Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Gesture detector with Leap Motion sensor

Anh Tran Viet

Department of Theoretical Computer Science
Supervisor: Tomáš Nováček

December 11, 2020

Acknowledgements

THANKS (remove entirely in case you do not with to thank anyone)

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on December 11, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Tran, Anh Viet. *Gesture detector with Leap Motion sensor.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Contents

List of Figures

Introduction

Mouse and keyboard are considered to be default devices for human-computer interaction nowadays. But with the maturity in technology, namely virtual and extended reality, the computer's need to understand human's body language is more and more present. Actions such as rotation or grabbing and moving an object in three-dimensional space with a computer mouse are unintuitive. They require a little understanding of the controls to execute the task. The movement is limited to the two-dimensional space of the mouse. Oppose to performing the desired action by hands in our three-dimensional space as we would in real life.

One of the proposed solutions for the issue is gesture recognition, where a general idea is for computers to have the ability to recognize gestures and perform actions based on them. Therefore, several devices were developed to process an image and yield useful data for gesture recognition.

Chapter 1

Neural Networks

An artificial neural network (ANN) is a mathematical model mimicking biological neural networks, namely their ability to learn and correct errors from previous experience.[1][2]

The ANN subject was first introduced by Warren McCulloch and Walter Pitts in "A logical calculus of the ideas immanent in nervous activity" published in 1943.[3] But it was not until recent years when ANN has gained popularity with still increasing advancements in technology and availability of training data. ANN had become one of the default solutions for complex tasks which were previously thought be unsolvable by computers.[4]

This chapter will briefly explore different types of neural units and their activation functions, along with some exemplary network architectures.

1.1 Artificial Neuron

As previously mentioned, artificial neurons are units mimicking behavior of biological neurons. Meaning it can receive as well as pass information between themselves.

1.1.1 Perceptron

Perceptron is the simplest class of artificial neurons developed by Frank Rosenblatt in 1958.[5]

Perceptron takes several binary inputs, vector $\vec{x} = (x_1, x_2, ..., x_n)$, and outputs a single binary number. To express the importance of respected input edges, perceptron uses real numbers called weights, assigned to each edge, vector $\vec{w} = (w_1, w_2, ..., w_n)$.

A step function calculates the perceptron's output. The function output is either 0 or 1 determined by whether its weighted sum $\alpha = \sum_i x_i w_i$ is less

or greater than its threshold value, a real number, usually represented as an incoming edge with a negative weight -1.

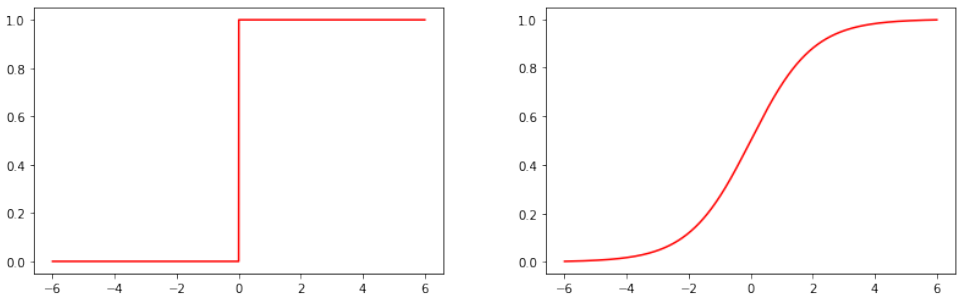
$$output = \begin{cases} 1, & \text{if } \alpha \geq threshold \\ 0, & \text{if } \alpha < threshold \end{cases} \tag{1.1}$$

PICTURE PERCEPTRON

1.1.2 Sigmoid Neuron

Sigmoid neuron, similarly to perceptron, has inputs $\vec{x}$ and weights. The key difference comes in once we inspect the output value and its calculation. Instead of perceptron's binary output 0 or 1, a sigmoid neuron outputs a real number between 0 and 1 using a sigmoid function.[6][7]

$$\sigma(\alpha) = \frac{1}{1 + e^{-\alpha}} \tag{1.2}$$



(a) Step function

(b) Sigmoid function

Figure 1.1: Comparison between step function and sigmoid function

As shown in Figure 1.1, the sigmoid function(1.1a) is a smoothed-out version of the step function(1.1b).

1.1.3 Activation Function

An artificial neuron's activation function defines that neuron's output value for given inputs, commonly being $f : \mathbb{R} \to \mathbb{R}$ [8]. A significant trait of many activation functions is their differentiability, allowing them to be used for Backpropagation, ANN algorithm for training weights. The activation function needs to have a derivative that does not saturate, heads towards 0, or explore, heads towards inf.

For such reasons, the usage of step function or any linear function is unsuitable for ANN.

1.1.3.1 Sigmoid Function

The sigmoid function is commonly used in ANN as an alternative to the step function. A popular choice of the sigmoid function is a logistic sigmoid. Its output value is in the range of 0 and 1.

$$\sigma(\alpha) = \frac{1}{1 + e^{-\alpha}} = \frac{e^x}{1 + e^x} \tag{1.3}$$

One of the reasons being the simplicity of derivative calculation:

$$\frac{d}{dx}\sigma(\alpha) = \frac{e^x}{(1 + e^x)^2} = \sigma(x)(1 - \sigma(x)) \tag{1.4}$$

One of its disadvantages being the vanishing gradient. A problem where for a given very high or very low input values, there would be almost no change in its prediction. Possibly resulting in training complications or performance issues.[9]

1.1.3.2 Hyperbolic Tangent

Hyperbolic tangent is similar to logistic sigmoid function with a key difference in its output, ranging between -1 and 1.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.5}$$

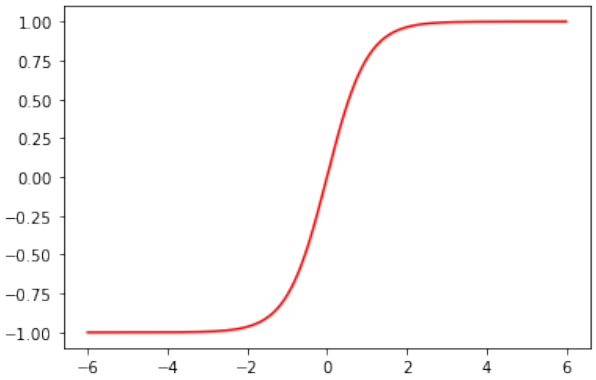


Figure 1.2: Hyperbolic tangent

It shares sigmoid's simple calculation of its derivative.

$$\frac{d}{dx}tanh(x) = 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - tanh^2(x) \tag{1.6}$$

By being only moved and scaled version of the sigmoid function, hyperbolic tangent does share sigmoid's advantages and its disadvantages.[8]

1.1.3.3 Rectified Linear Unit

The output of the rectified linear unit (ReLU) is defined as:

$$f(x) = max(0, x) \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{1.7}$$

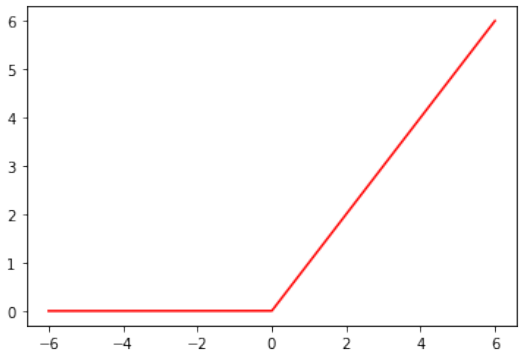


Figure 1.3: Rectified Linear Unit

ReLU popularity is mainly for its computational efficiency.[9] ReLu's disadvantages appear when inputs approach zero or are negative. Causing the so-called dying ReLu problem, where the network is unable to learn. There are many variations of ReLu to this date, e.g., Leaky ReLU, Parametric ReLU, ELU, ...

1.1.3.4 Softmax

Softmax separates itself from all the previously mentioned functions by its ability to handle multiple input values in the form of a vector $\vec{x} = (x_1, x_2, ..., x_n)$ and output for each x_i defined as:

$$\sigma(x_i) = \frac{e_i^x}{\sum_{j=1}^{n} e_j^x} \tag{1.8}$$

For output being normalized probability distribution, ensuring $\sum_i \sigma(x_i) = 1$.[10] It is being used as the last activation function of ANN to normalize the network's output into n probability groups.

1.2 Types of Neural Networks

To this day, there are many types and variations of ANN, each with its structure and use cases. Here we will briefly introduce the most common ones, such as feed-forward networks, convolutional neural networks, or recurrent neural networks.

1.2.1 Feed-forward Networks

Feed-forward network (FNN) was the first ANN to be invented and the simplest form of ANN. Its name comes from how the information flows through the network. Its data travels in one direction, oriented from the input layer to the output layer, without cycles.[11]

FNN may or may not contain several hidden layers of various widths. By having no back-loops, FNN generally minimizes error in its prediction by using the backpropagation algorithm to update its weight values.[12]

GRAPH

The input layer takes input data, vector $\vec{x}$, producing $\hat{y}$ at the output layer. The process of training weights consists of minimizing the loss function $\mathcal{L}(\hat{y}, y)$, y being the target output of input $\vec{x}$.[10]

1.2.1.1 Cost Function

Cost function $C(\vec{w})$ is used in ANN's training process. It takes all weights and biases of an ANN as its input, in the form of a vector $\vec{w}$ and calculates a single real number expressing ANN's incorrectness.[13] The number is high when the ANN performs poorly and gets lower when the ANN's output gets closer to the correct result. The main goal of training is then to minimize the cost function.

1.2.1.2 Backpropagation

Backpropagation, short of backward propagation of errors, is a widely used algorithm in training FFN using gradient descent to find a local minimum of a cost function and update ANN's weights.[14]

A gradient of a function with multiple variables gives us the direction of the steepest gradient ascent, where should we step to increase the output quickly and find the local maximum. Naturally, taking its negative will give the direction towards a local minimum.

The usual practice is to divide training samples into small batches of size n. We will calculate a gradient descent for each sample in the batch and use their average gradient descent to update the network's weights. The average gradient descent tells us which weights should be adjusted for the ANN to get closer to the correct results.[14]

$$-\gamma \nabla C(\vec{w}_i) + \vec{w}_i \to \vec{w}_{i+1} \tag{1.9}$$

Here, $\vec{w}_i$ are weights of the network at the current state (batch), $\vec{w_{i+1}}$ are updated weights, γ is the learning rate and $-\nabla C(\vec{w}_i)$ is the gradient descent.

1.2.2 Convolutional Neural Networks

Convolutional Neural networks (CNN) primary goal is to make a computer recognize images and objects. For such, it is primarily used for image classification or object recognition.

CNN was inspired by the biological processes of the human brain. Its connectivity patterns resemble the human's visual cortex. But an image is perceived differently by a human brain than by a computer. To a computer, an image is interpreted as an array of numbers. Thus CNN is designed to work with two-dimensional image arrays, although it is possible to work with one-dimensional or three-dimensional arrays too.[15]

CNN is a variation of FNN.[13]. It usually consists of the input layer followed by multiple hidden layers, typically several convolutional layers with standard pooling layers, and ending with the output layer.

1.2.2.1 Convolutional Layer

The convolutional layers' objective is to extract key features from the input image by passing a matrix known as a kernel over the input image abstracted into a matrix.[16]

IMAGE

The convolution result can be of two types depending on their size. One being the convolved feature is reduced in dimensions compared to the input, valid padding. For example, an input image of dimensions 8x8 being reduced to 6x6 after convolution operation. And the other type being where dimensions are either increased or remain the same, same padding. [17]

1.2.2.2 Pooling Layer

Similar to the previously mentioned convolutional layer, the pooling layer reduces the convolved feature's spatial size to decrease the computational power required for data processing. Furthermore, being useful by extracting dominant features, which are rotational and positional invariant, thus maintaining the process of effectively training the model.[17]

There are two types of pooling: max pooling and average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel. It performs as a noise suppressant, discarding the noisy activations altogether and performing de-noising and dimensionality reduction. Where average pooling returns the average of all the values from the same covered portion, performing dimensionality reduction as a noise suppressing mechanism. Hence, it is possible to note that max-pooling performs better.[17]

IMAGE

1.2.3 Recurrent Neural Networks

Recurrent Neural Network (RNN) is distinguished by its memory, taking input sequence with no predetermined size. Its past predictions influence currently generated output. Thus for the same input, RNN could produce different results depending on previous inputs in the sequence.[18].

RNNs features make it commonly used in fields such as speech recognition, image captioning, natural language processing, or language translation. Some of the popular being, for example, Siri, Google Translate or Google Voice search.[19]

As previously mentioned, RNN takes into consideration information from previous inputs. Let us look at the idiom "feeling under the weather", where for it to make sense, words have to be in a specific order. RNN needs to account for each word's positions and use its information to predict the next word in the sequence. Each time step represents a single word. In our case, the third timestep represents "the". Its hidden state holds information of previous inputs, "feeling" and "under".[19]

SCHEMATIC

Figure XX shows the network for each time step, i.e., at time t, the input $\vec{x_t}$ goes into the network to produce output $\hat{y}_t$, the next time step of the input is x_{t+1} with additional input from the previous time step from the hidden state h_t. This way, the neural network looks at the current input and has the context from the previous inputs. With this structure, recurrent units hold the past values, referred to as memory. Making it possible to work with a context in the data. [20]

The recurrent unit is calculated as follows:

$$h_t = f(W_x x_t + W_h h_{t-1} + \vec{b_h}) \tag{1.10}$$

$f()$ being the activation function, W_x, W_h are weight matrixes, x_t is the input, and $\vec{b_h}$ is the vector of bias parameters. Unit at time step $t = 0$ is initialized to $(0, 0, ..., 0)$. The output $\hat{y}_t$ is then calculated as:

$$\hat{y}_t = g(W_y h_t + \vec{b_y}) \tag{1.11}$$

$g()$ also being an activation function, usually being softmax to ensure the output is in the desired class range. W_y is the weight matrix and $\vec{b_y}$ being a vector of biases determined during the learning process.

Training RNNs uses a modified version of the backpropagation algorithm called backpropagation through time (BPTT), working by unrolling the RNN [13], calculating the losses across time steps, then updating the weights with the backpropagation algorithm. More on RNN in [10] by Liton et al.

Chapter 2

Gesture Recognition

Gestures are classified into static gestures and dynamic gestures. Group of static gestures consits of fixed gestures which are not relative to time, where group of dynamic gestures are time varying.

Hand and body gesture recognition had followed a conventional scheme of extracting key features via one or multiple preprocessing sensors and applying machine learning techniques on them.[21]

The field of gesture recognition gave birth to several image processing devices yielding useful data. One of them being Microsoft Kinect, a device where the main intention was to interpret whole-body movement, making it lacking in required accuracy for hand gesture recognition.

2.1 Leap Motion Controller

Another option would be using a Leap Motion Controller (LMC), developed specifically to track hand movements and extract its features, such as positions of fingers, hand rotation, and others.

LMC consists of two monochromatic IR cameras and three IR LEDs (emitters).

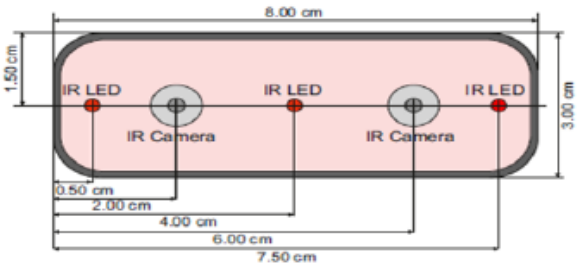


Figure 2.1: Schematic View of Leap Motion Controller

The LMC's current API, Leap Motion Service, yields positions of extracted hand features. All the positional data about the hand and its features are represented in the coordinate system relative to the LMC's center point, positioned at the middle IR LED.[22] The x- and z-axes lie in the camera sensors plane, with the x-axis running along the camera baseline. The y-axis is vertical, with positive values increasing upwards (in contrast to the downward orientation of most computer graphics coordinate systems). The z-axis has positive values increasing toward the user.[23]

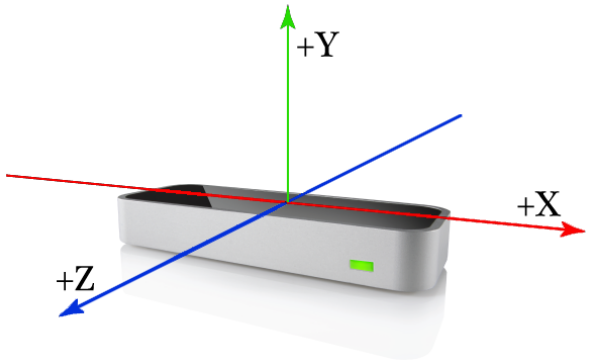


Figure 2.2: Leap Motion Controller Axes

Unfortunately, Leap Motion Controller has no official library for gesture recognition, limiting developers from utilizing the controller for its key features. Orion, Leap Motion tracking software build for virtual reality, used to have a gesture detector with its 3.0 version, but the detector is absent with the release of more accurate version 4.0.

2.2 Methods

Gestures classification should be taken into account when choosing appropriate methods due to their time-varying properties. As previously mentioned, gestures are classified into static and dynamic groups.

2.2.1 Static Gesture Recognition

Common methods for static gesture recognition are Support Vector Machines(SVM), ANN, or pattern techniques.[24]. Mapari and Kharat[25] proposed a method to recognize American Sign Language (ASL) by extracting data from LMC and computing 48 features (18 positional values, 15 distance values, and 15 angle values) for 4672 collected signs (146 users for 32 signs), eventually feeding them to an ANN using a Multilayer Perceptron (MLP).[26] Hasan et al.[27] proposed a method to recognize six sets of static gestures base on shape analysis using MLP. Filho et al.[28] compared the effectiveness between K-Nearest

Neighbors, SVM, and Decision Trees over a dataset of 1200 samples (6 uses for 10 gestures). They normalized positions of the five fingertips and the four angles between adjacent fingers as features to discover that the Decision Tree has performed the best.[26]

2.2.2 Dynamic Gesture Recognition

2.2.3 LSTM

Many of the proposed methods focus either on static gesture recognition or dynamic gesture recognition, but very few of them are actually utilized for both types at the same time.

Bibliography

[1] Chen, Y.-Y.; Lin, Y.-H.; et al. Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes. *Sensors*, 05 2019, doi:10.3390/s19092047.

[2] Bengio, Y.; Goodfellow, I.; et al. *Deep learning*, volume 1. Citeseer, 2017, ISBN 0262035618, 166–485 pp.

[3] McCulloch, W. S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, volume 5, no. 4, 1943: pp. 115–133, ISSN 0007-4985.

[4] Krishtopa. What Are Neural Networks, Why They Are So Popular And What Problems Can Solve. 2016. Available from: `https://steemit.com/academia/@krishtopa/what-are-neural-networks-why-they-are-so-popular-and-what-problems-can-solve`

[5] Rosenblatt, F. The Perceptron: A Probabilistic ModelFor Information Storage And Organization In The Brain. *Psychological Re-view*, 1958: p. 2047, doi:0.1037/h0042519.

[6] Nielsen, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.

[7] Rojas, R. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013, ISBN 9783642610684, 37–99 pp.

[8] Leskovec, J.; Rajaraman, A.; et al. *Mining of massive data sets*. Cambridge university press, 2020, ISBN 9781108476348, 523–569 pp.

[9] Maladkar, K. 6 Types of Artificial Neural Networks Currently Being Used in ML. Available from: `https://analyticsindiamag.com/6-`

types-of-artificial-neural-networks-currently-being-used-in-todays-technology/

[10] Lipton, Z. C.; Berkowitz, J.; et al. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015: pp. 5–25, ISSN 2331-8422. Available from: https://arxiv.org/pdf/1506.00019.pdf

[11] Feedforward Neural Networks. Available from: https://brilliant.org/wiki/feedforward-neural-networks/

[12] Team, T. A. Main Types of Neural Networks and its Applications-Tutorial. Aug 2020. Available from: https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e

[13] Goodfellow, I.; Bengio, Y.; et al. *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[14] Backpropagation. Available from: https://brilliant.org/wiki/backpropagation/

[15] How Do Convolutional Layers Work in Deep Learning Neural Networks? April 2020. Available from: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[16] Convolutional Neural Network. Available from: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[17] A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way. Dec 2018. Available from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[18] Recurrent Neural Networks. Jun 2019. Available from: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

[19] What are Recurrent Neural Networks? Available from: https://www.ibm.com/cloud/learn/recurrent-neural-networks

[20] Understanding Recurrent Neural Networks in 6 Minutes. Sep 2019. Available from: https://medium.com/x8-the-ai-community/understanding-recurrent-neural-networks-in-6-minutes-967ab51b94fe

[21] Avola, D.; Bernardi, M.; et al. Exploiting Recurrent Neural Networks and Leap Motion Controller for the Recognition of Sign Language and Semaphoric Hand Gestures. *IEEE Transactions on Multimedia*, volume 21, no. 1, 2019: pp. 234–245, doi:10.1109/TMM.2018.2856094.

[22] Weichert, F.; Bachmann, D.; et al. Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Sensors (Basel, Switzerland)*, volume 13, 05 2013: pp. 6380–6393, doi:10.3390/s130506380.

[23] Tomas Novacek, M. J., Christian Marty. Project MultiLeap: Fusing data from multiple Leap Motion sensors. *ACM Trans. Graph*, volume 37, 08 2020: pp. 1–5.

[24] Alexandre Savaris, A. v. W. A. Comparative evaluation of static gesture recognition techniques based on nearest neighbor, neural networks and support vector machines. *J Braz Comput Soc*, volume 16, 2010: p. 147–162, doi:10.1007/s13173-010-0009-z.

[25] Mapari, R. B.; Kharat, G. American Static Signs Recognition Using Leap Motion Sensor. 2016, doi:10.1145/2905055.2905125. Available from: `https://doi.org/10.1145/2905055.2905125`

[26] Lupinetti, K.; Ranieri, A.; et al. 3D Dynamic Hand Gestures Recognition Using the Leap Motion Sensor and Convolutional Neural Networks. 2020: pp. 420–439.

[27] Hasan, H. Static hand gesture recognition using neural networks. *Artificial Intelligence Review*, volume 41, 02 2014, doi:10.1007/s10462-011-9303-1.

[28] Stinghen, I.; Gatto, B. Gesture Recognition Using Leap Motion: A Machine Learning-based Controller Interface. 11 2018.

Appendix A

Acronyms

GUI Graphical user interface

XML Extensible markup language

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format