

# Developing and establishing a natural user interface based on Kinect sensor with artificial neural network

CEMIL OZ\*, M. ALI OZ<sup>a</sup>

*Sakarya University, Computer Engineering Department, Sakarya, Turkey*

*<sup>a</sup>Yildiz Technical University, Control and Automation Engineering Department, Istanbul, Turkey*

---

In this study, a real-time human motion classification system is developed by using Artificial Neural Networks (ANN) with negative correlation learning (NCL), to control windows games and applications. The system uses two Microsoft Kinects to extract human motion features. The x, y, z values of human skeleton joints are obtained from two Kinects using Microsoft Kinect Library and Microsoft c#. The data obtained from these devices are processed using noise reduction, feature extraction and classification modules. Four feature vectors; hand shape, hand locations, hand movement and hand distance are extracted for every human action and histograms of these feature vectors are used for classification. Kalman filter is used for noise reduction. Hand shapes are located and extracted using skeleton hand joints data, Kinect dept camera image and Kinect RGB camera image. Hands feature vectors are extracted with moment invariant method. The neural network is used as a classifier. The test results show that the developed and established system can be successfully used in real time recognition of human motion. This system is flexible and open for future extensions.

(Received August 12, 2014; accepted November 13, 2014)

*Keywords:* Human Motion Analysis, Microsoft Kinect, Artificial Neural Network, Natural User Interface

---

## 1. Introduction

Analysis of 3D human motion is an important tool for various interactive systems; it is used in virtual reality, video games, animation movies, medical analysis, scientific visualization, puppetry, and gesture-based control. To realize such interaction, the system has to estimate motion parameters of human bodies in real-time. An efficient 3D human motion analysis system should be able to recognize and understand the human behavior from observations alone. This task is very difficult, misinterpretations and mistakes are very common. The efficient human activity monitoring system should also provide independent movement ability and a secure environment to the user.

Real-time human body tracking research can be categorized into three major classes: (i) computer-vision based, (ii) special device (such as flock of birds, a data glove) based, and (iii) a combination of the two classes [1]. Most of the vision based human motion capture systems have multiple cameras which are specially located. The user wears a special body suit and places markers on various parts of the suit to capture the movement of the joints in the human body [2]. Recently, in parallel to the development of the computer hardware and software, some fully image-feature-based motion capturing systems without such restrictions have been developed as a computer vision application [3-5].

Marker-less motion capture technology has been harnessed for several years to track human movements for developing various applications. Shimosaka et al. (2010) introduced a robust framework for multiple people pose tracking [6]. The notable aspects of their approach are

real-time ensuring speed (up to 30 fps), and flexibility towards various complex motions and environments. Their framework successfully obtains multiple body pose estimation in real-time even when contacts between people occurs in the scene, which is addressed in the conventional approaches. They demonstrated the effectiveness of their approach with experiments on indoor cluttered scene sequences. Graf et al. (2011) presented a marker-less 3D motion capture system based on a volume reconstruction technique of non-rigid bodies [7]. It depicted a new approach for pose estimation in order to fit an articulated body model into the captured real-time information. They aimed at analyzing athlete's movements in real-time within a 3D interactive graphics system. Wang et al. (2010) provided a comprehensive survey of recent developments on gait recognition approaches [8]. Their survey emphasized on three major issues involved in a general gait recognition system, namely, gait image representation, feature dimensionality reduction, and gait classification. Also, a review of the available public gait datasets was given. Their conclusion outlined a number of research challenges and provided promising future directions for the field.

Recently with the launch of Microsoft Kinect researchers have been keenly interested in developing applications using this device [9, 11 12]. It is a low-cost and good substitute for the comparatively expensive other vision based motion capture systems. Although it is designed for home entertainment, numerous applications can be developed with the capabilities of Kinect. The skeleton data of a human being tracked by a single Kinect device is enough to obtain the human motion simulation in many cases. This paper aims to provide an application

based on ANN classifier using two Kinect that uses gestures to interact with windows application, windows games and virtual object in a Virtual reality application.

## 2. System overview

The system software written with Microsoft C# has four modules: data capturing, data selecting, ANN classifier training and testing and real-time motion classification. The data capturing module tracks the user skeleton joint data in 3D coordinate system with two Kinects using Microsoft Kinect Library and store the captured data in the database when the users move their arms. Not all of the captured data are suitable for classification. Therefore, the data selecting module is used for manual classification of the human motion to prepare training and testing data for the ANN model. Data selection module also increases robustness in classification. The recorded human skeleton is replayed step by step. If the data are suitable for a certain class, the extracted feature vector of the current skeleton data is stored in an input file and the classification script is recorded to an output file. Fig. 1 shows a block diagram of the system.

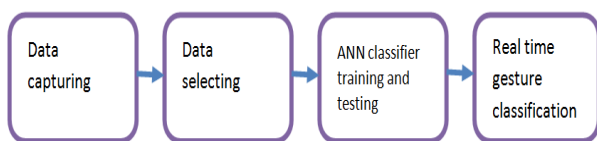


Fig. 1. Block diagram of the system.

### 2.1 System hardware

One of the primary means by which we physically connect to the world is through our limbs, especially the hands. We perform most of our everyday tasks with them; however, along with our hands, we also rely on devices such as a mouse, keyboard, or joystick to work with a computer and computer applications. Virtual Reality input devices like data glove, motion tracker, and Kinect could overcome the limitations of these devices [13, 14]. However, most commercially available virtual reality devices are very expensive.

The Kinect sensor was developed for the Microsoft Xbox 360 video game platform. Then it was transferred to the personal computers and this device offers portable 3-D motion capture capabilities that can overcome the limitations of most existing systems. The Kinect was designed to allow users to interact with the gaming system without the need for a traditional handheld controller. Instead, the sensor recognizes the user's gestures and voice commands. The Kinect device is shown in Fig. 2.

The key to gesture recognition is the device's "depth camera" which consists of an infrared laser projector and an infrared video camera mounted within the sensor bar.

The system uses the infrared camera to detect a speckle pattern projected onto objects in the sensor's field of view. The Kinect is able to create a 3-D map of these objects by measuring deformations in the reference speckle pattern. The sensor bar also houses a color video camera which provides color data to the depth map [15].

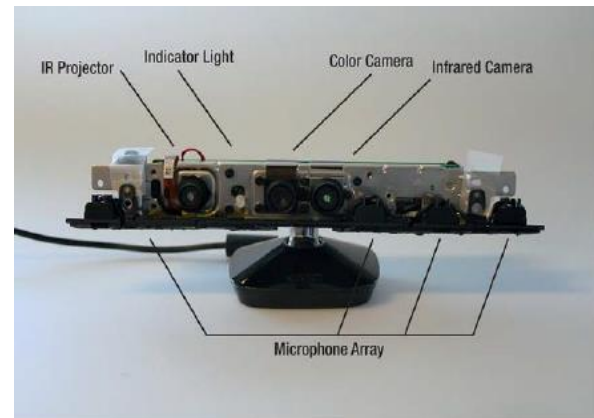


Fig. 2. The Microsoft Kinect.

Mostly, skeleton joints data are most accurate when the person faces the Kinect device, the orientation of the person with respect to the Kinect should be within  $\pm 50^\circ$ . If the user starts the turn either right or left, the accuracy of the skeleton data is decreases. A basic idea to increase the accuracy is to use multiple Kinect which are specially located in the working area. But increasing Kinect sensor will decrease the accuracy due to interference. Some studies solved this problem using multiple Kinect devices simultaneously but activate one device at a time [16]. The activation should depend on the orientation of the person being tracked. The device must switch the activation to another device and deactivate itself if a designed orientation is exceeded. Thus, since only one Kinect will track the human at a time the interference issue will be resolved. In this study, two Kinect are used to tract the human skeleton instead of single Kinect.

### 2.2. Camera calibration

Virtual Reality systems require multiple cameras to visually capture environments such as big rooms or even outdoors scenes like Immersive environments and surveillance systems. All cameras which are used in these kinds of systems should be calibrated in order to show the same world coordinate system [17]. Camera calibration is a difficult process, but, today there are some useful software tool kits for either single or multiple camera calibration to make it quite easy to achieve. [18]. However, multiple camera calibration is more difficult than single camera calibration because of lacking overlap of the field of views among cameras [19]. It is easy to realize that reference objects unless they are very elongated lines [20] will not be visible in all cameras.

There are number of ways to calibrate Kinect depth cameras. Though [21] describes a way to calibrate the depth cameras of multiple Kinect devices it is a very complex process. In this study, first, the RGB cameras of the two Kinect devices were calibrated with the help of MATLAB calibration toolbox [22] using images of the calibration board to obtain the extrinsic parameters. Then, the depth cameras of every individual Kinect device are calibrated with their respective RGB cameras with the help of the same calibration toolbox using IR image of the calibration board after illuminating with a halogen lamp while covering the IR beam from the Kinect IR source.

### 2.3 Input data selection for the system with two KINECT

An NUI system needs to use multiple Kinect simultaneously in order to get robust human skeleton data. However, Relation between the number of Kinect and image quality is exponential, if the number of Kinect increases the quality of image decreases. Our system uses two Kinect devices simultaneously and interference is tolerable using filter. We developed an algorithm for Kinect data selection. The objective of the algorithm is to decide which Kinect device should track the movements using the operator's orientation with respect to Kinect as a decisive factor. The algorithm is able to control both devices while tracking position without decreasing the frame rate.

Kinect SDK does not allow a single application to obtain skeletal data from more than one Kinect devices simultaneously. Hence, multi process application was designed to overcome this limitation. The application consisted of a parent process which enumerated the Kinect devices which are named D1 and D2, and spawned two child processes in parallel, one for each Kinect device. However, since the approach presented in this research suggested that only one Kinect device should considering at a time; it was assumed that the person was facing one of the Kinect devices (D1) initially. Thus the skeleton data from D1 was obtained and stored. Once the person started turning towards D2 and surpassed a certain angle  $\alpha=45^\circ$ , switch took place and skeleton data from D2 was obtained. There are no data losses during the switch.

### 2.4 Kalman filter

The Kalman filter is a set of mathematical equations that provides an efficient computational means to estimate the state variables of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown [23, 24]. In this paper since a control input cannot be used in the state transition model a very simple one dimensional Kalman filter will be used to estimate the position using the sensor data from Kinect. Since the model used for Kalman is very simple and is different from the real data instead of calculating estimate position Kalman filter will

be smoothing the position vector by limiting how fast the position can change thus reducing steep jumps between data.

State transition model describes how the state is expected to change from one time step to the next.  $\mathbf{A}$  matrix is the transition matrix and  $\mathbf{B}$  matrix is the control matrix. In addition  $w_{k-1}$  is the process noise and  $u_k$  is known exogenous control input. In this study control input is random depending on the Kinect user thus a detailed state transition model is not possible instead of this we will assume the position is constant an Kalman filter will be only smoothing the position vector.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (1)$$

$$A = 1, B = 0, w_{k-1} \quad (2)$$

Measurement model is given in eq. 3.

$$z_k = Hx_k + v_k \quad (3)$$

H is the observation matrix and we know that sensor information consists of only position and noise hence H matrix equals to 1. Respectively  $v_k$  is the measurement noise. Kalman filter consist of two phases time update (prediction) and measurement update (correction). Time update; At each time step filter calculates both position  $x_{k+1}$  and error covariance  $P_{k+1}$  this are also known as prior estimates.

$$x_{k+1} = Ax_k \quad (4)$$

$$P_{k+1} = AP_k A^t + Q \quad (5)$$

Q is the process noise covariance matrix which is unknown in this case it has been chosen using trial and error method.

Measurement Update: Measurement update equations, correction equations, provide feedback by introducing a new measurement value and thus producing a new position  $x_k$  and a new error covariance  $P_k$ .

$$K_{k+1} = P_{k+1} H^t (H P_{k+1} H^t + R)^{-1} \quad (6)$$

$$x_{k+1} = x_k + K_{k+1} (z_k - H x_k) \quad (7)$$

$$P_{k+1} = (1 - K_k H) P_k \quad (8)$$

R is the measurement noise covariance matrix. It is easily calculated since the environmental noise can be measured. Measurement noise and process noise have been assumed as constants. Initial values were as follows:  $Q = 1 \text{ mm}$ ,  $R = 5 \text{ cm}$ ,  $P = 1$

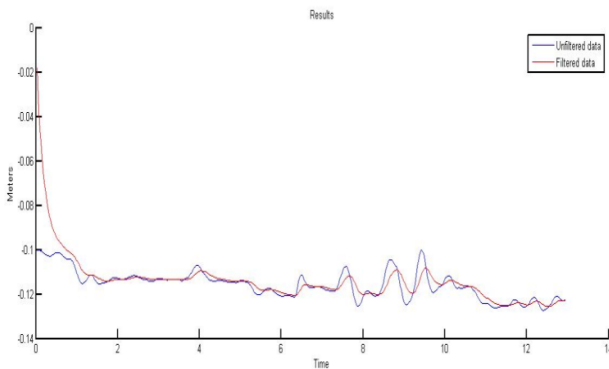


Fig. 3. Results.

### 3. Hand shape recognition using Artificial Neural Networks

Hand shape refers to the specific shape of the hand while forming an instruction. Hand Shape is a very important feature for Natural user Interface (NUI) design and Sign language (SL) recognition studies [25]. In total, there are 36 possible hand shapes according to SL investigator [26]. We use 13 hand shapes. Fig. 4 shows these hand shapes.



Fig. 4. Hand shapes.

#### 3.1 Hand shape extracting using skin color

Most of the studies like face recognition, gesture recognition and Sign language recognition systems, needs to locate and track patches of skin-colored pixels through an image sequence [27]. Generally, skin-color segmentation is the first stage of these systems. And it is used for localization of the human hands and face. Many researchers have been working on skin-color segmentation, but there are major difficulties due to the varying environmental effect such as illumination, complex background. Robust skin color pixels locating and tracing is a hard and a challenging problem under varying illumination conditions. There are some robust skin color locating and tracking studies like using luminance invariant color-spaces [28, 29]. However, some

studies used specially design rooms which keep skin color distribution within determining limits.

Skin-color segmentation approaches can be categorized into two basic classes: (i) physically based, and (ii) statistical based [30-34]. Different color spaces have been used in skin-color segmentation. Phung and his friends investigate how the choice of color space and the use of chrominance channels affect skin segmentation. They found out that there exist numerous color spaces but many of them share similar characteristics [35].

Hence, in this study, we focus on YCbCr color spaces which are commonly used in the image processing field. According to YCbCr color space, Colors are specified in terms of luminance (the Y channel) and chrominance (Cb and Cr channels). The transformation between YCbCr and RGB is linear. The simplicity of the transform which is given in Equation (9) and the explicit separation of luminance have made the model very attractive for skin color detection [36, 37]

$$\begin{aligned} Y &= 0.299R - 0.587G - 0.114B. & (9) \\ Cb &= R - Y. \\ Cr &= B - Y. \end{aligned}$$

As given in Fig. 5, first hands are located using skeleton data which is obtained from Kinect, then, average color values are calculated using a group of pixel around the hand joint, hand images are filtered using Gaussian filter. Then, the hand is extracted using the calculated average color, features vector are calculated using the last image.

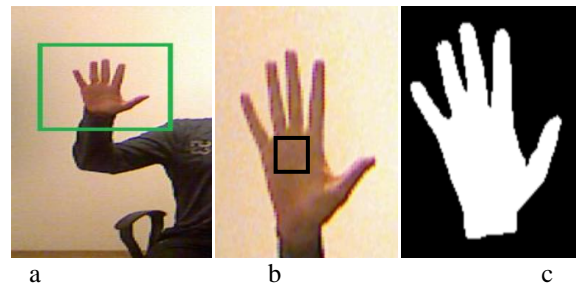


Fig. 5. a) Kinect RGB image, b) a group pixel around hand joint, c) binary hand image after passing Gaussian filter.

There are several skin color pixel classification algorithms in literature such as Gaussian classifiers [29, 38, 39], adaptive Gaussian classifiers[40], the multilayer perceptron [41], piecewise linear classifiers [36,42], the Bayesian classifier with the histogram technique [43], Fuzzy YCBCR Color Space with the Mamdani Inference [44], and genetic algorithm (GA)[45]. Some Depth Image based hand gesture recognition studies appear in literature lately [46].

#### 3.2 Moment invariant method

Moment invariants are properties of connected regions in binary images, six of these are invariant describe

translation, scaling and rotation. The seventh descriptor ensures skew invariance, which enables to distinguish between mirrored images. They can be easily calculated from region properties and they are very useful in performing shape classification and part recognition. One of the techniques for generating invariants in terms of algebraic moment was originally proposed by Hu [47]. Feature vectors used for hand shape recognition are generated using moment invariants. The moment invariants used in our research are computed using the following equations for all hand shape.

Table 1. (a) Formulas used for specific central moments  
b) List of the derived invariant moments.

Central Moments	Derived Invariant Moments
$\mu_{00}=m_{00}$	$I_1=\eta_{20} + \eta_{02}$
$\mu_{10}=0$	$I_2=(\eta_{20}-\eta_{02})^2+4\eta_{11}^2$
$\mu_{01}=0$	$I_3=(\eta_{30}-3\eta_{12})^2+(3\eta_{21}-\eta_{03})^2$
$\mu_{20}=m_{20}-x^2m_{00}$	$I_4=(\eta_{30}+\eta_{12})^2+(\eta_{21}-\eta_{03})^2$
$\mu_{02}=m_{02}-y^2m_{00}$	$I_5=(\eta_{30}-3\eta_{12})(\eta_{30}+\eta_{12})^2(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2+(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})(3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2)$
$\mu_{11}=m_{11}-xym_{10}$	$I_6=(\eta_{20}-\eta_{02})((\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2)+4\eta_{11}(\eta_{30}+\eta_{21})(\eta_{21}+\eta_{03})$
$\mu_{30}=m_{30}-3x^2m_{20}+2x^2m_{10}$	$I_7=(3\eta_{12}-\eta_{30})(\eta_{30}+\eta_{12})(3\eta_{30}\eta_{12}-3(\eta_{21}+\eta_{03})^2)+(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})(3\eta_{30}\eta_{12}^2-(\eta_{21}+\eta_{03})^2)$

A two stage ANN classifier is developed for hand shape recognition. The input layer consists of 7 nodes (p1 to p7), one for each of the moment invariant values. The output layer consists of 13 nodes (o1 to o13), each representing a different hand shape. The initial weights and biases are set to random values. The network has both the transfer functions as log-sigmoid and the output lies between 0 and 1. The target vector consists of 13 elements. All the elements are set to 0 except one, which is set to 1. The position of the element corresponding to 1 defines the hand shape.

Resilient backpropagation algorithm (TRAINRP) is used for training the classifier. Trials with different number of hidden layer nodes and different number of training samples of a single user are carried out. 20 hidden layers for TRAINRP algorithm give good performance in terms of accuracy and training time. The resilient backpropagation algorithm results are given in Figure 6 for training samples from six to 20. Some recognized hand shapes are showed in Fig. 7.

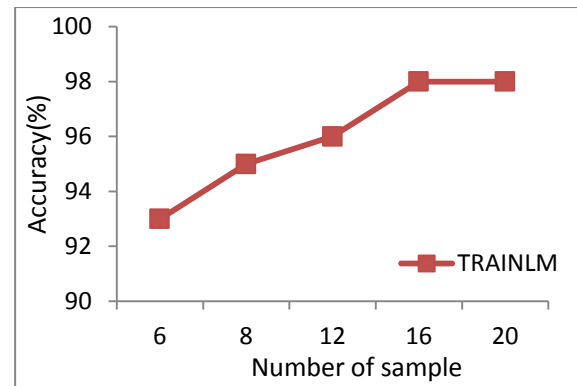


Fig. 6. Accuracy of hand shape recognition system for TRAINLM algorithm.

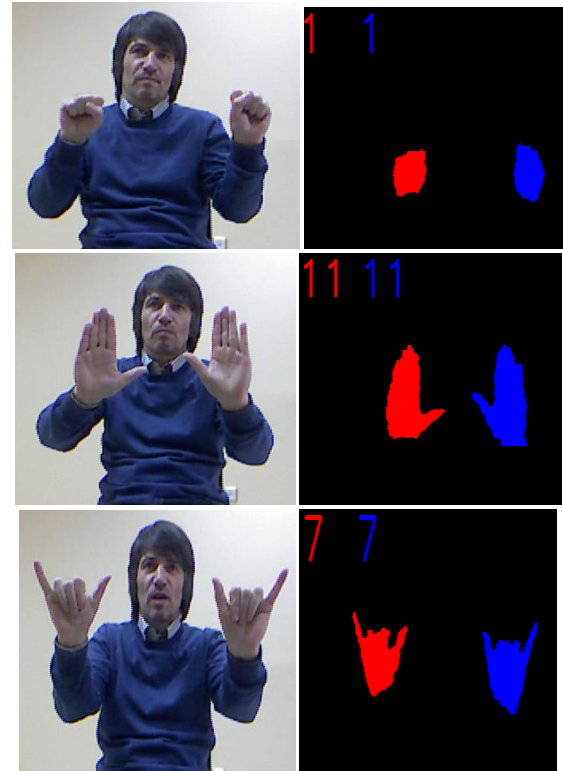


Fig. 7. Some recognized hand shapes.

#### 4. Feature extraction

The x, y, z values of twenty body joints are captured using the Kinect sensor. User's hands and face are located by using hand joints with YCbCr color space, and depth image values. Kinect sensor data first passes through Kinect filter, and then Kalman filter is used to obtain joints data. The names of these joints are given in Fig. 8. In this study, four feature vectors are extracted and used. These are hand shape, wrist location, wrist movement and distance between right hand wrist and left hand wrist.

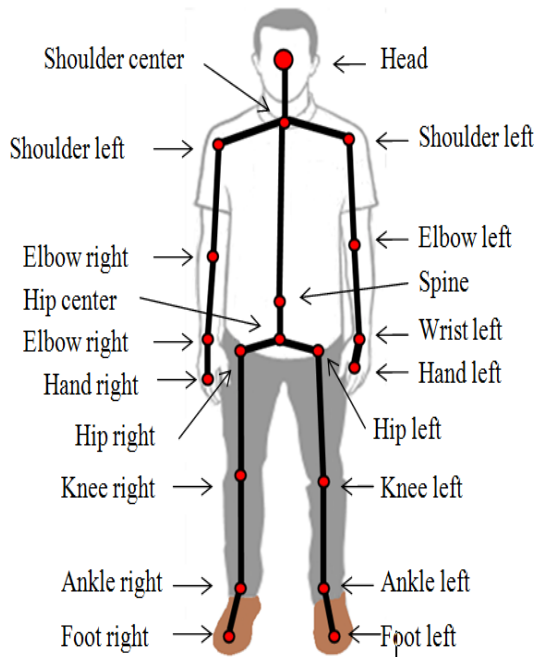


Fig. 8. Human skeleton joints tracked by Kinect.

Speed of gestures may vary from one person to another. This means that the size of the collected feature vector data of human gesture depends on the persons' gesturing speed. This is not desired in classification of gesture since artificial neural network classifier demands certain number of input. Therefore histograms of sign feature vectors are used to get a certain amount of input data the classifier.

A histogram is a graphical representation showing a visual impression of the distribution of data. It is an estimation of the probability distribution of a continuous variable [1]. Basically, the range of possible values is segmented into sub ranges, and then the number of instances of each sub range is counted. The histogram can be calculated in the following way: Let  $d$  be the number of divisions we wish to segment the range into and let  $h_i$ ,  $0 \leq i \leq d$ , be the "columns" of the histogram. The following equations show how a histogram can be calculated.

$$h = \sum_{j=0}^n \frac{1}{n} r_i(x_j), \quad \forall i, \quad 0 \leq i \leq d \quad (10)$$

$$r_i(x_j) = \begin{cases} 1 & \text{if } \frac{i(x_{\max} - x_{\min})}{d} \leq x_j < \frac{(i+1)(x_{\max} - x_{\min})}{d} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

We gave below, how four feature vectors were extracted and converted to histogram.

**Hand shape:** as given section 3, we designed an ANN Hand shape classifier to classify the hand image feature data that are obtained from the Kinects' RGB camera for each hand. We trained and tested our models for both single user and multi users. Right hand shape and left hand shape classifier accuracies are 96%, and 94% respectively. Two hand shape histograms are created using all results of the ANN hand shape classifier during the instruction signing period. The Histograms' data were used as a feature vector. Fig. 9 shows the right hand-shape histogram for action "while left hand is staying close to the body, right hand starts in front of the body and parallel to ground with open palm hand shape, arrives as a closed palm hand shape beside the body"

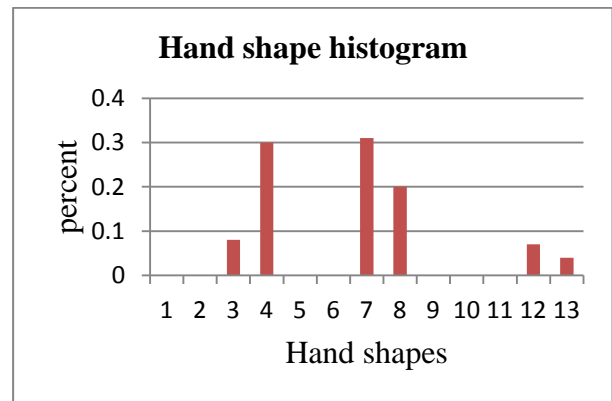


Fig. 9. An example of right hand shape histogram.

**Hand wrist location:** Hand location is an important feature for human motion analysis, gesture recognition and NUI systems. We designed parametric and flexible virtual rectangular prisms to determine user's both hand wrist positions which are based on four skeleton joints; head, left shoulder, right shoulder and spine joints. Shoulder center joint is determined as a body center. Before the integration hand wrists location to the project, we designed an interface for hand shape location. Instruction signing space is divided into 57 regions, which are cube shaped, for each hand by using four skeleton joints. According to our system, x coordinate is divided to five element; far left, left, middle, right, far right, and y coordinate is divided in to three element; top, middle, bottom, same as z coordinate is divided in to four element; back, front, far front, and farthest front. We reduced the number of regions from 57 to 14 for each hand. In total there are 28 regions for both hands. Fig. 10 shows the hand wrist location determine user interface and created regions around the body. Histograms of hands wrist locations are used for classification.

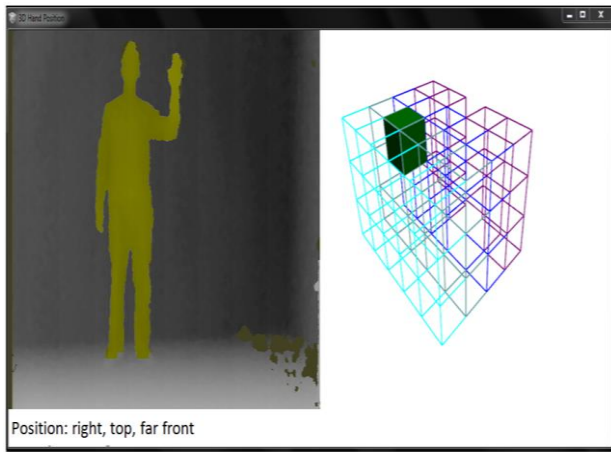


Fig. 10. Features for Right hand wrist position.

### Hand wrist movement:

The term movement describes the action of the hand wrist joints, such as moving in a circle, up and down, forward or backward. In this study, movement features are produced by calculating differences between present and previous coordinate values of every time instant. In our study, Hand wrist movement histogram was used. There are three axes and each has three states: zero, minus, and positive 27 features in total. We used threshold values to filter small changes, under determined threshold values, to classify them as a zero state. Fig. 11 shows the right hand wrist movement histogram for action “while left hand is staying close to the body, right hand starts in front of the body and parallel to ground with open palm hand shape, arrives as a closed palm hand shape beside the body”

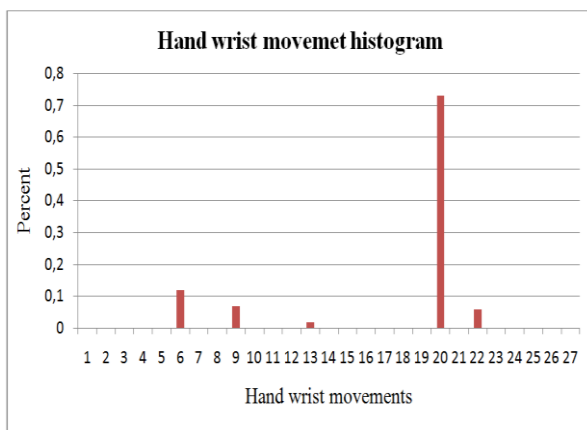


Fig. 11. An example right Hand wrist movement histogram.

### Distance between left hand wrist and right hand wrist

Another feature is distance between left hand wrist and right hand wrist distance. During the user action period, right hand and left hand are moving either symmetrically or asymmetrically depending on the

characteristic of the action. Distance values between right hand wrist and left hand wrist are calculated for each time cycle. The distance between two points on the three dimensions of the xyz-plane can be calculated using the following equation.

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (12)$$

Where point  $(x_1, y_1, z_1)$  is the right hand wrist joint, and point  $(x_2, y_2, z_2)$  is the left hand wrist joint on the 3D Cartesian coordinate. The distance histogram is segmented into 6 sub ranges between -30cm to 120 cm. Sub ranges values are (-30 to -15, -15 to 0, 0 to 15, 15 to 30, 30 to 60, and 60 to 120) respectively. If the distance value is bigger than 120, we consider it in sub range determining numbers between 60 and 120. There is also one more similar assumption. If the distance value is less than -30, we consider it in sub range determining values between -30 to -15. Hand distance value is more important when the hands come close. Therefore, sub ranges are not the same length.

## 5. Negative Correlation Learning for Artificial Network ensembles

Ensemble of learning machines is a collection of classifiers which are trained for solving the same task. There are many ensemble methods in the literature. One of them uses a penalty term in the error function to produce biased individual learners whose errors tend to be negatively correlated. It is called **Negative Correlation Learning (NCL)** algorithm [48].

NCL is a collective learning technique that introduces a correlation penalty term to the error function of each singular neural network in the NN ensembles. All individual networks can be trained interactively with the same training data set. Liu et al. was the first to propose NCL [49]. Thus, best result is obtained [50].

Given training set  $\{(x(1),d(1)),(x(2),d(2)), \dots, (x(N),d(N))\}$ ,  $x \in \mathbb{R}^+$ ,  $d$  is a target output and  $n$  is number of training patterns. The equation of NN average of results as given in equation 13.

$$F(n) = \frac{1}{M} \sum_{i=1}^M F_i(n) \quad (13)$$

$M$  is a number of individual NNs in the ensemble;  $F_i(n)$  is  $i^{\text{th}}$  networks output in  $n$ . training.  $F(n)$  is an ensemble output in  $n^{\text{th}}$  training. Thus whole networks are trained at the same moment and are combined with the same training set.  $E_i$  is a error function for  $i^{\text{th}}$  networks,

$$\begin{aligned} E_i &= \frac{1}{N} \sum_{n=1}^N E_i(n) \\ &= \frac{1}{N} \sum_{i=1}^N \frac{1}{2} [F_i(n) - d(n)]^2 + \frac{1}{N} \sum_{n=1}^N \lambda \rho_i(n) \end{aligned} \quad (14)$$

between  $F_i(n)$  and  $d(n)$ . That parameter  $\lambda$  is used range of  $0 \leq \lambda \leq 1$  determines errors strength. Error function  $p_i$  in equation 15;



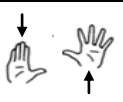
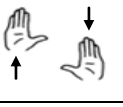

$$\rho_i(n) = (F_i(n) - F(n)) \sum_{j \neq i} (F_j(n) - F(n)) \quad (15)$$

The penalty term explicitly encourages the  $i^{\text{th}}$  NN to be Negatively Correlated in the whole NNs ensemble. Thus, diversity among the individual NNs is achieved. NCL has shown a number of experimental successes various applications [52].

### 6. Design of human gesture recognition system with ANN

Some human gestures used in training and testing set are given in Table 2. When we look at the explanation, we see the gesture begins with hand shapes in a start position and continues with changing hand shapes and positions.

Table 2. Some example command in training and testing set.

Gestures	Command	Definition
	Slow down	While Left hand stays constant with <i>hand shape 3</i> which is a palm facing front, right starts as <i>hand shape 7</i> , palm facing ground, moves from the front to the body, while opening until all its fingers are opened <i>hand shape 4</i> .
	Speed up	While right hand stays constant with <i>hand shape 3</i> which is palm facing front, left hand starts as shape 4, palm facing front, moves from front to body, while closing until all its fingers touch around its thumb.
	Turn right	both hands start as <i>hand shape 3</i> , palms facing front, while left hand moves up, right hands move down
	Turn left	both hands <i>hand shape 3</i> , palms facing front, while right hand left hand moves up, left hand move down
	next	Bot hands hand shape 3, palms facing front, while right hand staying stable, left hand moves left

A multi-layer ANN with Negative Correlation Learning is designed to recognize a set of human gestures. The gesture classification ANN model is trained and tested of line with Matlab program. The ANN classifier parameters are stored in a text file for future use with the real-time gesture classification module. The extracted feature vectors of user gestures, which are produced from data collected during the gesturing period, are inputs to the network. The gesturing period is the time period over which the output of trigger block is 1. At each time interval, the skeleton joint data from Kinect and

recognized hand shape are recorded. These data consist of 20 joints data form Kinect, as well as, recognized hand shape with ANN using image form Kinect RGB and depth cameras. Before we produce features vectors as mentioned in section 4, we pass the skeleton data through Kalman filter for every gestures, and a total of 110 features are extracted; 26 hand shapes histogram values 13 for each hand, 54 hand wrist movements histogram values, 27 for each hand wrist, 28 hand location histogram values 14 for each hand, and 6 distance histogram values between right hand and left hand. We used three ANN model, and weights of each individual network are randomly given different from each other. Also, learning coefficients are given independently from each other (0.1, 0.2, and 0.5). Theoretically, when network recognizes the whole gestures, the training ends. However, in practice stopping criteria (mean square error) is necessary to determine. The stopping criterion is 0.00001 in this study. The training continues until mean square error is 0.00001.

A Multi-layer feedforward ANN model classifies the user movements to 16 classes such as slow down, speed up, turn right, turn left, etc. The ANN classifier model has 110 inputs, 50 hidden and 16 output neurons. The system is designed to recognize a gesture as a whole at one time. An NCL algorithm is used for training. The Network output consists of 16 outputs, each representing one gesture, and commonly used in literature. Fig. 12 shows the human motion classification module block diagram. When the real-time motion classification module starts, first the ANN classifier data are read from the data file and then a real-time motion recognition loop starts.

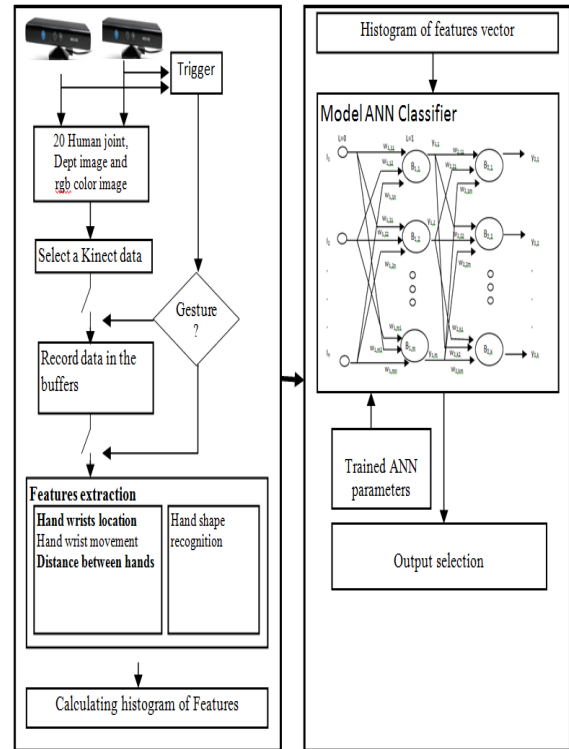


Fig. 12. The Real-time motion classification module block diagram.



## 7. Test results

The human gesture recognition system was trained with six, ten, sixteen and twenty samples of data with 16 human gestures, at the testing stage, real-time data were also used. In total, 160 actions (16×10) in the training set are used for the test. Training and testing data are acquired from several people while they try to perform the same gesture. Therefore, data from different people could vary. The testing results are given in Table 3. In this table, the ‘unknown’ come from the output selection procedure, as mentioned before, the system is designed to recognize a gesture as a whole at one time. But, we used a threshold value for output, first system find out the maximum output if the output is under the determined threshold value than it was regarded as an unknown. If the threshold value is increased, unknown gestures will also increase; otherwise, when the threshold value is decreased, the number of misrecognized signs will increase.

Table 3. ANN test results for Human motions recognition.

Number of training sample	Test data			
	6	10	16	20
Total test data	160(10x16)	160(10x16)	160(10x16)	160(10x16)
Missed (unknown)	20	19	13	10
Misrecognized	12	11	8	6
Recognized	128	133	139	144

## 8. Application

In the last decay there has been a great emphasis on Human-Computer Interaction research to create easy-to-use interfaces by facilitating natural communication and manipulation skills of humans. Adopting human gesture as an interface in HCI will allow the deployment of wide range of applications, such as virtual reality system, interactive gaming platform, and system with touch less interfaces. Now we propose using human gesture as an interface for windows based games and application.

The developed software is complicated and it consists of four modules. It is not practical for real time windows application control. Therefore a single alone user interface program is developed for real-time human gesture classification using trained network parameters and data. The developed application tracks the skeleton and hands of the human using the Kinect device. When the skeleton joints data are captured from Kinect device using the Microsoft Kinect SDK, Hands are located using Kinect RGB camera images. MLNN classifies the hand shape using hand shape feature vector. In order to recognize a human gesture, four feature vectors are extracted from the skeleton joints data and hands data; these are hand shapes, hand wrist location, hand wrist movement, and distance between hands. Features vectors are classified with an ANN classifier to determine the human gesture. The developed application converts the human gesture into the keyboard direction keys and mouse button functions. As shown in Fig. 13. In order to control an application with

the developed interface program, first, users have to execute the pc game or any application for windows, and then they need to make a connection between the program and the developed interface program by using the *connection tool* of the developed interface program. The connection can easily be made by dragging the find button with the mouse and dropping it on the running windows application. A connection is automatically form by the program. When a human gesture is recognized, NUI application produces the determined keyboard, mouse or special control for the program. As shown in Fig. 14, System has been tested with games and windows application. Interface program shows the captured video stream data and Kinect skeleton data in an image box, depending on the user’s chose, there are two more user option tools on the interface. One of them is showing classification result with a text, the second one is showing classification result with a graphical way. If check box labeled as a “Use recognized gesture as a NUI“ of the interface is not checked, keyboard and mouse control signal are not produced. The program also has a gesture recognition module based on captured images from a webcam which was published earlier [53].

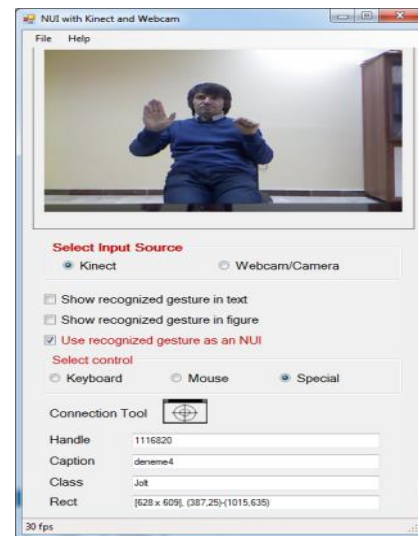


Fig. 13. Graphic user interface of the application.

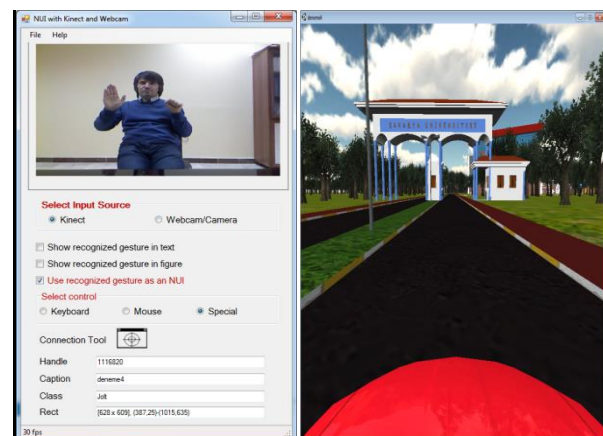


Fig. 14. An example Windows games control with NUI application.

## 9. Conclusion

The development and evaluation of a human gesture classification system is described in this paper. The data from two Kinects are processed by a noise reduction, feature extraction, and a classification network. Four important features are extracted for every human action and histograms of the feature vectors are used. Neural networks are used as a classifier of these feature vectors. Our developed and implemented System is capable of recognizing the human motions in real time. But, when users sign an instruction, they have to wait the system processing time. When users see the recognized instruction, system produces a control output. A NUI application is designed to control windows program and games using the developed system. The system is trained and tested, and the test results indicate that the recognition accuracy is about 83%.

## Acknowledgments

This research was partially supported by Sakarya University Scientific Research Project (BAP) awarded (2012-12-10-002) in Turkey.

## References

- [1] C. Oz, M. C. Leu, *Engineering Applications of Artificial Intelligence*, **24**, 1204 (2011).
- [2] Z. Q. Zhang, W. C. Wong, J. K. Wu, *IEEE Transactions on Information Technology in Biomedicine*, **15**, 513 (2010).
- [3] M. K. Fiaz, B. Ijaz, *Intelligent and Advanced Systems (ICIAS)*, 2010 International Conference on 2010; pg. 1 - 5.
- [4] J. Svendsen, A. B. Albu, *Workshops (CVPRW)*, 4296 (2010).
- [5] S. Ye, C. Ren, X. Wang, S. Y. Chen, *Third International Conference on Intelligent Human-Machine Systems and Cybernetics*; 224 (2010).
- [6] M. Shimosaka, Y. Sagawa, T. Sato, T. Mori, *Multimedia and Expo (ICME)*, 2010 IEEE International Conference on 167 (2010).
- [7] H. Graf, S. M. Yoon, C. Malerczyk, *Proceedings of IEEE 17th International Conference on Image Processing*, September; 223 (2010).
- [8] J. Wang, M. She, S. Nahavandi, *Digital Image Computing: Techniques and Applications*, 320 (2010).
- [9] E. Souza Santos, E. A. Lamounier, A. Cardoso, *XIII Symposium on Virtual Reality*, 112 (2011).
- [10] S. Qin, X. Zhu, Y. Yang, *J Sign Process System* **74**, 47 (2014).
- [11] Z. Ren, J. Yuan, J. Meng, Z. Zhang, *IEEE Transactions on Multimedia* **15**, 1110 (2013).
- [12] A. Jalal, M. Z. Uddin, J. T. Kim, T. S. Kim, 2011 :1-7 DOI: 10.1177/1420326X11423163.
- [13] D. J. Sturman, D. Zelter, *IEEE Computer and Applications*, 30 (1994).
- [14] C. Oz, M. C. Leu, *Neurocomputing*, **70**, 2891 (2007).
- [15] B. Freedman, A. Shpunt, M. Machline, Y. Arieli, Patent Application, US2010/0118123 A1, 2010.
- [16] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, M. Magnor, *Vision, Modeling, and Visualization*, 317 (2011).
- [17] P. B. Jo˜ao, K. Daniilidis, *The fifth Workshop on Omnidirectional Vision, Camera Networks and Non-classical cameras* 22 (2004).
- [18] J. Y. Bouguet, *Camera Calibration Toolbox for Matlab*, [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/) (2014).
- [19] Ferid Bajramovic, Marcel Brückner, Joachim Denzler, *Journal of Mathematical Imaging and Vision (JMIV)*, 1 (2011).
- [20] P. Baker, Y. Aloimonos, *OMNI'2003 – Workshop on Omnidirectional Vision and Camera Networks*. Madison, WI, June 2003.
- [21] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, M. Magnor, *Vision, Modeling, and Visualization*, 317 (2011).
- [22] *Matlab Camera Calibration Toolbox* [http://www.vision.caltech.edu/bouguetj/calib\\_doc/htmls/example5.html](http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html). (2014).
- [23] V. Lippiello, B. Siciliano, L. Villani, *Control Engineering Practice*; **15**, 123 (2007).
- [24] G. Welch, G. Bishop, *SIGGRAPH*, Los Angeles, CA, August, 12 (2001).
- [25] N. Sarawate, C. Oz, M. C. Leu, *Turkish Journal of Electrical Engineering & Computer Sciences*, DOI: 10.3906/elk-1303-167.
- [26] C. Oz, M. C. Leu, *Pattern Recognition and Machine Intelligence LNCS* **3776**, 280 (2005).
- [27] T. Kuremoto, Y. Kinoshita, L. Freng, S. Wantanabe, K. Kobayashi, M. Obayashi, *Neurocomputing* **116**, 291 (2013).
- [28] N. Oliver, A. P. Pentland, F. Berard, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 123 (1997).
- [29] J. Yang, L. Weier, A. Waibel, *Proc. Asian Conf. Computer Vision*, 1998; II: 687-694.
- [30] X. Zhu, J. Yang, A. Waibel, *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, 446 (2000).
- [31] K. Schwerdt, J. L. Crowley, *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, 90 (2000).
- [32] M. Soriano, B. Martinkauppi, S. Huovinen, M. Laaksonen, *Proc. Int'l Conf. Pattern Recognition*, **1**, 839 (2000).
- [33] W. Hafner, O. Munkelt, *Pattern Recognition and Image Analysis*, **7**, 47 (1997).
- [34] Y. Raja, S. J. McKenna, S. Gong, *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, 228 (1998).
- [35] S. L. Phung, A. Bouzerdoum, D. Chai, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27 (2005).
- [36] D. Chai, K. N. Ngan, *IEEE Trans. Circuits and Systems for Video Technology* **9**, 551 (1999).
- [37] C. Liu, *IEEE Transactions on Pattern Analysis and*

- Machine Intelligence, **25**, 725 (2003).
- [38] H. Greenspan, J. Goldberger, I. Eshet, Pattern Recognition Letters, **22**, 1525 (2001).
- [39] B. Menser, M. Wien, SPIE Visual Comm. and Image Processing; **4067**, 731 (2000).
- [40] R. Hassanpour, A. Shahbahrami, Wong Stephan, International Journal of Electrical and Electronics Engineering, **2**, 8 (2008).
- [41] S. L. Phung, D. Chai, A. Bouzerdoun, Proc. INNS-IEEE Int'l Joint Conf. Neural Networks, **4**, 2844 (2001).
- [42] C. Garcia, G. Tziritas, IEEE Trans. Multimedia, **1**, 264 (1999).
- [43] M. J. Jones, J. M. Rehg, Int'l J. Computer Vision, **46**, 81 (2002).
- [44] M. S. Iraj, A. Yavari, American Journal of Scientific Research, 131 (2011).
- [45] L. K. Lee, S. Y. An, S. Y. Oh, WCCI 2012 IEEE World Congress on Computational Intelligence 2012.
- [46] H. M. Zhu, C. M. Pun, Ninth International Conference on Computer Graphics, Imaging and Visualization, 49 (2012).
- [47] Z. Huang, J. Leng, Proceedings of 2nd International Conference on Computer Engineering and Technology (ICCET), Chengdu, China, 476 (2010).
- [48] H. Chen, X. Yao, IEEE Transactions on neural networks, **20**, 1962 (2009).
- [49] Y. Liu, X. Yao, T. Higuchi, IEEE Transactions on Evolutionary Computation, **4**, 380 (2000).
- [50] Y. Liu, X. Yao, IEEE Transactions on systems, man, and cybernetics Part B: Cybernetics, **29**, 716 (1999).
- [51] B. Kir, C. Oz, A. Gulbag, 20th signal processing and communications applications conference, 2012, 1-4.
- [52] F. F. Navarro, P. A. Gutierrez, IEEE Transactions on Neural Networks and Learning Systems, **11**, 1836 (2013).
- [53] S. Hizal, C. Oz, TOK, August, 217 (2010).

---

\*Corresponding author: coz@sakarya.edu.tr