# SIMULATION-BASED MODEL CONTROL USING STATIC HAND GESTURES IN MATLAB

*S. Kajan, J. Goga*

Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic

**Abstract**

**This paper deals with the domain of simulation-based models control using static hand gestures in the MATLAB environment. The aim of this paper was to design an algorithm for visual static hand gesture recognition with high classification accuracy. For this recognition task, different convolutional neural network models (CNN) were tested. For the successful training of CNN, stochastic backpropagation of error was used. Training of CNN was implemented on the graphic card using toolboxes such as Neural Network and Parallel Computing from the MATLAB program package. For the training and testing of CNN a database of 35 static hand gestures was used. The proposed CNN gesture recognition system has been implemented in the simulation scheme due to the need of setting different model parameters.**

## 1 Principle of Gesture Recognition

The hand gesture recognition itself can be implemented in a several consecutive steps. General scheme of hand gesture recognition system is shown in Fig. 1. In this recognition task, the *Kinect v2* sensor was used.

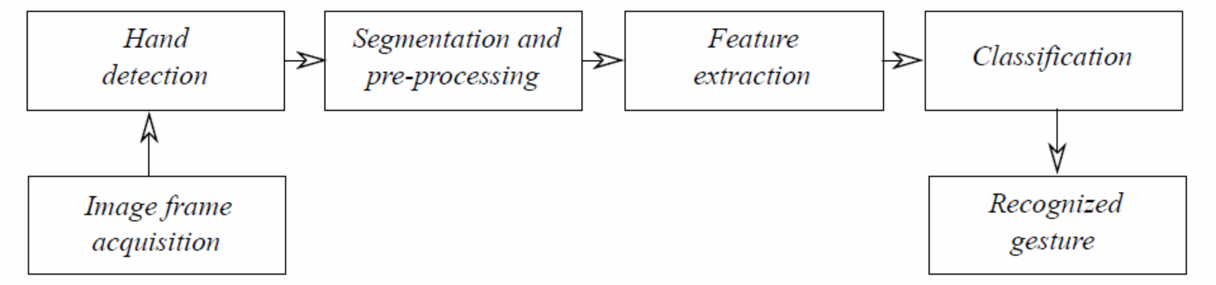

Figure 1: General scheme of gesture recognition system

The captured image from the sensors is processed and stored in appropriate form. Subsequently, the gesture in an acquired frame may be segmented or image features may be extracted from the entire input frame [1]. Under the extraction of features, we understand the evaluation of quantitative or statistical indicators which represent given gesture based on the suitable metrics. It can be the number of stretched fingers, angles between fingers, fingertip markings, fingertips positions [3], histograms, Voronoi diagrams, and other statistical and quantitative indicators. These extracted features are an input to a computational model, whose job is to correctly classify the given gesture. With regard to the complexity of this task, in this paper we dealt only with the recognition of static hand gestures.
Hand gesture recognition and associated problems such as hand segmentation were elaborated in many papers. Some authors have used a color-based image analysis approach such as a color histogram based on statistical methods [3], thresholding the tints of a color model [4], or gesture capturing with color gloves [5] that are easier to segment. Approaches based on depth analysis are in general more successful than color-based methods, but those approaches assume the hand is the closest object in the frame. One possible solution is mapping from the depth data to a corresponding part of a color image [3], or hand segmentation based on the distance limited by color bracelet [6].

Use of the latest deep neural network models does not always improve classification accuracy, which ranges from 70% - 90%, depending on the specific architecture of the neural network [2][8 -10]. In this paper, we used different architectures of convolutional neural networks for this challenging static hand gesture recognition task.

## 2 Gesture Recognition Using Convolutional Neural Network

The general structure of the convolutional neural network (CNN) is displayed in Fig. 2.

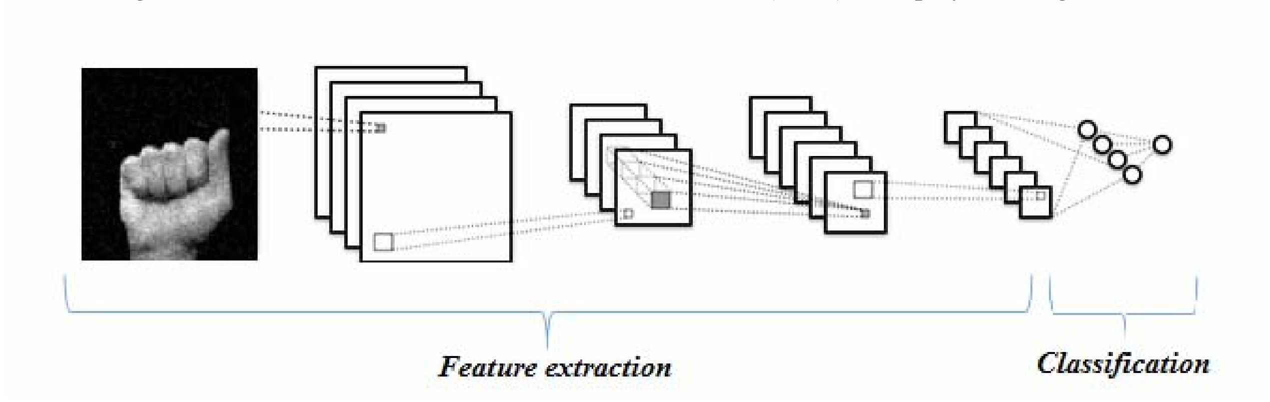

Figure 2: The general architecture of convolutional neural network used for gesture recognition

Convolutional neural networks are designed specifically for pattern recognition with a large degree of invariance to shift, change of scale or other forms of distortion. These properties are gained through learning process. However, the structure of convolutional neural networks involves certain forms of constraints. The most commonly used types of layers in the network architecture are *convolutional*, *pooling*, and *fully-connected* layers. By arranging these computational layers, we create the overall architecture of the convolutional neural network.

The convolutional layer is the main computing block in the overall network architecture. Its input is usually 3-dimensional image tensor, which contains 3 color image channels. As the title of this layer suggests, a discrete convolution of input with the kernel is performed there. When computing, we move the kernel in the direction of the width and height of the input image with the selected step, for all their mutual positions, creating a feature map. By learning, these feature maps are activated when different image patterns are detected, such as edges at a certain angle, color clusters, and others.

The pooling layer performs sub-sampling of the input tensor, thereby reducing the size of the feature map, but retaining the most important information contained therein. This greatly reduces the spatial magnitude of the feature map, as well as the number of parameters and the computational difficulty of the neural network.

The neurons in the fully-connected layer of the convolutional network have, as the name suggests, all connections to the neurons in the previous layer. This is, therefore, a classical multilayer perceptron network. Outputs from convolutional and pooling layers represent high-level features extracted from input images. These features are an input into the fully-connected layer of the convolutional network, and its role is to correctly classify them.

## 3 Training and Testing of the Convolutional Neural Network

For the training and testing of CNN a database of 35 static hand gestures was used. This database contains static gestures of the American Sign Language (ASL), which was changed due to the dynamic characters "J" and "Z". The database was created by 65 volunteers (60 men and 5 women), which consists of 5 frames per gesture [6]. Overall, we created 175 frames (35 gestures times 5 frames) for color, infrared and depth images. A total of 525 images per person. The resulting database has 34 125 images (65 people times 525 frames). Data from 50 people was used in the training process and data from 15 people was used in the testing process.

These images were then modified into a form suitable for training the convolutional neural network. In the original color image from the Kinect sensor, with a resolution of 1920x1080 pixels, we only segmented the hand area. In this way, we created a square image suitable for training with a resolution of 640x640 pixels. Segmentation of hand gestures was also done for original depth and infrared frames. The original resolution of 512x424 pixels was adjusted to a gesture frame with a size of 156x156 pixels.

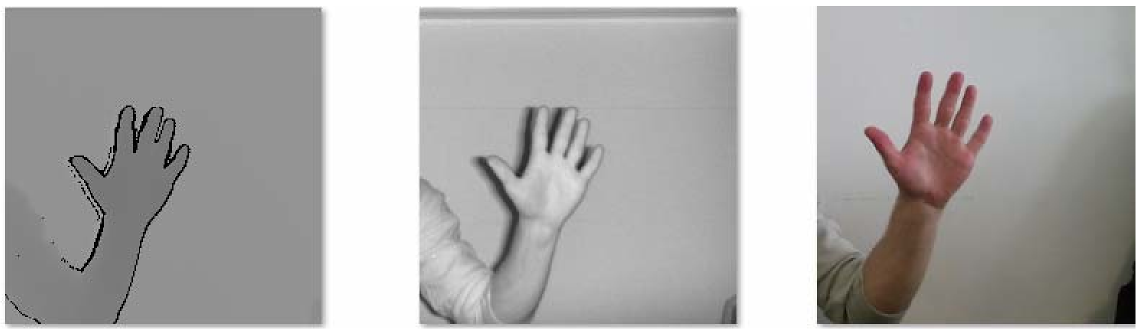
Figure 3: A preview of frames in the database

The first tested architecture of CNN (Table 1) consist of three convolutional layers. The second tested architecture (Table 2) has the same number of convolutional layers, but fully-connected layer with 50% dropout was added. This omission is used to make the neural network better distribute trained information throughout the network as any neuron may be omitted in the next epoch. The last tested architecture (Table 3) has been extended to four convolutional layers, where kernel size is the same in all layers.

Table 1: FIRST ARCHITECTURE OF CNN – A-1

| *No.* | *Layer type* | *Parameters* |
|---|---|---|
| 1 | Image Input | 156x156x3 images |
| 2 | Convolution | 3x3@18 kernels with stride [1 1] |
| 3 | ReLU | - |
| 4 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 5 | Convolution | 6x6@36 kernels with stride [1 1] |
| 6 | ReLU | - |
| 7 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 8 | Convolution | 9x9@72 kernels with stride [1 1] |
| 9 | ReLU | - |
| 10 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 11 | Fully-Connected | 35 neurons, fully-connected layer |
| 12 | Softmax | - |
| 13 | Classification Output | 35 classes, cross-entropy error |

Table 2: SECOND ARCHITECTURE OF CNN – A-2

| *No.* | *Layer type* | *Parameters* |
|---|---|---|
| 1 | Image Input | 156x156x3 images |
| 2 | Convolution | 5x5@18 kernels with stride [1 1] |
| 3 | ReLU | - |
| 4 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 5 | Convolution | 6x6@36 kernels with stride [1 1] |
| 6 | ReLU | - |
| 7 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 8 | Convolution | 9x9@72 kernels with stride [1 1] |
| 9 | ReLU | - |
| 10 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 11 | Fully-Connected | 35 neurons, fully-connected layer |
| 12 | ReLU | - |
| 13 | Dropout | 50% dropout |
| 14 | Fully-Connected | 35 neurons, fully-connected layer |
| 15 | Softmax | - |
| 16 | Classification Output | 35 classes, cross-entropy error |

Table 3: LAST ARCHITECTURE OF CNN – A-3

| *No.* | *Layer type* | *Parameters* |
|---|---|---|
| 1 | Image Input | 156x156x3 images |
| 2 | Convolution | 5x5@18 kernels with stride [1 1] |
| 3 | ReLU | - |
| 4 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 5 | Convolution | 5x5@36 kernels with stride [1 1] |
| 6 | ReLU | - |
| 7 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 8 | Convolution | 5x5@72 kernels with stride [1 1] |
| 9 | ReLU | - |
| 10 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 11 | Convolution | 5x5@72 kernels with stride [1 1] |
| 12 | ReLU | - |
| 13 | Max Pooling | 2x2 max pooling with stride [2 2] |
| 14 | Fully-Connected | 128 neurons, fully-connected layer |
| 15 | ReLU | - |
| 16 | Dropout | 50% dropout |
| 17 | Fully-Connected | 35 neurons, fully-connected layer |
| 18 | Softmax | - |
| 19 | Classification Output | 35 classes, cross-entropy error |

Table 4: COMPARISON OF DIFFERENT CNN ARCHITECTURES – CLASSIFICATION SCORE (ACCURACY)

| *Architecture* | *Accuracy [%]* | |
|---|---|---|
| | *Train* | *Test* |
| A-1 | 100.00 | 72.14 |
| A-2 | 90.63 | 84.43 |
| A-3 | 98.44 | 81.14 |

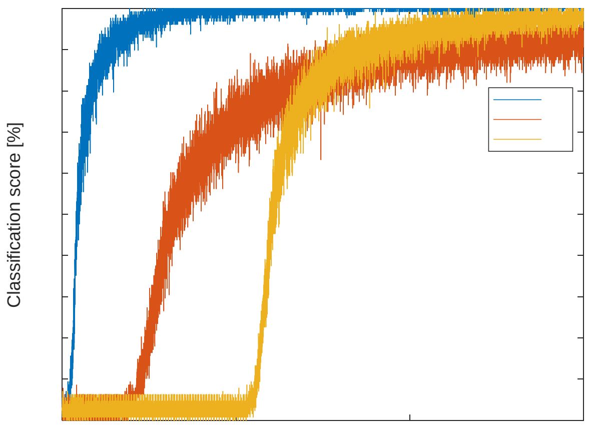

Figure 4: Classification score during the training process for different CNN architectures

For the successful training of CNN, stochastic backpropagation of error was used [13]. Training of CNN was implemented on the graphic card using toolboxes such as *Neural Network* and *Parallel Computing* from the MATLAB program package. In Figure 4 is shown the comparison of classification score during the training process for all tested neural network architectures. The comparison of classification accuracy of those architectures is displayed in Table 4. Examples of learned features acquired by transition of the random input color image through individual convolutional layers of trained CNN with architecture A-2 is shown in Figure 5 [6].

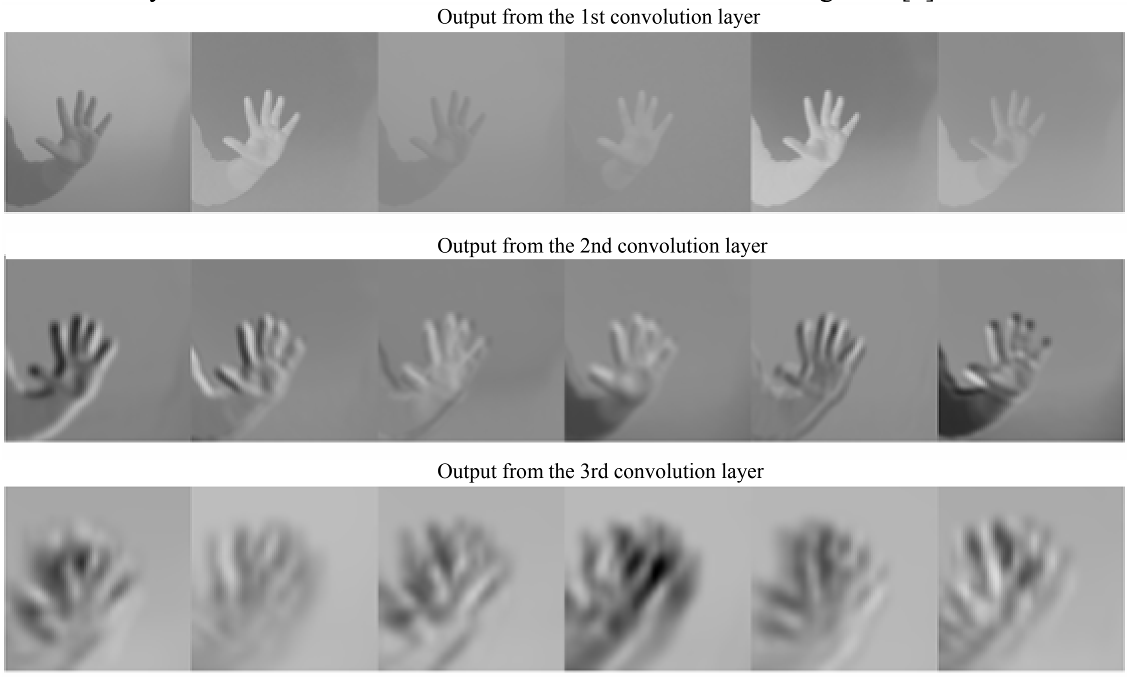
Output from the 1st convolution layer
Output from the 2nd convolution layer
Output from the 3rd convolution layer

Figure 5: 6 examples of learned features by convolutional layers of trained CNN with architecture A-2

## 4 Simulation-based Model Control Using Static Hand Gesture

### 4.1 Example of Hand Gesture Recognition Using Kinect SDK

The Kinect functions are accessed using the *From Video Device* block in the Simulink graphical simulation environment. This block allows you to obtain images from an RGB or depth camera along with the tracking metadata. For the hand gesture recognition, we used the metadata *HandRightState* property, which identifies a recognized right-hand gesture. In figure 6 are displayed hand gestures, which can be recognized by Kinect SDK. In figure 7 is shown simulation scheme for PID control loop, in which setpoint is changed based on a right-hand gesture recognized by the Kinect sensor [6].

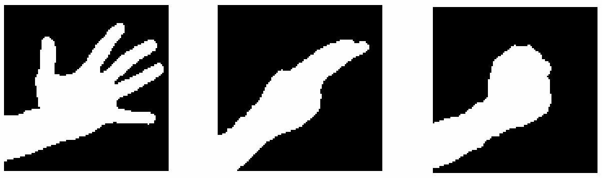

Figure 6: Standard hand gestures of Kinect SDK

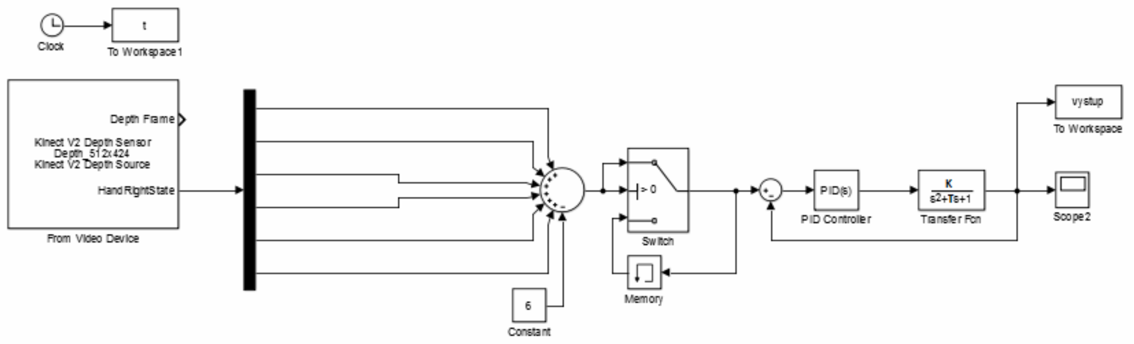

Figure 7: Simulation scheme for a PID control loop (setpoint is changed based on a right-hand gesture recognized by the Kinect sensor)

## 4.2 Example of Hand Gesture Recognition Using CNN

A trained CNN with A-2 architecture was used to control the simulation model. To demonstrate hand gesture recognition, a simulation model of four independent water tanks with PI controllers was created [6]. We designed the entire control system scheme according to the *Model-View-Controller* (MVC) software architecture. MVC divided the program into 3 independent units (*Simulation Model*, *Visualization*, *Controller*) with minimal interconnections.

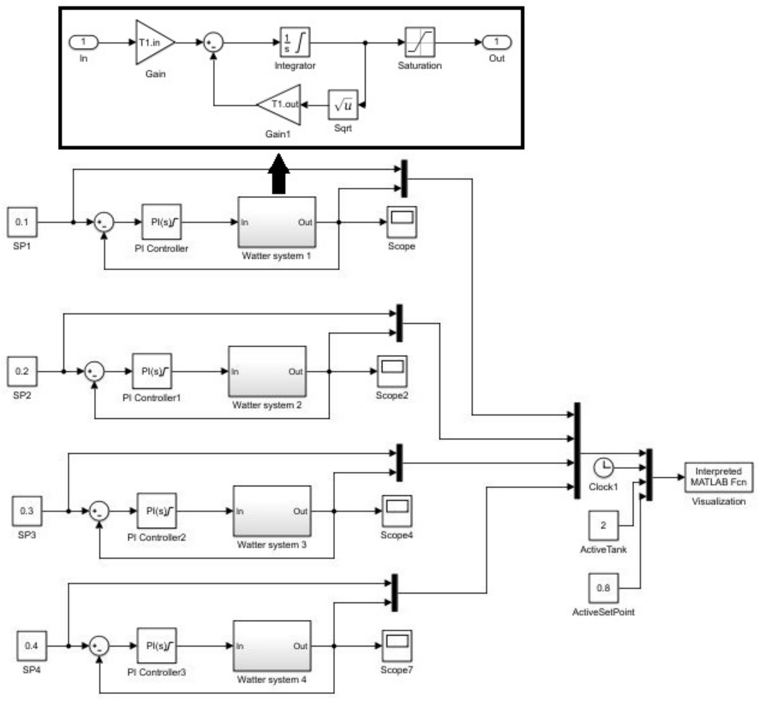

Figure 8: Simulation scheme of water tanks system with PI controllers

The simulation model is represented by a simulation scheme of four independent water tanks (Fig. 8). In the simulation scheme are four independent control loops with PI controllers for level control in tanks. Inputs to the simulation model are the desired level values (*SetPoints*) in the tanks.

For a better presentation of the results achieved and the current state of simulation, a simple visualization was created (Fig. 9). Simulation model is connected to the water tank visualization (Fig. 8), to which it sends the desired level (*SP*) and current level values in the individual tanks. Additionally, active tank number (*ActiveTank*), new desired value (*ActiveSetPoint*) and simulation time are sent to the visualization subroutine.

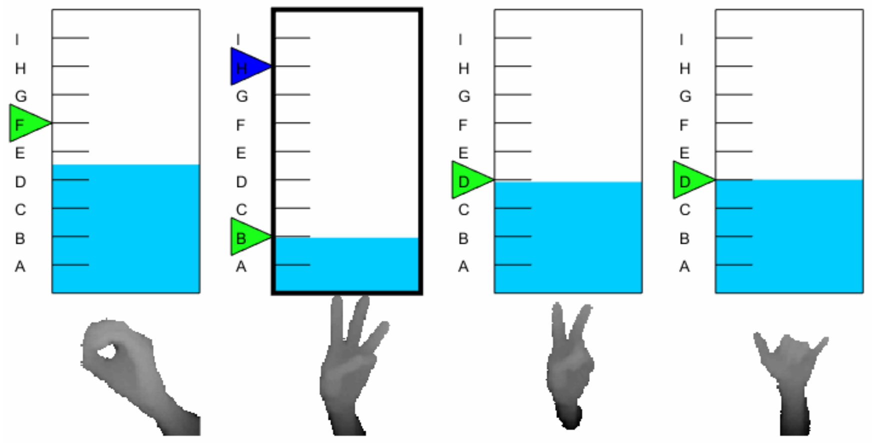

Figure 9: Visualization of the water tanks

The last part of the MVC is a controller, that is linked to all MVC, Kinect sensor, and the neural network. Controller is running independently from the simulation. In the cycle, images and metadata from the Kinect sensor are collected and the simulation-based model is controlled. Both hands are used in the control process. The left hand is used to control parameter changes. Actions such as start and stop of right hand recognition, confirmation, and cancellation of set points are controlled by left-hand gesture. The right-hand gesture controls the desired level value of the selected tank. In the first phase of the simulation, the gesture symbolizing a particular tank is expected (Fig. 9). When the gesture is recognized, the algorithm comes into the second phase - adjusting the desired level on the selected tank. Setpoint value can by controlled by ASL alphabet gestures from A to I.

The main subject of the final simulation test was to verify accuracy of the gesture recognition. We have created a test script with 14 gestures. 3 people participated on the testing phase, and everyone repeated the test 3 times. The average gesture recognition accuracy was 92.86%.

## 5 Conclusion

Various architectures of CNN were tested in this paper. The CNN classification models showed a very good classification accuracy. The proposed CNN gesture recognition system has been implemented in the simulation scheme due to the need of setting different model parameters. After successful testing of different CNN architectures, we verified the suitability of their use in the static hand gesture recognition task such as simulation-based control.

## References

[1] P. K. Pisharady, M. Saerbeck. *Recent methods and databases in vision-based hand gesture recognition: A review.* Computer Vision and Image Understanding, 2015, 141: 152-165.

[2] A. Tang, K. Lu, Y. Wang, J. Huang and H. Li, *A realtime hand posture recognition system using deep neural networks.*, ACM Transactions on Intelligent Systems and Technology (TIST), 2015, 6(2):21

[3] M. Van den Bergh and L. Van Gool. *Combining rgb and tof cameras for real-time 3d hand gesture interaction.* Applications of Computer Vision (WACV), 2011 IEEE Workshop on. IEEE, 6–72.

[4] M. Fagiani, E. Principi, S. Squartini, and F. Piazza. *A new system for automatic recognition of italian sign language.* Neural Nets and Surroundings, 2013, 69–79.

[5] R. Y. Wang and J. Popović. *Real-time hand-tracking with a color glove*, ACM transactions on graphics (TOG), 2009, vol. 28, 63

[6] Z. Ren, J. Yuan, and Z. Zhang. *Robust hand gesture recognition based on fingerearth mover's distance with a commodity depth camera.* Proceedings of the 19th ACM international conference on Multimedia, 2011, ACM:1093–1096

[7] S. Kajan, D. Pernecký, A. Hamad. *Hand gesture recognition using multilayer perceptron network.* 23th Annual Conference Proceedings, Technical Computing Prague, 2015

[8] G. Strezoski, D. Stojanovski, I. Dimitrovski, and G. Madjarov. *Hand gesture recognition using deep convolutional neural networks.*

[9] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. *Hand gesture recognition with 3D convolutional neural networks,* 2015, Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 1–7.

[10] P. Barros, S. Magg, C. Weber, and S. Wermter. *A multichannel convolutional neural network for hand posture recognition.* 2014, International Conference on Artificial Neural Networks, 403–410.

[11] F. Špaldoň. *The control of simulation models using Kinect sensor.* Bachelor thesis FEI STU in Bratislava, 2017, (in Slovak)

[12] J. Goga, F. Špaldoň, S. Kajan, J. Pavlovičová, and M. Oravec. *Static hand gesture database of FEI STU Bratislava.* http://www.uim.elf.stuba. sk/kaivt/MLgroup, 2017.

[13] M. Beale, M. Hagan, H .Demuth. *Neural Network Toolbox, User's Guide*, 2017

## Acknowledgments

The research described in this paper was done within the project No. 1/0867/17 of the Slovak Grant Agency VEGA.

Ing. Slavomír Kajan, PhD.:
Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Ilkovičova 3, 812 19 Bratislava, E-mail: slavomir.kajan@stuba.sk

Ing. Jozef Goga:
Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Ilkovičova 3, 812 19 Bratislava, E-mail: jozef.goga@stuba.sk