Measurement 94 (2016) 852–861

Contents lists available at ScienceDirect

Measurement

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

Gesture imitation and recognition using Kinect sensor and extreme learning machines

Emrehan Yavşan [a], Ayşegül Uçar [b,*]

[a] Mechatronics Engineering Department, Konya Necmettin Erbakan University, 42090 Konya, Turkey
[b] Mechatronics Engineering Department, Firat University, 23119 Elazig, Turkey

ARTICLE INFO

Article history:
Received 6 August 2015
Received in revised form 12 February 2016
Accepted 14 September 2016
Available online 15 September 2016

Keywords:
Human action recognition
NAO humanoid robot
Xbox 360 Kinect
Extreme learning machines

ABSTRACT

This study presents a framework that recognizes and imitates human upper-body motions in real time. The framework consists of two parts. In the first part, a transformation algorithm is applied to 3D human motion data captured by a Kinect. The data are then converted into the robot's joint angles by the algorithm. The human upper-body motions are successfully imitated by the NAO humanoid robot in real time.

In the second part, the human action recognition algorithm is implemented for upper-body gestures. A human action dataset is also created for the upper-body movements. Each action is performed 10 times by twenty-four users. The collected joint angles are divided into six action classes. Extreme Learning Machines (ELMs) are used to classify the human actions. Additionally, the Feed-Forward Neural Networks (FNNs) and K-Nearest Neighbor (K-NN) classifiers are used for comparison. According to the comparative results, ELMs produce a good human action recognition performance.

1. Introduction

In the recent past, robots have been used in factories for various jobs requiring speed, sensitivity, and power, but now they are involved in our daily lives. Besides emulating human behavior, robots are able to do almost everything we can do. Similarities between humans and humanoid robots also increase the cooperation between them. Human-Robot Interaction (HRI) and Human-Computer Interaction (HCI) have risen as research area, gaining attention in academics and industry [1,2]. They have many common areas such as computer science, mathematics, physiology, and bioinformatics [3]. Therefore, HRI studies target natural communication with the robots and recognizing human behavior. Humans want to interact easily and quickly with robots as they do with other people. Voice and text communications are widely used for HRI and HCI, but psychologists say that humans communicate through non-verbal cues about 60% more than through other methods [4,5]. Humans usually use motions or gestures in many cases, like pointing at objects and while speaking. Hence, many researchers are currently working on motions for interacting with robots [6–10]. The aim is to construct natural and intuitive interaction with minor training in real time. According to this perspective, using the techniques of artificial intelligence, recognizing and tracking human actions are required to improving human-robot cooperation. This topic is important for the disabled, older adults, children, and people needing rehabilitation.

Conventional vision-based action or gesture recognition methods cannot recognize and accurately imitate motions because the images captured by optical sensors are sensitive to lighting conditions, shadows, occlusions, and cluttered backgrounds [11,12]. The wearable sensors capturing the motions are used for more robust gesture recognition and imitation [13], as they are more reliable and less insensitive to lighting conditions and cluttered backgrounds. However, the users have to wear sensors and have to make calibrations. Moreover, they are usually more expensive than optical sensors, i.e., cameras. At this point, the 3D Kinect camera has become the preferred tool to get rid of these disadvantages in many recent applications [14–18].

This paper introduces a HRI system allowing the user to communicate with a humanoid robot using nonverbal cues. The Xbox 360 Kinect sensor is used for teaching an NAO humanoid robot human actions by using the artificial intelligence techniques. NAO is a robot that was produced by Aldebaran Robotics. It can perform various movements with 25 degrees of freedom. The human actions data collected from the Kinect are processed and transferred to the NAO robot. The human upper-body actions are then imitated in real time by the NAO robot. In addition, the

* Corresponding author.
E-mail addresses: eyavsan@konya.edu.tr (E. Yavşan), agulucar@firat.edu.tr (A. Uçar).

http://dx.doi.org/10.1016/j.measurement.2016.09.026
0263-2241/

upper-body action recognition algorithm is produced by Extreme Learning Machine (ELM) classifiers [19–24].

Action recognition is quite an old topic, but it is still an open problem to create the best classification algorithm [25,26]. In the literature, many researchers studied how to improve recognition performance by using the conventional classifiers like Feed-Forward Neural Networks (FNNs) [27–29], K-Nearest Neighbor (K-NN) [28], Support Vector Machines (SVMs) [30], and Convolutional Neural Networks (CNNs) [31]. Of these techniques, ELM is the most preferred classifier for the recognition and tracking in terms of learning, performance, implementation, and human intervention [32–35]. ELM is a type of FNNs. The FNNs are capable of approximating a nonlinear function by nonlinear mappings using input samples. The parameters of FNNs are iteratively determined by gradient-based learning algorithms. The FNNs have a very slow learning speed and need a number of iterative learning steps in order to obtain better learning accuracy. In the ELM, the weights of hidden nodes and biases are randomly chosen and the output weights are analytically determined [19–21]. In this study, to eliminate these disadvantages for human action recognition, the ELM proposed by Huang et al. [19–21] is used to classify human upper-body actions defined by joint angles. Additionally, to evaluate the effectiveness of the preferred recognition approach, ELM is compared with FNN and K-NN-based classification approaches.

In many applications, the NAO robot and Kinect were employed together to find the medical and social requirements of a person with mobility difficulties, older adults, or children [36–43]. In [36], an inverse Kinematics model was used on the learning stage by demonstration of NAO for teleoperation. A similar inverse Kinematics model was introduced to control the upper-body of NAO by using the Kinect in [37]. In [38], an NAO controlled by Kinect sensor was used to cure patients subject to physical treatments. A learning method was introduced for the upper-body actions of children with hearing disabilities by using the NAO robot and the Kinect in [39]. In [40], the NAO robot imitated whole-body motions of human by a motion capture system consisting of inertial sensors attached to the body. In [41], simple mathematical techniques were presented for NAO to mimic a person in real time. In [42], a system was presented to control the NAO robot by preferring the body structure comprised of different Kinect joint points, unlike [36], for supporting physiotherapy. In [43], it was realized that new capabilities could be transmitted to the NAO robot by using the Kinect. This paper relates to the referenced papers through the imitation of humanoid robots.

This study proposes a new HRI system based on ELM. Different from the other studies, the contribution of this paper is twofold. Firstly, an easy system for imitating human upper-body motions in real time without any learning process is proposed. Secondly, the study develops an ELM classifier as an efficient classification method in the human action recognition area. So far, both the real time imitation and the motion recognition have not been applied to the NAO robot.

The rest of this paper is organized as follows: in Section 2, the proposed system is introduced; the motion transfer algorithm based on computing the robot's joint angles is described in Section 3; K-NN, FNN, and ELM are shortly reviewed in Section 4; in Section 5, the comparatively experimental results are given; this paper is concluded in Section 6.

2. Structure of the proposed system

Fig. 1 shows the proposed system architecture. The system is composed of a computer, an Xbox 360 Kinect, and the NAO humanoid robot. The Kinect sensor is used to detect human motions. Users stand in the front of Kinect sensor and perform various motions. The Kinect sensor simultaneously detects and saves the points relating to the skeleton of users in all motions. The human skeleton data gathered by the Kinect are sent to a computer through its USB interface. The skeleton data are not directly provided to the NAO robot, however NAO can't imitate the human motions by using the data because the sizes of the human and the robot are quite different from each other and the Cartesian coordinates obtained from the Kinect can't be directly transformed into NAO's coordinate space. Hence, the data are first analyzed and then converted into the control command for each joint of NAO in the computer and are sent to the remote NAO over Wireless Fidelity (Wi-Fi) or a conventional network. Thus, the NAO robot simultaneously, i.e., in real time, imitates users' motions without requiring an iterative learning stage, but rather with a little computational effort and time.

In order to perform the action imitation in the proposed system, we use a Kinect RGB sensor. Several applications have been employed with the Kinect, many of them in the robotics field, but the Kinect was originally designed for the Xbox 360 game console by Microsoft. This sensor has an RGB camera, 3D depth sensor on the front and multiple microphone arrays at the sides. The sensor also consists of a motorized tilt in the range of $\pm 27^{\circ}$. The algorithm in the Kinect detects the joints of the human in the sensor field of view and represents them as a position (x,y,z) of a 3D space. It can measure the distance from an object 1.2–3 m away in the order of 1 cm accuracy. It is also cheaper than other RGB-D sensors. The Kinect is used with a powerful computer and the Microsoft Kinect SDK. C# is the programming language for using the SDK. It gives skeletal information with 20 joint-points for each person.

NAO is a humanoid robot developed by the French company Aldebaran Robotics in 2006 [45]. It is an ideal candidate humanoid robot for an HRI task due to its human-like appearance. Fig. 2 shows the size and a picture of the robot. The NAO robot has an AMD Geode processor at 550 MHz. In addition, it includes equipment such as 45-min life battery, Wi-Fi and Ethernet, speakers, LEDs in the eyes and ears, infrared emitters, sonars, tactile sensors, force sensing resistors, two cameras, gyroscopes, and accelerometers. The NAO robot runs with the NaoQi operating system, which allows for easy programming through Choregraphe software using C++, Python, Java, MATLAB, Urbi, C#, and .Net. The NaoQi software contains basic tools such as joint control, walking, speaking, and face tracking.

3. Processing of motion data and motion transfer algorithm

Considering that the body parts consist of a combination of two different joint points at the Kinect, the Cartesian coordinates are defined by means of the terminal points of the body parts. Each body part defined between Kinect joint points are evaluated as a position vector that start and end are previously known coordinates in the 3D space. Fig. 3 shows the definition of the human body part.

The position vectors defined by the Kinect joint positions cannot be directly transferred to the NAO humanoid robot. They have to be transferred to the joint angles of the NAO as shown in Fig. 4. In this paper, a transformation algorithm is used to obtain the arm joint angles of the NAO from the Kinect joint positions [42].

The vectors relating to each human body part generated on the Kinect are transformed into suitable ones for the NAO robot by the following equation:

$$\vec{r}_{s,f_{NAO}} = A \cdot (\vec{r}_{f_{Kinect}} - \vec{r}_{s_{Kinect}}) = A \cdot (\vec{r}_{s,f_{Kinect}}),$$
$$s,f \in \{0,1,2,\ldots,19\} \quad (1)$$

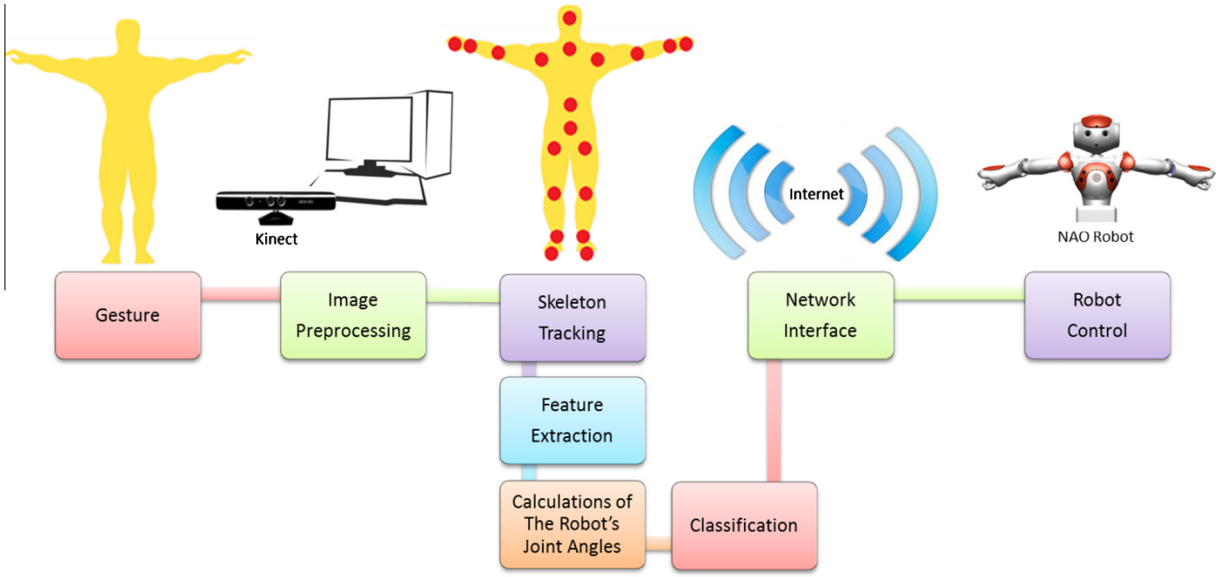


Fig. 1. The proposed system architecture.

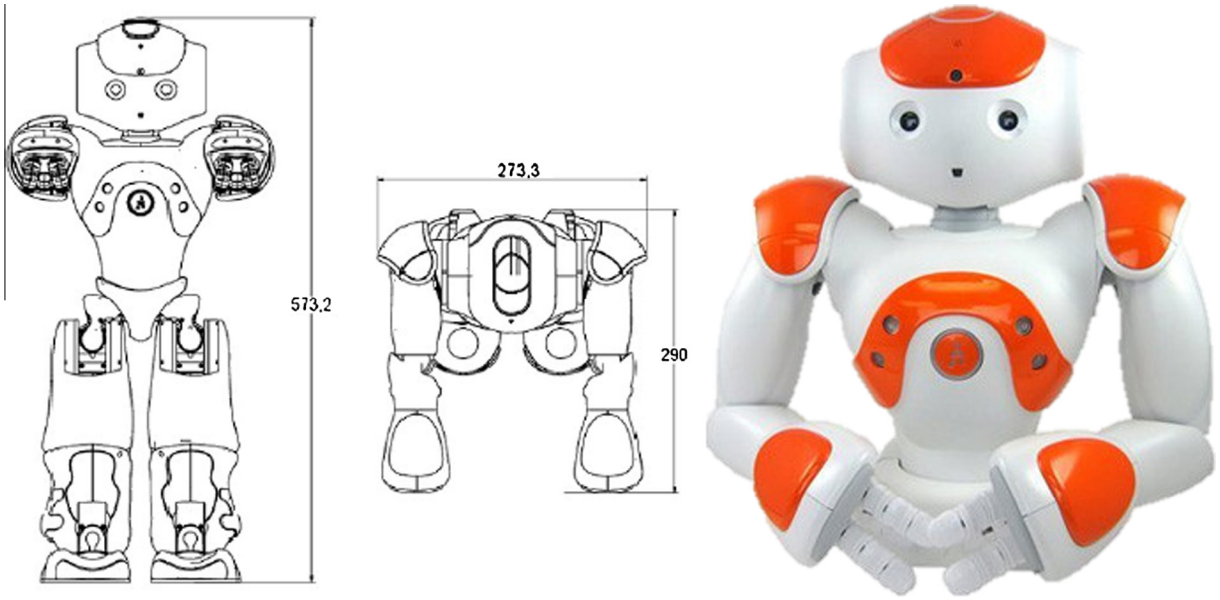


Fig. 2. NAO robot dimensions [45] (left). Torso NAO robot used in this paper (right).

where $\vec{r}_{f_{Kinect}}$ and $\vec{r}_{s_{Kinect}}$ are final and starting points of vector $\vec{r}_{sf_{Kinect}}$ in the Kinect coordinate system, respectively. A is the transformation matrix and $\vec{r}_{sf_{NAO}}$ is the vector in the coordinate system of the NAO robot. The skeleton stream consists of position data belonging to the Kinect joints and the skeleton model is expressed by the triangle of $\Delta\vec{r}_2\vec{r}_{12}\vec{r}_{16}$ in Fig. 4 [42,44,45]. Because the z-axis of the skeleton model is orthogonal to the defined triangle in Fig. 4a, it can be calculated by the vector product as follows:

$$\vec{z} = \frac{(\vec{r}_{16_{Kinect}} - \vec{r}_{12_{Kinect}}) \times (\vec{r}_{16_{Kinect}} - \vec{r}_{2_{Kinect}})}{|(\vec{r}_{16_{Kinect}} - \vec{r}_{12_{Kinect}}) \times (\vec{r}_{16_{Kinect}} - \vec{r}_{2_{Kinect}})|} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}. \quad (2)$$

The x-axis can be defined as the vector between the Kinect joint points (12, 16)

$$\vec{x} = \frac{(\vec{r}_{16_{Kinect}} - \vec{r}_{12_{Kinect}})}{|\vec{r}_{16_{Kinect}} - \vec{r}_{12_{Kinect}}|} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad (3)$$

and the y-axis can be calculated by:

$$\vec{y} = \vec{z} \times \vec{x} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}. \quad (4)$$

The transformation matrix A is generated as follows:

$$A = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix}. \quad (5)$$

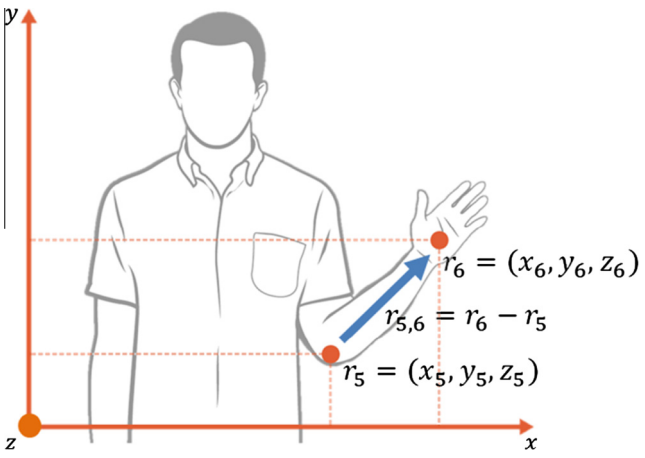


Fig. 3. Body portion definition.

All coordinates on the Kinect are transferred to the NAO robot. The arm joint angles of the robot can be calculated based on geometric calculations. The NAO robot's right arm joints angles are obtained as follows [42,44,45]:

$$Right\ elbow\ roll = \cos^{-1}\left(\frac{\vec{r}_{8,9_{NAO}} \cdot \vec{r}_{9,10_{NAO}}}{|\vec{r}_{8,9_{NAO}}| \cdot |r_{9,10_{NAO}}|}\right), \tag{6}$$

$$Right\ should\ erroll = -\left(\frac{\pi}{2} - \cos^{-1}\left(\frac{(\vec{y} \times \vec{z}) \cdot \vec{r}_{8,9_{NAO}}}{|\vec{y} \times \vec{z}| \cdot |\vec{r}_{8,9_{NAO}}|}\right)\right), \tag{7}$$

$$Right\ shoulder\ pitch = -\tan^{-1}\left(\frac{r_{3_{right}}}{r_{1_{right}}}\right), \vec{r}_{8,9_{NAO}} = \begin{pmatrix} r_{1_{right}} \\ r_{2_{right}} \\ r_{3_{right}} \end{pmatrix}. \tag{8}$$

The coordinates $r_{1_{right}}, r_{2_{right}}$, and $r_{3_{right}}$ represent the projections of vector $\vec{r}_{8,9_{NAO}}$ on the x-axis, y-axis, and z-axis, respectively. If the right shoulder roll and pitch angles are respectively expressed with α and γ, the right elbow yaw angle is calculated by:

$$\vec{b}_{right_{NAO}} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\gamma) & 0 & \sin(\gamma) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) \end{pmatrix} \cdot \vec{z} \tag{9}$$

$$\vec{r}_{right_{NAO}} = \frac{\vec{r}_{8,9_{NAO}} \times \vec{r}_{9,10_{NAO}}}{|\vec{r}_{8,9_{NAO}} \times \vec{r}_{9,10_{NAO}}|} \tag{10}$$

$$Right\ elbow\ yaw = \left(\frac{\pi}{2} - \cos^{-1}\left(\frac{\vec{b}_{right_{NAO}} \cdot \vec{r}_{right_{NAO}}}{|\vec{b}_{right_{NAO}}| \cdot |\vec{r}_{right_{NAO}}|}\right)\right). \tag{11}$$

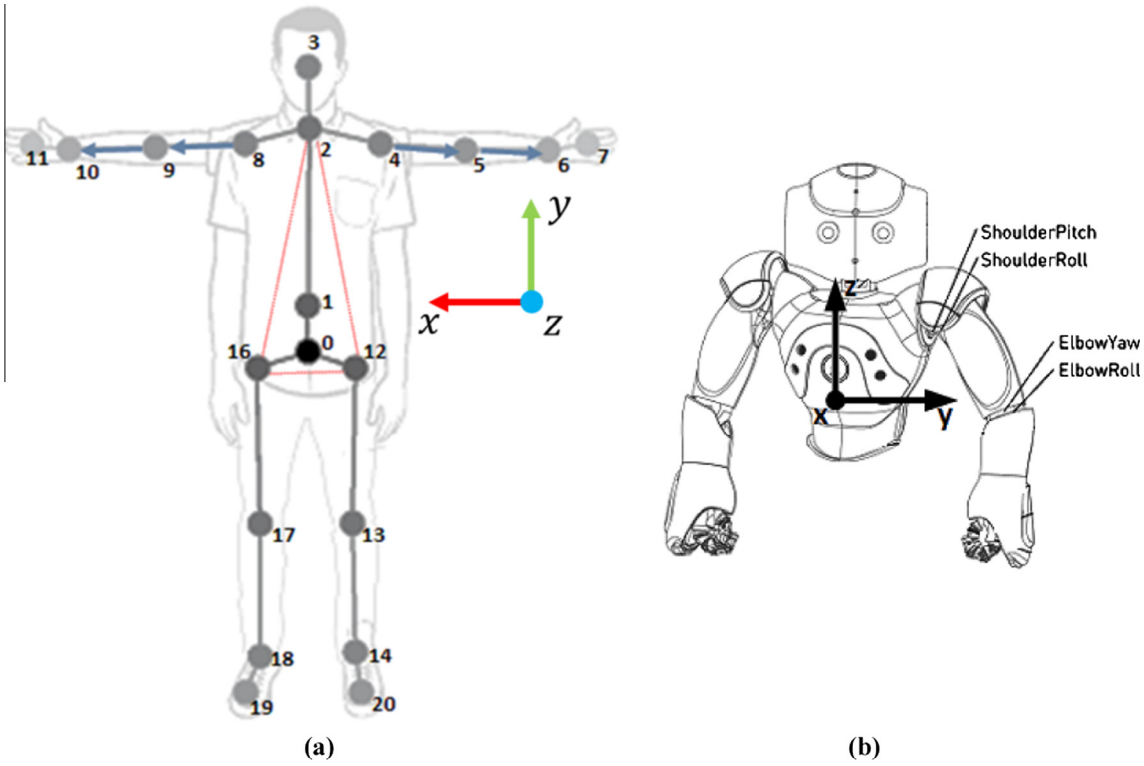
All equations relating to the joint angles of the left arm are calculated like as the right arm.

4. Methodology

4.1. K-nearest neighbor (K-NN) classifiers

The K-NN classifier is a sample-based learning algorithm. To classify an unknown pattern, the classifier first finds the training patterns closest to it in the feature space and then assigns a class by a majority vote of its *k*-nearest neighbors, where *k* is a positive integer [28]. The majority vote rule means that a pattern should be assigned to the class most common among its *k* nearest neighbors.

In this paper, the K-NN classifier was applied using Euclidean distance metrics to locate the nearest neighbor in the feature



(a) (b)

Fig. 4. Transformation to NAO platform from the Kinect coordinate system [44]. (a) The skeleton model with Kinect joint points in the Kinect coordinate system. (b) The joint angles of the Torso NAO robot.

space. Given two pattern vectors o^1 and o^2, the Euclidean distance ($\|o^1 - o^2\|$) between the vectors is defined as:

$$\|o^1 - o^2\| = \sqrt{\sum_{i=1}^{N} (o_i^1 - o_i^2)^2} \tag{12}$$

where N is the number of samples describing o^1 and o^2. A correct selection of the number of neighbor k is important to obtain a high classification performance. Having k be too large or too small influences the generalization capability.

4.2. Feed-forward neural networks (FNNs)

FNNs are based on a simplified mathematical representation of the biological nervous system. The FNN architecture usually consists of an input layer, one or more hidden layers, and an output layer. Each layer includes parallel processing elements (neurons) and the layers are fully connected to the next layer by synaptic interconnection weights. In Fig. 5, the structure of an FNN with one hidden layer is illustrated. The neurons in the input layer include the input values obtained from the training data. Each neuron in the hidden layer processes the inputs into the neuron outputs.

For a set of training samples (o_i, d_i), $i = 1, \ldots, N$ with $o_i \in \mathscr{R}^n$ and $d_i \in \mathscr{R}^m$, the output formulation of an FNN including C neurons in the hidden layer is defined by:

$$\hat{d}_i = \sum_{j=1}^{C} v_j g(w_j \cdot o_i + b_j), \quad i = 1, \ldots, N \tag{13}$$

where w_j is the input weight vector connecting the input layer to the *jth* hidden node, b_j is the bias weight on the *jth* hidden node, v_j is the weight vector connecting a *jth* hidden node to output nodes, and g is the activation function of the hidden layer. The linear activation function or one of the nonlinear activation functions for hidden layers, such as sigmoid $g(w, b, o) = 1/1 + \exp(-(w \cdot o + b))$ and hyperbolic tangent $g(w, b, o) = (1 - \exp(-(w \cdot o + b)))/(1 + \exp(-(w \cdot o + b)))$ can be used, but the most common choice in real applications is a sigmoid function. In addition, any activation function may be applied to the output neurons in Eq. (13). For purposes of this paper, the FNN uses the linear activation function for output neuron.

The training process of the FNNs involves the tuning of free parameters that are composed of weights and biases. The most extensively used training method is the back propagation algorithm [27–30]. In this algorithm, the objective function is composed of errors.

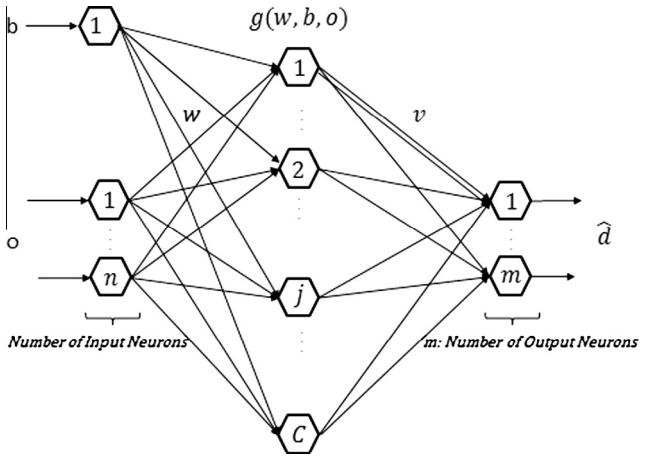


Fig. 5. The structure of the FNN.

The error signal at the output of neuron p in the output layer of the network at an iteration l is defined by:

$$e_p(l) = d_p(l) - \hat{d}_p(l). \tag{14}$$

where d_p refers to the desired response for neuron p and $\hat{d}_p$ refers to the function signal appearing at the output of neuron p.

The instantaneous value of the total error energy over all neurons in the output layer is thus written:

$$E(l) = \frac{1}{2} \sum_{p=1}^{m} e_p^2(l), \tag{15}$$

For a given training set, $E(l)$ represents the objective function. The objective of the training process is to adjust the free parameters of the network to minimize the objective function. To achieve this minimization, the negative gradients of the objective function are computed during the training stage. The adjustments to the set of all initial weights W_j consisting of v_j, w_j, and b_j are made in accordance with respective gradients computed with respect to each weight of the objective function as follows:

$$W_j(l+1) = W_j(l) - \eta(l) \frac{\partial E(l)}{\partial W_j(l)}, \tag{16}$$

where $\eta > 0$ is learning rate. The standard backpropagation algorithm requiring the first order gradient is very time consuming. Although using an adaptive learning rate or momentum term can make the algorithm faster, these endeavours don't make it faster than the Newton method requiring the second order gradient [30]. The Newton method requiring the second order gradient method provides good results regarding higher computational cost. In this paper, the Levenberg-Marquardt method was applied. The method use an approximation to the Hessian matrix without calculating the second gradient [28,29]. It is faster and more accurate than the backpropagation algorithm.

The Newton method updates the all parameters by:

$$W_j(l+1) = W_j(l) - \nabla^2 E(l)^{-1} \nabla E(l), \tag{17}$$

where $\nabla^2 E(l)$ is the local Hessian matrix and $\nabla E(l)$ is the local gradient. If the Taylor series expansion is applied to the objective function around the operating point, the final form of the Newton-like update formula for the Levenberg-Marquardt method is described by:

$$W_j(l+1) = W_j(l) - [J^T(l)J(l) + \mu I]^{-1} J^T(l) e_j(l), \tag{18}$$

where J is the Jacobian matrix that contains first derivatives of network errors with respect to all free parameters, I is the identity matrix, and μ is a scalar factor. For $\mu > 0$, the μ is multiplied by some parameter β (normally $\beta = 10$) whenever a step would result in an increased objective function. Otherwise, μ is divided by β. When the scalar μ is sufficiently large, the algorithm becomes a gradient descent with a small step $1/\mu$. For small μ, the algorithm becomes Gauss-Newton.

4.3. Extreme learning machines (ELMs)

The ELM is a kind of FNN with C hidden neurons and common activation functions in Fig. 5 [19–23]. The input weight w_i and bias b_i values of ELMs are randomly generated according to continuous probability distributions that are different from the FNN. Thus, the output parameters v_i of the ELM represented by the linear system

are learned by solving the minimum norm least-squares formulation. Thus, zero error is approximately provided for N samples as follows:

$$\sum_{i=1}^{N}\|d_i-\hat{d}_i\|=0 \tag{19}$$

or

$$\sum_{j=1}^{C}v_j g_j(w_j\cdot o_i+b_j)=d_i,\ i\in\{1,2,\ldots,N\}. \tag{20}$$

ELM output formulation is compactly expressed as follows:

$$D=GV \tag{21}$$

$$G=\begin{bmatrix} g(w_1\cdot o_1+b_1) & \ldots & g(w_C\cdot o_1+b_C)\\ \vdots & \ldots & \vdots\\ g(w_1\cdot o_N+b_1) & \ldots & g(w_C\cdot o_N+b_C)\end{bmatrix}_{NxC} \tag{22}$$

$$V=[v_1,v_2,\ldots,v_m]^T_{Cxm}\text{ and }D=[d_1,d_2,\ldots,d_N]^T_{Nxm}. \tag{23}$$

where G is the hidden layer output matrix and g is a nonlinear piecewise continuous function satisfying the universal approximation capability theorems of ELMs [21].

For the fixed w_i and b_i, the training of ELMs is equal to solving the primal optimization problem:

$$\min_V\|D-GV\|. \tag{24}$$

If G is a non-square matrix for $C\ll N$, then the unique solution of output weights can be determined by the minimum norm least-squares method the linear system in Eq. (21):

$$V=G^{\dagger}\mathrm{D}=(G^T\mathrm{G})^{-1}G^T\mathrm{D}, \tag{25}$$

where $G^{\dagger}$ is the Moore-Penrose generalized inverse of matrix G. The smallest training error is attained by:

$$\min_V\|D-GV\|=\|D-GV\|=\|D-GG^{\dagger}\mathrm{D}\|. \tag{26}$$

The ELMs have the following benefits thanks to the least-squares solution:

1. The first-order gradient-based methods used for the FNNs require a very long training time and result in low accuracy. The adaptive learning rate reduces these disadvantages. However, a small learning rate represents a very slow convergence while a larger one may cause bad local minima. The second-order gradient-based methods and their approximations have many more computational methods than the first order methods. On the other hand, the ELMs are extremely fast and have stronger generalization ability than the FNN.
2. Compare to K-NN, K-NN is fast for a testing stage. Moreover, it doesn't require any effort for training, which means it can be implemented for real time applications. However, the K-NN exhibits low testing accuracy. On the other hand, the ELM with a fixed hidden neuron number can be tested in real time with high accuracy. Even the training stage can be embedded into the program as a pre-stage relating to the testing stage. Thus, a complete ELM can be applied in real time without user intervention, with the exception of determining the hidden neuron number.
3. The ELM produces not only the smallest training error, but also the smallest generalization error due to the smallest norm of output weights, like with SVMs.

5. Experimental results

In the proposed system, a computer with an Intel Core i5-2400 3.10 GHz CPU was used. The program was written in Microsoft Visual Studio 2010 and the Microsoft.NET Framework 4.0 by using both Kinect for Windows SDK 1.8 and the NAOqi.NET SDK. The communication interface was built with WPF (Windows Presentation Foundation) and the software codes were programmed in C#. The communication between the NAO robot and the Kinect sensor was conducted through the Wi-Fi network.

The experiments were realized in backgrounds under varying lighting conditions in the laboratory. To show the system's efficiency, users with different sizes were selected. Then the users were asked to stand in the front of Kinect sensor. The distance between the users and the Kinect sensor was 0.3–0.5 m.

The system was constructed in two parts for two different aims. In the first part, different user commands were directly given to the NAO robot in real time. Six upper-body actions relating to the user's upper-body were selected: arms at sides, hands up, right hand up, left hand up, hands in front, and hands down. Fig. 6 illustrates an implementation interface generated by using Choregraphe and C#. Fig. 7 shows the snapshots taken from our real robot imitation system. The figures demonstrate that the NAO robot successfully follows the user's upper-body actions in real time. However, there are some limitations due to the differences between the NAO robot and the human (e.g., the human arm has seven degrees of freedom, whereas the NAO's arm has just six). Furthermore, the Kinect sensor is not able to track human fingers movements and rotations of the hand/forearm [42].

In the second part, an algorithm for recognizing human actions was constructed as a six-class classification problem. A human action data set composed of six upper-body actions was created: arms at sides, hands up, right hand up, left hand up, hands in front, and hands down. In order to build different body actions, 24 different users were selected for the experiment. Each action was repeated 10 times by each user and all joint angles relating to each action were obtained by the Kinect sensor. After the collected joint angles were transformed into the joint angles of the NAO robot, they were divided into six classes according to the upper-body actions. In this paper, ELM was used because it is an efficient classifier. In order to show the effectiveness of the ELM-based classification algorithm, the K-NN and FNN classifiers were used for comparison purposes as well.

In the experiments, the multi-class classification problem was applied by using a single ELM and FNN. A network structure of multi-output nodes equal to the class number, six, was selected. A pattern of class i was labelled as "1" and the patterns belong to the other classes were labelled as "−1". After completing the training stage, the max operation was performed to six output values to apply the winner-takes-all method that classifies the input pattern to a winner class.

The joint angles generated by the first eight users were used for the training set, while those of the next two sets of eight users were used for the testing and validation sets. All simulation results were obtained by MATLAB. In the FNN, the hyperbolic tangent and linear activation functions were used for the hidden and output layers, respectively. The sigmoid activation function was used for the hidden layer of the ELM. In order to find the finest network architecture of ELM and FNN, the number of hidden neurons was gradually increased from 1 to 60. The hidden neuron numbers producing the best validation accuracies were searched. The ELM training was carried out by calculating the matrix inverse in the least-squares method. The FNN was trained by the Levenberg-Marquardt method [28,29]. The FNN was trained for 400 epochs, as the errors on the validation and training sets after the epoch

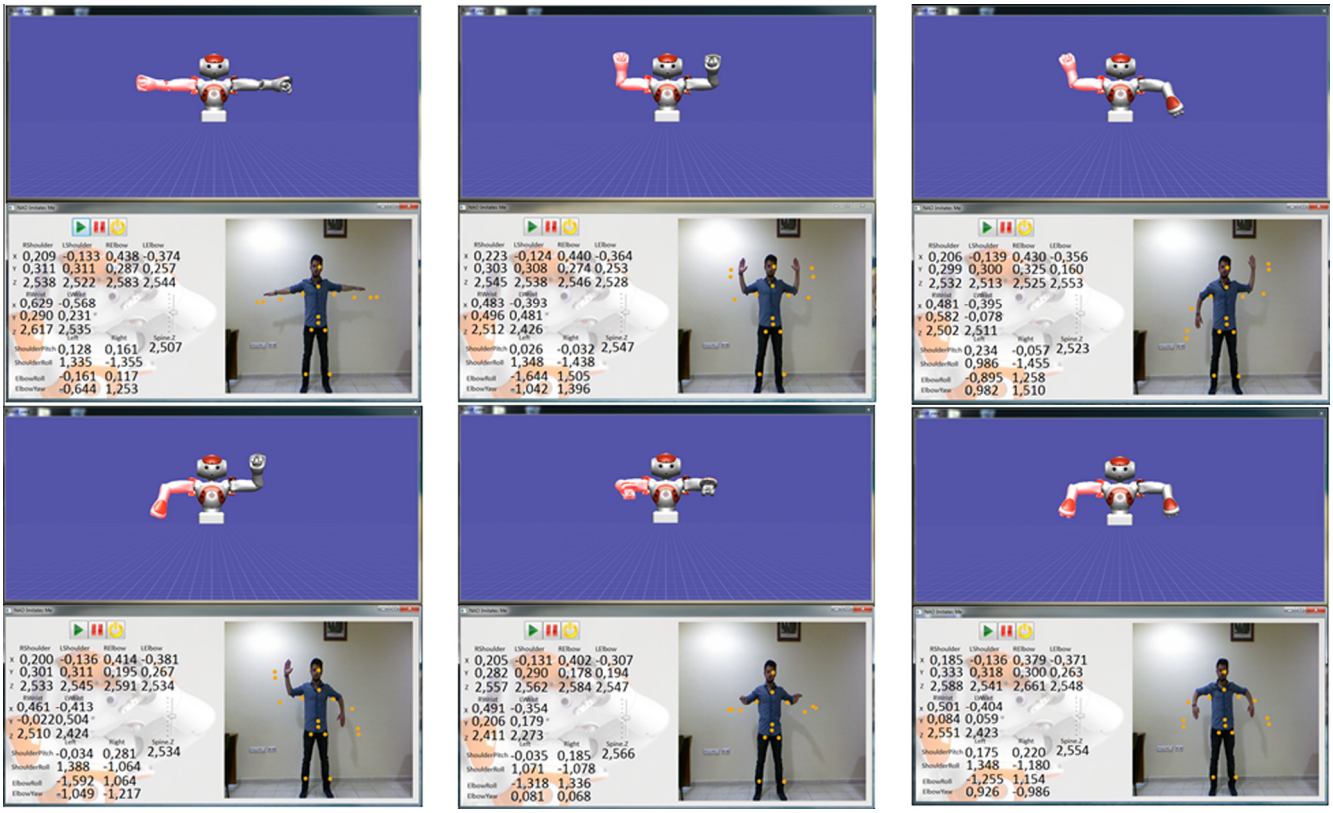


Fig. 6. The developed implementation interface for imitation.

Fig. 7. The snapshots taken in the real human imitation experiments.

were too small. For K-NN, the finest k value was determined by searching for the best accuracy on independent validation set for different k values. The value of k was varied in a range of 1–10. Fig. 8 shows the training and validation performances of ELM, FNN, and K-NN with respect to the hidden neuron number and k, respectively. As can be seen from Fig. 8, the best validation accuracies were obtained by the hidden neurons of 21 for the ELM classifiers and 36 for the FNN classifiers. K-NN met the best validation accuracy by eight of its nearest neighbors. After the numbers of hidden neurons and the nearest neighbors were fixed for the best option, the effectiveness of the proposed approaches was evaluated both in terms of recognition accuracy and speed. Fig. 9 shows the performance of ELM, FNN and K-NN on the training, testing, and validation sets. The highest accuracies were obtained by ELM classifiers with 99.5833%, 98.5417%, and 96.4583% in the training, validation, and testing stages, respectively. Table 1 shows the durations for each classifier at all stages. The ELM classifier achieved the smallest training, testing, and validation times, shown in Table 1. To show the recognition performance of the algorithm, the confusion matrixes were calculated for the ELM, FNN, and K-NN classifiers using 80 sequences relating to eight users. In Tables 2–4, all assignments in the confusion matrixes are represented in the corresponding box. The ELM classifier revealed only 7 errors. The results relating to FNN and K-NN

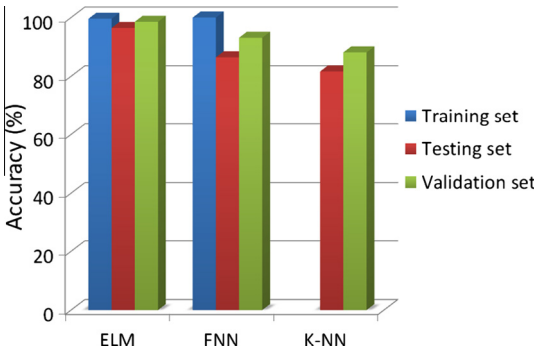
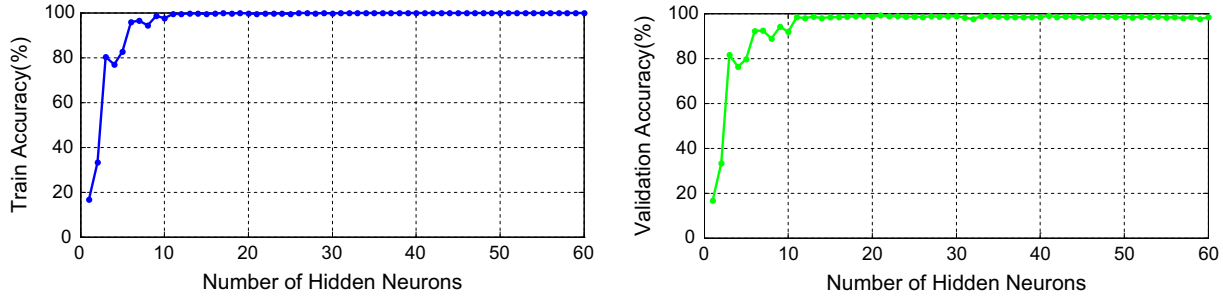


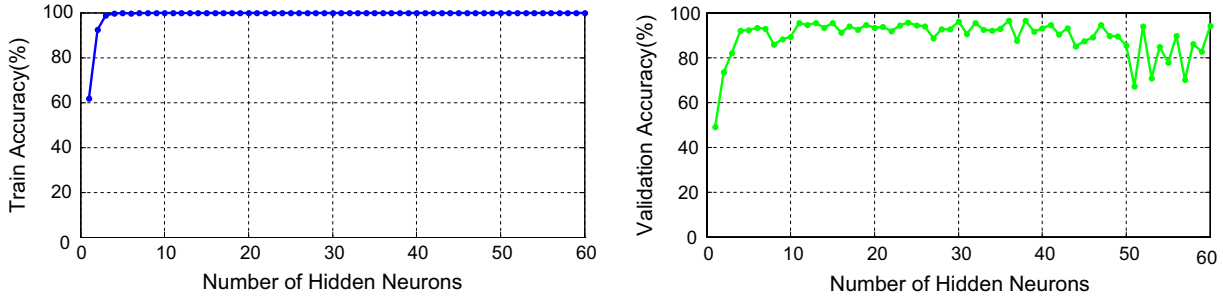
Fig. 9. Classification accuracies for each classifier.

Table 1
The learning times relating to the classifiers.

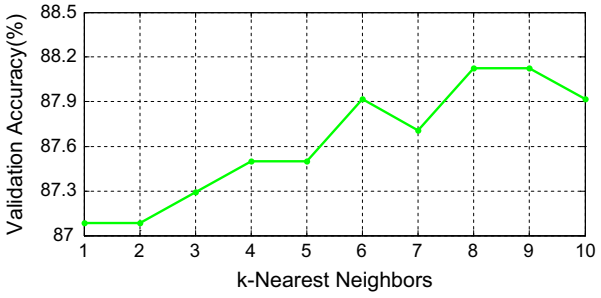
Classifier	ELM	FNN	K-NN
Training time (s)	0.0010	3.3317	–
Testing time (s)	0.0054	0.0284	0.0933
Validation time (s)	0.0016	0.0275	0.0462



(a) ELM classifier



(b) FNN classifier



(c) K-NN classifier

Fig. 8. The training and validation performances of the classifiers.

Table 2
Confusion matrix relating to the ELM classifier.

Known class	Predicted class	Arms at sides	Hands up	Right hand up	Left hand up	Hands in front	Hands down
Arms at sides		79	0	1	0	0	0
Hands up		0	79	0	0	1	0
Right hand up		0	0	80	0	0	0
Left hand up		0	0	0	78	0	2
Hands in front		0	1	0	0	79	0
Hands down		0	0	0	0	2	78

Table 3
Confusion matrix relating to the FNN classifier.

Known class	Predicted class	Arms at sides	Hands up	Right hand up	Left hand up	Hands in front	Hands down
Arms at sides		28	31	9	12	0	0
Hands up		2	76	0	1	1	0
Right hand up		0	0	80	0	0	0
Left hand up		0	0	1	75	0	4
Hands in front		0	1	0	1	78	0
Hands down		0	0	1	0	1	78

Table 4
Confusion matrix relating to the K-NN classifier.

Known class	Predicted class	Arms at sides	Hands up	Right hand up	Left hand up	Hands in front	Hands down
Arms at sides		80	0	0	0	0	0
Hands up		2	56	14	4	0	4
Right hand up		0	12	55	2	0	11
Left hand up		0	3	0	64	0	13
Hands in front		0	0	1	0	76	3
Hands down		0	5	0	12	2	61

classifiers proved the goodness of the ELM-based proposed algorithm since FNN and K-NN provided 65 errors and 88 errors, respectively. The results show that the proposed ELM algorithm is robust against illumination and user dimensions.

All results exhibit that the ELM has the advantages, specifically regarding training time, applicability in real time, and high recognition performance in the application of human action recognition on the NAO robot by using a Kinect sensor. Therefore, the proposed system is more practical than current usable applications.

6. Conclusions

Interaction of robots with older adults, disabled individuals, and children is very challenging because of the requirements on the constructed system, accuracy, and implementation speed. In this paper, two different applications were proposed for teaching the humanoid robots how to mimic human behaviors. The first application compromises an imitation system in which the user's upper-body human actions were simultaneously realized by the NAO humanoid robot by using an Xbox 360 Kinect sensor. In this application, three steps were monitors: collection of the user's motion data, transforming the data into the robot's coordinates, and sending the converted data to the robot. The Kinect sensor was first used for collecting the motion data relating to user joint points. A transformation algorithm was then utilized to calculate the robot's joint angles. The data was finally transferred to the robot. Several tests were performed on a set of users having different body measurements in environments with different lighting levels and backgrounds. Results showed that the robot is able to imitate the motions regardless of the user who performed the motion and the surrounding illumination. The users interacted with the NAO robot in a natural and fast way, similar to interactions between other humans. The constructed system by the Kinect could be used for children or older people without exterior intervention by physiotherapists thanks to the developed computational tools.

In the second application, an upper-body human action recognition system was developed. The proposed ELM-based algorithm for action recognition was successfully applied in real time. The effectiveness of the algorithm was shown by classification accuracy and speed. In addition, the experimental results were comparatively demonstrated with respect to K-NN and FNN. It was observed that the proposed ELM recognition algorithm achieved the highest accuracy level on the training, testing, and validation stages. Moreover, since the proposed algorithm is the fastest one, it can be used in real time. The system can be used comfortably by older adults or disable peoples. More motions or gestures could be added to the database of the proposed system. Thus, the robots could more easily and naturally perform the tasks given to them by humans. In the present system architecture, the NAO robot and the Kinect sensor are connected to each other via a discrete powerful computation system. If the Kinect sensor and all of the developed algorithms could be embedded as hardware and software within the NAO robot itself in future, then the imitation and gesture recognition processes would be more practical.

Future work will consider improving the human-robot interaction on software developed by using a walking NAO robot, network technologies, and the Kinect.

Acknowledgements

This paper was supported by the Firat University Scientific Research Projects Foundation (no. MF.12.33).

References

[1] B. Gates, A robot in every home, Sci. Am. 296 (1) (2007) 58–65.
[2] K. Dautenhahn, Socially intelligent robots: dimensions of human–robot interaction, Philos. Trans. R. Soc. Lond., B, Biol. Sci. 362 (1480) (2007) 679–704.

[3] M.A. Goodrich, A.C. Schultz, Human–Robot Interaction: A Survey, Foundations and Trends in Human–Computer Interaction, Now Publishers Inc., 2008.
[4] T. Fong, I. Nourbakhsh, K. Dautenhahn, A survey of socially interactive robots, Rob. Auton. Syst. 42 (3) (2003) 143–166.
[5] C. Breazeal, C.D. Kidd, A.L. Thomaz, G. Hoffman, M. Berlin, Effects of nonverbal communication on efficiency and robustness in human-robot teamwork, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, Alberta, Canada, 2005, pp. 708–713.
[6] J.A. DeVito, M.L. Hecht, The Nonverbal Communication Reader, third ed., Waveland Press, 1990.
[7] M. Riley, A. Ude, K. Wade, C.G. Atkeson, Enabling real-time full-body imitation: a natural way of transferring human movement to humanoids, in: Proc. of IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 2003, pp. 2368–2374.
[8] L. Molina-Tanco, J.P. Bandera, R. Marfil, F. Sandoval, Real-time human motion analysis for human-robot interaction, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, Alberta, Canada, 2005, pp. 1402–1407.
[9] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell, A.G. Billard, Learning and reproduction of gestures by imitation, IEEE Trans. Rob. Autom. Magn. 17 (2) (2010) 44–54.
[10] F.G. Pereira, R.F. Vassallo, E.O.T. Salles, Human–robot interaction and cooperation through people detection and gesture recognition, J. Control Autom. Electr. Syst. 24 (3) (2013) 187–198.
[11] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, X. Twombly, Vision-based hand pose estimation: a review, Comput. Vis. Image Underst. 108 (1) (2007) 52–73.
[12] S. Mitra, T. Acharya, Gesture recognition: a survey, IEEE Trans. Syst. Man. Cybern. C Appl. Rev. 37 (3) (2007) 311–324.
[13] G. Dewaele, F. Devernay, R. Horaud, Hand motion from 3d point trajectories and a smooth surface model, in: T. Pajdla, J. Matas (Eds.), 8th European Conference on Computer Vision, Volume I of LNCS 3021, Springer, 2004, pp. 495–507.
[14] Z. Ren, J. Yuan, J. Meng, Z. Zhang, Robust part-based hand gesture recognition using Kinect sensor, multimedia, IEEE Trans. Multimed. 15 (5) (2013) 1110–1120.
[15] R. Ibañez, Á. Soria, A. Teyseyre, M. Campo, Easy gesture recognition for Kinect, Adv. Eng. Softw. 76 (2014) 171–180.
[16] Y. Xiao, Z. Zhang, A. Beck, J. Yuan, D. Thalmann, Human-robot interaction by understanding upper-body gestures, Presence 23 (2) (2014) 133–154.
[17] D.U. Guanglong, P. Zhang, Human–manipulator interface using hybrid sensors with Kalman filters and adaptive multi-space transformation, Measurement 55 (2014) 413–422.
[18] M. Bueno, L. Díaz-Vilariño, J. Martínez-Sánchez, H. González-Jorge, H. Lorenzo, P. Arias, Metrological evaluation of KinectFusion and its comparison with Microsoft Kinect sensor, Measurement 73 (2015) 137–145.
[19] G.B. Huang, C.K. Siew, Real-time learning capability of neural networks, IEEE Trans. Neural Network 17 (4) (2006) 863–878.
[20] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[21] G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man. Cybern. B Cybern. 42 (2) (2012) 513–529.
[22] A. Uçar, Color face recognition based on steerable pyramid transform and extreme learning machines, Sci. World J. (2014) 1–15, http://dx.doi.org/10.1155/2014/628494 628494.
[23] A. Uçar, E. Yavşan, Behavior learning of a memristor – based chaotic circuit by extreme learning machines, Turk. J. Elec. Eng. Comp. Sci. 24 (1) (2016) 121–140.
[24] A. Uçar, Y. Demir, C. Güzeliş, A new facial expression recognition based on curvelet transform and online sequential extreme learning machine initialized with spherical clustering, Neural Comput. Appl. 27 (1) (2016) 131–142.
[25] Z. Zhang, D. Tao, Slow feature analysis for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 436–450.
[26] J.K. Aggarwal, L. Xia, Human activity recognition from 3d data: a review, Pattern Recognit. Lett. 48 (2014) 70–80.
[27] Y. Demir, A. Uçar, Modelling and simulation with neural and fuzzy-neural networks of switched circuits, COMPEL, Int. J. Comput. Math. Electr. Electro. Eng. 22 (2) (2003) 253–272.
[28] S. Haykin, Neural Networks and Learning Machines, third ed., Prentice Hall, Upper Saddle River, New Jersey, 2008.
[29] A. Ebrahimzadeh, A. Khazaee, Detection of premature ventricular contractions using MLP neural networks: a comparative study, Measurement 43 (2009) 103–112.
[30] A. Uçar, Y. Demir, C. Güzeliş, A penalty function method for designing efficient robust classifiers with input space optimal separating surfaces, Turk. J. Elec. Eng. Comp. Sci. 22 (6) (2014) 1664–1685.
[31] K.R. Konda, A. Königs, H. Schulz, D. Schulz, Real time interaction with mobile robots using hand gestures, in: Proc. of the 7th ACM/IEEE International Conference on Human-Robot Interaction, Boston, USA, 2012, pp. 177–178.
[32] R. Minhas, A. Baradarani, S. Seifzadeh, Q.J. Wu, Human action recognition using extreme learning machine based on visual vocabularies, Neurocomputing 73 (10) (2010) 1906–1917.
[33] S. Decherchi, P. Gastaldo, A. Leoncini, R. Zunino, Efficient digital implementation of extreme learning machines for classification, IEEE Trans. Circuits Syst. II Express Briefs. 59 (8) (2012) 496–500.
[34] X. Chen, M. Koskela, Online RGB-D gesture recognition with extreme learning machines, in: Proc. of the 15th ACM on International Conference on Multimodal Interaction, Sydney, Australia, 2013, pp. 467–474.
[35] X. Chen, M. Koskela, Skeleton-based action recognition with extreme learning machines, Neurocomputing 149 (2015) 387–396.
[36] E. Mota, A.P. Moreira, T.P. do Nascimento, Motion and teaching of a NAO robot, in: Proc. of Provas de Dissertacao do MIEEC, Portugal, 2011, pp. 1–2.
[37] A.R. Ibrahim, W. Adiprawita, Analytical upper-body human motion transfer to NAO humanoid robot, Int. J. Electr. Eng. Inf. 4 (4) (2012) 563–574.
[38] A.A. Manasrah, Human motion tracking for assisting balance training and control of a humanoid robot, Master Thesis, University of South Florida, 2012.
[39] I.I. Itauma, H. Kivrak, H. Kose, Gesture imitation using machine learning techniques, in: Proc. of 20th IEEE Signal Processing and Communications Applications Conference, Istanbul, Turkey, 2012, pp. 1–4.
[40] J. Koenemann, M. Bennewitz, Whole-body imitation of human motions with a NAO humanoid, in: Proc. of 7th ACM/IEEE International Conference on Human-Robot Interaction, Boston, USA, 2012, pp. 425–425.
[41] F. Zuher, R. Romero, Recognition of human motions for imitation and control of a humanoid robot, in: Proc. of IEEE Robotics Symposium and Latin American Robotics Symposium, Brazilian, 2012, pp. 190–195.
[42] S. Franz, R. Nolte-Holube, F. Wallhoff, NAFOME: NAO Follows Me – tracking, reproduction and simulation of human motion, in: Proc. of 4th European Conference on Technically Assisted Rehabilitation, Berlin, 2013, pp. 1–4.
[43] S. Michieletto, A. Rizzi, E. Menegatti, Robot learning by observing humans activities and modeling failures, in: Proc. of 2nd International Workshop on Cognitive Robotics Systems: Replicating Human Actions and Activities, IEEE, Tokyo, Japan, 2013.
[44] E. Yavşan, A. Uçar, Teaching human gestures to humanoid robots by using Kinect sensor, in: Proc. of IEEE 23th Signal Processing and Communications Applications Conference, Malatya, Turkey, 2015, pp. 1208–1211.
[45] Aldebaran, March 2012. <https://www.aldebaran.com >.