Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 384865, 12 pages
http://dx.doi.org/10.1155/2013/384865

Research Article

A Multilayer Hidden Markov Models-Based Method for Human-Robot Interaction

Chongben Tao and Guodong Liu

Key Laboratory of Light Industry Process Advanced Control, School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China

Correspondence should be addressed to Chongben Tao; tom1tao@163.com

Received 31 May 2013; Revised 6 August 2013; Accepted 8 August 2013

Academic Editor: Vishal Bhatnaga

To achieve Human-Robot Interaction (HRI) by using gestures, a continuous gesture recognition approach based on Multilayer Hidden Markov Models (MHMMs) is proposed, which consists of two parts. One part is gesture spotting and segment module, the other part is continuous gesture recognition module. Firstly, a Kinect sensor is used to capture 3D acceleration and 3D angular velocity data of hand gestures. And then, a Feed-forward Neural Networks (FNNs) and a threshold criterion are used for gesture spotting and segment, respectively. Afterwards, the segmented gesture signals are respectively preprocessed and vector symbolized by a sliding window and a K-means clustering method. Finally, symbolized data are sent into Lower Hidden Markov Models (LHMMs) to identify individual gestures, and then, a Bayesian filter with sequential constraints among gestures in Upper Hidden Markov Models (UHMMs) is used to correct recognition errors created in LHMMs. Five predefined gestures are used to interact with a Kinect mobile robot in experiments. The experimental results show that the proposed method not only has good effectiveness and accuracy, but also has favorable real-time performance.

1. Introduction

Human-Robot Interaction (HRI) is one of the hottest research directions in robot field, and activity recognition is one of the most important markers of HRI. As gesture recognition is a branch of activity recognition, methods used in activity recognition are also suitable for gesture recognition. Generally, depending on sensor types, gesture recognition is divided into two control modes: motion sensor-based [1–3] and vision-based [4, 5]. Both of them have their own characteristics, where gesture recognitions based on motion sensors are easy to capture the motion data of a body, but most of them need to be fixed on human's body, which may make users feel uncomfortable. While vision-based gesture recognition also has some disadvantages. For instance, illumination conditions and background scenery of an environment may cause the difficulty of recognition. Additionally, it is difficult to obtain orientation information from a frame of two-dimensional (2D) image.

According to different gesture recognition methods used in HRI, they can be divided into four categories, which include generative method, discriminative method, heuristic analysis method and combinations of these methods [6]. Generative method uses generative models for randomly generating observed data, typically given some hidden parameters. It specifies a joint probability distribution over observation and label sequences [7]. Generative method attempts to build a complete description of the input or data space, usually with a probabilistic model. Hidden Markov Models (HMMs) is one of the most popular generative methods, which is a probabilistic model with a specific structure. Once the model is established, it can be learned from data and used to analyze the data. Discriminative method analyzes features extracted from sensor data or segments without considering sequential connections in the data. Common discriminative methods include Neural Networks (NNs) [8], Conditional Random Fields (CRFs) [9] and Support Vector Machines (SVM) [10]. Heuristic analysis method is based on the direct characteristic analysis and description of the data from sensors [11]. However, each individual has different characteristics. Thus, this method is difficult to find a consistency way for observation. Since generative method and discriminative method are both machine learning algorithms, their parameters can be training by using different types of individual

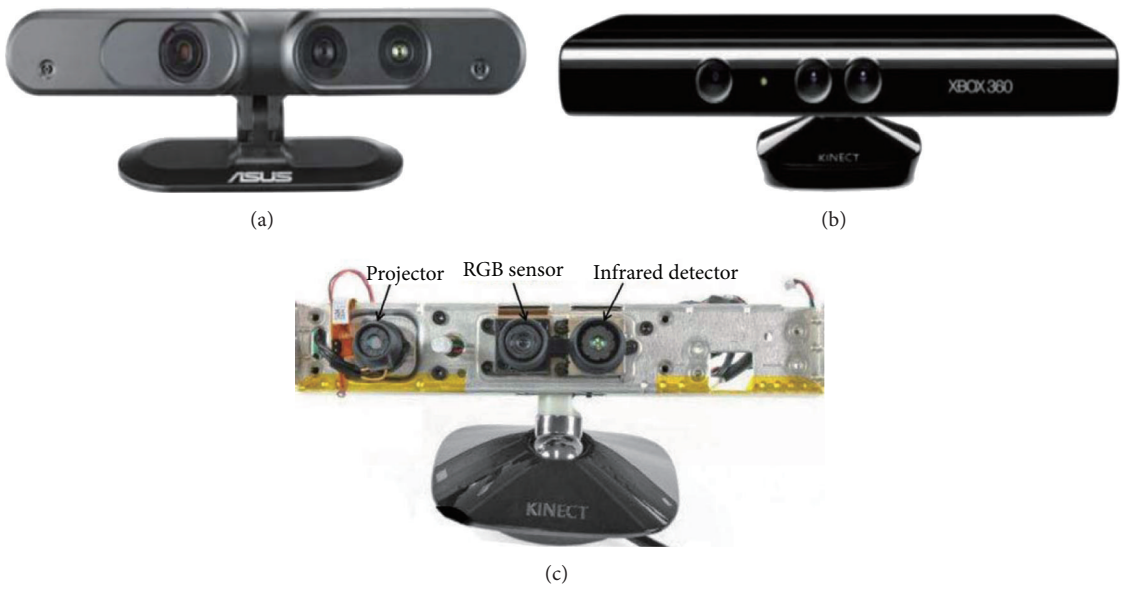

Figure 1: (a) Xtion Pro Live sensor, (b) Kinect sensor, and (c) Internal structure of Kinect sensor.

data. However, one drawback of generative approach is that enough data must be available to learn the complete probabilistic representations that are required. Moreover, both of these two types of methods have the problem of high-computational complexity. Currently, researchers have tried to integrate the advantages of different methods to address complex activity recognition problems [12].

In recent years, a new type of RGB-Depth (RGB-D) sensor has appeared, which is not only able to extract RGB feature and depth information with position and direction, but is also able to obtain three-dimensional (3D) acceleration and 3D angular velocity information of the body's motion. Asus Xtion Pro Live sensor and Microsoft Kinect sensor are two types of famous RGB-D sensors [13], as shown in Figures 1(a) and 1(b), respectively. The internal structure of a Kinect sensor is shown in Figure 1(c). Since the advent of Kinect sensor in November 2010, researchers have used it to do extensive researches in the field of Human-Robot Interaction (HRI). Giraldo et al. [14] proposed an approach based on machine learning to identify predefined gestures based on depth images obtained from a Kinect sensor. Additionally, they also adopted a kernel-based learning method to discover the relationship among gestures and samples, which can be used to correctly describe gestures. Ren et al. [15] detected the shape of a hand from color images and depth images obtained from a Kinect sensor and then proposed a shape distance metric method called Finger-Earth Mover's Distance for dissimilarity measurement. Finally, they recognize predefined gestures by template matching. Bauer et al. [16] proposed a Human-Robot Interaction system. Firstly, a person makes a predefined gesture, and then a robot try to understand the action and respond accordingly in realtime.

To overcome the shortcomings of large computation and poor real-time performance, a continuous gesture recognition method is proposed for HRI. The rest of the paper is organized as follows: Section 2 describes a Feed-forward Neural Networks (FNNs) for gesture spotting and the proposed Multilayer Hidden Markov Models (MHMMs) for continuous gesture recognition. And then gives the proposed continuous hand gesture recognition procedure, which contains hand spotting and gesture recognition based on FNNs and MHMMs. The experimental results are shown in Section 3. Finally, Section 4 shows the summary and then the acknowledgments.

2. Mhmms-Based Continuous Gesture Recognition

In this paper, the proposed gesture recognition algorithm consists of two parts: the gesture spotting and segment module and the continuous gesture recognition module. An important problem in gesture recognition is how to distinguish gestures and nongestures, which is called *gesture spotting problem*, namely, how to detect a start point and an end point of a gesture [17]. Firstly, a Feed-Forward Neural Networks (FNNs) is used to detect gestures in an action signal, and then a threshold criterion is used to segment gestures from the signal. Finally, MHMMs is used to recognize segmented continuous gestures.

FNNs are firstly used to distinguish whether a particular action is a gesture. When the FNNs determine a certain action is a gesture, then the output is 1, otherwise the output is 0. Features can be well combined together in FNNs for classification of gestures and nongestures. Additionally, through training, the weights and deviations of FNNs can be optimized. Two counters are used to detect a start point and an end point of a gesture in accordance to a preset threshold value. When the end point of a gesture is detected, the spotting and segment module will trigger the recognition module. As MHMMs is a probabilistic model which has a high computational complexity, FNNs-based segment module will be used as a switch to control data flows, thus computation time can be

saved and the real-time performance of the whole algorithm can be approved. Sequential constraints among gestures will be modeled by MHMMs in recognition module. Forward propagation program in the Bayesian filtering will be used to produce a posterior gesture decision and update the recognition result. In this paper, the flow chart of MHMMs-based continuous gesture recognition method is shown in Figure 2.

2.1. Gesture Spotting and Segment. In this paper, a Feed-forward Neural Networks (FNNs) is designed to discover gesture data from signals [18]. The structure of a three-layer FNNs module is shown in Figure 3. The number of input nodes is n, the input variable of the input layer is: $Y = (y_1, y_2, \ldots, y_n)^T$, which indicates an original input signal of actions; the output vector of hidden layer is $a_i^2 = (a_1^2, a_2^2, \ldots, a_m^2)^T$; the output vector of output layer is $a_i^3 = (a_1^3, a_2^3, \ldots, a_m^3)^T$, which means a gesture spotting result; the weight matrix from the input layer to the hidden layer is $W_i^1 = (W_1^1, W_2^1, \ldots, W_m^1)$; the weight matrix from the hidden layer to the output layer is $W_i^2 = (W_1^2, W_2^2, \ldots, W_m^2)$; and Sigmoid function $f(x) = (1 + e^{-x})^{-1}$ is used as an excitation function.

A 3D acceleration $\alpha_i = [\alpha_x, \alpha_y, \alpha_z]^T$ is one kind of data extracted by RGB-D sensor, which is trained by FNNs. Let a_i^3 be the output of FNNs:

$$a_i^3 = \text{hard lim}\left(f^2\left(W^2 f^1\left(W^1 \mu_i + b^1\right) + b^2\right) - 0.5\right), \quad (1)$$

where W^1, W^2, b^1, and b^2 are parameters of FNNs.

Both of the network functions f^1 and f^2 are log-sigmoid functions; function hard lim is a hard limiting function. Therefore, parameters of FNNs can be trained by a back-propagation method. In the output layer, set $W^3 = 1$ and $b^3 = 0$ in order to produce a discrete output. The output of FNNs is a binary number (1 or 0), which, respectively, represents a gesture or a nongesture. The first layer and the second layer constitute a two-layer feed-forward network which can be obtained optimized parameters through training. And then, weights and deviations are fixed in the output layer in order to produce a discrete output.

Moreover, through training the neural networks, weights and deviations of the FNNs can be optimized. Two digital counters are used for continuous recording of the output of FNNs. When both of them exceed a preset threshold value, the start point and the end point of a gesture can be detected, thereby erroneous classification of the FNNs can be prevented. When an end point of a gesture is inspected, the spotting and segment module will trigger the recognition module.

2.2. Multilayer Hidden Markov Models. In our daily life, people usually use some gestures to command pets. And these gestures are corresponded to some reasonable modes which reflect sequential restrictions among continuous gestures. Therefore, these restrictions can be used to improve the accuracy of gesture recognition. According to this inspiration, this paper proposed a continuous gesture recognition algorithm based on MHMMs which is a statistical model derived from HMMs. HMMs is a famous statistical model for continuous data recognition, which has been widely used in speech recognition, handwriting recognition, and pattern recognition [19]. It is characterized by a set of parameters $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, where $\mathbf{A}$, $\mathbf{B}$ and $\boldsymbol{\pi}$ are a state transition probability distribution, a probability distribution of observation symbols in each state and an initial state distribution, respectively. In this paper, a forward-backward algorithm is used to estimate a series of likelihood values of observation $\mathbf{P}(\mathbf{O} \mid \boldsymbol{\lambda})$ when given a specific HMMs [20]. For a given observation sequence $\mathbf{O}$ in the test mode, a Viterbi algorithm is used to find the best state sequence $\mathbf{Q}$ [21]. An Expectation Maximization (EM) method is used to train the parameters of HMMs [22].

The proposed MHMMs are a kind of generalization for a segmented model, where each segment has subsegments. MHMMs are divided into two layers, where LHMMs are used to identify individual gestures in the continuous gesture signal, and a Bayesian filter with sequential constraints in UHMMs is used to correct recognition errors created in LHMMs. Figure 4 shows the basic idea of MHMMs. A time sequence is layered into several segments, where S_i^1 represents the state sequence of UHMMs and S_i^2 represents the state sequence of LHMMs. S_i^2 is the sub-HMMs of state sequence S_i^1. $\mathbf{O}_i^2$ means the decision of LHMMs.

2.2.1. Data Preprocessing. Before MHMMs are used to training and recognition gesture data segmented by FNNs, a preprocessing step needs to be completed firstly, which includes removing high-frequency noise and finding the stroke duration of a gesture. Each raw data obtained by a Kinect sensor is composed of a 6-component vector $\mathbf{U}$: a 3D acceleration $[\alpha_x, \alpha_y, \alpha_z]^T$ and a 3D angular velocity $[\omega_x, \omega_y, \omega_z]^T$, where

$$\mathbf{U} = \left[\alpha_x, \alpha_y, \alpha_z, \omega_x, \omega_y, \omega_z\right]^T. \quad (2)$$

When the robot computer receives a set of data, as the original data contains high-frequency noise, which may affect gesture recognition, a low pass filter with a cutoff frequency of 5 Hz is used to remove the noise and meanwhile smooth the data, and then a sliding window with the length of 100 ms is used to calculate the average to remove the DC components in the 3D acceleration in time domain and generate a vector $\mathbf{w} = [d_x, d_y, d_z]^T$. Additionally, a Fast Fourier Transform (FFT) is applied to the deviation vector $\mathbf{w}$ in frequency domain, and the maximum frequency among d_x, d_y, and d_z is used as the fundamental frequency of a gesture. Finally, the data point $\mathbf{V}$ composing of a 3D angular velocity vector, a 3D acceleration vector, and a 3D deviation vector of acceleration is fed into MHMMs for gesture recognition, where

$$\mathbf{V} = \left[\alpha_x, \alpha_y, \alpha_z, \omega_x, \omega_y, \omega_z, d_x, d_y, d_z\right]^T. \quad (3)$$

Then a K-means clustering method is used to train centroids in order to quantize vectors into observable symbols. The process of symbolization converts feature vectors into a finite number of symbols, which can be used in the discrete

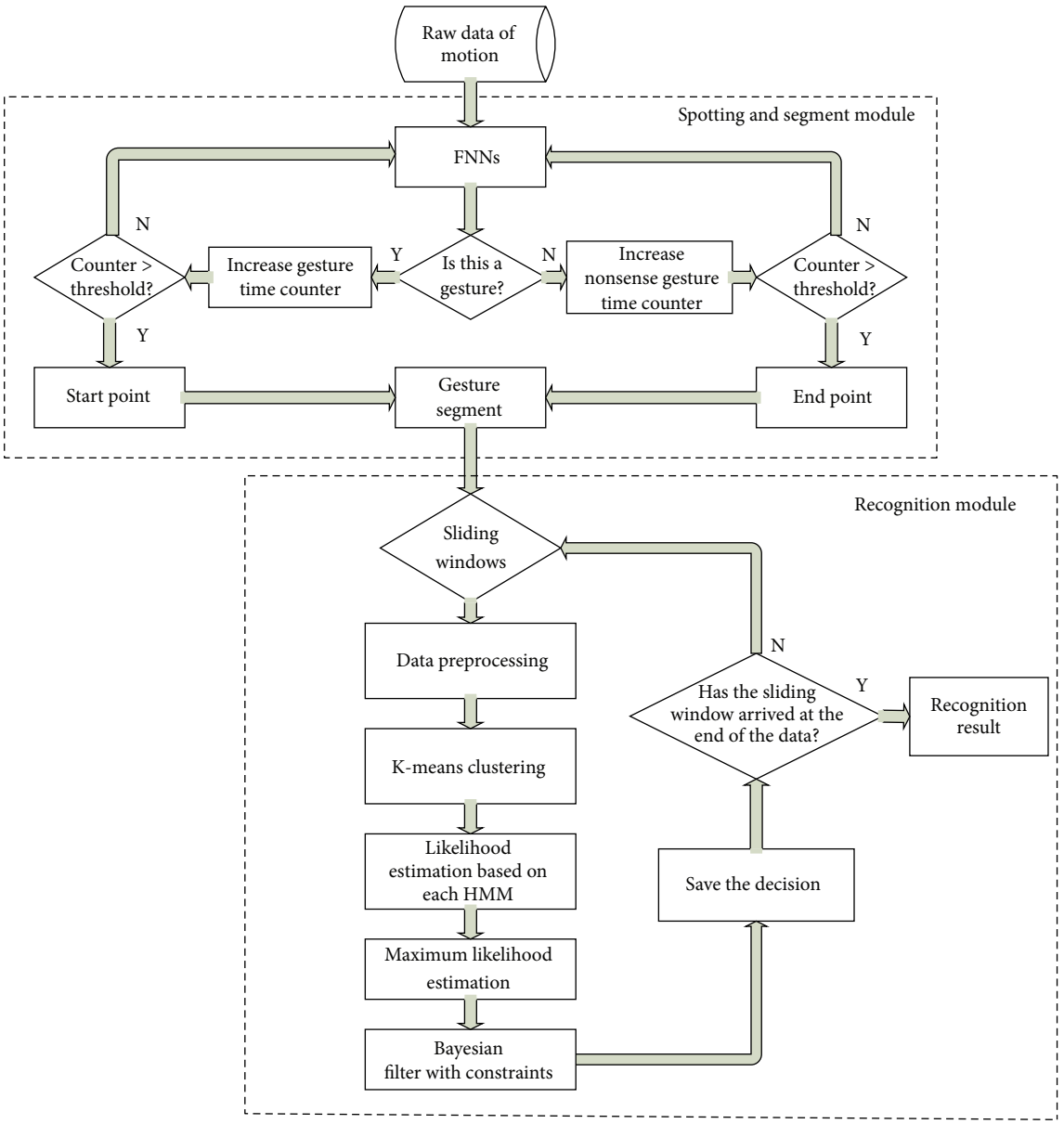

Figure 2: Flow chart of MHMMs-based continuous gesture recognition.

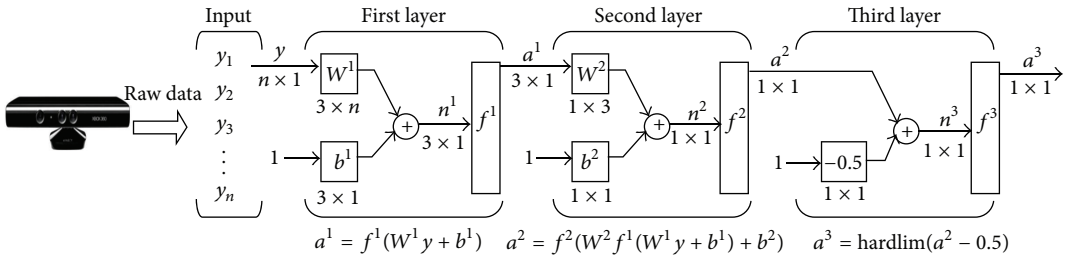

Figure 3: The structure of a three-layer FNNs.

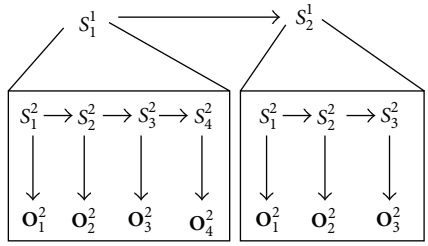

FIGURE 4: The architecture of the proposed MHMMs.

HMMs. Namely, the K-means clustering method is applied on the vector $\mathbf{V}$ to get the partition value for each vector and also a set of centroids for clustering the data into observable symbols in the recognition phase. The K-means clustering method clusters n objects into k partitions, where $k < n$, which is based on the properties of objects.

As shown in Figure 5, a sliding window of 1 second (150 data points) is moving with a speed of 30 data points each step, which is simultaneously estimating likelihood of each set of HMMs parameters. Each sliding window generates a likelihood value. Therefore, for a certain gesture, the model which maximizes the likelihood over other HMMs to be the recognized type is chosen as the output decision of the sliding window.

2.2.2. LHMMs-Based Individual Gestures Recognition

(1) Training Phase. The training phase is divided into four steps, which include finding the stroke duration of a gesture; quantifying the vectors into observable symbols; setting up initial HMMs parameters; and iterating for Expectation Maximization.

Step 1 (find the stroke duration of a gesture). FFT is used to find the stroke duration of the gesture for HMMs.

Step 2 (quantify the vectors into observable symbols). After preprocessing the raw data, these data are used for the observation sequence $\mathbf{O} = \mathbf{O}_1\mathbf{O}_2 \dots \mathbf{O}_T$ and the parameters of HMMs $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, respectively. Given parameters of the model, conditional probability of the observation sequence can be calculated. Namely, the problem is given a specific HMMs to assess the probability of an observation sequence $\mathbf{P}(\mathbf{O} \mid \boldsymbol{\lambda})$, a forward-backward algorithm is used to realize the task as follows [20]. Define the forward variables

$$\alpha_t(i) = \mathbf{P}\left(\mathbf{O}_1\mathbf{O}_2 \dots \mathbf{O}_t, q_t = S_i \mid \boldsymbol{\lambda}\right); \tag{4}$$

that is, given a model, the output observable sequence is $\mathbf{O}_1\mathbf{O}_2 \dots \mathbf{O}_T$ to the time t, and the probability of the state is S_i at time t. The forward variables can be recursive calculated by the following formula:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\right] b_j\left(\mathbf{O}_{t+1}\right), \tag{5}$$

where $1 \le t \le T-1$, $1 \le j \le N$.

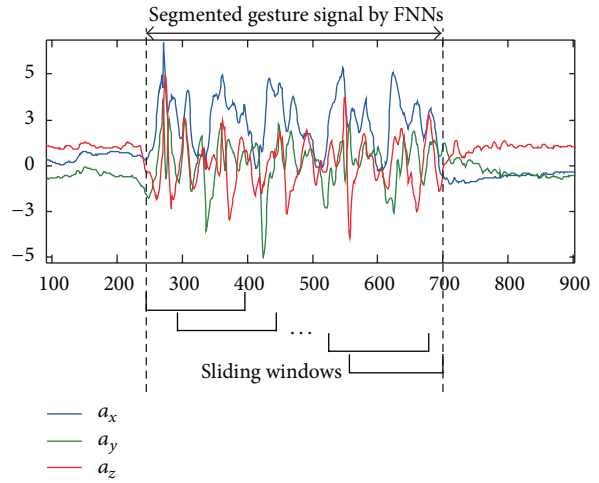

FIGURE 5: Sliding windows for preprocessing.

Define the backward variables

$$\beta_t(i) = \mathbf{P}\left(\mathbf{O}_{t+1}\mathbf{O}_{t+1} \dots \mathbf{O}_T \mid q_t = S_i, \boldsymbol{\lambda}\right). \tag{6}$$

That is, given model parameters at time t, the state is S_i. From the moment $t+1$ to the end of the sequence, the probability of the output observation sequence is $\mathbf{O}_1\mathbf{O}_2 \dots \mathbf{O}_T$. Similarly, a recursive method can be used to calculate $\beta_t(i)$ as follows:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j\left(\mathbf{O}_{t+1}\right) \beta_{t+1}(j), \tag{7}$$

where $t = T-1, T-2, \dots, 1$, $1 \le j \le N$.

Then, a K-means clustering method is applied to a 9D vector $\mathbf{V}$ to get the partition value for each vector and also a set of centroids for clustering the data into observable symbols in the recognition phase.

Step 3 (set up initial HMMs parameter). Set up state numbers in the model. The initial value of different observable symbols $\mathbf{O} = \mathbf{O}_1\mathbf{O}_2 \dots \mathbf{O}_T$ and $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ in each state need to be iterative, which should meet the randomness constraints of HMMs parameters. Then a Viterbi algorithm is used to find the single best state sequence $\mathbf{Q} = q_1 q_2 \dots q_T$ [21].

Step 4 (expectation maximization iteration). A Baum-Welch algorithm is used to calculate an auxiliary function and maximize the likelihood value of $\overline{\boldsymbol{\lambda}}$ [22]. N iterations are preceded until the likelihood approaches to a steady value. Expectation value $\mathbf{Q}(\boldsymbol{\lambda}, \overline{\boldsymbol{\lambda}})$ and the maximized likelihood $\overline{\boldsymbol{\lambda}}$ are calculated as follows:

$$\begin{aligned} \mathbf{Q}\left(\boldsymbol{\lambda}, \overline{\boldsymbol{\lambda}}\right) &= \sum_{\mathbf{Q}} \mathbf{P}(\mathbf{Q} \mid \mathbf{O}, \boldsymbol{\lambda}) \log\left[\mathbf{P}\left(\mathbf{O}, \mathbf{Q} \mid \overline{\boldsymbol{\lambda}}\right)\right], \\ \max_{\overline{\boldsymbol{\lambda}}}\left[\mathbf{Q}\left(\boldsymbol{\lambda}, \overline{\boldsymbol{\lambda}}\right)\right] &\Longrightarrow \mathbf{P}\left(\mathbf{O} \mid \overline{\boldsymbol{\lambda}}\right) > \mathbf{P}(\mathbf{O} \mid \boldsymbol{\lambda}). \end{aligned} \tag{8}$$

(2) Recognition Phase. During the recognition phase, after training a group of centroids for K-means clustering, several

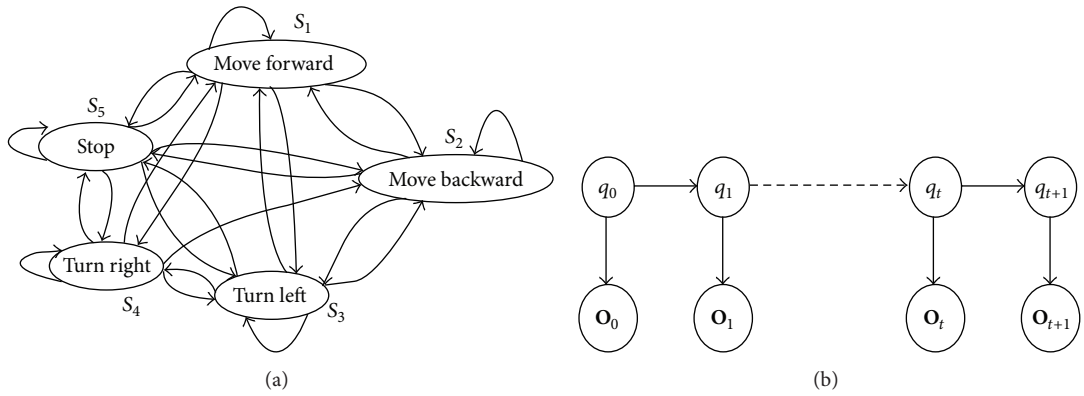

Figure 6: (a) transitions among UHMMs consider sequential constraints of context information; (b) update gesture recognition result by UHMMs.

groups of HMMs are formed. Then we calculate the likelihood of each group of HMMs parameters. Given observable sequence $\mathbf{O}$, the Viterbi algorithm is used to find the single best state sequence $\mathbf{Q}$, and the HMMs with the greatest likelihood can be identified as the most likely gesture type.

2.2.3. UHMMs-Based Continuous Gesture Update. In order to improve accuracy in decision making, we consider the sequential conditions of "Context" in UHMMs, where a Bayesian filter is used to update results from LHMMs. Therefore, five gestures are predefined, which includes "*Move forward*", "*Move backward*", "*Turn left*", "*Turn right,*" and "*Stop*". Meanwhile, the "Context" is used as sequential constraints among different types of gestures. For example, the event that someone continuously sends a same command to a robot has a small probability; someone firstly sends a "*Move forward*" command, and before he wants to send a "*Move backward*" command, he should send a "*Stop*" command. Figure 6(a) shows a transition among UHMMs, which is a discrete first-order HMMs with five states and observation symbols, respectively. UHMMs can be described as a sequence of gestures and at any time as being in one of a set of N ($N = 5$) distinct states: $S_1, S_2, \ldots, S_5$. The system undergoes a change of state according to a set of probabilities associated with the state. Each arch represents a probability of transition between two adjacent states. The moment associated with change of states is indicated as $k = 1, 2, \ldots, N$ and the kth actual state is q_k. The probability a_{ij} described below relates the current state with the previous state:

$$a_{ij} = \mathbf{P}\left[q_k = S_j \mid q_{k-1} = S_i\right]. \tag{9}$$

The initial state distribution represents the probability distribution of the first order, which is defined as

$$\pi = \mathbf{P}\left[q_1 = S_i, (i = 1, 2, \ldots, N)\right]. \tag{10}$$

Another element of LHMMs is the state probability distribution of observation symbol S_j

$$b_j(k) = \mathbf{P}\left[\mathbf{O}_k \mid q_t = S_j\right], \tag{11}$$

b_j means the probability which is identified as different observation symbols, where $\mathbf{O}_k$ represents decisions made by UHMMs.

Figure 6(b) shows a sequence in UHMMs, where the state q_t represents a tth gesture, and $\mathbf{O}_t$ is a majority decision result given by LHMMs. A Forward propagation method [23] is used to update the joint probability observation gathered until time t. In this paper, the model is defined as

$\mathbf{P}(\mathbf{O}_{t+1} \mid q_{t+1} = S_j)$ represents $b_i(\mathbf{O}_{t+1})$, where the state at time $t + 1$ is S_j and the observation is $\mathbf{O}_{t+1}$;

$\mathbf{P}(q_{t+1} = S_j \mid q_t = S_i)$ means the transition probability a_{ij} in matrix $\mathbf{A}$.

The forward variable $\alpha_t(i)$ is defined as the probability of the observation sequence $\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t$, and state S_i at time t, given the model $\boldsymbol{\lambda}$.

$$\alpha_t(i) = \mathbf{P}\left(\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t, q_t = S_i \mid \boldsymbol{\lambda}\right), \tag{12}$$

where $1 \le i \le N$.

According to the network structure shown in Figure 6(b), we can conclude

$$\alpha_{t+1}(j) = \mathbf{P}\left(\mathbf{O}_{t+1} \mid q_{t+1} = S_j\right) \times \sum_{S_i} \mathbf{P}\left(q_{t+1} = S_j \mid q_t = S_i\right) \alpha_t(i) \tag{13}$$

with the initial condition of uniform distribution

$$\alpha_0(i) = \mathbf{P}\left(q_0 = S_i\right) = \pi_i. \tag{14}$$

For the UHMMs, the training parameters $\boldsymbol{\lambda}(\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$ are obtained by observing interaction between experimenter and robot, so these parameters vary from person to person. The transition matrix $\mathbf{A}$ is estimated from the statistical results of actually observed gestures. Probability distribution of observation symbol $\mathbf{B}$ is an accuracy matrix of individual gestures from LHMMs. The forward variable is updated after having obtained the current observation value, which represents the posterior probability of a current state in the case of given context constraints of UHMMs. Therefore, the updated result is the maximum posteriori probability of the state.

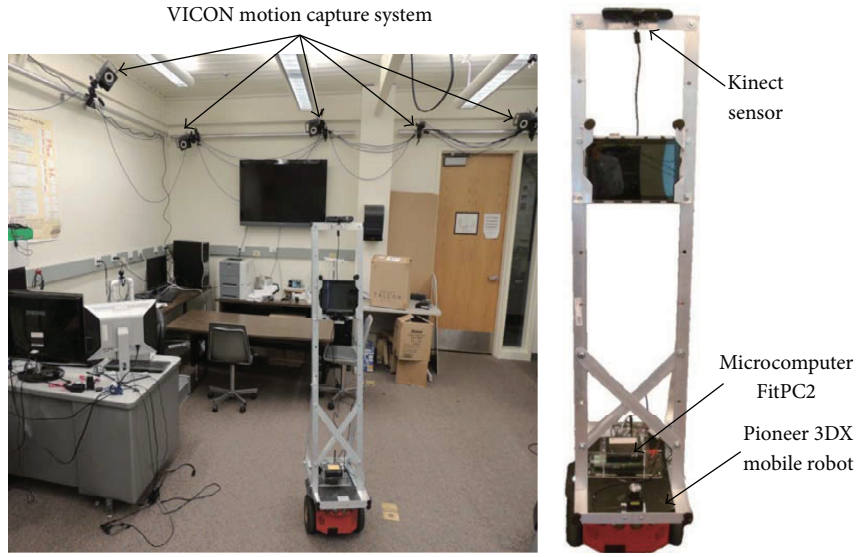

Figure 7: Experimental environment and a Kinect mobile robot.

3. Experiments

3.1. Introduction of Hardware Platform. An experimental environment where a VICON motion capture system [24] and a Kinect mobile robot platform are installed is shown in Figures 7(a) and 7(b), respectively. The mobile robot platform mainly includes: a Pioneer 3DX mobile robot, a microcomputer called FitPC2, and a Kinect sensor which is developed from a RGB camera and a depth sensor. The effective sensing range of the Kinect sensor is from 0.4 meters to 4 meters, vertical viewing angle range is ±430, horizontal range is ±570, and frame rate is 30 fps. Microcomputer FitPC2 is a fanless minicomputer which is able to install Windows or Linux operating systems, and equipped with WiFi. Besides the main components mentioned above, an external battery is used to supply power for both of FitPC2 and Kinect sensor.

Robot Operating System (ROS) is installed in the Linux operating system of FitPC2 microcomputer, which is an open source metaoperating system and provides services like a real operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [25]. In this paper, all processes are developed in accordance to the ROS framework, which includes a driver of Kinect sensor, a motion data receiver program, a motion data processing program, and a display program.

3.2. Motion Data Evaluation Based on VICON. In order to assess accuracy of the motion data captured by a Kinect sensor, a VICON motion capture system was used to evaluate. As angular velocity data is similar to acceleration data, we select angular velocity for evaluation. Firstly, several markers of the VICON system were fixed on the wrist of an experimenter, and then the experimenter made a variety of predefined gestures in front of the Kinect sensor. Finally, we sent pitch, roll, and yaw data, respectively recorded from the VICON system and the Kinect sensor into a PC, which was used to plot these data on the same graph, as shown in Figure 8. The data from VICON system was plotted by solid lines, while the data from the Kinect sensor was plotted by dotted lines. From these Figures, we find that the Kinect sensor has accurately measured motions in most of the time, except that a period of time did not match as shown in Figure 8(a), which was due to the arm swings out of the scope of the Kinect sensor. And the mismatch range between these two types of data was always less than 10 degrees. By comparison with the VICON system, we have confirmed that motion data captured from the Kinect sensor meets the accuracy requirement of gesture recognition.

3.3. Continuous Gesture Recognition Test. In order to test the performance of the proposed algorithm, we predefine five gestures for experiments. They are "*Move forward*", "*Move backward*", "*Stop*", "*Turn left,*" and "*Turn right*", where the gesture "Waving front and back" means "*Move forward*"; the gesture "Waving left and right" means "*Move backward*"; the gesture "Waving up and down" means "*Stop*"; the gesture "Swing counterclockwise" means "*Turn left*"; and the gesture "Swing clockwise" means "*Turn right*", as shown in Figure 9.

We respectively recorded 10 sets of data from three experimenters for training and testing, and these experiments were carried out in accordance to the following three steps.

Step 1. In the scope of a Kinect sensor, repeat gesture Type 1 for 15 times and take a 5 seconds break. Continue performing the rest of the types following in the same way until Type 5 is done, and then save the data.

Step 2. In the scope of a Kinect sensor, perform a sequence of 20 gestures with a break of about 2 seconds between gestures, and save the data.

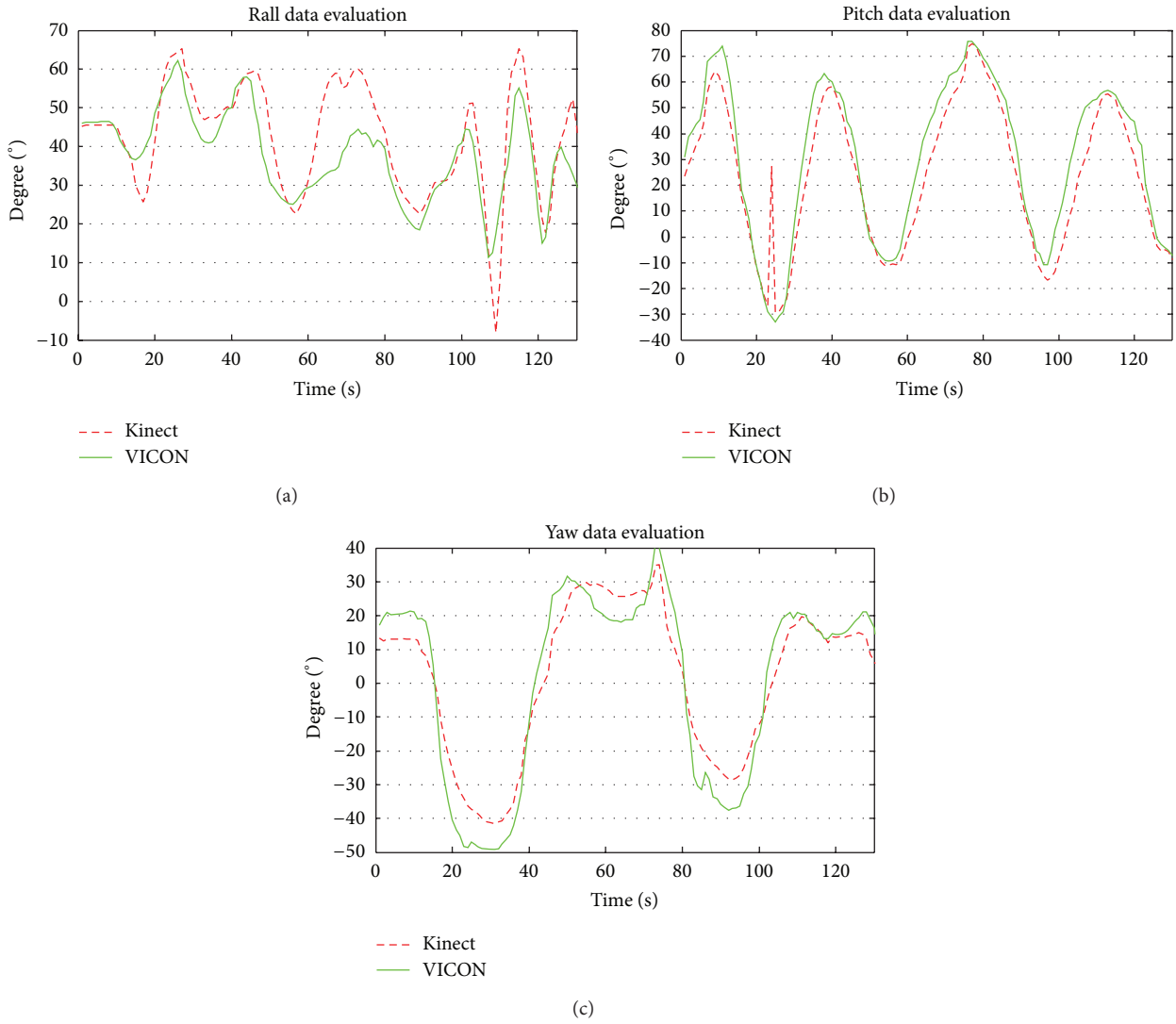

Figure 8: Comparisons of Roll (a), Pitch (b), and Yaw (c) data between a VICON system and a Kinect sensor.

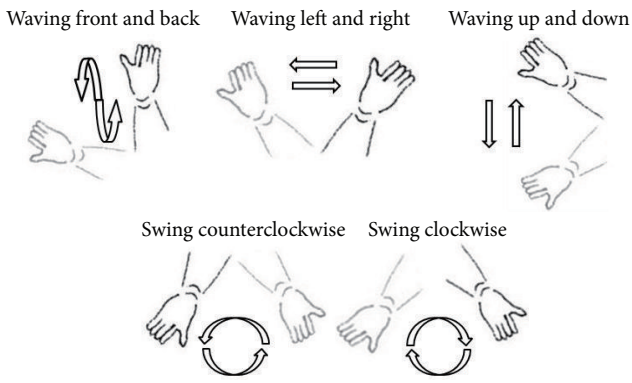

Figure 9: Five predefined gestures for Human-Robot Interaction.

Step 3. Process the training data and the testing data. Firstly, training the FNNs to distinguish gestures and nongestures; secondly, use each block of training data to train the LHMMs.

To trade off the computational complexity with efficiency and accuracy, set up the number of states 20 in the LHMM. Meanwhile set up the number of distinct observation symbols 20 and then use the trained LHMMs to recognize individual gestures in the test data. The output of each test is a sequence of recognized gestures. Finally, a Bayesian filtering with sequential constraints in UHMMs is used to generate a most likely underlying gesture sequence.

3.3.1. Performance Evaluation of FNNs. A MATLAB neural network toolbox [26] is used for training in the first and second layer of FNNs. The initial values of weights and deviations are randomly selected. Different initial values have different performances. If the performance does not reach the target, then the training phase needs to restart a new neural network until it reaches a sufficient accuracy. For instance, within 180 iterations, two different initial values got two different performances, as shown in Figure 10, which gave two results of both good and bad neural network training.

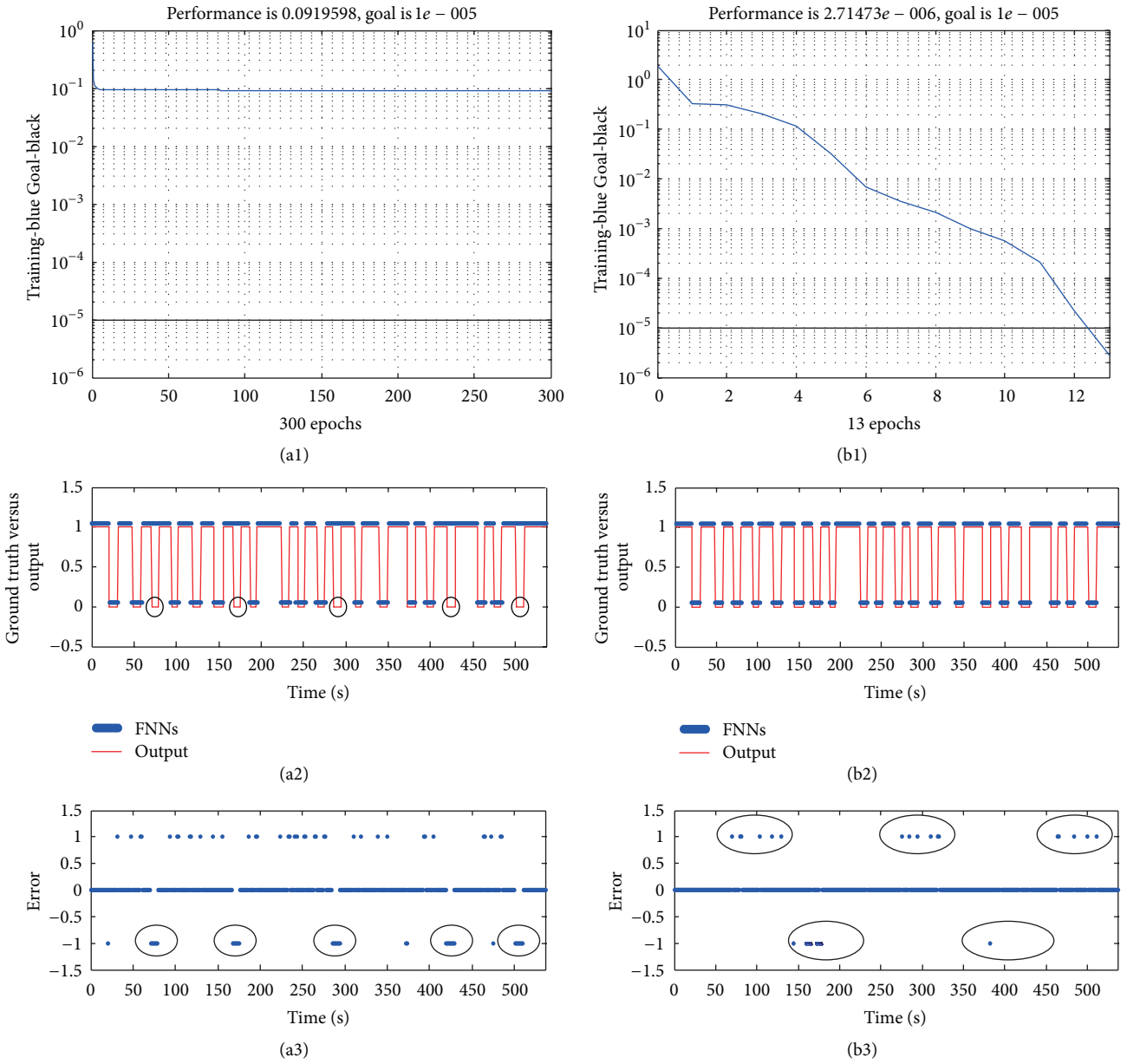

Figure 10: Performance comparison of gesture spotting between two FNNs. (a1) the performance goal is not met within 180 iterations. (a2) the output of FNNs, accuracy = 75.69%. (a3) the error of FNNs. (b1) the performance goal is met within 26 iterations. (b2) the output of FNNs, accuracy = 96.86%. (b3) the error of FNNs.

After 180 iterations, the FNNs did not reach the target as shown in Figure 10(a1). These parts circled in Figure 10(a3) showed continuous error, which led to detection errors in Figure 10(a2). Thus, it can be seen that only if the performance curve reaches the standard, the FNNs can achieve sufficient accuracy. As the performance curve cannot meet its target in Figure 10(a1), the training program needs to be restarted. In Figure 10(b1) we find the FNNs achieved the requirement after 26 iterations. Although there were some discrete errors in Figure 10(b3), they did not generate any detection error in Figure 10(b2). Therefore, we can conclude that several discrete errors on the edge of gesture signal do not affect detection results, but consecutive errors can cause detection errors.

3.3.2. Performance Evaluation of MHMMs. We respectively recorded 10 sets of data from three experimenters for training and testing again, and every set of sequence data has 20 gestures. The test includes two steps. We firstly performed training LHMMs to recognize individual gestures in a sequence. Secondly, the decision obtained from training was used in a Bayesian filter with sequential restrictions to generate a most likely potential command sequence as the final result. The test results of 3D angular velocity and 3D acceleration are shown in Figures 11 and 12, respectively. In Figure 11(a), the signal was a 3D angular velocity which included twenty gestures captured by a Kinect sensor. While the circled region (A) and (B) in Figure 11(b) indicated two errors generated by FNNs, which caused the size of the segmentation to be shorter

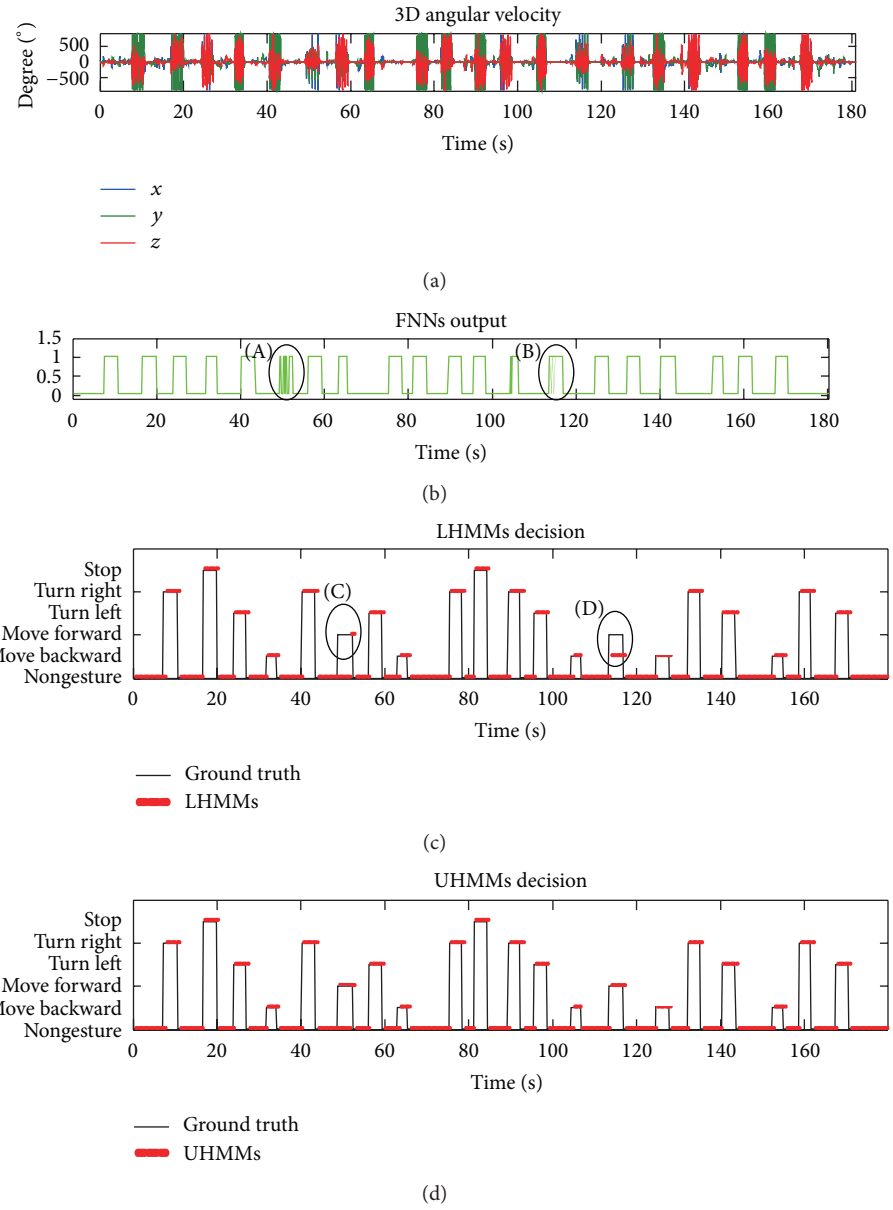

Figure 11: Results of FNNs, LHMMs, and UHMMs. (a) the raw 3D angular velocity; (b) the output of FNNs; (c) the LHMMs decision results compared with the ground truth; (d) the UHMMs decision results compared with the ground truth.

than its actual length after detecting the start and end points of the gesture. After using LHMMs to recognize gestures, errors appeared as shown in the circled regions (C) and (D) of Figure 11(c), which indicate the length of segment affects the accuracy of gesture recognition. While, after, UHMMs were used to update recognition results from LHMMs, the output obtained accurate recognition results. Similarly, 3D acceleration data was also used for testing, recognition errors also appeared in LHMMs, as four circled regions shown in Figure 12(c). However, after updating in UHMMs, accurate recognition results were also obtained.

We have calculated the average recognition rate and average time of 30 sets of gestures which were composed by the five predefined gestures. Table 1 lists the average recognition accuracy and the average recognition time of HMMs and MHMMs, respectively. From the table, we can conclude that the recognition accuracy of MHMMs which use Bayesian filtering is better than that which only uses HMMs, and the real-time performance of the MHMMs-based gesture recognition meets the requirement of online gesture recognition.

4. Conclusion

In this paper, a novel MHMMs continuous gesture recognition algorithm is proposed for Human-Robot Interaction, which uses raw motion data captured by a Kinect sensor. Therefore it is different from traditional vision-based methods. Firstly, a Kinect sensor was used to obtain 3D

Table 1: Comparisons of average recognition accuracy and average recognition time between HMMs and MHMMs.

Gesture tyep	HMMs accuracy (%)	MHMMs accuracy (%)	HMMs recognition time (s)	MHMMs recognition time (s)
1	86.83%	95.71%	0.564	0.637
2	91.24%	97.28%	0.629	0.784
3	78.43%	90.63%	0.468	0.615
4	85.92%	93.42%	0.693	0.812
5	73.97%	89.85%	0.503	0.759

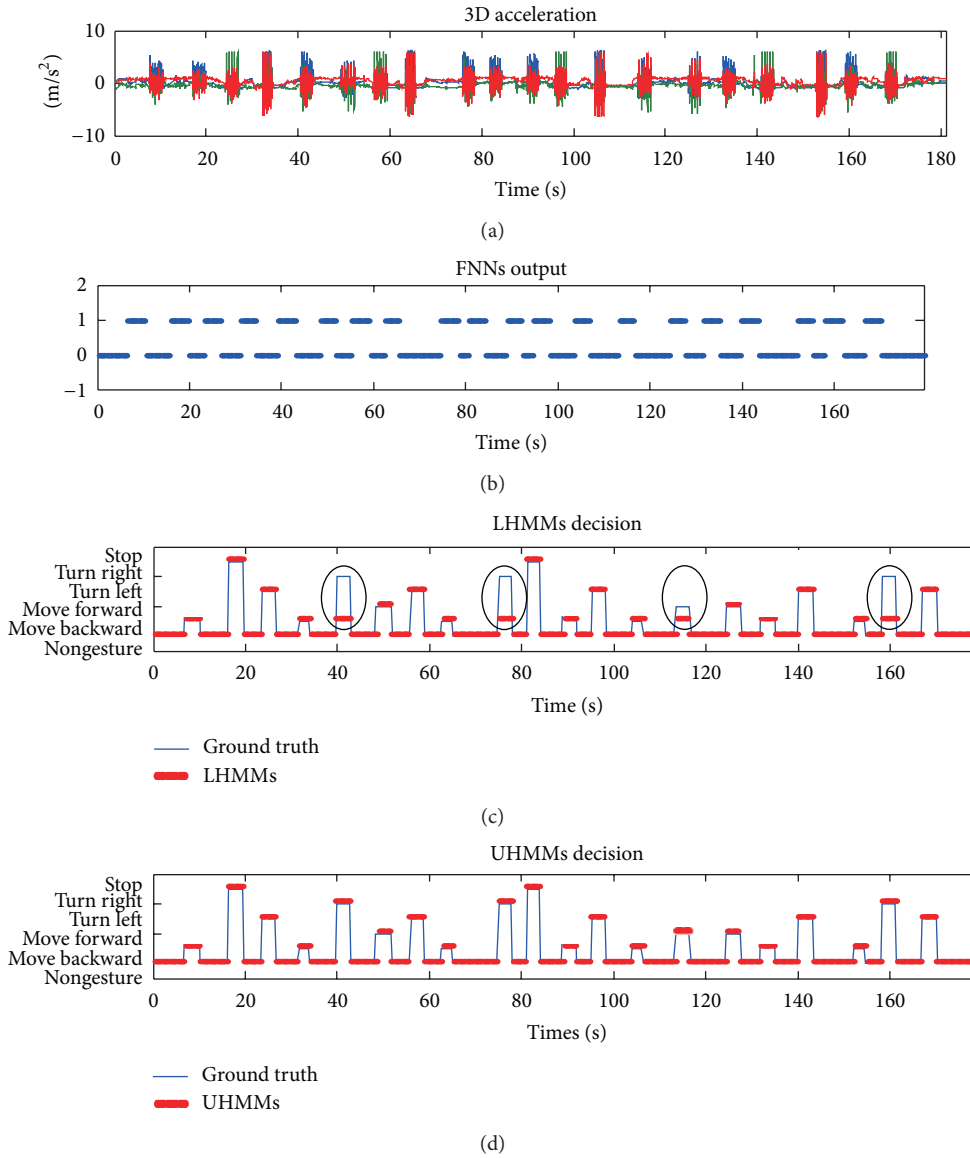

Figure 12: Results of FNNs, LHMMs and UHMMs. (a) the raw 3D acceleration; (b) the output of FNNs; (c) the LHMMs decision results compared with the ground truth; (d) the UHMMs decision results compared with the ground truth.

acceleration and 3D angular velocity of an arm. Secondly, a FNNs and a threshold criterion were, respectively, used for gesture spotting and segment. And then the segmented gesture signals were sent into LHMMs to recognize individual gestures. After that, the identified gesture sequence was fed into UHMMs, where a Bayesian filter with sequential constraints was used to correct errors generated in LHMMs. The experimental results verify the proposed algorithm not only effectively improves the accuracy of continuous gestures, but also has a good real-time performance.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (60277605).

References

[1] X. Wang, M. Xia, and H. Cai, "Hidden markov models based dynamic hand gesture recognition," *Mathematical Problems in Engineering*, vol. 2012, Article ID 986134, 11 pages, 2012.

[2] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 40, no. 1, pp. 1–12, 2010.

[3] J. Yang, J. Lee, and J. Choi, "Activity recognition based on RFID object usage for smart mobile devices," *Journal of Computer Science and Technology*, vol. 26, no. 2, pp. 239–246, 2011.

[4] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 42, no. 6, pp. 790–808, 2012.

[5] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[6] L. Chen, J. Hoey, and C. D. Nugent, "Sensor-based activity recognition," *Transactions on Systems, Man, and Cybernetics*, vol. 42, pp. 790–808, 2012.

[7] S. Bilal, R. Akmeliawati, A. A. Shafie, and M. J. E. Salami, "Hidden Markov model for human to computer interaction: a study on human hand gesture recognition," *Artificial Intelligence Review*, vol. 8, pp. 1–22, 2011.

[8] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Computing*, vol. 3001, pp. 1–17, 2004.

[9] C. Sutton, A. McCallum, and K. Rohanimanesh, "Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data," *Journal of Machine Learning Research*, vol. 8, no. 2, pp. 693–723, 2007.

[10] O. Brdiczka, J. L. Crowley, and P. Reignier, "Learning situation models in a smart home," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 39, no. 1, pp. 56–63, 2009.

[11] F. Omar, M. Sinn, J. Truszkowski, P. Poupart, J. Tung, and A. Caine, "Comparative analysis of probabilistic models for activity recognition with an instrumented walker," in *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI '10)*, pp. 392–400, July 2010.

[12] F. J. Ordonez, P. D. Toledo, and A. Sanchis, "Activity recognition using hybrid generative/discriminative models on home environments using binary sensors," *Sensors*, vol. 13, no. 1, pp. 5460–5477, 2013.

[13] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multi-Media*, vol. 19, no. 2, pp. 4–10, 2012.

[14] R. Giraldo, D. M. Giraldo, and S. A. Meza, "Kernel based hand gesture recognition using kinect sensor," in *Proceedings of the 17th Symposium of Image, Signal Processing, and Artificial Vision*, pp. 158–161, 2012.

[15] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM International Conference on Multimedia ACM Multimedia 2011 (MM '11)*, pp. 759–760, December 2011.

[16] A. Bauer, K. Klasing, and G. Lidoris, "The autonomous city explorer: towards natural human-robot interactivity in urban environments," *International Journal of Social Robotics*, vol. 1, no. 3, pp. 127–140, 2009.

[17] H.-D. Yang, A.-Y. Park, and S.-W. Lee, "Gesture spotting and recognition for human-robot interaction," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 256–270, 2007.

[18] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*, PWS Publishing Company, 1996.

[19] L. R. Rabiner, "Tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[20] S.-Z. Yu and H. Kobayashi, "Practical implementation of an efficient forward-backward algorithm for an explicit-duration hidden Markov model," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 1947–1951, 2006.

[21] W. Liu and W. Han, "Improved Viterbi algorithm in continuous speech recognition," in *International Symposium in Information Technology*, pp. 542–545, October 2010.

[22] O. Mourad, "Hmms parameters estimation using hybrid baum-welch genetic algorithm," in *Proceedings of the International Conference on Computer Application and System Modeling*, pp. 207–209, 2010.

[23] K. Hirasawa, M. Ohbayashi, M. Koga, and M. Harada, "Forward propagation universal learning network," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 353–358, June 1996.

[24] VICON, http://www.vicon.com/.

[25] M. Quigley, K. Conley, and P. Gerkey B, "Ros: an open-source robot operating system," in *Proceedings of the ICRA Workshop on Open Source Software*, 2009.

[26] Neural, http://www.mathworks.com/products/neuralnet/.