 to convert it to spikes. The second group of 500 neurons (Classification in Fig. 5(a)) was trained with the prerecorded training data at start to give responses of the classified fingers. Then, a third group of 500 neurons (Classification refined in Fig. 5(a)) was used to separate and amplify signals further. A final group of 500 neurons (Hand activation signal, Fig. 5(a)) was trained to give out one single signal for a specific finger and was connected to 5 groups representing the different fingers for the robot hand.

In Fig. 11, we present samples of the SNN classifying the activation of the different fingers. As it can be seen, the eight channels of sEMG have different data for each finger. The signals are processed with the SNN, and the activation of the different populations can also be observed. The output of the classification is a dominant activation of one of the populations representing each finger.

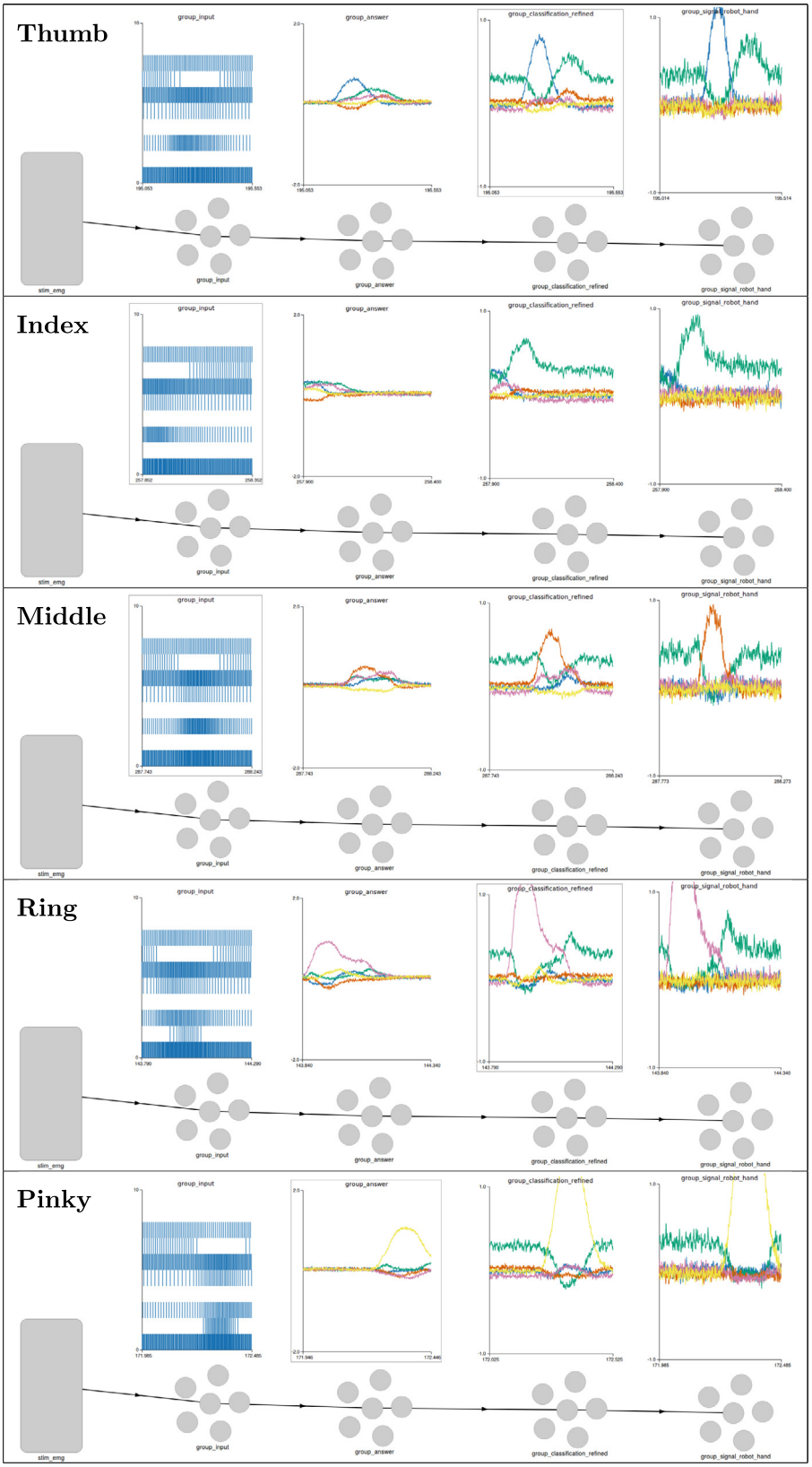


Fig. 11. sEMG activation and classification for different fingers. The first graph from the left shows the encoded sEMG signals to spikes. The second plot shows the classification output. The other plots show the refined classification signal and the hand activation signal respectively with one finger active. From top to bottom thumb, index, middle, ring and pinky are shown. Notice the different activations in the eight channels.
Source: Adapted from [19].

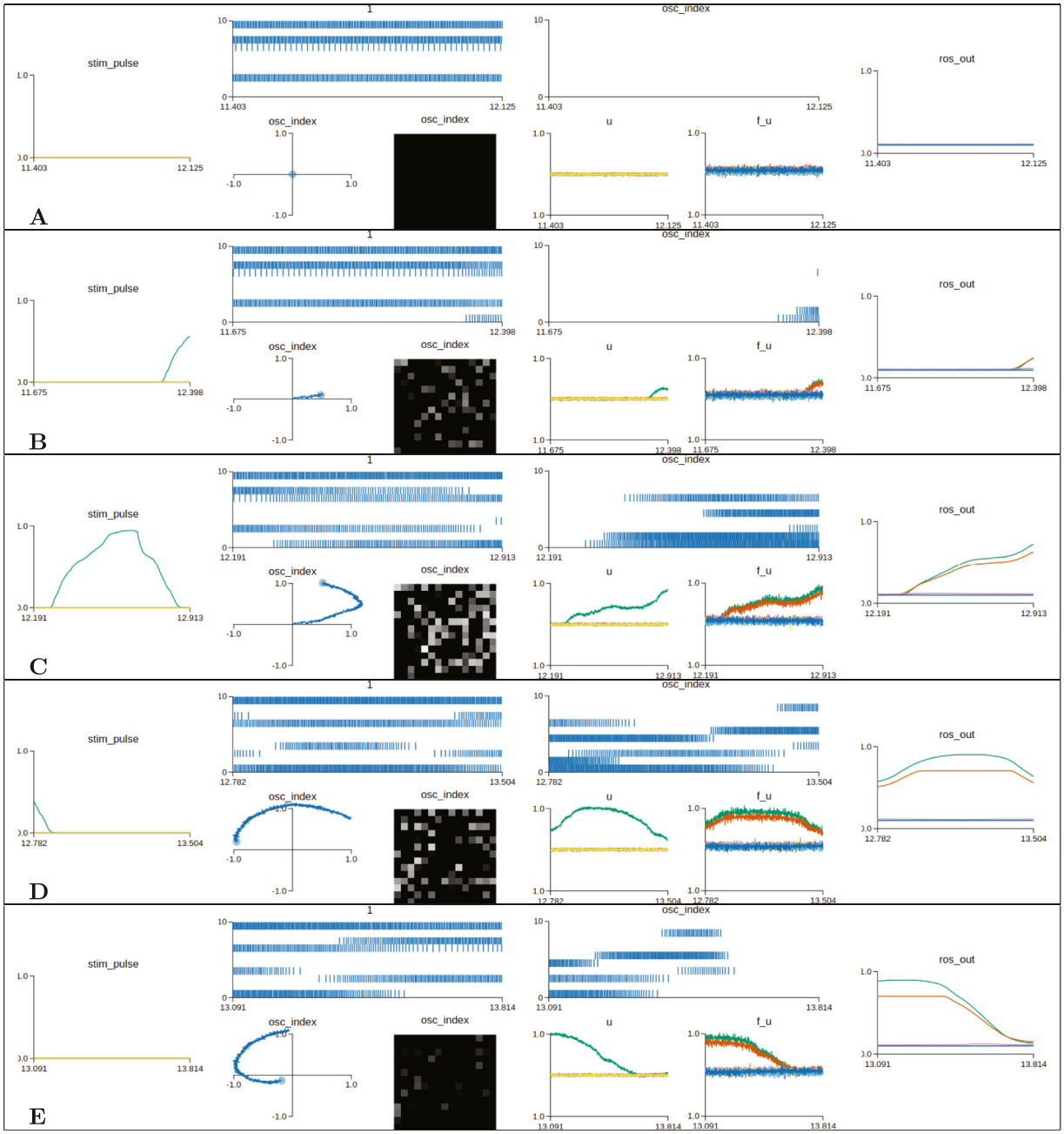


Fig. 12. Motion generation from the SNN for a reflex in the index finger. The frames A to E correspond to the fingers from the thumb to the pinky respectively. From left to right the plots represent: (*left*) Finger activation signal coming form the sEMG classification. (*middle left, group of three plots*) Show the activity in the oscillator with a spike train plot, the decoded activity of the population in a plane *XY*, and a raster plot color coded by the neuron's activity. (*middle right, group of three plots*) Show the decoding to u, and the mapping $g(f(u))$ to the robot kinematics. Note that the mapping goes to two different joints. (*right*) A continuous plot of the current motor commands being sent to the robot.
Source: Adapted from [19].

3.6. Motion generation

After receiving the classification signal the reflex is activated and fully executed once. The resulting motion of the robot is presented as a frame sequence in Fig. 10(d). The data presented corresponds to a reflex motion of the index finger. The corresponding activity of the SNN is presented in Fig. 12 with sufficient information to illustrate the functioning of the SNN. The signal that triggers motion generation comes from the classification part. An oscillator is activated for each finger. Observe the circular activation of the oscillator population when decoded in a plane *XY* in Fig. 12. From this circular activation u is decoded and mapped to one or more joints in the robot hand. Observe that the mapping in Fig. 12 is performed to two joints of the index finger. Finally the neural activity is decoded and send over ROS to the robot hand.

4. Discussion

We presented a SNN that activates motion reflexes on a robotic hand based on human sEMG data online. After training, the network classifies the sEMG signals to detect finger activation. Based on it, single finger reflexes are triggered. The finger reflexes are modeled with motion primitives and mapped to the robot kinematics.

Modeling robot motion with motor primitives is a promising approach as it provides a clear framework in combination with a model free representation of the kinematics. Motions can be learned for different tasks and the approach has been successfully extended to model different activation modalities [17] target reaching [30] and locomotion [32]. A notable capability of our approach is that it works with no modifications with a left or a right arm.

As can be seen in Figs. 4 and 11, the index finger showed in the raw sEMG data a signal that is not clear and the output is sometimes ambiguous. As a consequence, the classification step delivers a weak and low activation signal that is propagated throughout the following populations and leads to false classification of other fingers. We noticed that for most people it is hard to move the fingers independently in a consistent way. One important note, the EMG dataset is of no use if the sensor is later used in a different position or in a different user or arm. In other words a trained SNN cannot be evaluated if you do not have the same user and a similar sensor placement. In that case a new dataset is required to train the network again.

The focus of this work was on single finger movements, so data with multiple fingers was not considered, only movements in quick succession of single fingers. We observed that the amount of flexion during the motion of one finger is proportional to the sEMG signal (see Section 2.1), but the signal is not stable if there is no motion or if it is too slow. The sEMG signals were used only to trigger the execution of the reflexes, and for the classification a time sequence of the whole motion was used. We currently work on modeling to the whole hand by representing grasping affordances as a higher coordination layer to the finger primitives. In order to detect the affordances based on event-based visual information we use a spiking variant of back-propagation to train a SNN and trigger the respective grasping motion [36]. A method similar to [37] could be incorporated togheter with microsaccades [38] to get depth information from the event-based visual information to adapt the motion to the position of the objects.

Brain-inspired technologies are attracting interest from the industry, especially neuromorphic hardware and event-based computation. New paradigms are required to program SNN to efficiently. By implementing our methods with SNN we are bridging the gap between neuromorphic technologies and robotics. Additionally, other recent methods using reservoir computing to learn muscle control [8] or to learn how to classify sEMG signals [11] in neuromorphic hardware, could provide the foundation to learn more complex muscle control using SNN. We want to explore a mechanism to use the sEMG signals from the Myo armband to detect the amount of flexion of one finger in the human and perform discrete control of the position with the robot finger. For this we need to figure out a representation that characterizes a position with the sEMG signals in a unique way. Using a second sEMG sensor could provide additional input from different muscle areas of the arm and improve classification results. Ideally the second sEMG sensor would be located close to the wrist [39] closer to the fingers and the hand.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2).

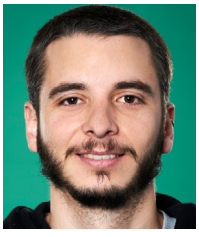
Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2020.103566.

References

[1] R.H. Chowdhury, M.B. Reaz, M.A.B.M. Ali, A.A. Bakar, K. Chellappan, T.G. Chang, Surface electromyography signal processing and classification techniques, Sensors (2013).
[2] M. Simão, N. Mendes, O. Gibaru, P. Neto, A review on electromyography decoding and pattern recognition for human-machine interaction, IEEE Access (2019).
[3] M.S. Johannes, J.D. Bigelow, J.M. Burck, S.D. Harshbarger, M.V. Kozlowski, T. Van Doren, An overview of the developmental process for the modular prosthetic limb, Johns Hopkins APL Tech. Dig. (2011).
[4] K.-J. You, K.-W. Rhee, H.-C. Shin, Finger motion decoding using EMG signals corresponding various arm postures, Exp. Neurobiol. (2010).
[5] A.H. Al-Timemy, G. Bugmann, J. Escudero, N. Outram, Classification of finger movements for the dexterous hand prosthesis control with surface electromyography, IEEE J. Biomed. Health Inform. (2013).
[6] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural Netw. (1997).
[7] A. Grüning, S.M. Bohte, Spiking neural networks: Principles and challenges, in: ESANN, 2014.
[8] J.C.V. Tieck, M.V. Pogančić, J. Kaiser, A. Roennau, M.-O. Gewaltig, R. Dillmann, Learning continuous muscle control for a multi-joint arm by extending proximal policy optimization with a liquid state machine, in: Conf. on Artificial Neural Networks ICANN, 2018.
[9] S. Lobov, V. Mironov, I. Kastalskiy, V. Kazantsev, A spiking neural network in sEMG feature extraction, Sensors (2015).
[10] E. Donati, M. Payvand, N. Risi, R. Krause, K. Burelo, G. Indiveri, T. Dalgaty, E. Vianello, B. Circuits, Processing EMG signals using reservoir computing on an event-based neuromorphic system, in: IEEE Biomedical Circuits and Systems Conf., BioCAS, 2018.
[11] E. Donati, M. Payvand, N. Risi, R.B. Krause, G. Indiveri, Discrimination of EMG signals using a neuromorphic implementation of a spiking neural network, IEEE Trans. on Biomed. Circuits Syst. (2019).
[12] E. Bizzi, V. Cheung, A. d'Avella, P. Saltiel, M. Tresch, Combining modules for movement, Brain Res. Rev. (2008).
[13] N. Bernstein, The co-ordination and regulation of movements, Pergamon-Press, 1967.
[14] A. d'Avella, P. Saltiel, E. Bizzi, Combinations of muscle synergies in the construction of a natural motor behavior, Nat. Seurosci. (2003).
[15] E. Chinellato, A. Pobil, The Visual Neuroscience of Robotic Grasping: achieving Sensorimotor Skills Through Dorsal-Ventral Stream Integration, in: Cognitive Systems Monographs, Springer, 2016.
[16] J.C.V. Tieck, H. Donat, J. Kaiser, I. Peric, S. Ulbrich, A. Roennau, M. Zöllner, R. Dillmann, Towards grasping with spiking neural networks for anthropomorphic robot hands, in: Conf. on Artificial Neural Networks, ICANN, 2017.
[17] J.C.V. Tieck, L. Steffen, J. Kaiser, R. Arne, R. Dillmann, Multi-modal motion activation for robot control using spiking neurons, in: IEEE Conf. Biomedical Robotics and Biomechatronics, BioRob, 2018.
[18] J. Knierim, Spinal reflexes and descending motor pathways, Neuroscience (2019) Online (Accessed 9 April 2019).
[19] J.C.V. Tieck, S. Weber, T.C. Stewart, A. Roennau, R.R. Dillmann, Triggering robot hand reflexes with human EMG data using spiking neurons, in: Conf. Intelligent Autonomous Systems IAS-15, Springer, Cham, 2018.
[20] S. Furber, S. Temple, A. Brown, High-performance computing for systems of spiking neurons, in: AISB'06 Workshop. GC5: Archit. Brain Mind, 2006.
[21] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S.H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Loihi: A neuromorphic manycore processor with on-chip learning, IEEE Micro (2018).
[22] P.A. Merolla, J.V. Arthur, R. Alvarez-Icaza, A.S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science (2014).
[23] C. Eliasmith, C.H. Anderson, Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems, MIT Press, 2003.
[24] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T.C. Stewart, D. Rasmussen, X. Choo, A. Voelker, C. Eliasmith, Nengo: a Python tool for building large-scale functional brain models, Front. Neuroinform. (2014).
[25] C. Eliasmith, C.H. Anderson, Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems, MIT Press, 2003.
[26] ThalmicLabs, Myo diagnostics, 2019, http://diagnostics.myo.com/. Online (Accessed 9 April 2019).

[27] dzhu, Myo Python API, 2019, https://github.com/dzhu/myo-raw/. Online (Accessed 9 April 2019).
[28] T. Bekolay, C. Kolbeck, C. Eliasmith, Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks, Cogn. Sci. (2013).
[29] D. MacNeil, C. Eliasmith, Fine-tuning and the stability of recurrent neural networks, PLoS One (2011).
[30] J.C.V. Tieck, L. Steffen, J. Kaiser, D. Reichard, A. Roennau, R. Dillmann, Combining motor primitives for perception driven target reaching with spiking neurons, in: Cognitive Informatics and Natural Intelligence, Vol. 13, IJCINI, IGI Global, 2019.
[31] J.C.V. Tieck, T. Schnell, J. Kaiser, F. Mauch, A. Roennau, Generating pointing motions for a humanoid robot by combining motor primitives, Front. Neurorobot. (2019).
[32] J.C.V. Tieck, J. Rutschke, J. Kaiser, M. Schulze, T. Buettner, D. Reichard, A. Roennau, Combining spiking motor primitives with a behavior-based architecture to model locomotion for six-legged robots, in: IEEE Conf. Intelligent Robots and Systems, IROS, 2019.
[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, in: IEEE Conf. Robotics and Automation, ICRA, 2009.
[34] G. Heppner, Schunk_svh_driver, 2019, http://wiki.ros.org/schunk{_}svh{_}driver. Online (Accessed 9 April 2019).
[35] L. Peng, Z.-G. Hou, N. Kasabov, G.-B. Bian, L. Vladareanu, H. Yu, Feasibility of neucube spiking neural network architecture for EMG pattern recognition, in: Conf. Advanced Mechatronic Systems, ICAMechS, 2015.
[36] J. Kaiser, A. Friedrich, J.C.V. Tieck, D. Reichard, A. Roennau, E. Neftci, R. Dillmann, Embodied neuromorphic vision with event-driven random backpropagation, 2019, arXiv preprint arXiv:1904.04805.
[37] G. Haessig, X. Berthelon, S.-H. Ieng, R. Benosman, A spiking neural network model of depth from defocus for event-based neuromorphic vision, Sci. Rep. 9 (1) (2019).
[38] J. Kaiser, J. Weinland, P. Keller, L. Steffen, J.C.V. Tieck, D. Reichard, A. Roennau, J. Conradt, R. Dillmann, Microsaccades for neuromorphic stereo vision, in: Biomedical Robotics and Biomechatronics (BioRob), 2018 IEEE International Conference On, Springer, 2018, pp. 244–252.
[39] F. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, N.V. Thakor, Towards the control of individual fingers of a prosthetic hand using surface EMG signals, in: EMBC, 2007.



Juan Camilo Vasquez Tieck is a research scientist at FZI Research Center for Computer Science and a Ph.D. student on the field of neurorobotics at (KIT) with Prof. Rüdiger Dillmann in Karlsruhe, Germany. He works on motion representation for manipulation and grasping with spiking neural networks and works in the SP10 Neurorobotics of the Human Brain Project. He graduated as Mechanical Engineer and Computer Scientist at the EAFIT University in Medellin. He worked designing, developing and building robots for protein christalization at Formulatrix Inc. in Boston. After that, he worked as solution developer in the fields of biomechanics and telecommunications at Ilimitada S.A in Medellin. And later, he worked in research on interactive digital television at Artica in Medellin. He finished his MSc degree in Informatics with focus on cognitive systems and humanoid robots at the Karlsruhe Institute of technology (KIT).

Sandro Weber is a Ph.D. student at the chair for Augmented Reality at TUM. Key interests are HCI with a focus on ubiquitous adaptive interaction systems in multi-device environments and virtual reality re-embodiment. Working for the HBP on the Neurorobotics Platform. Developing virtual reality re-embodiment systems with the goal to integrate users as human-like avatars in a physical simulation.

Terrence C. Stewart is a post-doctoral research associate working with Chris Eliasmith at the Centre for Theoretical Neuroscience at the Universit of Waterloo. My initial training was as an engineer (B.A.Sc. in Systems Design Engineering, University of Waterloo, 1999), my masters involved applying experimental psychology on simulated robots (M.Phil. in Computer Science and Artificial Intelligence, University of Sussex, 2000), and my Ph.D. was on cognitive modeling (Ph.D. in Cognitive Science, Carleton University, 2007). He is also a co-founder of Applied Brain Research, a research-based start-up company based around using low-power hardware (neuromorphic computer chips) and adaptive neural algorithms.

Jacques Kaiser graduated with an international Master degree in computer graphics, vision and robotics, Jacques Kaiser is now working as a robotics engineer in FZI and simultaneously pursuing a Ph.D. within the Human Brain Project. His research focuses on embodying synaptic plasticity rules in real world neurorobotic setups. He is particularly interested in visuomotor tasks and learning from event-based vision.

Arne Roennau is the department manager of the Interactive Diagnosis and Service Systems department and head of the FZI Living Lab Service Robotics. He studied Electrical Engineering and Information Technology at the Karlsruhe Institute of Technology, specialized in the fields of feedback control, automation and robotics. Since 2011, he is department manager and head of the FZI Living Lab Service Robotics. In this role, he is leading more than more than 10 national and 4 European research projects. His main fields of research are robot motion control, human robot collaboration (HRC), and the design of innovative service robots.

Rüdiger Dillmann received his Ph.D. from University of Karlsruhe in 1980. Since 1987 he has been Professor of the Department of Computer Science and is Director of the Research Lab. Humanoids and Intelligence Systems at KIT. 2002 he became director of an innovation lab. at the Research Center for Information Science (FZI), Karlsruhe. Since 2009 he is spokesman of the Institute of Anthropomatics and Robotics at the Karlsruhe Institute of Technology. His research interest is in the areas of service robotics with special emphasis on intelligent, autonomous and interactive robot behavior based on machine learning methods and programming by demonstration (PbD). Other research interests include machine vision for mobile systems, manmachine interaction, computer supported intervention in surgery and related simulation techniques. He is author/co-author of more than 950 scientific publications, conference papers, several books and book contributions. He is Editor of the journal"Robotics and Autonomous Systems", Elsevier, and Editor in Chief of the book series COSMOS, Springer. He is IEEE Fellow.