In: Proc. of the IEEE-RAS International Conference on Humanoid Robots (Humanoids 14), pp. 83, Madrid, Spain (2014)

Real-time Gesture Recognition Using a Humanoid Robot with a Deep Neural Architecture

Pablo Barros, German I. Parisi, Doreen Jirak and Stefan Wermter

Abstract— **Dynamic gesture recognition is one of the most interesting and challenging areas of Human-Robot-Interaction (HRI). Problems like image segmentation, temporal and spatial feature extraction and real-time recognition are the most prominent issues to name in this context. This work proposes a deep neural model to recognize dynamic gestures with minimal image preprocessing and real-time recognition in an experimental set up using a humanoid robot. We conduct two experiments with command gestures in an offline fashion and for demonstration in a Human-Robot-Interaction (HRI) scenario with the robot giving audio feedback for the user. Our results show that the proposed model achieves high classification rates of the gestures executed by different subjects, who perform them with varying speed. We demonstrate that our system performs in real-time.**

I. Introduction

There is a trend in the robotic community towards integration of robots into assistive systems in everyday life situations and different areas, e.g. in medical surgery or helping systems for elderly people [1].

In the context of humanoid robots, the demand for intuitive communication interfaces for natural Human-Robot-Interaction (HRI) is increasing. The most significant aspects for HRI comprise the correct understanding and processing of the user input, robustness concerning varying environmental situations and realtime recognition performance.

In the area of gesture recognition several different approaches have been established using different input devices and learning algorithms. Among them, development of neurally inspired algorithms is of special interest as they offer a model for brain-like stimuli processing, which allows adaption to a changing environment, is robust against noise and provides instantaneous reactions. Before performing gestures ourselves, we first perceive them as visual stimuli over time,which motivates us using neural computational models, in particular Convolutional Neural Networks (CNN).

CNN has been introduced as a computational approach which mirrors the hierarchical stages for visual processing in the visual cortex. An incoming visual percept is processed in a simple cell layer extracting edge features. The outcome of this extraction is then passed to complex cells in the next layer, where these features are pooled together. These computations alternate between the cell types until the neurons show high coding specificity and are invariant to scale, rotation and noise. This approach has been successfully applied to object recognition where only single images were used for classification. A prominent example is the cipher classification introduced in [2]. A similar approach called HMAX [3] performs template matching and pooling operations from Gabor filtering at different scales and with different orientations. It showed very good recognition rates on benchmark object databases, which underlines the crucial impact of implementing a cascading visual system to obtain invariant and insensitive features.

Deep learning architectures are also prominently applied in different areas of HRI. For autonomous robotic behaviour the work presented in [4] demonstrated higher recognition rates using a max-pooling CNN (MPCNN) compared to other learning methods like SVM. The idea was to provide a swarm of mobile robots with reliable vision information derived from counting gestures which are performed using a coloured glove. Although the authors could show superior performance of their architecture in terms of 96% recognition rate and real-time processing they did not model the temporal dimension.

To bridge the gap, a more recent approach for action recognition [5] extends CNN with 3D kernels capturing motion information along the frames of an action performance stream. Instead of using single images for convolution, the whole computation is performed on a frame cube of predefined size, i.e. frames to consider in the video. The feature maps are created using different kernels to increase the diversity of features. Moreover, the author described extraction of high-level motion information by defining auxiliary features presented to the architecture as bag-of-words and added to the output layer. The system was evaluated with an airport surveillance database that contains three action types and showed superior results compared to other models like spatial pyramidal matching features. Also for the KTH benchmark DB for action recognition, e.g. boxing and hand clapping, the results were competitive to [6] and superior to an unsupervised probabilistic learning approach on extracted time-space points of interest [7].

As our work focuses on dynamic gestures, we also have to deal with different timescales and long-range dependencies to capture the gestures' semantics. The different demands on a gesture recognition system combined in a model are described in [8], based on data captured with a Kinect depth sensor. The stream is separated into depth information employed as input to a four-layer CNN for each hand and into a skeleton part for the whole body motion. The first stage is referred to as dynamic pose and catches only local information. The output of the different computational steps is then aggregated and serves as input to a Recurrent Neural

The authors are with University of Hamburg - Department of Computer Science, Vogt-Koelln-Strasse 30, 22527 Hamburg - Germany.
`barros,parisi,jirak,wermter @informatik.uni-hamburg.de`

Network. They evaluated their system on the ChaLearn 2013 data set for fused audio and vision recognition and could compete well with other methods, resulting in rank 6 of 20 final presentations.

Although the presented work showed promising results, we point out some critical issues: standard methods for gesture recognition is to use specifically coloured objects like a red ball to be tracked or as in [4] to use gloves. This, however, introduces an additional preprocessing step of reduce the input to a particular colour or learning a specific colour distribution. We also recognized that in literature only a few papers deal with CNN in the domain of dynamic gestures. As mentioned above, this is due to the fact that CNN were introduced rather for single image processing and classification. The extension of this approach as in [5] for action recognition motivates us to this more neurobiological approach for commanding gestures comprising motions.

Therefore, we extend the idea of CNNs in two ways: first, we enhance the computational capacity of the model by introducing 3D kernels suitable for noise-affected images as is the case when captured in a stream in ambient living situations and sensor limitations. Second, we extend CNN to the temporal domain by adding the capability to generate and learn motion representations. In our paper, we focus on gestural communication with the Nimbro robot platform.

Therefore, we defined 5 command gestures: 'Circle', 'Point Left', 'Point Right', 'Stop' and 'Turn Around', and 'Stand Still' as our reference standing position. In our scenario, we also embed a simple speech response toprovide user feedback.

The paper structure is as follows: In the first section, we introduce our deep neural architecture and processing stages in the network. We then explain our experimental methodology in section 2 and present our results derived from the experiments in the following section. Finally, we provide the reader with a discussion part and some future work suggestions.

II. Deep temporal and spatial feature extraction learning framework

In order to extract the temporal and spatial features of a gesture sequence, we use a deep neural network architecture. This architecture is able to create a representation of motion and uses a series of deep layers to identify and extract the features that represent the changes during the gesture execution as this underlies subject variability. The architecture is divided into two steps: Motion representation and motion feature extraction. Both steps work together and are part of the proposed architecture, illustrated in Figure 1.

A. Motion representation

The first layer of the architecture is responsible to create the motion representation. This layer receives N gray scale frames, without the application of any preprocessing step, and creates a representation based on the difference of each pair of frames. The layer works in a sequential way, receiving the frames one after the other. After receiving a pair of frames, this layer computes an absolute difference and sums up the resulting frame to a stack of frames. This operation is represented by M:

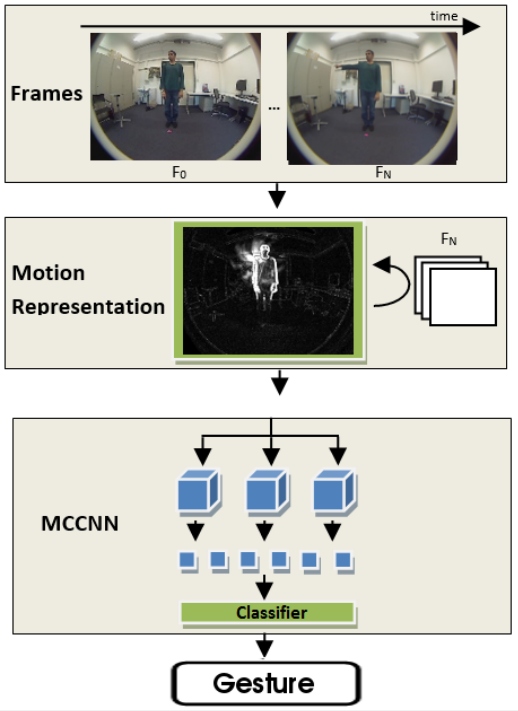

Fig. 1: Proposed deep neural architecture used to recognize dynamic gestures. The first layer receives a number of frames and generates a motion representation. An MCCNN implementation is used to learn and extract the features from the motion representation and use them to classify the gesture.

$$M = \sum_{i=1}^{N} |(F_{i-1} - F_i)| Ws, \tag{1}$$

where N is the number of frames, F_i represents the current frame and Ws is the weighted shadow. The absolute difference of each pair of frames removes irrelevant parts of the gesture execution, being able to extract the background or any other detail in the image that is not part of the gesture motion. By summing up the results it is possible to create a shape representation of the motion, imprinting it in a motion representation image. The weighted shadow is used to create different gray scale shadows in the final representation according to the time that each frame is presented. The weighted shadow is defined as $Ws = i/t$, where t is the memory size. The memory size defines how many frames will be important in the gesture execution. For example, a "Stop" gesture will be faster than a "Turn Around" gesture, meaning that it will have a smaller memory size. An adjustment of the memory size based on each gesture execution makes the motion representation robust against gestures with varying amounts of frames. The utilization of weighted shadows to create the motion representation

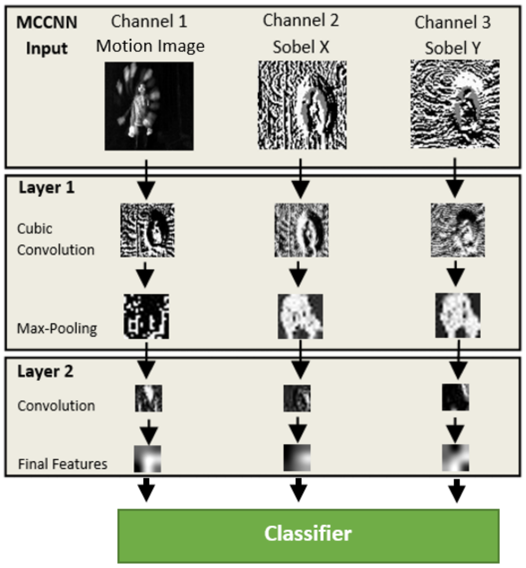

Fig. 2: Illustration of the output of each pair of convolution and max-pooling operations. This example uses a two layers MCCNN. Using a motion representation as input, the MC-CNN applies a Sobel operator in both directions, horizontal (X) and vertical (Y). At the end a representation containing 3 feature sets, each one with 3x3 pixels is generated.

is responsible for creating a time feature imprinted in the final motion image. This representation contains the shape of the motion and, with the help of the weighted shadows, the information of when each single posture happened.

B. Spatial and time features fusion

The second step is responsible for identifying and extracting relevant features in the motion representation. To achieve this, the previous layer is attached to a Multi Channel Convolutional Neural Network (MCCNN) [9]. The MCCNN receives an image containing the motion representation and applies a series of convolutional and max-pooling operations that, at the end, will generate a feature vector. In the proposed model an MCCNN with three channels is implemented. Each channel receives a different version of the motion image. The first one receives the original motion image. The second and third receive the resulting image after applying a Sobel operator in both directions, horizontal(S_x) and vertical(S_y).The Sobel operators encode our prior knowledge on features, and our experiments showed a improve in the model performance. The idea behind the multichannel implementation is to use the different information provided by each channel input in different ways. As each channel has its own filter maps and weights, it is possible to learn which features are more important for the final feature set in each channel. The Sobel operator uses two different 3x3 kernel filters, defined as:

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, S_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \quad (2)$$

The utilization of the Sobel filters helps in the discrimination of the motion shape in the image. When applied in both directions these filters can detail the shape aspects of the motion. The channel that receives the original image is responsible to extract the time structure created by the weighted shadows.

To create a feature set invariant to position and scale, a cubic kernel implementation is used. In the CNN implementation the convolution layers are applied as 2D filter maps, each one with independent weights, to be robust against noise and small changes in the pixels intensities [2]. The cubic kernel allows the application of a 3D filter in a stack of images. Each filter map has 3D kernels, still with independent weights, that are applied to a sequence of images of the same motion representation. In the cubic implementation, the value of each unit(x,y,z) at the nth filter map in the cth layer is defined as:

$$v_{nc}^{xyz} = tanh(b_{cn} + \sum_{m} \sum_{h=0}^{H_i-1} \sum_{w=0}^{W_i-1} \sum_{r=0}^{R_i-1} w_{i(c-1)m}^{hwr} v_{(m-1)}^{(x+h)(y+w)(z+r)}) \quad (3)$$

where tanh is the hyperbolic tangent function, b_{cn} is the bias for the nth filter map of the cth layer, m indexes over the set of feature maps in the (c-1) layer connected to the current layer c. In the equation, w_{ijm}^{hwr} is the weight of the connection between the unit (h,w,r) within a region, or kernel, connected to the previous layer ($c-1$). H_i and W_i are the height and width of the kernel and z indexes the image in the image stack, R_i is the amount of pictures stacked together representing the new dimension of the kernel.

To produce invariant features using the cubic kernel, the image stack is created with different examples of the same class. This way, the cubic kernel receives a stack of different motion images, performed with the user in different positions and distance to the camera.

In the MCCNN implementation each convolutional layer is followed directly by a max-pooling layer. Each max-pooling layer compresses the data from the convolutional layer in smaller images. Each filter map is divided into regions, and each region is used as input for a unit in the max-pooling layer. This operation enhances invariance to scale and distortion of the input [10].

To train the model, the MCCNN is attached to a hidden layer and then connected to a logistic regression classifier. The result of the classifier is used to calculate the error for the weights update using the backpropagation algorithm. This means that all the filters in the channels are updated individually, but the final error is calculated as a whole. Using this strategy, the filters in each channel can be trained to have different roles in the feature extraction but must be synchronized with the final feature representation.

After the training, the channel receiving the original image will be mostly responsible for identifying the different pixel intensities in the image, being able to differentiate between the weighted shadows imprinted in the motion representation

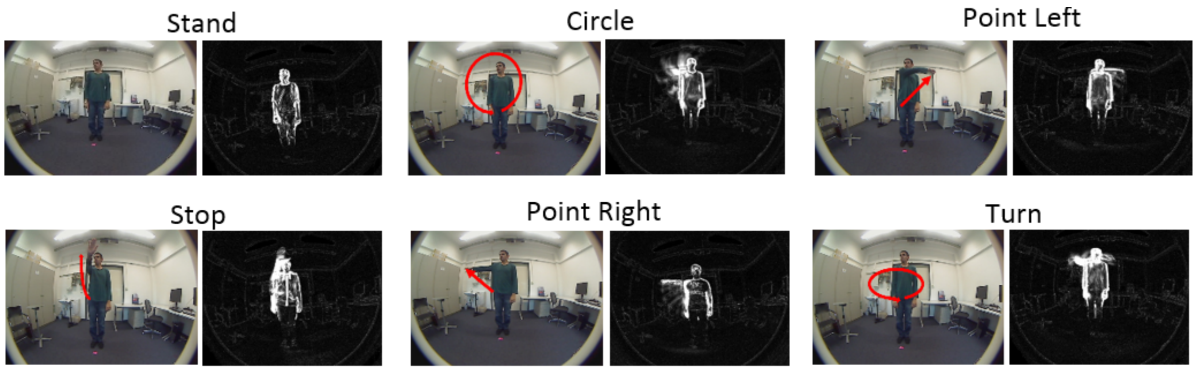

Fig. 3: Illustration of the five commands and standing position in the recorded dataset. Following each sequence it is an example of the motion image generated by the proposed model.

image. The two channels with the Sobel filters will be strongly trained to extract the motion shape patterns, with highlighted edges in both directions, horizontal and vertical. After the filters are trained, the hidden layer and logistic regression can be detached of the model and the filters can be used to extract features for other classifiers. Figure 2 illustrates the whole process of feature extraction after the model is trained. In the example, the model has 2 layers, each one composed of a pair of a convolutional and a max-pooling layer. At the end, three feature sets, each one with 3x3 pixels, are used to represent the full gesture.

III. Experiments

We set up two experiments to evaluate our model. One, aimed to evaluate the recognition rate, training time and recognition time for the proposed model, was called the offline experiment. The other one applied the trained filters in a real-time recognition scenario with a humanoid robot called the Nimbro experiment.

For the offline experiment, a data set containing six classes was recorded. The classes represent five gesture commands and a standing position with no gesture execution. The commands are as follows: 'Circle', 'Point Left', 'Point Right', 'Stop', and 'Turn'. The data set contains 60 examples for each gesture, executed by one subject. Each frame has a resolution of 640x480 pixels and each gesture sequence has varying amounts of frames. In sum, 360 videos were recorded, and illustrations of the gestures are shown in Figure 3.

The learning was executed with the dataset separated into 60% of the gestures for training and 40% for testing. The frames were resized to 100x100 pixels and all the frames were used to compose the motion representation. The MCCNN was implemented with a depth of two layers and connected to the hidden layer and logistic regression classifier for training the filters. We performed experiments with different parameter values, following the indications of [11], and the best performing parameters are shown in Table I.

The experiment was performed 30 times and the mean and standard deviation of the F-Score, recognition and training times were collected and are shown in the next section. In each execution, the data for either training or testing is randomly chosen. We implemented the system using Python and Theano[1], running on a machine with an Intel Core I5 processor and 8GB of RAM memory.

TABLE I: Parameters of the MCCNN for training session.

Parameters	Layer 1	Layer 2
Filters	20	50
Kernel size	5x5x5	4x4
Sub sampling size	5x5	5x5
Neurons hidden layer	500	
Learning rate	0.01	

For the second experiment we applied our framework to an HRI scenario with the Nimbro humanoid robot. In this scenario, the robot is positioned in a room so that it captures the performing subject frontal to the RGB camera. The robot gives voice feedback for each recognized gesture in real-time. The Nimbro is 95cm tall and has a weight of 6.6kg including the battery [12]. It uses a Zotac Zbox Nano XS PC, running a Linux Ubuntu distribution. This PC has a 1.65 GHz Dual-Core AMD E-50 processor and has 2GB of RAM. It contains a Logitech C905 USB camera with customized wide-angle lens, that produces images with a resolution of 640x480 pixels.

The model was deployed in the robot, after training the filters using the recorded dataset, and used for real-time recognition. To provide continuous classification, every frame was collected and sent to the MCCNN. The experiments with the robot were conducted with 3 different subjects, each one executing 10 times each gesture in a random order. None of the subjects executed the gestures in the recorded dataset. The mean of the F-Score is shown in the next session. Figure 4 illustrates the scenario for this experiment.

IV. Gesture Recognition System Interface

To be able to give spoken feedback, a communication system was developed and deployed in the robot. The system

[1]http://deeplearning.net/software/theano/

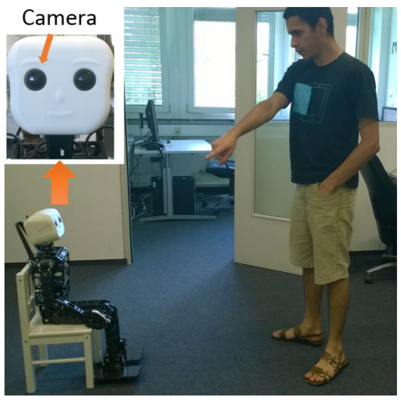

Fig. 4: Scenario for the second experiment, using the Nimbro robot. The Robot is positioned in front of the human and recognizes the gestures continuously. After each recognition, the Nimbro gives a spoken feedback with the recognized gesture.

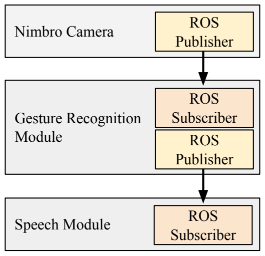

Fig. 5: Diagram of system interface over ROS network with Publisher-Subscriber nodes.

is composed of two main modules: one module for recognizing the learned gestures from visual input and the other for speaking the recognized gestures with Nimbro. To interface the different modules and devices of our architecture we use the Robot Operating System (ROS).

For our interface implementation, we rely on a synchronous RPC-style communication over a ROS network implemented with publisher-subscriber nodes. The publisher node will continually broadcast a message. We broadcast a message over the network using a message-adapted class. The subscriber node receives the messages on a given topic via a master node, which keeps a registry of who is publishing and who subscribing. This architecture represents a robust interface to connect different applications, e.g. written in different programming languages, over a common network of communication.

A diagram of our system interface over the ROS network is illustrated in Figure 5. The gesture recognition module receives input, i.e. raw RGB images, from the Nimbro camera over the ROS network. Results of the recognition, i.e. gesture labels, are published for the speech module.

V. Results

The offline experiment used the recorded dataset to train the model and to perform a classification of the gestures. The results showed that the mean F-Score for all the gestures was 96.85%. In Figure 6 we show that Point Left and Standing produced a lower F-Score compared to the others, due to the fact that the motion images generated from both are very similar. Even with the visual similarity, the model was still able to identify and classify them with an elevated F-Score.

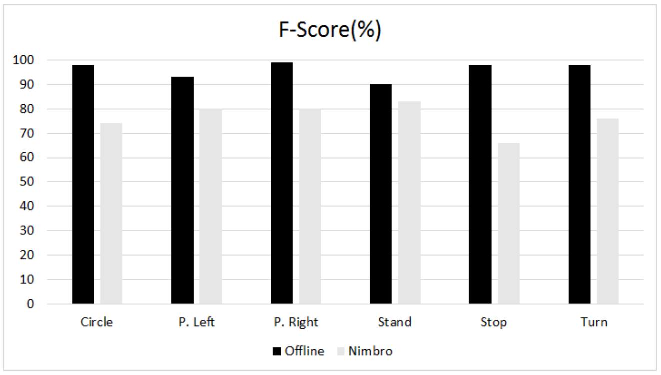

Fig. 6: F-Score obtained in both experiments. Nimbro experiment presented lower results, but still demonstrate the capability of the model in recognizing dynamic gestures in real-time using the humanoid robot.

One of the problems of a deep neural architecture is the necessary time to train the model. One of the advantages of the MCCNN is to be able to use smaller images and still be able to classify the presented patterns. In our experiments, each training routine, with 100 epochs, took 165.60 minutes to complete, with a standard deviation of 0.027 after 30 executions. The recognition time was 0.039 seconds, with a standard deviation of 0.0022. With such a smaller recognition time, it is possible to use the proposed model in a real-time classification scenario.

In the Nimbro experiment, the model was trained using the recorded dataset, in the same way as for the first experiment. After the training, it was deployed in the Nimbro and used to classify the input obtained by its camera. Three different persons executed the gestures, none of them present in the previously recorded dataset. The F-Scores for this experiment are also shown in Figure 6.

Figure 7 shows the confusion matrix obtained for the Nimbro experiment. In the confusion matrix we see that most of the misclassification are related to the ”Stand” position. This happens because most of the gestures contain the ”Stand” position in the beginning and ending of their execution. If the gesture is executed too fast or too slow, the ”Stand” position will still be captured and will be highlighted by the motion representation layer.

When using the live classification with Nimbro the results are lower compared to the offline experiment, but still have a significant F-score value when applied to a real-world scenario. The experiment shows that the model was able to recognize gestures with different persons, that were not present in the training set, and in real-time.

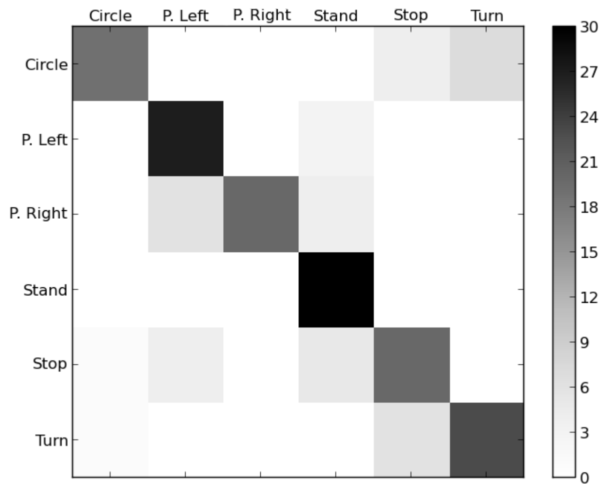

Fig. 7: Confusion matrix obtained by Nimbro experiment, using the Nimbro‘s camera to recognize the gestures in a real-time recognition scenario.

VI. Conclusion and Future Work

We presented a neural framework extended from CNNs for the recognition of commanding gestures in a HRI-scenario. To test our approach we set up two experiments and extended the recognition with audio feedback for the user. Opposed to [4] our framework needs no additional input devices nor specific colour definitions, which is beneficial for a natural user interface. Our work shows that the proposed model is suited to be used in a real-world scenario. After the training, the recognition time is small and the spatial-temporal features are extracted and learned. Although we conducted rather initial experiments, we could show promising results for gesture classification. We detected very good accuracy for the ’circle’, ’point right’, ‘stop‘ and the ’turn’ gesture, but lower accuracy for ’point left’ and ’stand’, our baseline gesture. This is due to the fact, that our system relies on motion information and has only minor focus for hand shape. The ’point left’ gesture executed with the same hand as the ’point right’ carries more motion information as the arm crosses the body. For the ’point right’ on the contrary the arm is raised up almost equal to the ’stop gesture’. The only difference is the orientation of the hand. So, we suggest to also provide a shape- and orientation representation for such low-motion gestures as can be found in the dichotomized streams in the visual cortex. Another interesting point worth mentioning is that in our live experiments with the Nimbro we could show that our command vocabulary is recognized in real-time indifferent to a specific user, which puts emphasis on the generalization capabilities of the MCCNN. In line with that our approach also allows us to cope with intra- and inter-subject variability in gesture performance, as our subjects performed each gesture several times and had no information about the execution speed. No time warping mechanisms or similar are necessary as is standard in variable-length sequence processing, thus avoids additional preprocessing. Therefore, we would like to use our extended gesture database with gestures performed with one- and two hands from multiple subjects. Until now, our setting is restricted to one-hand gestures and simplified illumination conditions. As the model generates the motion features using raw images, light conditions may affect the final results and our system is limited to one moving person in the scene. Thus, we are working on a flexible mechanism for changes in lighting and additionally extend the system for multi-person scenarios. In line with the neural architecture proposed so far we are going to look into unsupervised methods for substituting the labelling of training data. This will further increase the autonomous nature of learning, and thus robotic behaviour.

Acknowledgement

This work was partially supported by CAPES Brazilian Federal Agency for the Support and Evaluation of Graduate Education (p.n.5951-13-5), the DAAD German Academic Exchange Service (Kz:A/13/94748) - CASY Project (Cognitive Assistive Systems), and by the DFG German Research Foundation (grant #1247) - International Research Training Group CINACS (Cross-modal Interaction in Natural and Artificial Cognitive Systems).

References

[1] G. Parisi and S. Wermter, “Hierarchical som-based detection of novel behavior for 3d human tracking,” in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, Aug 2013, pp. 1–8.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[3] T. Serre, A. Oliva, and T. Poggio, “A feedforward architecture accounts for rapid categorization,” *Proceedings of the National Academy of Sciences*, vol. 104, no. 15, pp. 6424–6429, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1073/pnas.0700622104

[4] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella, “Max-pooling convolutional neural networks for vision-based hand gesture recognition,” in *Proc. IEEE Int Signal and Image Processing Applications (ICSIPA) Conf*, 2011, pp. 342–347.

[5] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, Jan 2013.

[6] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, “A biologically inspired system for action recognition.” in *ICCV*. IEEE, 2007, pp. 1–8. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2007.htmlJhuangSWP07

[7] J. C. Niebles, H. Wang, and L. Fei-Fei, “Unsupervised learning of human action categories using spatial-temporal words,” *Int. J. Comput. Vision*, vol. 79, no. 3, pp. 299–318, Sept. 2008. [Online]. Available: http://dx.doi.org/10.1007/s11263-007-0122-4

[8] N. Neverova, C. Wolf, G. Paci, G. Sommavilla, G. W. Taylor, and F. Nebout, “A multi-scale approach to gesture detection and recognition,” in *ICCV Workshop on Understanding Human Activities: Context and Interactions (HACI 2013)*, Dec. 2013, pp. 484–491. [Online]. Available: http://liris.cnrs.fr/publis/?id=6330

[9] P. Barros, S. Magg, C. Weber, and S. Wermter, “A multichannel convolutional neural network for hand posture recognition,” in *Proceedings of the 24th international conference on Artificial neural networks - To apear*, ser. ICANN’14, 2014.

[10] D. C. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” *CoRR*, vol. abs/1202.2745, 2012.

[11] P. Simard, D. Steinkraus, and J. C. Platt, “Best practices for convolutional neural networks applied to visual document analysis,” in *International Conference on Document Analysis and Recognition*, Aug 2003, pp. 958–963.

[12] M. Schwarz, J. Pastrana, M. Schreiber, M. Missura, and S. Behnke, “Humanoid teensize open platform nimbro-op.”