symmetry

MDPI

Article

Advanced Machine Learning for Gesture Learning and Recognition Based on Intelligent Big Data of Heterogeneous Sensors

Jisun Park, Yong Jin, Seoungjae Cho, Yunsick Sung and Kyungeun Cho *

Department of Multimedia Engineering, Dongguk University, Seoul 04620, Korea
* Correspondence: cke@dongguk.edu; Tel.: +82-2-2260-3834

Received: 7 June 2019; Accepted: 12 July 2019; Published: 16 July 2019

Abstract: With intelligent big data, a variety of gesture-based recognition systems have been developed to enable intuitive interaction by utilizing machine learning algorithms. Realizing a high gesture recognition accuracy is crucial, and current systems learn extensive gestures in advance to augment their recognition accuracies. However, the process of accurately recognizing gestures relies on identifying and editing numerous gestures collected from the actual end users of the system. This final end-user learning component remains troublesome for most existing gesture recognition systems. This paper proposes a method that facilitates end-user gesture learning and recognition by improving the editing process applied on intelligent big data, which is collected through end-user gestures. The proposed method realizes the recognition of more complex and precise gestures by merging gestures collected from multiple sensors and processing them as a single gesture. To evaluate the proposed method, it was used in a shadow puppet performance that could interact with on-screen animations. An average gesture recognition rate of 90% was achieved in the experimental evaluation, demonstrating the efficacy and intuitiveness of the proposed method for editing visualized learning gestures.

Keywords: machine learning; gesture learning; gesture recognition; editing; heterogeneous sensors

1. Introduction

The development of technologies, which are based on intelligent big data such as virtual reality and invoked reality, have contributed to the increase in research interest in natural user interfaces (NUI) and natural user experience (NUX). Gesture-based systems maximize immersion by allowing end-users to intuitively manipulate contents. Numerous applications recognize end-user gestures such as the ability to remotely control a television using hand gestures or to activate stage effects triggered by body gestures. The successful implementation of applications based on gesture recognition requires a high gesture recognition accuracy, which requires defining and learning end-user gestures by immense collected gestures in advance. However, obtaining adequate gesture recognition accuracies is difficult because of the constraints imposed by different system components. First, each sensor has a limited sensory range in terms of capturing and measuring body parts in motion, reducing the accuracy possible for capturing complex gestures. Second, the end-users encounter a difficulty when they attempt to integrate their own gestures into the systems. For example, during the learning process applied to incorporate actual end-user gestures. Non-experts often face difficulty in gathering training gestures and learning gestures because this process requires a certain degree of programming knowledge.

In this study, a generic gesture recognition and learning framework is developed, which utilizes heterogeneous sensors, and enables end-users to modify their gestures based on intelligent big data. There are disadvantages associated with the use of heterogeneous sensors such as data synchronization

and data fusion. However, it is difficult to accurately identify gestures without using heterogeneous sensors because of the limited range and recognition ability of each sensor. More complex gestures can be recognized using multiple sensors. Therefore, a suitable approach for utilizing heterogeneous sensors to increase the gesture recognition accuracy must be identified. In addition, for heterogeneous sensors, the most popular and low-cost sensors such as Kinect, Myo, and Leap Motion must be integrated for a better utilization of the proposed method. The approach must enable end-users to learn and recognize their specific gestures. To recognize gestures correctly, they must be learned by applying an appropriate learning algorithm using good training gestures. If the obtained results are not satisfactory, the end-user must be able to re-edit the learning data.

Therefore, a novel gesture recognition and learning method was developed and experimentally evaluated. The method comprises four end-user interfaces (UIs): Body selection, gesture learning, gesture editing, and gesture recording.

The remainder of this paper is organized as follows. In Section 2, the studies conducted on gesture learning and recognition are reviewed. Section 3 describes the structure of the proposed generic gesture recognition and learning framework. In Section 4, the implementation of the gesture learning and recognition process is described. Section 5 presents the experimental methods, results, and analysis of those results. Finally, Section 6 presents the conclusions of the study and the scope for future work.

2. Related Works

This section reviews the recent studies conducted in the field of gesture recognition. Moreover, active developments in gesture recognition techniques have led to the recent developments in virtual reality (VR) technology [1–16]. The gesture recognition accuracy, which is a key metric to evaluate the effectiveness of gesture recognition systems, has witnessed steady improvements using input data from multiple sensors [17–22]. Moreover, researchers have studied visual recognition and learning systems for more intuitive learning of gesture recognition. In particular, gesture recognition and learning using multiple sensors is being studied actively.

For example, Ibañez et al. [23] proposed a Kinect-based gesture recognition tool that uses only the skeleton generated by Kinect as input data, and it can visualize the results. However, this tool cannot support a variety of other sensors or perform partial recognition tasks. Further, the editing of the obtained results is not supported by the tool. Signer et al. [24] proposed a 2D-data-based gesture recognition tool with gesture registration and learning features; this tool also supports visualization and simple editing tasks. Zhang et al. [25] proposed a gesture recognition framework that uses electromyography and gyroscope data and can learn from these two datasets simultaneously. The proposed framework utilizes the hidden Markov model (HMM) algorithm as its learning algorithm. Truong et al. [26] proposed a 3D gesture recognition framework using Laban movement analysis and HMM models. Ma et al. [27] proposed an enhanced HMM model that could recognize handwritten characters in real time. In addition, the framework classified gestures into long-term/short-term gestures and dynamic/static gestures. Borghi et al. [28] also proposed a gesture recognition algorithm, which was based on the HMM algorithm, using 3D skeleton data. The proposed method divides the skeleton into upper, lower, left, and right parts and recognizes the corresponding gestures. Suma et al. [29] proposed a Kinect-based gesture recognition tool that included a network communication structure, and it could visualize end-user gestures. The learned gestures could be applied in game control using the communication system. However, this tool could not learn or recognize gestures. Gillian et al. [30] developed a gesture recognition library based on C++ that had high universality, and it supported learning and recognition and various types of input data. However, it did not offer editing features for visualizing and learning data. Gris et al. [31] used Kinect to develop a gesture recognition tool that could collect data by specifying the body parts required for the gesture. However, the tool only supported the recognition of static gestures. Maes et al. [32] proposed a recognition system based on learning gestures that enabled users to practice dancing. However, the proposed system only focused on learning and recognition, and it did not include a module to edit the data used to learn the gesture.

Yavşan et al. [33] proposed an interface for gesture recognition and an NAO robot control using Kinect. In the proposed method, the learning and recognition experiments were conducted using K-nearest neighbor (K-NN) and feed-forward neural networks (FNNs). However, only static gestures were recognizable, and the system lacked editing capabilities for learning data.

In this study, we develop a method wherein various gestures are precisely learnt by collecting them from multiple sensors. This method offers GUI-based editing features in which even non-experts can easily edit the collected gestures. Moreover, the learning accuracy can be improved through the collection of static and dynamic gesture data and the editing of this collected data. After reviewing the visual learning results, the learning data can be re-edited to achieve better recognition. Further, the learning results can be applied to interactive systems such as smart TVs through a communication network.

3. Generic Gesture Learning and Recognition Framework

The proposed generic gesture recognition and learning framework is designed to obtain and learn various gestures based on sensing data by multi-sensor fusion. In this framework, end-users can define customized gestures by performing the corresponding gestures, which are learnt by the framework to recognize user-specific gestures. Moreover, the defined gestures can be edited after the end user analyzes the accuracy of the recognized gesture.

3.1. Overview of the Proposed Generic Gesture Recognition and Learning Framework

The proposed framework comprises five stages: Gesture registration, editing, learning, recognition, and transfer. Figure 1 illustrates the working of the proposed generic gesture recognition and learning framework.

First, in the gesture registration stage, the proposed method obtains gesture data from the multi-sensor fusion. Using two or more sensors, various types of gestures can be learnt and precise and accurate gesture recognition can be realized [17–22]. Moreover, the proposed method allows the end user to limit the range of gestures to be registered for recognizing specific body parts, thereby improving the recognition accuracy. Second, if the learnt gestures are not found to be satisfactory during the gesture editing stage, the end user can edit the collected gestures using the GUI editor. Thus, non-experts can easily define and learn various gestures. Third, during the gesture learning stage, the end user can also select the most appropriate learning algorithm. Subsequently, during the gesture recognition and gesture transfer stages, the learnt data can be applied to various interaction systems. In short, the proposed framework can realize accurate gesture recognition and learning through multi-sensor data fusion and gesture data.

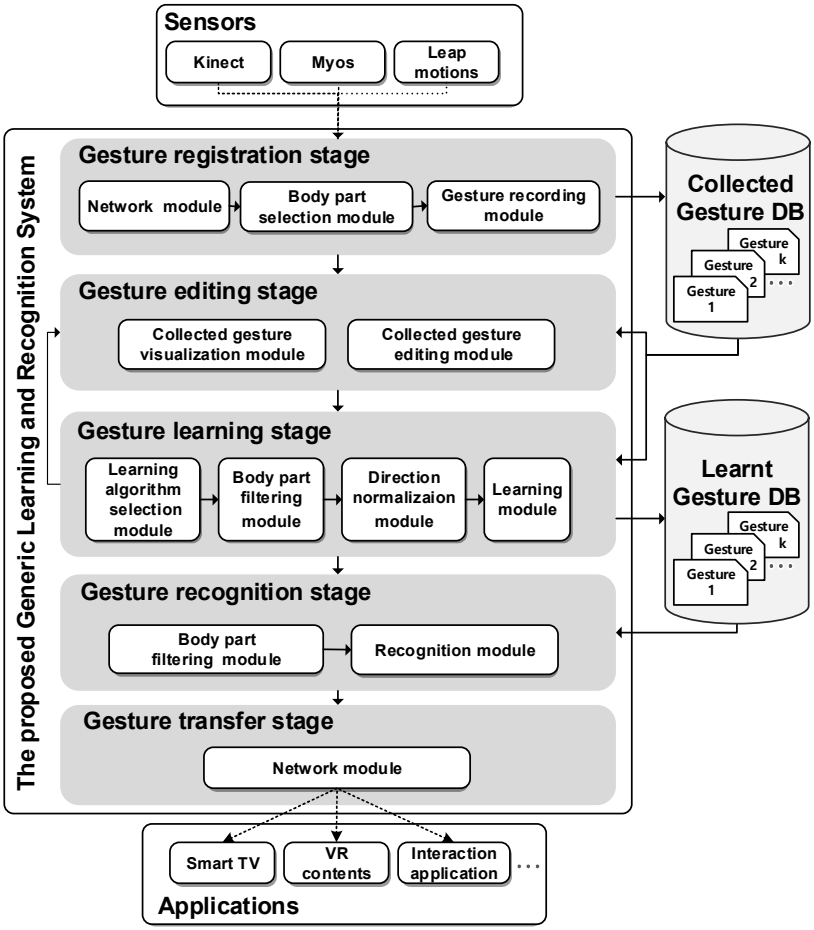


Figure 1. Overall process of the proposed generic gesture recognition and learning framework.

3.2. Gesture Registration Stage

The proposed framework obtains gestures from Kinect, Myo, and Leap Motion. The gesture formats of Kinect, Myo, and Leap Motion are shown in Table 1. The gesture registration stage consists of a module for selecting the relevant body part and a gesture recording module to resister a new gesture.

The body part selection module selects the body parts by taking the recognition accuracy into consideration. For example, when the end-user wants for the proposed method to learn gestures using the right arm, the gestures identified from body parts other than the right arm are excluded, and only the relevant gestures are learnt. Figure 2 shows six body parts recognized by Kinect, four body parts recognized by Myo, and two body parts recognized by Leap Motion.

Table 1. Data format for each sensor.

Sensor	Data Format
Kinect	Skeleton orientation (x, y, z)
	Depth image (d)
	RGB image (r, g, b)
Myo	Electromyography (e_0 – e_7)
	Gyroscope (= orientation) (x, y, z)
	Angular speed (s)
Leap Motion	Skeleton orientation (x, y, z)
	Depth image (d)

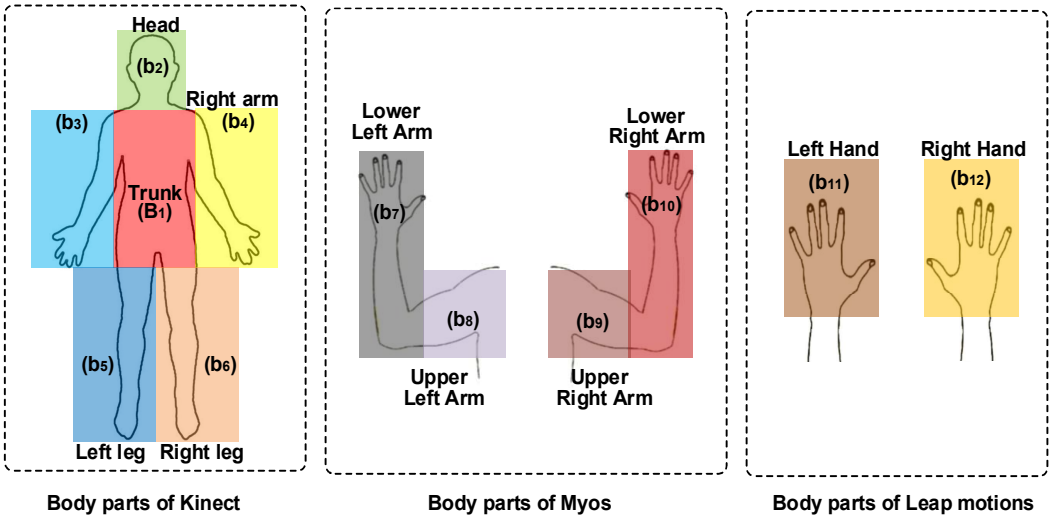


Figure 2. Body parts recognized by Kinect, Myo, and Leap motions.

In the proposed framework, human gestures, expressed using arms and legs, are recognized based on the movement of the twelve learning body parts: Head, trunk, left arm, right arm, left leg, right leg, lower left arm, lower right arm, upper left arm, upper right arm, left hand, and right hand. Mathematically, let B be the set of all body parts, $B = \{b_1, b_2, \cdots, b_{12}\}$; let B^*_t be the subset of B, i.e., body parts selected by the end-user at time t, $B^*_t \in B$.

The gesture recording module obtains gestures from the sensor values of Kinect, Myo, and Leap Motion. Given that the sensor values in the proposed method consider only the orientation of the user, the values of each sensor values at time t are expressed by $s_{i,j,t} = \{x_{i,j,t}, y_{i,j,t}, z_{i,j,t}\}$, where i is the index of a body part in B, and j is the index of the sensors of the i^{th} body part. Let S be the set of all sensor values, and S^*_t be the set of all sensor values for the selected body parts at time t, B^*_t. $s_{i,j,t} \in S$. $S^*_t = \{s_{1,1,t}, s_{1,2,t}, \cdots\} \cup \{s_{2,1,t}, s_{2,2,t}, \cdots\} \cup \cdots\}$. $S^*_t \subset S$. Table 2 represents available joints acquired by each sensor.

Table 2. Available joints acquired by each sensor.

Device	Body part (Symbol)	Joints (Symbol)
Kinect	Trunk (b_1)	Base spine ($s_{1,1}$), Mid spine ($s_{1,2}$), Shoulder spine ($s_{1,3}$)
	Head (b_2)	Neck ($s_{2,1}$), Head ($s_{2,2}$)
	Left arm (b_3)	Left shoulder ($s_{3,1}$), Left elbow ($s_{3,2}$), Left wrist ($s_{3,3}$), Left hand ($s_{3,4}$), Left hand tip ($s_{3,5}$), Left thumb ($s_{3,6}$)
	Right arm (b_4)	Right shoulder ($s_{4,1}$), Right elbow ($s_{4,2}$), Right wrist ($s_{4,3}$), Right hand ($s_{4,4}$), Right hand tip ($s_{4,5}$), Right thumb ($s_{4,6}$)
	Left leg (b_5)	Left hip ($s_{5,1}$), Left knee ($s_{5,2}$), Left ankle ($s_{5,3}$), Left foot ($s_{5,4}$)
	Right leg (b_6)	Right hip ($s_{6,1}$), Right knee ($s_{6,2}$), Right ankle ($s_{6,3}$), Right foot ($s_{6,4}$)
Myo	Lower left arm (b_7)	Lower left arm ($s_{7,1}$)
	Upper left arm (b_8)	Upper left arm ($s_{8,1}$)
	Lower right arm (b_9)	Lower right arm ($s_{9,1}$)
	Upper right arm (b_{10})	Upper right arm ($s_{10,1}$)

Table 2. *Cont.*

Device	Body part (Symbol)	Joints (Symbol)
Leap Motion	Left hand (b_{11})	Palm ($s_{11,1}$), Wrist ($s_{11,2}$), Thumb distal ($s_{11,3}$), Thumb intermediate ($s_{11,4}$), Thumb proximal ($s_{11,5}$), Thumb metacarpal ($s_{11,6}$) Index distal ($s_{11,7}$), Index intermediate ($s_{11,8}$), Index proximal ($s_{11,9}$), Index metacarpal ($s_{11,10}$) Middle distal ($s_{11,11}$), Middle intermediate ($s_{11,12}$), Middle proximal ($s_{11,13}$), Middle metacarpal ($s_{11,14}$) Ring distal ($s_{11,15}$), Ring intermediate ($s_{11,16}$), Ring proximal ($s_{11,17}$), Ring metacarpal ($s_{11,18}$) Pinky distal ($s_{11,19}$), Pinky intermediate ($s_{11,20}$), Pinky proximal ($s_{11,21}$), Pinky metacarpal ($s_{11,22}$)
	Right hand (b_{12})	Palm ($s_{12,1}$), Wrist ($s_{12,2}$), Thumb distal ($s_{12,3}$), Thumb intermediate ($s_{12,4}$), Thumb proximal ($s_{12,5}$), Thumb metacarpal ($s_{12,6}$) Index distal ($s_{12,7}$), Index intermediate ($s_{12,8}$), Index proximal ($s_{12,9}$), Index metacarpal ($s_{12,10}$) Middle distal ($s_{12,11}$), Middle intermediate ($s_{12,12}$), Middle proximal ($s_{12,13}$), Middle metacarpal ($s_{12,14}$) Ring distal ($s_{12,15}$), Ring intermediate ($s_{12,16}$), Ring proximal ($s_{12,17}$), Ring metacarpal ($s_{12,18}$) Pinky distal ($s_{12,19}$), Pinky intermediate ($s_{12,20}$), Pinky proximal ($s_{12,21}$), Pinky metacarpal ($s_{12,22}$)

When the gesture recording module records and stores a gesture, the gesture at time t, g_t, includes B_t^* and S_t at time t, $g_t = \{B_t^*, S_t\}$. Although the proposed method stores S_t, $S^*_t \subset S_t \subset S$, during the gesture registration stage, the learning model learns and recognizes only S^*_t during the gesture learning and recognition stage in order to reduce the processing time required. Figure 3 illustrates the visual relationship between these notations and the gestures collected by Kinect, Myo, and Leap Motion sensors.

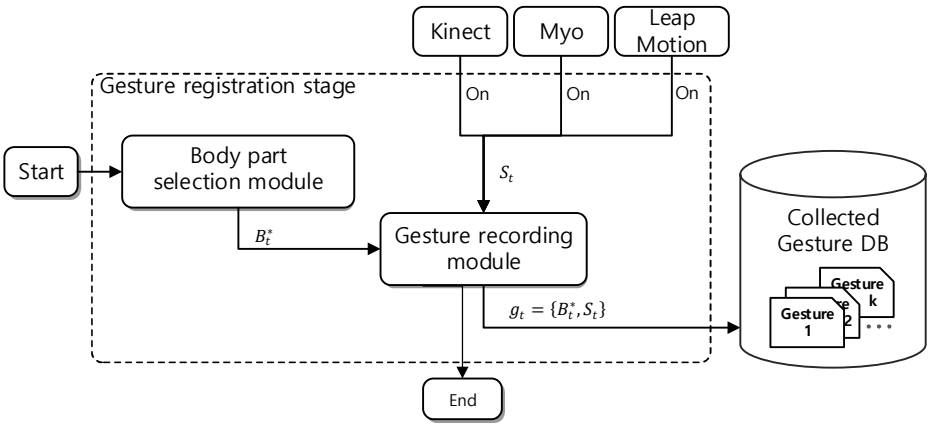


Figure 3. Gesture registration process using Kinect, Myo, and Leap Motion sensors.

For multi-sensor data fusion, the framework obtains sensor values using different devices such as Kinect, Myo, and Leap motion. Therefore, the proposed method is susceptible to port collision when data is collected from various gesture recognition sensors through the communication model. To address this susceptibility, a port assignment module is designed that assigns a unique ID to each sensor such that the sensor can receive data from different ports, as shown in Figure 4.

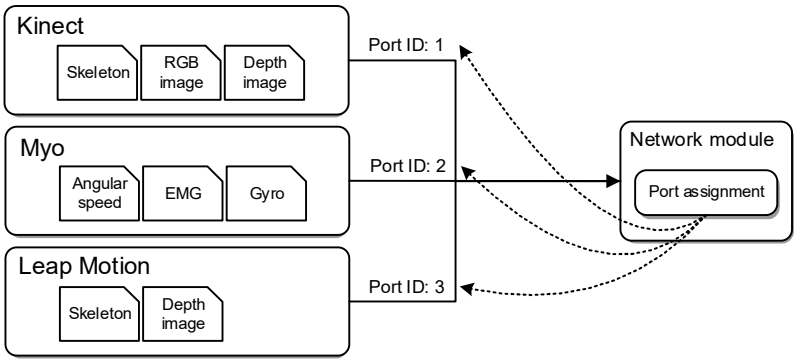


Figure 4. Port assignment for Kinect, Myo, and Leap Motion sensors.

As shown in Table 1, the sensor value type of each sensor is unique. However, in this framework, only the skeleton orientation information is used for training.

3.3. Gesture Editing Stage

The gesture editing stage comprises the visualization module and collected gesture editing module. Figure 5 illustrates the structure of the two modules in this stage. As the name suggests, the visualization module enables the visualization of the gestures when editing the collected gestures in order to realize a more intuitive editing experience for the end-user. It consists of a collected gesture viewer and learning information viewer. Each collected gesture is associated with a name, similarity, and trajectory, and the learning information includes the names, similarities, and counts of the gestures to be learnt.

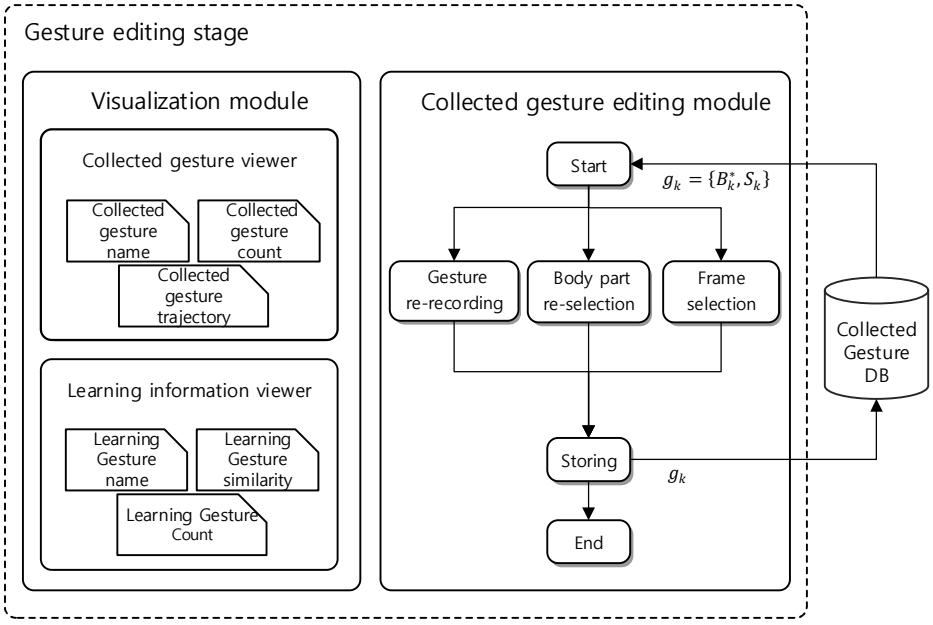


Figure 5. Modular structure of the gesture editing stage.

The collected gesture editing module supports functions such as gesture re-recording, body part re-selection, and frame reselection for higher recognition accuracies. If the results obtained after learning and recognition are deemed unsatisfactory, the inadequate gesture can be edited, deleted, or re-recorded through editing to obtain the appropriate gesture. In Figure 5, the collected gestures are expressed as $g_k = \left\{B_k^*, S_k\right\}$, where k is the index of the collected gestures in the database.

3.4. Gesture Learning Stage

As shown in Figure 6, the gesture learning stage comprises a learning algorithm selection module, body part filtering module, direction normalization module, and learning module. First, the end-user selects an algorithm to learn the collected gestures. Second, the body part filtering increases the learning accuracy by focusing on a selected body part when collecting gestures. Third, the direction normalization module normalizes each sensor's orientation along eight directions in order to accelerate the learning accuracy. Finally, the learning module learns the normalized gestures using the selected algorithm.

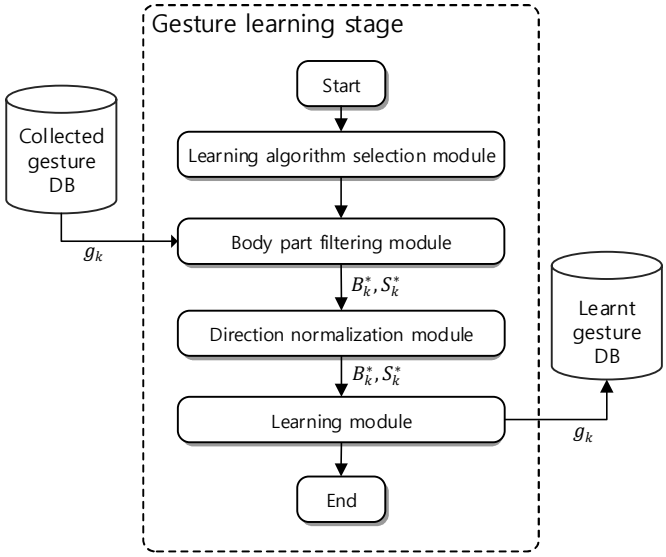


Figure 6. Modular structure of the gesture learning stage.

In the proposed framework, the end-user can select from several learning algorithms including hidden Markov models (HMMs), dynamic recurrent neural networks (Dynamic RNNs), and dynamic time warping (DTW) algorithms. To enhance the gesture recognition accuracy, the appropriate learning algorithms can be selected through the learning algorithm selection module. In a previous study, it was shown that the recognition accuracies of HMMs or RNNs are higher than that of DTW [34–36]. Moreover, in our experiment, the highest accuracy was obtained when using HMMs. Therefore, the proposed framework uses HMM by default, and the user has the option of selecting another algorithm.

HMM is a statistical Markov model that comprises two elements: A hidden state and an observable state. HMM is suitable for tasks that involve recognizing patterns that change over time. Equation (1) represents hidden Markov model:

$$\lambda_{B^*} = \{P_{B^*}, G_{B^*}, \pi, A, B\}, \quad (1)$$

where P represents the observable states of a posture, and G represents the hidden states of a gesture. π indicates a matrix of the initial probabilities of hidden states, while A indicates a matrix of the transition probabilities of hidden states; B indicates a matrix of the emission probabilities of hidden states. In this stage, gesture recognition is performed using B.

The HMM uses the following three algorithms. First, the observation probabilities are calculated using forward and backward algorithms. Second, the Viterbi algorithm is used to find the most appropriate state transition sequence for the observable result sequence. This algorithm selects the most probable transition state from the previous states and outputs a result by backtracking to the initial state. Third, the initialization, state transition, and observation probabilities—which are HMM parameters—are optimized to determine the maximum value of the observation probability. To optimize these parameters, the Baum–Welch algorithm is used to generate an optimized HMM for the observation sequence.

In the proposed method, gestures recognition using HMM is achieved by using one integer to represent the direction of one sensor. The directions, namely, up, down, left, right, front, and back, are represented by corresponding bits, i.e., 00100(2), 01000(2), 00010(2), 00001(2), and 10000(2), respectively. Therefore, one or more directions can be represented by one integer. For example, up/left/front is represented as 10110(2). Equation (2) represents the algorithm for calculating direction d of a sensor:

$$\begin{aligned} &\text{if } \left|x_{i,j,t} - x_{i,0,t}\right| > \left|x_{i,j,t-1} - x_{i,0,t-1}\right|, \text{ then } d_{i,j,t} \leftarrow d_{i,j,t} + 2^0 \\ &\qquad \text{else } d_{i,j,t} \leftarrow d_{i,j,t} + 2^1 \\ &\text{if } \left|y_{i,j,t} - y_{i,0,t}\right| > \left|y_{i,j,t-1} - y_{i,0,t-1}\right|, \text{ then } d_{i,jt} \leftarrow d_{i,j,t} + 2^2 \\ &\qquad \text{else } d_{i,j,t} \leftarrow d_{i,j,t} + 2^3 \\ &\text{if } \left|z_{i,j,t} - z_{i,0,t}\right| > \left|z_{i,j,t-1} - z_{i,0,t-1}\right|, \text{ then } d_{i,j,t} \leftarrow d_{i,j,t} + 2^4 \\ &\qquad \text{else } d_{i,j,t} \leftarrow d_{i,j,t} + 2^5. \end{aligned} \tag{2}$$

The direction is determined based on the differences in the 3D coordinates of the sensor values between previous and current frames. The direction sequence of the sensor values used in the gesture is calculated, and an optimized HMM for the gesture is created using the Baum–Welch algorithm. The gesture recognition result is obtained by calculating the observation probability of the HMM for each gesture using the Viterbi algorithm with the gesture direction sequence as the input value.

3.5. Gesture Recognition Stage and Gesture Transfer Stage

In the gesture recognition and transfer stage, the sensing module obtains gestures using Kinect, Myo, and Leap Motion. The body part filtering module filters the sensing values by considering the body parts selected by the end-user for the learnt gestures. Subsequently, the recognition module calculates the similarity of all learnt gestures, and then, the most similar learnt gesture is selected. The selected learnt gesture is deliveries provided as output to smart TVs and VR sets through the network module. Figure 7 illustrates the module structure chart of the gesture recognition and gesture transfer stage.

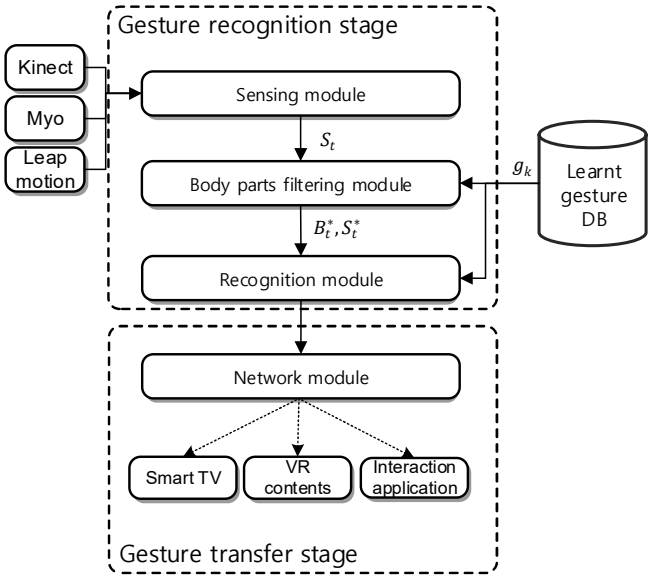


Figure 7. Modular structure of the gesture recognition and gesture transfer stage.

4. Generic Gesture Learning and Recognition Approach

The method developed in this study is based on the generic gesture learning and recognition framework. Figure 8 shows the UI Architecture of the proposed generic gesture learning and recognition framework.

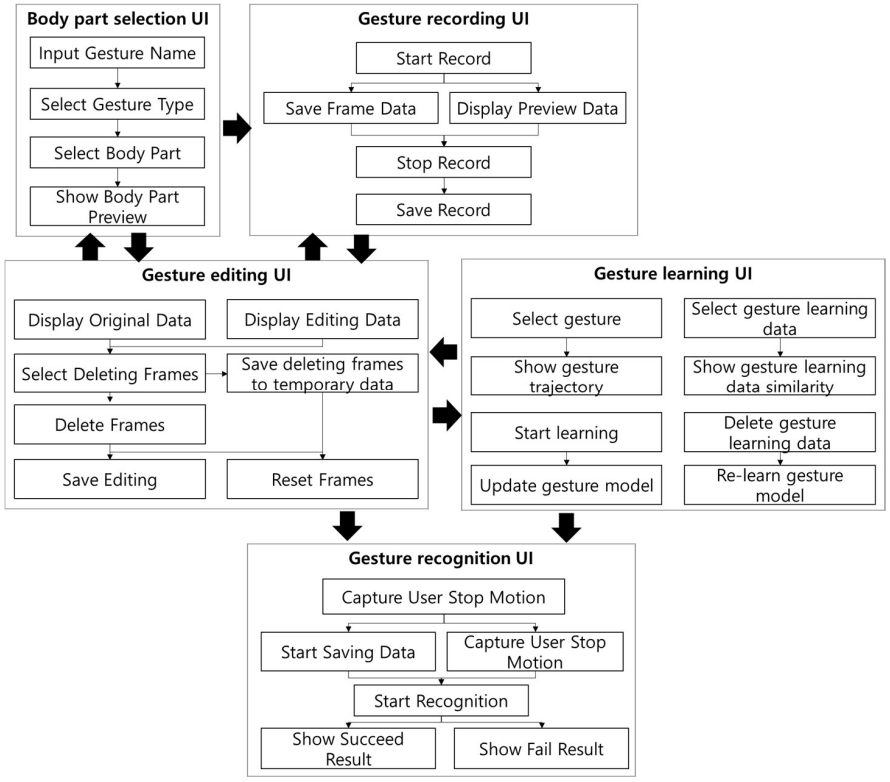


Figure 8. UI Architecture of the proposed generic gesture learning and recognition framework.

4.1. Generic Gesture Learning and Recognition Overview

Figure 9 shows a snapshot of the UI designed for the gesture learning application. As shown in Figure 9, the end-user is displayed a list of learnt gestures. The end-user can select a gesture on the gesture list for editing. To add a new gesture to the gesture list, the end-user can define a new gesture in the gesture creation and option window, select body parts to learn, and start recording. The end-user can also re-learn or re-edit the collected gestures for learning.

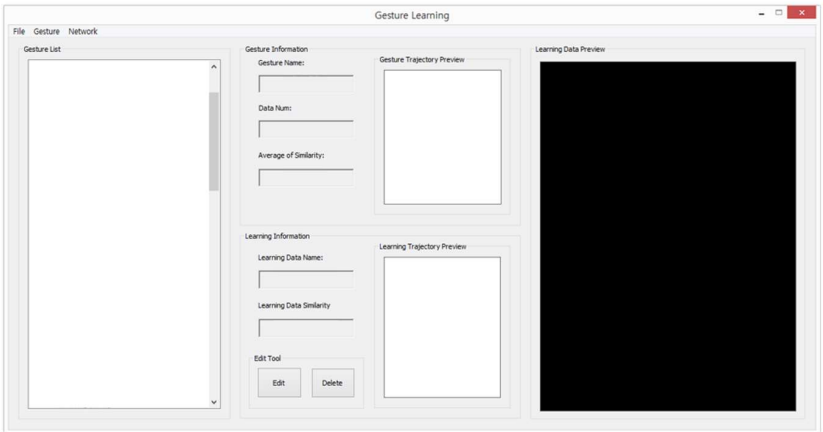


Figure 9. UI of the proposed generic gesture learning and recognition application.

4.2. Implementation of User Interface

In the body part selection UI, the end-user can select the body part(s) that will be used for recording the gestures. Figure 10 shows the body selection UI for Kinect. In the UI, the end-user can select up to six parts of the body—the head, trunk, left arm, right arm, left leg, and right leg—during

the gesture learning stage. As shown in the right side Figure 10, the end-user can directly select the relevant body parts.

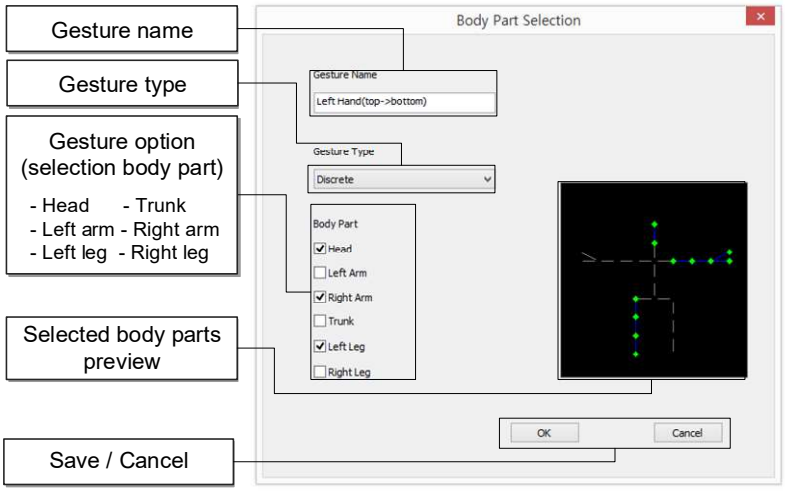


Figure 10. Body part selection UI.

After choosing the body part, the end-user can record and save the gesture. The gesture recording UI is shown in Figure 11. In this UI, the end-user can visualize the original color video as well as the skeletal representation. The end-user can see if the recorded skeletal representation is not representative of the original color video. For example, if a frame displays an inappropriate motion trajectory of the target gesture, the inaccurate frame, in which a specific sensed value is measured incorrectly during the measurement or an empty frame is present between the recorded frames, it can be manually excluded. Moreover, learning using the obtained gestures that contain such frames leads to low recognition accuracies. Thus, the proposed method supports partial deletion and re-recording of such inappropriate frames, thereby improving the recognition accuracy.

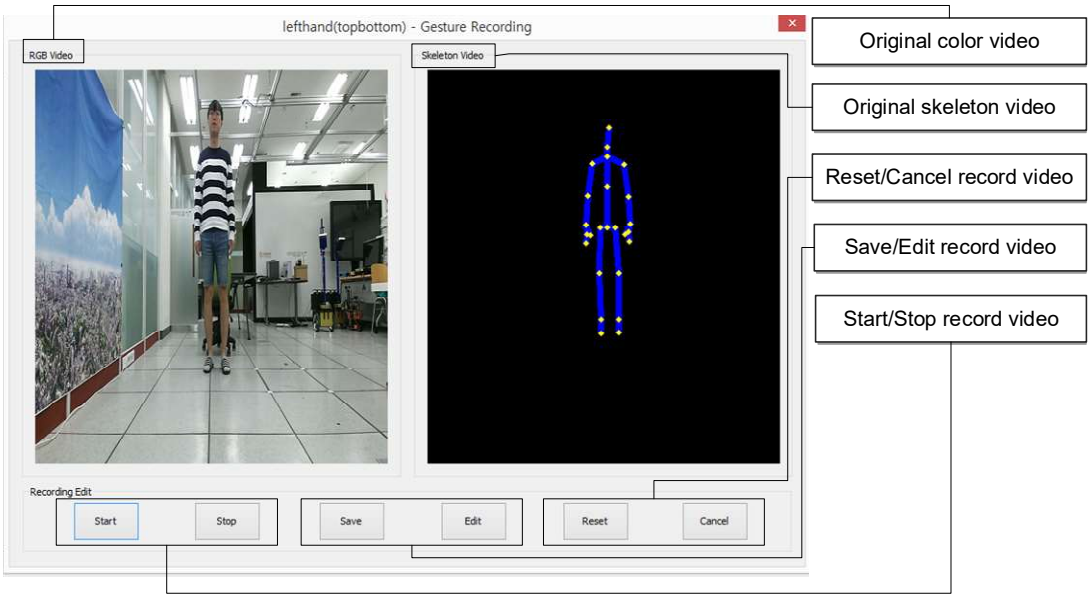


Figure 11. Gesture recording UI.

As shown in Figure 12, by using the gesture editing UI, the end-user can edit the recorded gestures. The UI displays two previews—original and edited. Therefore, the end-user can compare the two previews and adjudge whether the editing is appropriate. Moreover, the end-user can use the slide

bar to preview the recorded video, set the range of the video for deletion, and save the gesture to the gesture database.

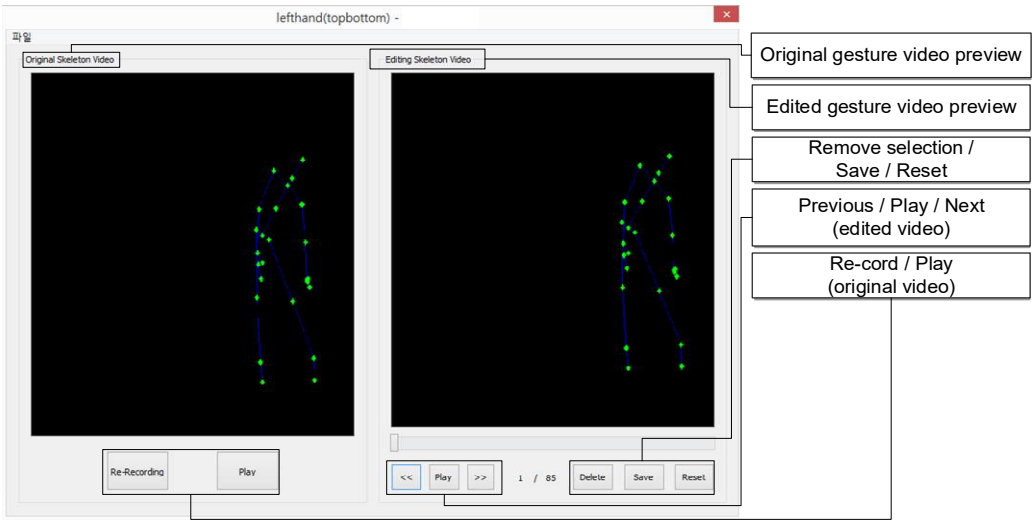


Figure 12. Gesture visualization and editing UI.

After editing the recorded gestures, the end-user generates the HMM module using the gesture learning UI. The UI displays the set of gestures selected by the end-user for learning, as shown in Figure 13. In the gesture learning UI, the end-user can visualize the trajectory of the gestures and compare it against other gestures for the same type of data. Moreover, if differences arise in the two trajectories obtained from the data because of unstable sensors, the end-user can edit the data to reduce the noise. After the end-user finishes editing the gestures, the proposed method automatically learns all the gestures, and the end-user can visualize the similarity of each data point corresponding to each gesture. Here, similarity refers to the similarity score of the gesture data and the trained HMM model by the Viterbi algorithm. After training the HMM model, we use the dissimilarities to identify and correct noisy learning data, thereby enhancing the recognition accuracy of the HMM model. If the similarity is significantly lower than the average of other training data's similarity, the corresponding data editing is required. In the proposed method, the GUI-based gesture learning UI allows the end-user to obtain improved recognition accuracy by the deleting and re-recording of low similarity data.

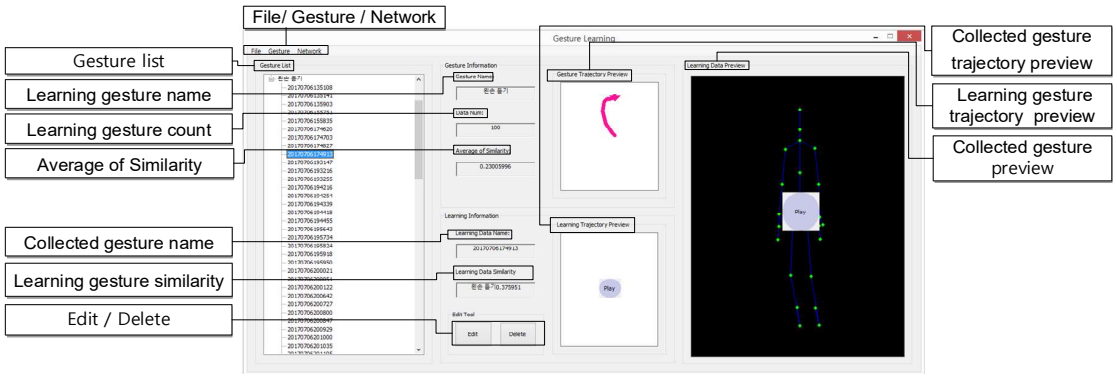


Figure 13. Gesture learning UI.

The gesture recognition UI shown in Figure 14 is used for the gesture recognition and gesture transfer stages. For the gesture recognition stage, the UI can be used to test whether the learnt gesture is accurate. The end-user can visualize the raw data obtained from sensors in the gesture data log panel. In the gesture succeed log, the end-user can identify the gesture recognized by the

proposed method. If the recognition accuracy is lower than the threshold, the performed gesture is considered inadequate. For the gesture transfer stage, the end-user can transfer the recognized gesture result to other applications through the network module. Subsequently, after the HMM model is generated during the gesture learning stage, the Viterbi algorithm is utilized to calculate the similarity of end-user input data for each gesture. The gesture with the maximum similarity is chosen as the recognition result.

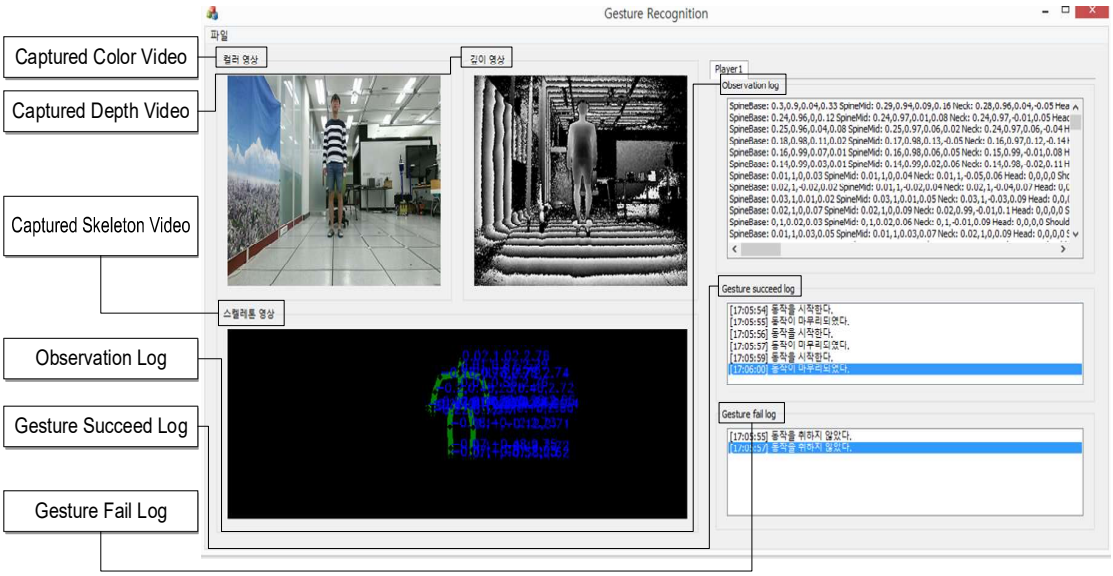


Figure 14. Gesture recognition UI.

5. Experiments

In this study, the proposed method was implemented and evaluated. Kinect was used to identify and record gestures. In the experiments, the HMM algorithm was used as the learning algorithm.

5.1. Performance Show

The proposed method was experimentally evaluated using a shadow puppet performance show. As shown in Figure 15, the actors create certain animations on the screen and interact with it using hand gestures.

Figure 15. Shadow puppet performance show stage.

For the experiments, ten hand gestures were defined based on the storyline of the shadow puppet performance show. The defined gestures were hand gestures which were intended to activate stage effects during the performance show. Table 3 lists the gestures used in the performance show.

Table 3. The defined gesture list for the experiments.

Gesture 1. Left hand (top→bottom)	Gesture 2. Left hand (bottom→top)
	
Gesture 3. Left hand (bottom→middle)	Gesture 4. Right hand (top→bottom)
	
Gesture 5. Right hand (bottom→middle)	Gesture 6. Both hands (bottom→top)
	
Gesture 7. Both hands (middle→top)	Gesture 8. Both hands (bottom→middle)
	
Gesture 9. Both hands (bottom→top→bottom)	Gesture 10. Both hands (middle→bottom)
	

The proposed method recognizes the gesture performed by the actor, and the performance show contents are organized by animating a corresponding virtual character based on this gesture information.

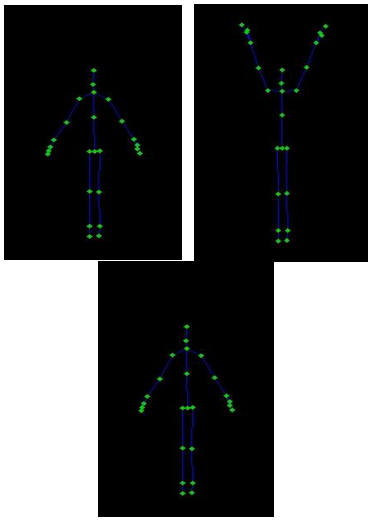
5.2. Implementation of the Gesture Learning and Recognition Approach

The gesture learning and recognition approach was implemented using the Microsoft Foundation Class library (MFC), which is based on the C++ language, in a Windows 8.1 environment. We stored all gesture data in Hadoop Distributed File System (HDFS). The OpenCV library was used to visualize each gesture image.

5.3. Gesture Registration Stage Result

In the experiment, the recognition rates of gestures performed by three subjects were compared. Each participant provided 120 examples for each of the 10 types of gestures (a total of 1200 gestures), which served as learning data. Table 4 shows the gestures recorded by the proposed method for different body parts as selected by the end-user.

Table 4. Differences in the rendering of the selected body parts.

One Gesture During Testing	Collected Gesture Based on Each Selected Body Part		
			
	Left arm	Right arm	Both arms

The user can visualize the specific body parts selected for the gesture data entered into the proposed system by using the learning gesture trajectory preview panel in the gesture recognition UI.

Table 5 presents right hand coordinates (x,y,z) of the ten gesture types. The graphs in the table represent the mean values of x, y, and z for the right hand data of three participants performing 100 repetitions of each gesture. The horizontal axes of all graphs indicates the frame count, and the vertical axes represents the x, y, and z values.

Table 5. Average sensing values of the right hand of 10 gestures per participant.

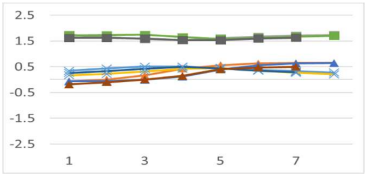
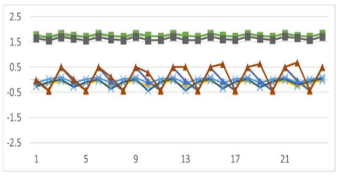
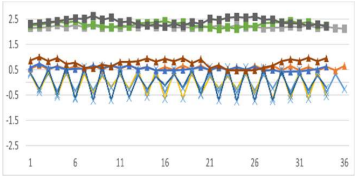
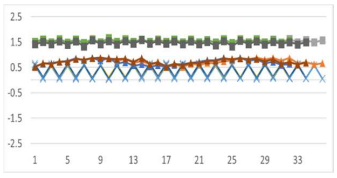
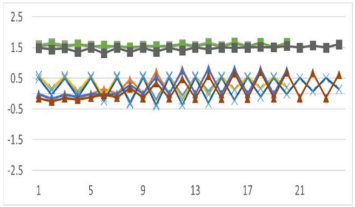
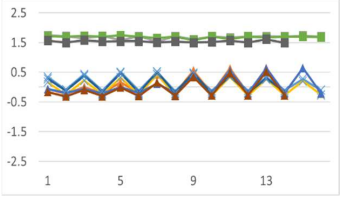
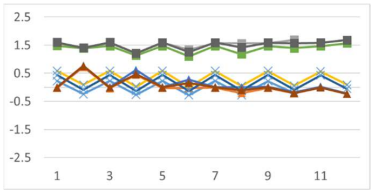
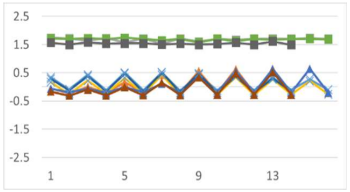
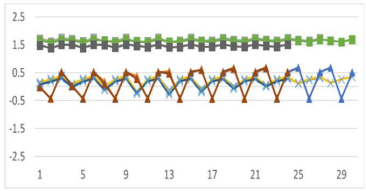
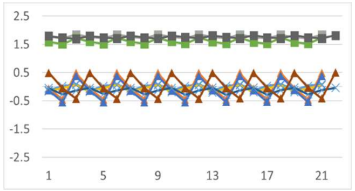
Gesture type	Average Sensing Values for Three Participants	Gesture Type	Average Sensing Values for Three Participants
Gesture 1		Gesture 6	
Gesture 2		Gesture 7	
Gesture 3		Gesture 8	

Table 5. *Cont.*

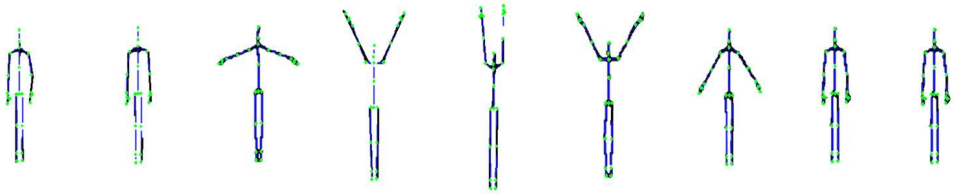
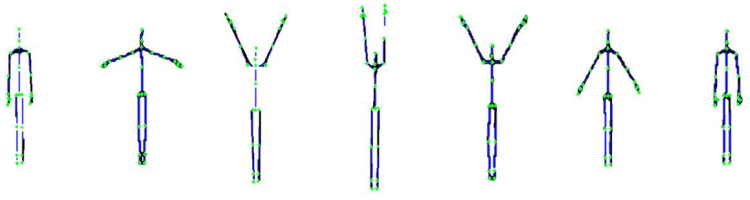
Gesture type	Average Sensing Values for Three Participants	Gesture Type	Average Sensing Values for Three Participants
Gesture 4		Gesture 9	
Gesture 5		Gesture 10	
 horizontal axis: frame count, vertical axis: *x*, *y*, *z* value			

We confirmed that the coordinates of a particular body part are similar even when different subjects perform the same gesture. If the user selects specific body parts to be used for learning and recognition, gesture learning and recognition tasks are performed only for the corresponding body parts.

5.4. Gesture Editing Stage Result

To improve the accuracy of gesture recognition, the user can eliminate duplicated or invalid frames from the acquired gesture data. Table 6 shows the results of an end user selecting specific frames of each gesture. These include the specific frames of the end-user's gestures.

Table 6. First key frame of the original skeleton and key frames of the transparent skeleton for different gestures.

Editing	Images of Frames								
Before editing									
Frame	1–9	10–19	20–29	30–39	40–49	50–59	60–69	70–79	80
After editing									
Frame	1–9	10–19	20–29	30–39	40–49	50–59	60		

As can be inferred from the above table, frames 1–9, 10–19, 70–79, and 80 are similar before the gesture is edited. The table also shows the results obtained after eliminating frames 10–19 and 70–79 by using gesture visualization and the editing UI.

5.5. Gesture Learning Stage Result

The criterion for determining the gesture recognition stage is the recognition accuracy of the learnt gestures. The gesture recognition accuracy is calculated as a percentage of successfully recognized gestures in the test data, as shown in Equation (3):

$$\text{Accuracy} = \frac{\text{Number of correctly recognized gestures}}{\text{Total number of samples}} \times 100. \quad (3)$$

In the experiment, the performance of each algorithm was compared. The number of participants was 21. For each participant, 100 examples were recorded for each gesture. Among these, 60 examples were used for learning, and the remaining 40 examples were used for testing. The threshold for the similarity of a gesture performed by a participant and the target gesture was set to 40%. If the similarity was below 40%, the gesture recognition task was considered unsuccessful. Table 7 lists the average gesture recognition accuracy for each gesture using HMM, Dynamic RNN and DTW. These were 92.00%, 91.98%, and 82.62% for HMM, Dynamic RNN, and DTW, respectively. The highest accuracy was obtained using HMM.

Table 7. Comparison results of recognition accuracy among the hidden Markov model (HMM), dynamic recurrent neural networks (Dynamic RNN), and dynamic time warping (DTW).

Algorithm / **Gesture Type**	**HMM**	**Dynamic RNN**	**DTW**
Gesture 1	94.25	94.17	87.75
Gesture 2	91.92	92.75	82.58
Gesture 3	91.75	91.50	91.67
Gesture 4	92.33	92.08	88.58
Gesture 5	91.50	90.42	77.75
Gesture 6	90.75	90.17	82.75
Gesture 7	92.25	91.08	78.25
Gesture 8	91.58	90.75	78.67
Gesture 9	92.08	91.08	74.17
Gesture 10	91.58	92.92	74.92
Total average recognition accuracy	92.00%	91.69%	81.71%

5.6. Gesture Recognition Stage Result

In addition to the above experiments, a scenario was designed using the 10 defined gestures to evaluate the recognition performance of the proposed method for each gesture. Table 8 shows the testing performance scenario. Between two gestures, a 1 s interval was added to distinguish between the start and end of a gesture.

Table 8. Testing scenarios to evaluate recognition performance.

1	2	3
		

Table 8. *Cont.*

4	5	6
		
7	8	9
		
10	11	12
		
13	14	15
		
16	17	18
		
19	20	21
		
22	23	24
		
25	26	27
		
28	29	30
		

Table 9 lists the recognition rates for the testing scenario, which suggest that all gestures were recognized correctly even when multiple gestures were performed consecutively. Therefore, the recognition performance rate was 90% on average.

Table 9. Gesture recognition results for scenarios.

Scene No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Similarity rate	86.3%	97.8%	24.3%	65.4%	74.3%	85.4%	24.3%	74.3%	94.3%	54.3%
Recognition result	True	True	False	True	True	True	False	True	True	True
Scene no.	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Similarity rate	84.3%	75.3%	43.2%	95.5%	43.4%	83.2%	64.3%	74.3%	62.1%	77.2%
Recognition result	True	True	True	True	True	True	True	True	True	True
Scene no.	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Similarity rate	85.2%	4.3%	64.3%	86.5%	48.1%	84.3%	94.3%	84.3%	74.3%	79.9%
Recognition result	True	False	True	True	True	True	True	True	True	True
Avg. recognition accuracy							90.0%			

6. Conclusions

In this paper, a method for gesture recognition and learning was developed that can learn precise and detailed gestures using intelligent big data obtained from multiple sensors and an appropriate learning algorithm. This proposed method was integrated with interaction systems by using a communication network. In the proposed method, the collected gestures were visualized, and a GUI-based UI was developed to enable non-experts to input their gestures into the proposed method. The proposed method also allows end users to re-edit, delete, and re-record gestures to improve the gesture recognition rate. The experimental results showed that the end user can learn gestures more intuitively. Moreover, the proposed method achieved a recognition rate of 90% on average.

The proposed gesture recognition and learning method has applications in gesture recognition tasks in various interaction systems because of its integration with different types of sensors. Therefore, in the future, the proposed method can potentially interface with new types of sensors to learn and recognize more complex and detailed gestures.

Author Contributions: J.P. and Y.J. wrote the paper; Y.J. performed experiments; S.C. provided full guidance; and Y.S. and K.C. analyzed the results of the experiments and revised the paper.

Funding: This research was supported by BK21 Plus project of the National Research Foundation of Korea Grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Preechasuk, J.; Piamsa-nga, P. Event Detection on Motion Activities Using a Dynamic Grid. *J. Inf. Process. Syst.* **2015**, *11*, 538–555. [CrossRef]
2. Song, W.; Sun, G.; Feng, N.; Fong, S.; Cho, K. Real-time infrared LED detection method for input signal positioning of interactive media. *J. Converg.* **2016**, *7*, 1–6.
3. Lin, J.; Ding, Y. A temporal hand gesture recognition system based on hog and motion trajectory. *Optik Int. J. Light Electron. Opt.* **2013**, *124*, 6795–6798. [CrossRef]
4. Zeineb, A.; Chaala, C.; Frikha, T.; Hadriche, A. Hand gesture recognition system. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 449–453.
5. Khan, R.Z.; Ibraheem, N.A. Comparative study of hand gesture recognition system. In Proceedings of the International Conference of Advanced Computer Science & Information Technology in Computer Science & Information Technology, Bangalore, India, 2–4 January 2012; Volume 2, pp. 203–213.
6. Da Gama, F.; Chaves, M.; Figueiredo, S.; Baltar, A.; Meng, M.; Navab, N.; Teichrieb, V.; Fallavollita, P. MirrARbilitation: A clinically-related gesture recognition interactive tool for an AR rehabilitation system. *Comput. Methods Programs Biomed.* **2016**, *135*, 105–114. [CrossRef] [PubMed]

7. Hachaj, T.; Ogiela, M.R. Rule-based approach to recognizing human body poses and gestures in real time. *Multimed. Syst.* **2014**, *20*, 81–99. [CrossRef]
8. Ren, Z.; Meng, J.; Yuan, J. Depth camera based hand gesture recognition and its applications in human-computer-interaction. In Proceedings of the 2011 8th International Conference on Information, Communications and Signal Processing (ICICS), Singapore, 13–16 December 2011; pp. 1–5.
9. Bautista, A.; Hernández-Vela, A.; Escalera, S.; Igual, L.; Pujol, O.; Moya, J.; Violant, V.; Anguera, M.T. A Gesture Recognition System for Detecting Behavioral Patterns of ADHD. *IEEE Trans. Cybern.* **2016**, *46*, 136–147. [CrossRef]
10. Zou, Y.; Xiao, J.; Han, J.; Wu, K.; Li, Y.; Ni, M. Grfid: A device-free rfid-based gesture recognition system. *IEEE Trans. Mob. Comput.* **2017**, *16*, 381–393. [CrossRef]
11. Kellogg, B.; Talla, V.; Gollakota, S. Bringing Gesture Recognition to All Devices. In Proceedings of the NSDI 14, Seattle, WA, USA, 2–4 April 2014; pp. 303–316.
12. Prakash, A.; Swathi, R.; Kumar, S.; Ashwin, T.S.; Reddy, G.R.M. Kinect Based Real Time Gesture Recognition Tool for Air Marshallers and Traffic Policemen. In Proceedings of the 2016 IEEE Eighth International Conference on Technology for Education (T4E), Mumbai, India, 2–4 December 2016; pp. 34–37.
13. Mi, J.; Sun, Y.; Wang, Y.; Deng, Z.; Li, L.; Zhang, J.; Xie, G. Gesture recognition based teleoperation framework of robotic fish. In Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, China, 3–7 December 2016; pp. 137–142.
14. Xiao, Y.; Zhang, Z.; Beck, A.; Yuan, J.; Thalmann, D. Human-robot interaction by understanding upper body gestures. *Presence Teleoper. Virtual Environ.* **2014**, *23*, 133–154. [CrossRef]
15. Bang, G.; Yang, J.; Oh, K.; Ko, I. Interactive Experience Room Using Infrared Sensors and User's Poses. *J. Inf. Process. Syst.* **2017**, *13*, 876–892. [CrossRef]
16. Sung, Y.; Choi, R.; Jeong, Y.-S. Arm Orientation Estimation Method with Multiple Devices for NUI/NUX. *J. Inf. Process. Syst.* **2018**, *14*, 980–988. [CrossRef]
17. Lara, D.; Labrador, A. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1192–1209. [CrossRef]
18. Hasan, H.; Abdul-Kareem, S. Human-computer interaction using vision-based hand gesture recognition systems: A survey. *Neural Comput. Appl.* **2014**, *25*, 251–261. [CrossRef]
19. Oyuntungalag, C.; Jiro, T. Gesture Input as an Out-of-band Chanel. *J. Inf. Process. Syst.* **2014**, *10*, 92–102. [CrossRef]
20. Song, W.; Liu, L.; Tian, Y.; Sun, G.; Fong, S.; Cho, K. A 3D localisation method in indoor environments for virtual reality applications. *Hum. Centric Comput. Inf. Sci.* **2017**, *7*, 1–11. [CrossRef]
21. Zhu, J.; San-Segundo, R.; Pardo, J.M. Feature extraction for robust physical activity recognition. *Hum. Centric Comput. Inf. Sci.* **2017**, *7*, 1–16. [CrossRef]
22. Gao, H.; Xia, S.; Zhang, Y.; Yao, R.; Zhao, J.; Niu, Q.; Jiang, H. Real-Time Visual Tracking with Compact Shape and Color Feature. *Comput. Mater. Contin.* **2018**, *55*, 509–521. [CrossRef]
23. Ibañez, R.; Soria, Á.; Teyseyre, A.; Campo, M. Easy gesture recognition for Kinect. *Adv. Eng. Softw.* **2014**, *76*, 171–180. [CrossRef]
24. Signer, B.; Kurmann, U.; Norrie, M. iGesture: A general gesture recognition framework. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, Brazil, 23–26 September 2007; pp. 954–958.
25. Zhang, X.; Chen, X.; Li, Y.; Lantz, V.; Wang, K.; Yang, J. A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2011**, *41*, 1064–1076. [CrossRef]
26. Truong, A.; Zaharia, T. Laban movement analysis and hidden Markov models for dynamic 3D gesture recognition. *EURASIP J. Image Video Process.* **2017**, *2017*, 52. [CrossRef]
27. Ma, M.; Park, D.; Kim, S.; An, S. Online Recognition of Handwritten Korean and English Characters. *J. Inf. Process. Syst.* **2012**, *8*, 653–668. [CrossRef]
28. Borghi, G.; Vezzani, R.; Cucchiara, R. Fast Gesture Recognition with Multiple Stream Discrete HMMs on 3D Skeletons. In Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; pp. 997–1002.
29. Suma, E.A.; Lange, B.; Rizzo, A.S.; Krum, D.M.; Bolas, M. Faast: The flexible action and articulated skeleton toolkit. In Proceedings of the 2011 IEEE Virtual Reality Conference, Singapore, 19–23 March 2011; pp. 247–248.

30. Gillian, N.; Paradiso J, A. The gesture recognition toolkit. *J. Mach. Learn. Res.* **2014**, *15*, 3483–3487.
31. Gris, I.; Camacho, A.; Novick, D. Full-Body Gesture Recognition for Embodied Conversational Agents: The UTEP AGENT Gesture Tool. In Proceedings of the Gesture and Speech in Interatcion (GESPIN 4), Nantes, France, 2–4 September 2015; pp. 131–136.
32. Pieter-Jan, M.; Denis, A.; Marc, L. Dance-the-Music: And educational platform for the modeling, recognition and audiovisual monitoring of dance steps using spatiotemporal motion templates. *EURASIP J. Adv. Signal. Process.* **2012**, *2012*, 35. [CrossRef]
33. Yavşan, E.; Uçar, A. Gesture imitation and recognition using Kinect sensor and extreme learning machines. *Measurement* **2016**, *94*, 852–861. [CrossRef]
34. Du, Y.; Wang, W.; Wang, L. Hierarchical Recurrent Neural Network for Skeleton Based Action Recognition. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
35. Choi, H.; Kim, T. Modified Dynamic Time Warping Based on Direction Similarity for Fast Gesture Recognition. *Math. Probl. Eng.* **2018**, *2018*. [CrossRef]
36. Alba, R.; Dawid, W.; Mariusz, O. An Approach to Gesture Recognition with Skeletal Data Using Dynamic Time Warping and Nearest Neighbour Classifier. *Int. J. Intell. Syst. Appl.* **2016**, *6*, 1–8. [CrossRef]