Received October 30, 2020, accepted November 17, 2020, date of publication November 19, 2020,
date of current version December 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3039401

Deep Learning Based Real-Time Recognition of Dynamic Finger Gestures Using a Data Glove

MINHYUK LEE AND JOONBUM BAE, (Member, IEEE)
Department of Mechanical Engineering, Ulsan National Institute of Science and Technology, Ulsan 44919, South Korea

Corresponding author: Joonbum Bae (jbbae@unist.ac.kr)

This work was supported by the National Research Foundation (NRF) of Korea Grant, Korean Government (MSIT), under Grant NRF-2019R1A2C2084677 and Grant NRF-2016R1A5A1938472.

ABSTRACT In this article, a real-time dynamic finger gesture recognition using a soft sensor embedded data glove is presented, which measures the metacarpophalangeal (MCP) and proximal interphalangeal (PIP) joint angles of five fingers. In the gesture recognition field, a challenging problem is that of separating meaningful dynamic gestures from a continuous data stream. Unconscious hand motions or sudden tremors, which can easily lead to segmentation ambiguity, makes this problem difficult. Furthermore, the hand shapes and speeds of users differ when performing the same dynamic gesture, and even those made by one user often vary. To solve the problem of separating meaningful dynamic gestures, we propose a deep learning-based gesture spotting algorithm that detects the start/end of a gesture sequence in a continuous data stream. The gesture spotting algorithm takes window data and estimates a scalar value named gesture progress sequence (GPS). GPS is a quantity that represents gesture progress. Moreover, to solve the gesture variation problem, we propose a sequence simplification algorithm and a deep learning-based gesture recognition algorithm. The proposed three algorithms (gesture spotting algorithm, sequence simplification algorithm, and gesture recognition algorithm) are unified into the real-time gesture recognition system and the system was tested with 11 dynamic finger gestures in real-time. The proposed system took only 6 ms to estimate a GPS and no more than 12 ms to recognize the completed gesture in real-time.

INDEX TERMS Artificial neural network, data glove, data compression, dynamic gesture recognition, human-computer interaction, pattern recognition, real time system, recurrent neural network.

I. INTRODUCTION

Devices and techniques that facilitate human-computer interaction (HCI) have attracted a great deal of interest. Hand gesture recognition is accepted as an effective natural interface, and both static and dynamic gesture recognition have been studied. Hand gesture recognition can be achieved via vision-based and data glove-based approaches. To date, most work on hand gesture recognition has employed vision sensors because few data gloves are commercially available, and most are expensive and hinder natural hand motion [1]–[3].

Hand gestures can be divided into static and dynamic hand gestures. A static hand gesture is a particular hand configuration and pose without hand movement, whereas a dynamic hand gesture is a moving gesture that includes movements of the fingers or hand. Since the two types of gestures have different characteristics, different types of algorithms are required to recognize each type of gesture. Static hand gesture recognition can be achieved by applying standard pattern recognition techniques such as template matching [4], whereas dynamic hand gesture recognition requires time-series pattern recognition algorithms such as hidden Markov models (HMMs) or dynamic time warping (DTW) algorithms [5]. Recently, there have been some studies that focused on recognition of dynamic hand gestures by applying recurrent neural networks (RNNs) [6].

Unlike static hand gesture recognition, dynamic hand gesture recognition must overcome a gesture spotting problem [7]. Gesture spotting segments meaningful gestures from the continuous data stream. In other words, a dynamic hand gesture recognition system must detect the start and end of gestures in a continuous data stream. One of the easiest ways to solve the gesture spotting problem is to define a certain static gesture that indicates the start or end of dynamic gestures [8]. However, this approach disturbs the natural flow of an intended sequence of gestures. Thus, other approaches

The associate editor coordinating the review of this manuscript and approving it for publication was Junhua Li.

have been studied that naturally detect the start and end of dynamic gestures in continuous sequence of hand motion.

One of the most common approaches to gesture spotting is to detect a moment when the hands are stationary [9]–[12]. Simao *et al*. proposed a gesture segmentation method that involves analyzing velocity and acceleration and does not require previous training or knowledge of the gestures or the sequence in which they are made [9]. Molchanov *et al*. assumed that a true gesture occurred only when a radar sensor detected significant motion with a velocity above a configurable threshold (0.05 m/s) [10]. Neverova *et al*. introduced a binary classifier to distinguish resting moments from periods of activity [11]. Li *et al*. proposed a dynamic gesture spotting algorithm based on evidence theory after analyzing changes in the speed of hand movements during dynamic gestures [12]. A limitation of these approaches is a false-positive problem: when the hands are stationary and in a meaningless posture, these techniques may interpret this as the start or end of known dynamic gestures. Thus, analyzing the data patterns in a given window using only ‘engineered features’ such as velocity or acceleration cannot provide a perfect solution for gesture spotting. Recently, there have been studies of gesture spotting using convolutional neural networks (CNNs) [13], [14]. A CNN is a well-known feature extraction method that has been widely used in computer vision. The role of the CNN in gesture spotting is to extract spatio-temporal features that cannot be defined physically, such as hand shape or finger configuration. However, using CNNs requires a graphics processing unit (GPU), the size of which makes it difficult to use the gesture recognition system as a portable application.

In this article, we introduce real-time dynamic gesture recognition using a soft sensor embedded data glove. The data glove measures the angles of metacarpophalangeal (MCP) and proximal interphalangeal (PIP) joints of each finger directly; thus, there is no need to apply any further feature extraction methods to obtain information about finger configuration. Furthermore, we developed three algorithms to achieve real-time dynamic gesture recognition.

Contributions of this article are as follows; First, a gesture spotting algorithm was developed which estimates a gesture progress sequence (GPS). A GPS is a scalar value between 0 and 1 that indicates the extent of gesture progress. For instance, a small GPS value indicates when a gesture is about to start, and a value close to one indicates when a gesture is about to end. The gesture spotting algorithm takes a short data sequence and estimates the current GPS value using a deep learning architecture consisting of long-short term memory (LSTM) and fully connected layers. The deep learning architecture learns features for the gesture spotting by itself and the number of features can be adjusted by changing the number of hidden units in each network layers. Thus, it is possible to detect start/end of dynamic gestures based on more various features compared to previous methods which use limited number of engineered features that cannot cover all aspects of start/end of gestures. From the estimated GPS, a complete data sequence of the current gesture is made.

Second, a sequence simplification algorithm was developed. After a gesture sequence is separated from a continuous data stream, the sequence simplification algorithm removes data in the gesture sequence that do not has notable changes. Thus, the output of the sequence simplification algorithm is not a continuous time series data anymore. It helps to reduce variation resulting from the speed of performing gestures. Furthermore, the sequence simplification enables gesture recognition more faster and more accurately.

Third, a gesture recognition algorithm was developed. The gesture recognition algorithm uses a deep learning architecture to recognize the gesture class from the output of the sequence simplification algorithm. The deep learning architecture was applied to the gesture recognition to cover other types of variation of dynamic gestures which cannot be removed by the sequence simplification. 10 dynamic hand gestures used in American Sign Language (ASL) and one pinching gesture were selected to verify the performance of the proposed three algorithms.

The remainder of this article is organized as follows. Section II introduces previous studies related to our work. Section III describes a data collection process of 11 gestures using the data glove. In Section IV, the gesture spotting algorithm, the sequence simplification algorithm, and the gesture recognition algorithms are introduced. In Section V, off-line and real-time experimental results are presented. In Section VI, limitations of this study are discussed, and Section VII presents our conclusions regarding this study.

II. RELATED WORKS

Our work is related to two research areas: 1) data glove-based gesture recognition, and 2) gesture spotting for continuous dynamic gesture recognition.

Over the few decades, various types of data glove-based gesture recognition system have been developed [15]–[18]. Recently, Pawel *et al*. proposed a system for quick and effective recognition of gestures of hand body language based on data from a glove equipped with ten sensors [19]. They used 22 hand gestures including shifting and rotation of hand and movement of fingers. The data glove consists of five finger flexion sensors, three accelerometers, and two gyroscopes. However, there was no consideration on real-time recognition or continuous gesture recognition that is the challenging problem on real world applications. Jakub *et al*. presented an accelerometer glove-based sign language gesture recognition system. They used seven accelerometers in total, five on the each fingers, one on the wrist, and one on the arm. For the sign language recognition, they adopted Parallel Hidden Markov Model (PaHMM) which was firstly used in a Automatic Speech Recognition (ASR) systems. Although their recognition system can classify 40 gestures, the system was only verified on offline experiments. Chaithanya *et al*. developed a data glove embedding a gesture classifier which detects 22 static hand gestures taken from the French sign language alphabet [20]. They used five IMU sensors on each fingertips and applied the complementary filter to obtain angular positions of each fingers. For the classification algorithms, several machine learning approaches such as the support vector machine, the naive Bayes, the multi layer perceptrons, and the random forest were examined.

In terms of the gesture spotting for continuous dynamic gesture recognition, most studies have been done on the vision sensor based gesture recognition. Nguyen *et al.* proposed a continuous dynamic hand gesture recognition for RGB video input [21]. The recognition system contains the gesture spotting module and the gesture classification module. The gesture spotting module that consists of bidirectional long short-term memory (Bi-LSTM) receives the motion of the hand palm and finger movements and determines whether the current frame is a gesture frame or a transition frame. Although the approach for continuous dynamic gesture recognition of [21] is similar to ours in the view of using different deep learning architectures on gesture spotting and gesture recognition, the gesture spotting method is different from our method. While the gesture spotting module proposed in [21] requires to use a whole data stream containing continuous dynamic gestures because of using Bi-LSTM, our gesture spotting algorithm takes a short sequence of past data. Thus, our recognition system works in real-time, whereas the system proposed in [21] can only be used in offline. Recently, Gibran *et al.* proposed a continuous finger gesture spotting and recognition setup which considers the driving distractions [22]. In [22], the author addressed three challenging tasks of recognition of finger gestures, i) similarities between gesture and non-gesture frame, ii) the difficulty in identifying the temporal boundaries of continuous gestures, and iii) the intraclass variability of gestures' duration. To overcome the tasks i) and ii), the author in [22] proposed gesture spotting method where continuous gestures are segmented by detecting boundary frames and evaluating hand similarities between the start and end boundaries of each gesture. For the task iii), a gesture recognition based on a temporal normalization of features extracted from the set of spotted frames was developed. The main idea of gesture spotting method in [22] is to detect a certain boundary posture enclosing each dynamic gesture. Thus, there is a limitation to selecting recognizable gestures because each gestures must have same postures at the start and the end.

Regarding the gesture spotting method, Becattini *et al.* introduced a concept of *action progress prediction* [23]. *Action progress prediction* is quantitatively defined into two ways, i) a linear interpretation which is versatile and can be applied to any sequence annotated for action detection, and ii) a phase-based interpretation where actions are precisely split into sub-events and manually annotated to obtain a richer representation that captures non-linear dynamics. Among these two definitions, i) is similar to the concept of GPS proposed in our work. Moreover, to predict the *action progress prediction*, [23] used LSTM layers as well. However, the prediction system proposed in [23] was applied to computer images and the target actions were not gestures, but the human activities such as *diving*, *golf swing*, or *surfing*.

III. GESTURE DATA COLLECTION

A. HARDWARE DESCRIPTION

Fig. 1(a) shows how the gesture data were collected with the data glove. Ten soft sensors were embedded in the data glove to measure the MCP and PIP joint angles of the five fingers (Fig. 1(b)). The sensors were made via direct writing of eutectic Gallium Indium (eGaIn); the detailed fabrication process is well described in [24]. As a finger flexes, the resistance of the corresponding sensor, which is linearly fitted from the clenched fist (90° flexion) to flat position (0°), increases (Fig. 1(c)). The changes in resistance are collected and amplified by a customized circuit and the data are transmitted to a computer via Bluetooth. MATLAB was used for collecting gesture data and developing the proposed algorithms. The sampling frequency was 100 Hz.

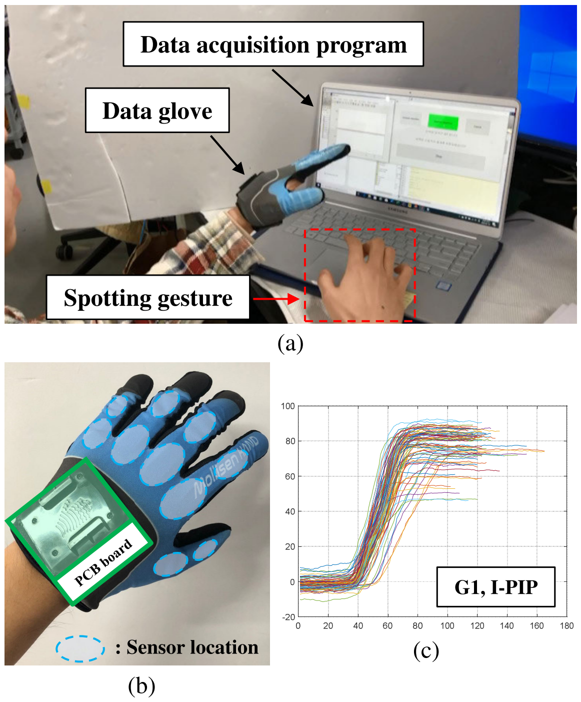


FIGURE 1. (a) The data collection process: the subject manually determines start/end of each gesture from a continuous data stream. (b) The data glove used in this study. (c) An example of collected data obtained from the sensor located at PIP joint of the index finger when performing G1.

B. DATASET GENERATION

Figure 2 shows the 11 dynamic finger gestures used in this study. Except for gesture G6, the gestures were selected from ASL. Notice that the data glove measures only finger joint angles, not three-dimensional hand movements. Thus, the dynamic finger gestures were selected by considering existence of movements of finger flexion/extension and whether a gesture is distinguishable from other gestures only using finger movements.

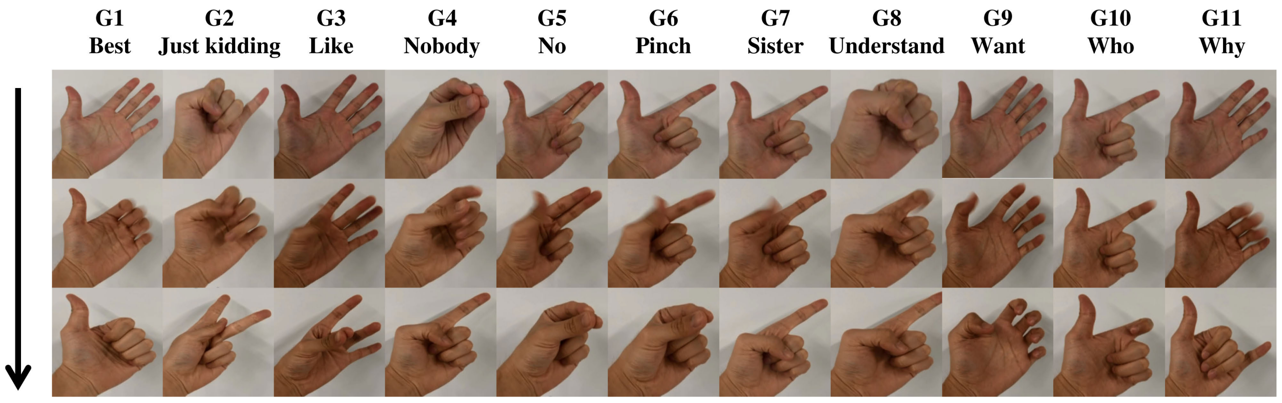


FIGURE 2. **Eleven dynamic hand gestures. G6 was defined intuitively and the other 10 gestures were selected from American Sign Language (ASL).**

We made training and test dataset from several sets of experiment for a few days. For each set of experiment, a subject put on the data glove and took a calibration process before collecting gesture data. During the calibration process, 10 resistance signals of a flat (0°) and fist (90°) hand positions were stored. Then, we linearly fitted resistance data into finger joint angle data using the stored resistance data of both hand positions.

As described in Fig. 1(a), a subject performed dynamic finger gestures using his left hand and manually determined start/end of each gesture by clicking a graphical user interface (GUI) button on a monitor using his right hand during the experiment. Before starting a gesture and after finishing a gesture, the subject did not move his fingers for a short time to ensure the start/end of each gesture. In one set of experiment, 11 gestures were performed in an order and each gesture was performed five times repetitively. Thus, we obtained 55 gesture data (11 gestures$\times$5 repetition=55 gesture data) after finishing a set of experiment. Each gesture samples was saved by separating it from the continuous data stream using the start/end marking determined by the subject. In total, sixteen sets of gesture data were used as a training dataset (80 samples per each gesture) and four sets of gesture data were used as a test dataset (20 samples per each gesture). Note that the training and test dataset consist of isolated gesture samples with corresponding gesture labels. Additionally, to test the gesture spotting performance from continuous data stream, another test data consisting of 11 gestures performed successively including gesture transition was made. Both training and test dataset were used for the gesture spotting algorithm and the gesture recognition algorithm. Although the gesture spotting algorithm and the gesture recognition algorithm share the same dataset, they used the dataset in a different way. Detail explanation is addressed in Section III.A and III.C.

Uncontrolled variation of each gesture samples, such as a moment that the gesture was started/ended, speed of joint angle changes, or the duration of maintaining the start/end pose are contained in the collected dataset. Due to the variation, three problems are arisen. First, quantitative definitions of start/end of each gesture samples are required. Second, there is an imbalance of duration of gesture stages (start/middle/end of each gesture) in the dataset. Third, different patterns of a same gesture caused by a speed variation makes difficult to train the gesture recognition algorithm. First and second problems are addressed in Section III.A, and third problem is discussed in Section III.B.

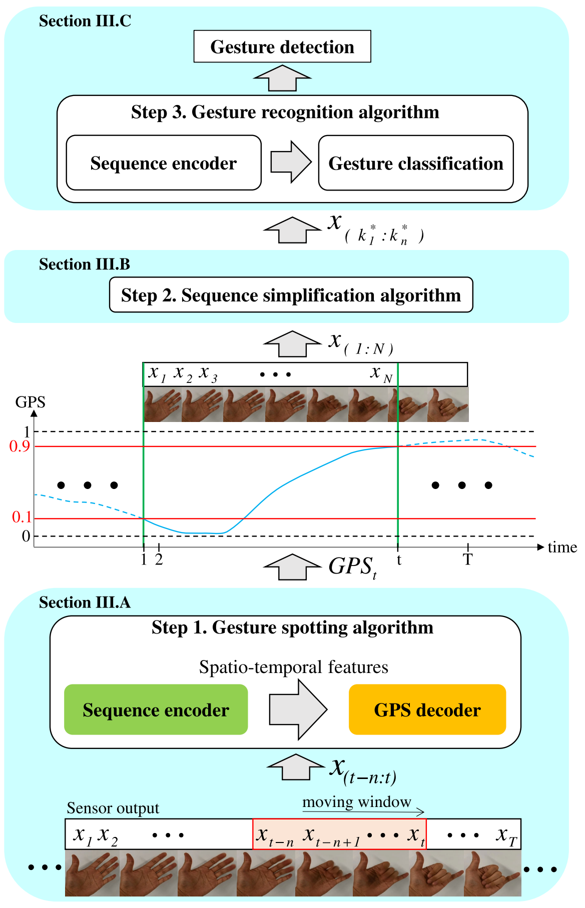


FIGURE 3. **Proposed real-time gesture recognition algorithm.**

IV. ALGORITHMS FOR REAL-TIME GESTURE RECOGNITION

Figure 3 shows the proposed real-time gesture recognition algorithm. As shown in the figure, the recognition system consists of three different algorithms, the gesture spotting algorithm, the sequence simplification algorithm, and the gesture recognition algorithm. The gesture spotting algorithm is operated in real-time to estimate GPS for every time step. Estimated GPS enables to determine start/end of a current gesture, thus a complete gesture sequence can be separated in the continuous data stream. The gesture sequence is pre-processed by passing through the sequence simplification algorithm, and the gesture recognition algorithm takes the pre-processed gesture sequence to detect the gesture class. In Section III, each algorithm is introduced in detail at each subsection.

A. GESTURE SPOTTING ALGORITHM

The gesture spotting algorithm detects the start and end points of gestures in a continuous data stream. To achieve gesture spotting, most previous studies have proposed certain features applying physical quantities such as a velocity, an acceleration, or a frequency. However, only using such 'engineered features' for gesture spotting has a limitation because the number of features is not enough to distinguish start/end of true gestures from natural hand motions including non-gestures or unconscious hand motions. Thus, we propose the new gesture spotting algorithm to find more features by employing a deep learning architecture which can find features on its own during training.

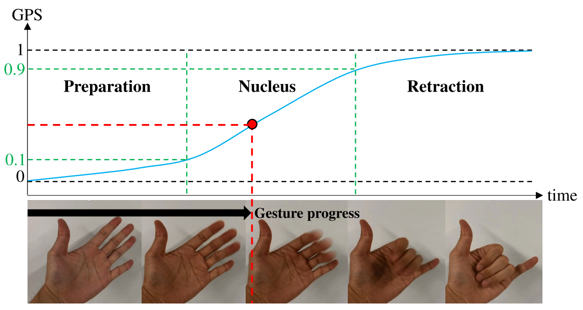


FIGURE 4. **A conceptual illustration of the Gesture Progress Sequence (GPS).**

The proposed deep learning based gesture spotting algorithm estimates the gesture progress sequence (GPS) which is a new concept defined in this study. The GPS is a scalar value between 0 and 1. This is used to demarcate the start and end point of gestures. For instance, when a gesture is about to start, the GPS value is close to zero. In contrast, when the gesture is about to end, the GPS value is close to one. Figure 4 shows the concept of the GPS. Basically, the GPS is defined as the cumulative sum of the velocity norms for each time step. The mathematical expression for the GPS is as follows.

$$v(t) = |p(t) - p(t-1)|, (t = 2, 3, \dots, T, v(1) = 0)$$
$$V_{sum}(t) = \sum_{i=1}^{t} v(i)$$
$$GPS = \{\frac{V_{sum}(1)}{V_{sum}(T)} = 0, \frac{V_{sum}(2)}{V_{sum}(T)}, \dots, \frac{V_{sum}(T)}{V_{sum}(T)} = 1\} \quad (1)$$

In (1), $p(t) \in \mathbb{R}^{10}$ is a finger joint angle vector at time step t, $v(t) \in \mathbb{R}$ is the second norm of the difference between the finger joint angle vector at time steps t and $t-1$. Please note that the GPS only expresses an amount of gesture progress regardless of gesture classes. As can be seen in Fig. 4, the gesture can be divided into three different stages based on the change in the slope of the GPS: 'preparation', 'nucleus', and 'retraction'. Li *et al.* stated that each stage can be distinguished by the change in the speed of the hand motion [12]. This supports the definition of the gesture stages using the GPS that is related to the speed of the finger joint angle vector. As described in Fig. 4, we decided to define the three gesture stages using the GPS thresholds 0.1 and 0.9 based on our experiments. A gesture sequence can be obtained in continuous data stream based on these two GPS thresholds.

The gesture spotting algorithm uses a window of finger joint angle vectors to estimate the GPS. Figure 5 illustrates the architecture of the gesture spotting algorithm consisting of two long-short term memory (LSTM) layers, six fully connected layers, and one output layer. x represents the finger joint angle vector, n represents the window size, t represents the current time step, h represents the hidden unit, and r represents the concatenation of the hidden unit and the last finger joint angle vector of a given window. For both LSTM layers, the number of hidden units was 128. Thus, the dimension of r was 138 because of the concatenation. The concatenation was applied to represent both spatial and temporal features simultaneously. Thus it can be said that r contains the spatiotemporal features of a current finger movement and the GPS is determined based on the spatiotemporal features. For the fully connected layers, a rectified linear unit (ReLU) was used as an activation function. The number of hidden units for all fully connected layers was 128. Finally, for the output layer, a sigmoid function was used as an activation function because the GPS is defined between 0 and 1.

To train the gesture spotting algorithm, the window size which is specified as n in Fig. 5 must be determined appropriately. However, it is not a good approach to set the window size as a fixed length because there is an imbalance among gesture stages for every gesture samples. This is caused by uncontrollable gesture speed and duration of preparation and retraction stages. As shown in Fig. 6, there are variations between gesture stages and even within each gesture stage. This implies that it is inappropriate to set a fixed length window for all gesture data when training the gesture spotting algorithm.

To construct a balanced training dataset, we gathered training data for each gesture stage respectively. For each gesture stages of each gesture samples, the window size was set to be half of the data length of each gesture stage. Then, we set the number of training data that we wanted to generate for each stages to 10. Finally, depending on the amount of training data that we wanted to generate, the window size, and the duration of each stage, a space length between adjacent windows was determined as $d = \lfloor \frac{L-w}{N-1} \rfloor$, where L represents the

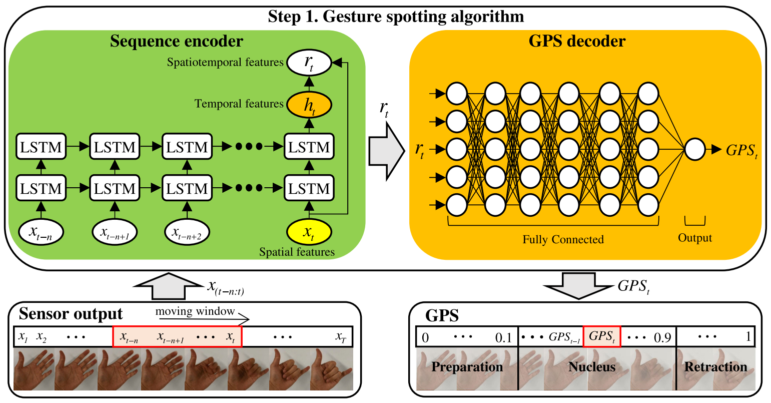


FIGURE 5. The proposed gesture spotting algorithm. For each time step, GPS is estimated from the moving window having a length of *n*.

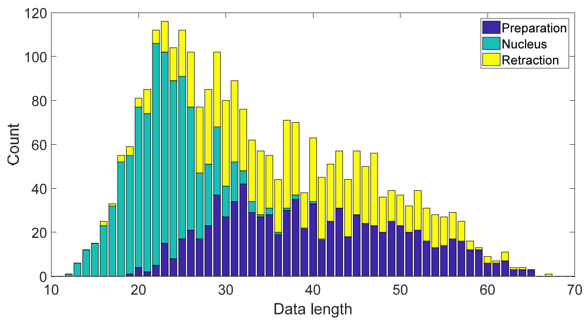


FIGURE 6. Distribution of data length for each gesture stages in training dataset. Note that the maximum data length of gesture stages contained in the training dataset does not exceed 70 because half of this sequence length is used as a fixed window size of the gesture spotting algorithm in the real-time test.

total length of each gesture stages, w represents the window size, and N represents the number of training data that we wanted to generate for each stage. Although a window size of training dataset is adaptively set depending on duration of each gesture stages, it is impossible to adaptively change the window size when we test the algorithm in real-time because a length of gesture stages cannot be known in a continuous data stream. Thus, the window size is fixed as 35 (half of 70) when we test the gesture spotting algorithm because any data lengths of gesture stages in training dataset do not exceed 70.

After generating the training data, the gesture spotting algorithm was trained with Adam optimization algorithm [25], and the learning rate was 0.001. As the cost function, mean square error (MSE) loss was used. The number of epochs for training was 10, and the batch size was 2640.

B. SEQUENCE SIMPLIFICATION ALGORITHM

In this section, we introduce the sequence simplification algorithm which takes a role to remove speed variation of a gesture sequence. The sequence simplification algorithm is a simple feature extractor that finds notable changes in the gesture sequence. The sequence simplification algorithm operates as follows. First, the algorithm finds one sensor out of ten sensors which has the largest difference between the maximum and minimum points of the given gesture pattern measured by the sensors. The mathematical expression of this step is as follows:

$$i^* = \underset{i}{\operatorname{argmax}}(\max_t S^i(t) - \min_t S^i(t)) \tag{2}$$

In (2), i is the sensor index, which ranges from 1 to 10, and S is a set of sensor measurements in the gesture sequence, so $S^i(t)$ is the sensor output at time step t measured by sensor i. Second, as described in Fig. 7(a), a line passing through the start and end points of S^{i^*} is defined. Third, as shown in Fig. 7(b), the location of the data point that has the maximum distance from the line is searched. This can be expressed mathematically as follows:

$$k_1 = \underset{1<k<T}{\operatorname{argmax}}\, d(S^{i^*}(k), \overline{S^{i^*}(1)S^{i^*}(T)}) \tag{3}$$

In (3), T represents the end of the sensor measurement, and the function $d(a, b)$ calculates the distance between point a and line b. Fourth, two lines $\overline{S^{i^*}(1)S^{i^*}(k_1)}$ and $\overline{S^{i^*}(k_1)S^{i^*}(T)}$ are drawn and (3) is applied to both lines to find k_2 and k_3 (Fig. 7(c)). This procedure is repeated until $d_{max}(k_n)$ reaches a pre-defined tolerance. The set $K = \{k_1^*, k_2^*, \ldots, k_n^*\} = sort(\{k_1, k_2, \ldots, k_n\}, ascending)$ is the notable locations of the gesture sequence and the set is applied to other sensor measurements to complete the sequence simplification for the 10-dimensional vector sequence. Figure 7(d) shows an example of the performance of the sequence simplification algorithm, comparing the original sensor output with the simplified result. The data length of the original sensor output is larger than 200, but that of the simplified output is less than 25, with a pre-defined tolerance of 2. If the tolerance is increased, then fewer data points remain, and vice versa. Thus, by changing the tolerance, the simplification scale can be controlled.

The proposed sequence simplification method has two advantages for real-time gesture recognition. First, a length of a given pattern is decreased after processing the sequence simplification, thus computational cost for analyzing gesture

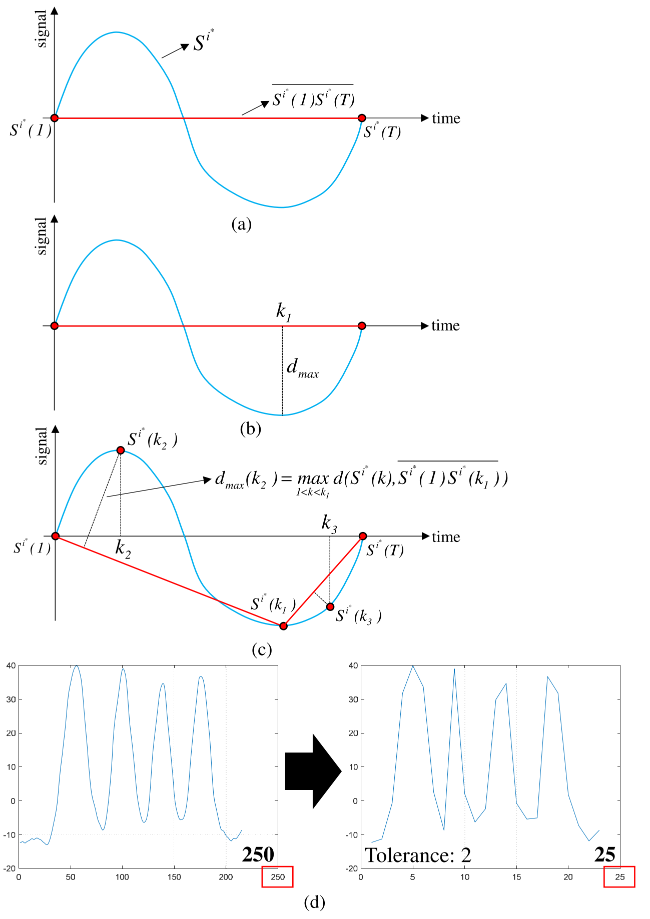

signal
time
(a)
(b)
(c)
(d)
k_1
d_{max}
$d_{max}(k_2) = \max_{1<k<k_1} d(S^{i^*}(k), \overline{S^{i^*}(1)S^{i^*}(k_1)})$
k_2
k_3
250
Tolerance: 2
25

FIGURE 7. (a)~(c) Process of the sequence simplification. (d) An example of applying the sequence simplification algorithm to real sensor output.

patterns is reduced. Second, the sequence simplification only leaves inflection, convex, and concave points and removes the others. Thus, it is possible to effectively diminish variation caused by gesture speed. This is the difference between down-sampling and the proposed sequence simplification method.

C. GESTURE RECOGNITION ALGORITHM

The gesture recognition algorithm takes a simplified pattern obtained by the sequence simplification algorithm and classifies the gesture. A deep learning architecture was adopted to the gesture recognition algorithm to make the gesture recognition more robust to variation of a same gesture which cannot be filtered after sequence simplification. The gesture recognition algorithm consists of two LSTM layers with 64 hidden units for each layer, three fully connected layers with 64 hidden units for each layer, and one output layer that has an output dimension of 11 (Fig. 8). The activation function of the three fully connected layers is ReLU, and a softmax activation function is used for the output layer. One of the differences from the gesture spotting algorithm is that there is no concatenation in the LSTM layer. For the gesture recognition algorithm, it is important to analyze the context of the gesture sequence. Thus, there is no need to concatenate the current hand shape. Notice that the gesture recognition algorithm does not consider the entire gesture sequence. Instead, it considers the gesture sequence from initiation to the moment that the GPS value reaches 0.9 because there is no further changes of hand shapes after GPS passes over 0.9.

The gesture recognition algorithm was trained independently to the gesture spotting algorithm. Both algorithms were trained separately, but shared the same training dataset. When training the gesture recognition algorithm, the GPS was computed for each gesture samples and we removed data that GPS exceeds 0.9. Then, the sequence simplification algorithm was applied for each gesture samples. The output of the sequence simplification was used as the input of the gesture recognition algorithm during training. In training, a cross-entropy loss function was used as the loss function, Adam optimization algorithm was used with a learning rate of 0.001 [25], the number of epochs was 10, and k-fold cross validation was applied with $k = 5$. The cross validation result showed 100% accuracy for the training set and 98.5% accuracy for the validation set.

V. EXPERIMENTAL RESULTS

A. GPS RECOGNITION

Two types of test dataset were used to test the gesture spotting algorithm. First, a test dataset consisting of isolated gestures was applied to the algorithm. The window size was set as 35 and it moves one time step forward for each gesture samples. The algorithm took 6ms in average for calculating the GPS from one window. Examples of the GPS recognition result is shown on Fig. 9(a). In Fig. 9(a), G1, G2, G5, and G7 are selected among 11 gestures because they have different finger poses at preparation and retraction stages. As shown in the figure, the GPS is below 0.1 during the preparation stage and higher than 0.9 during the retraction stage.

Second, a test dataset which contains successively performed 11 gestures was applied. Figure 9(b) shows the experimental result of this test. The ground truth of start/end of each gestures was set by the subject when he decided to start/end each gestures. Note that the GPS exceeds 0.9 before the gesture is completely finished. Thus, it is possible to save some time to operate the sequence simplification algorithm because the sequence simplification is operated right after the estimated GPS reaches to 0.9. Furthermore, although we never trained the algorithm with the data collected during gesture transition, such as changing a gesture from G1 to G2, the algorithm estimated the GPS lying in between 0.1 and 0.9 at the gesture transition. This was possible because the algorithm was trained with the data of the nucleus stage which includes joint angle changes and various hand shapes not included in the preparation and retraction stages. Thus, it can be interpreted that training the GPS recognition algorithm with more dynamic gestures can prevent false-positive problem caused by non-gestures or gesture transition.

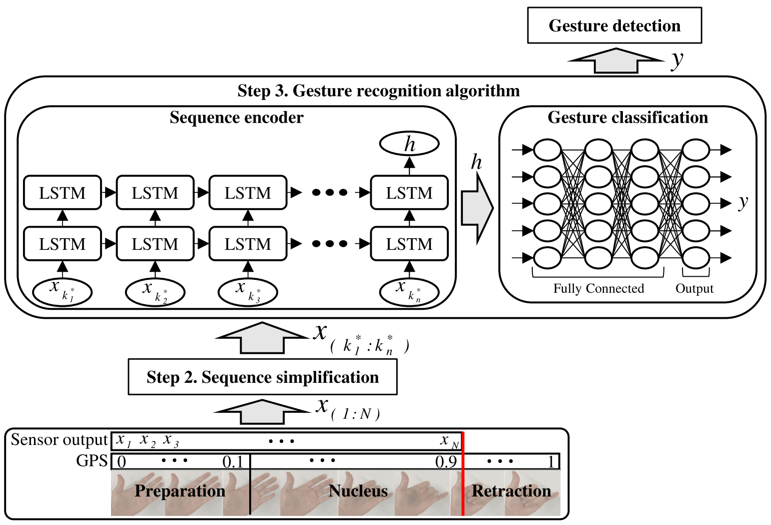


FIGURE 8. The proposed gesture recognition algorithm. Note that there is no concatenation at the output of the LSTM layer.

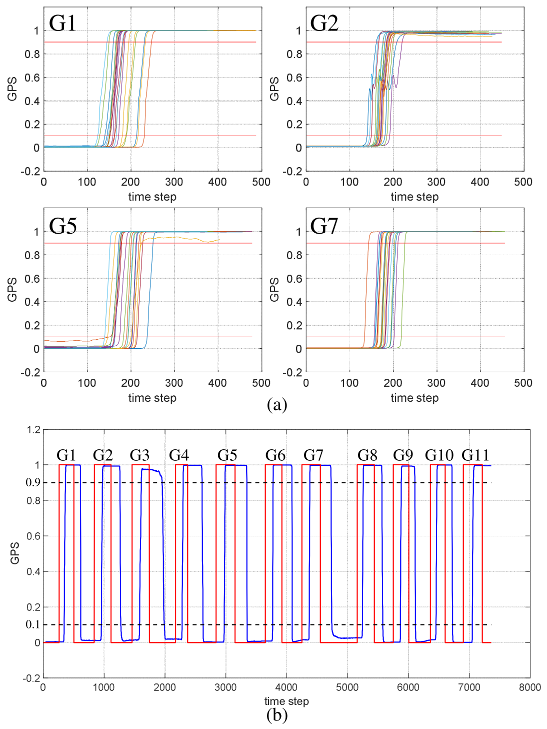
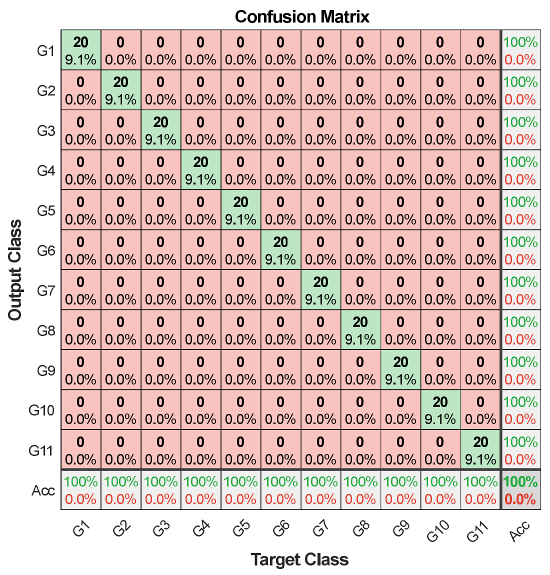


FIGURE 9. (a) GPS recognition for each gesture. Red line indicates the threshold distinguishing three gesture stages and others indicate gesture samples. It is possible to know when each gesture samples is started/ended by observing each GPS graphs passing over the two thresholds. (b) GPS recognition using successively performed 11 gestures. The red line indicates true start/end of each gestures, and the blue line indicates the estimated GPS. Note that the GPS never reaches to higher than 0.9 or lower than 0.1 during gesture transition.


Confusion Matrix

Output Class \ Target Class	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	Acc
G1	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G2	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G3	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G4	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
G9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	0 0.0%	100% 0.0%
G10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	0 0.0%	100% 0.0%
G11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	20 9.1%	100% 0.0%
Acc	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

FIGURE 10. Confusion matrix of the gesture recognition algorithm. Note that the trained gesture recognition algorithm takes inputs that are processed the sequence simplification.

B. GESTURE RECOGNITION

The performance of the gesture recognition algorithm was verified using the test dataset consisting of isolated gestures. To construct same conditions as training, the test dataset was pre-processed by calculating the GPS and cutting each gesture sequence at the GPS value 0.9. After that, the sequence simplification was also applied. The recognition result showed 100% accuracy (Fig. 10) and each gesture sample took 3 ms in average during processing the sequence simplification and the gesture recognition. The sequence simplification algorithm dramatically reduced a length of each gestures as shown in Fig. 11, thus it was possible to achieve fast recognition.

To verify a performance of the sequence simplification algorithm at the gesture recognition step, we tested the gesture recognition algorithm without the sequence simplification process. The sequence simplification process

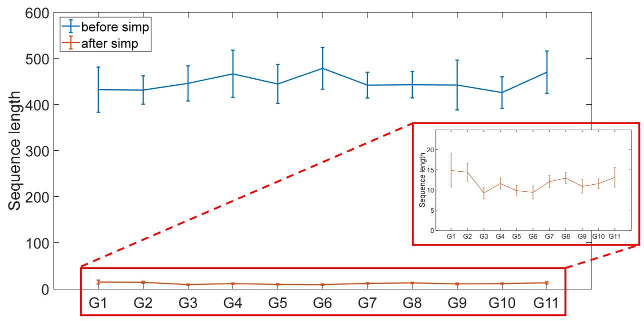


FIGURE 11. Sequence length comparison of before and after applying the sequence simplification algorithm to the test dataset. Blue line indicates a mean sequence length of each gesture classes before applying the sequence simplification. Red line indicates a mean sequence length of each gesture class after applying the sequence simplification. Note that mean sequence lengths of all gesture classes are dramatically reduced (about 4%) and deviations of the sequence lengths are also reduced. The sequence simplification procedure makes the gesture recognition algorithm focus only on features related to a shape of a given sequence, not a length of the sequence.

was removed from both training and testing the gesture recognition algorithm. The test result showed a decrease of recognition accuracy of 55% (G4), 95% (G5) and 75% (G8). Among the three gestures, G4 showed dramatic decrease of recognition accuracy and it was misclassified to G8 mostly. G4 and G8 have similar finger motions, but they are clearly distinguished at the preparation stage. However, if a sequence of G4 or G8 becomes longer than a certain amount, then features at the preparation stage can be faded out at the end of the sequence when applying the LSTM model. As shown in Fig. 11, G4 has a long sequence length in average and large deviation comparing with other gestures. Thus, it can be summarized that the sequence simplification algorithm removes data which have no meaning except connecting a given sequence and improves the gesture recognition performance.

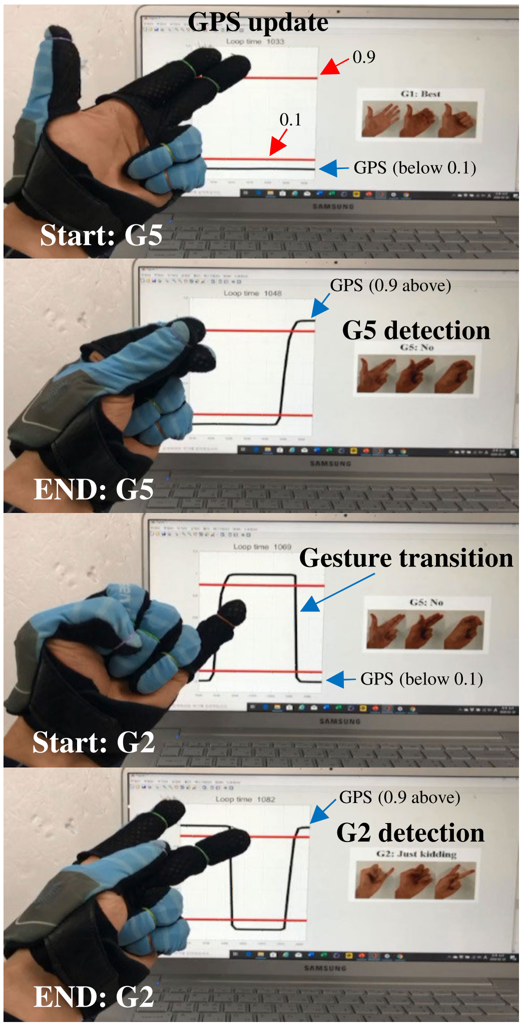


FIGURE 12. Real-time gesture recognition test. The GPS graph changes in real-time and the gesture recognition result is shown on the right side when the gesture is completed.

C. REAL-TIME GESTURE RECOGNITION

After verifying the performance of the three algorithms separately, we combined them into a real-time gesture recognition system. For each time step, a joint angle vector is measured by the data glove and the vector is stored in a fixed size window. The window can keep its size to 35 because old data is removed if new data is entered. The window is passed to the gesture spotting algorithm for every time step and a gesture sequence is generated if the estimated GPS is changed from lower than 0.1 to upper than 0.9 in an order. After a complete gesture sequence is generated, the gesture sequence is transmitted to the sequence simplification algorithm. Finally, the gesture recognition algorithm takes the simplified gesture sequence and the current gesture is classified.

Figure 12 shows images of real-time gesture recognition. In Fig. 12, when the hand was in the start pose of G5, the GPS value is lower than 0.1. Then, if the hand completed G5, the GPS value increased to above 0.9 and the gesture recognition result was shown on the right side. Figure 12 also shows a clean gesture transition from G5 to G2. This implies that there is no problem of gesture transition if gestures are performed in an random order. During real-time test, GPS estimation took only 6 ms on average and gesture recognition took no more than 12 ms; thus, the proposed gesture recognition system can be characterized as a real-time recognition system.

Figure 13 shows examples of GPS output at random hand poses. As shown in the figure, the estimated GPS does not change in between 0.1 and 0.9 because the finger configurations are not included in the preparation or retraction stages of known gestures and the hand is in static. If start/end of a gesture is determined only by engineered features such as velocity, acceleration or frequency related quantities, then all static hand shapes would be regarded as start/end of gestures. On the other hand, the proposed gesture spotting algorithm determines start/end of gestures with more features that cannot be described physically, thus the false-positive problem of the gesture spotting can be prevented.

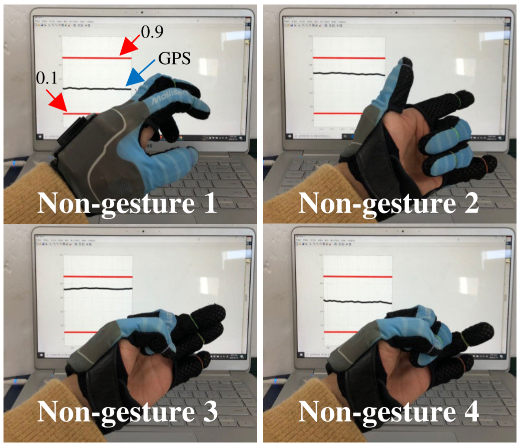


FIGURE 13. Examples of non-gestures. Note that the GPS does not change in between 0.1 and 0.9.

VI. DISCUSSION

In the gesture recognition field, there are two types of approaches: vision-based gesture recognition and data glove-based gesture recognition. To date, the vision-based gesture recognition has been studied widely than the data glove-based gesture recognition because vision sensors are easily available in markets. Most studies of the vision-based gesture recognition commonly use specific types of data such as depth, color or stereo-IR data [26], [27]. Although each data types represents different physical quantities, they are all image data which have 2D or 3D data structures. Thus, there are various public dynamic gesture datasets for developing gesture recognition algorithms and comparing performance of the algorithms. On the other hand, most studies of the data glove-based gesture recognition have developed their own data gloves or have customized data gloves using commercial sensors such as IMUs, optical sensors, tactile sensors, magnetic sensors or flex sensors [18]. Thus, data types obtained from data gloves are varied depending on sensors that composing the data gloves. Since the recognition algorithms and types of recognizable gestures (static/dynamic, finger gesture/hand gesture) highly depend on the input data that data gloves measure, most studies of data glove-based recognition have been selected gesture types and collected gesture data by themselves. Moreover, studies of data glove-based gesture recognition have been focused on classifying static gestures mostly because the classification of static hand gestures is easier than that of dynamic hand gestures [18]. On the other hand, recently, the studies of vision-based gesture recognition have concentrated on recognition of continuous dynamic gestures for the application of sign language translation [28]. Thus, there are public datasets of dynamic gestures for vision based gesture recognition, whereas there is no public dynamic gesture dataset for data glove-based gesture recognition [26], [29]. Lack of large size of public datasets for training and verifying gesture recognition models is a crucial problem for data glove-based gesture recognition research field. Furthermore, public datasets containing continuous dynamic gestures that the start and the end boundaries of each gestures are annotated are required for verifying the performance of the gesture spotting model.

In this study, we proposed two machine learning algorithms for gesture spotting and gesture recognition, respectively. During the real-time operation, the gesture spotting model extracts a complete gesture sequence from a continuous data stream and the gesture recognition model classifies the gesture by receiving the simplified version of the gesture data. Since the gesture spotting model stores a gesture data until the gesture is finished, the gesture recognition model has no choice but to output the recognition result after the gesture is finished. We determined to show the recognition result right after the gesture was finished because we assumed that finger or hand movements can be regarded as a gesture if and only if the starting and the ending postures of a gesture is clear. However, most of state-of-the-art real-time dynamic gesture recognition algorithms have been developed as an end-to-end deep learning model to predict a gesture class for every time step [29], [30]. This means that the recognition result is shown for every time step. Therefore, since the proposed gesture spotting strategy is different from other methods, it is hard to compare the performance of the proposed gesture spotting algorithm with other gesture spotting methods in a fair way.

VII. CONCLUSION

In this study, we proposed a real-time gesture recognition system that uses a data glove. In the gesture recognition research field, the gesture spotting problem has been one of the biggest challenges to real-time recognition. To overcome the problem, we proposed a deep learning based gesture spotting algorithm that estimates a new concept, the gesture progress sequence (GPS). The GPS is a scalar quantity between 0 and 1 that is defined by a cumulative sum of velocity norm for the 10-dimensional finger joint angle vector. Depending on the estimated GPS, it is possible to detect start/end of a current dynamic gesture in a continuous data stream. Next, the sequence simplification algorithm was proposed to compress a gesture sequence into a short sequence by selecting data from the sequence that change notably. This process helps to reduce variation of gestures caused by gesture speed. Finally, the gesture recognition algorithm, which also employed a deep learning architecture, was developed to classify 11 gestures. The three algorithms are unified into one real-time gesture recognition system. The real-time test showed that the proposed method segments a gesture sequence in a continuous data stream that non-gestures are included and detects the known gestures fast and accurate.

Performance of the GPS based gesture spotting approach will be verified with more dynamic gestures including repetitive gestures as a future work. Furthermore, additional sensors will be added such as motion trackers or IMUs to use hand movement information for gesture recognition.

REFERENCES

[1] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Comput. Vis. Image Understand.*, vol. 108, nos. 1–2, pp. 52–73, Oct. 2007.

[2] H. Cooper, B. Holt, and R. Bowden, ''Sign language recognition,'' in *Visual Analysis of Humans*. Berlin, Germany: Springer, 2011, pp. 539–562.
[3] S. Mitra and T. Acharya, ''Gesture recognition: A survey,'' *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 3, pp. 311–324, May 2007.
[4] L. Yun, Z. Lifeng, and Z. Shujun, ''A hand gesture recognition method based on multi-feature fusion and template matching,'' *Procedia Eng.*, vol. 29, pp. 1678–1684, 2012.
[5] G. Plouffe and A.-M. Cretu, ''Static and dynamic hand gesture recognition in depth data using dynamic time warping,'' *IEEE Trans. Instrum. Meas.*, vol. 65, no. 2, pp. 305–316, Feb. 2016.
[6] X. Chai, Z. Liu, F. Yin, Z. Liu, and X. Chen, ''Two streams recurrent neural networks for large-scale continuous gesture recognition,'' in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 31–36.
[7] S. Escalera, V. Athitsos, and I. Guyon, ''Challenges in multi-modal gesture recognition,'' in *Gesture Recognition*. Berlin, Germany: Springer, 2017, pp. 1–60.
[8] K. Murakami and H. Taguchi, ''Gesture recognition using recurrent neural networks,'' in *Proc. SIGCHI Conf. Human Factors Comput. Syst. Reaching Through Technol. (CHI)*, 1991, pp. 237–242.
[9] M. A. Simao, P. Neto, and O. Gibaru, ''Unsupervised gesture segmentation by motion detection of a real-time data stream,'' *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 473–481, Apr. 2017.
[10] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, ''Multi-sensor system for driver's hand-gesture recognition,'' in *Proc. 11th IEEE Int. Conf. Workshops Autom. Face Gesture Recognit. (FG)*, vol. 1, May 2015, pp. 1–8.
[11] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout, ''Multi-scale deep learning for gesture detection and localization,'' in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2014, pp. 474–490.
[12] Q. Li, C. Huang, Z. Yao, Y. Chen, and L. Ma, ''Continuous dynamic gesture spotting algorithm based on dempster-shafer theory in the augmented reality human computer interaction,'' *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 14, no. 5, p. e1931, Oct. 2018.
[13] F. Ordóñez and D. Roggen, ''Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition,'' *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016.
[14] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, ''Deep learning for sensor-based activity recognition: A survey,'' *Pattern Recognit. Lett.*, vol. 119, pp. 3–11, Mar. 2019.
[15] D. L. Quam, ''Gesture recognition with a DataGlove,'' in *Proc. IEEE Conf. Aerosp. Electron.*, May 1990, pp. 755–760.
[16] J. Weissmann and R. Salomon, ''Gesture recognition for virtual reality applications using data gloves and neural networks,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, Jul. 1999, pp. 2043–2046.
[17] R.-H. Liang and M. Ouhyoung, ''A real-time continuous gesture recognition system for sign language,'' in *Proc. 3rd IEEE Int. Conf. Autom. Face Gesture Recognit.*, Apr. 1998, pp. 558–567.
[18] M. A. Ahmed, B. B. Zaidan, A. A. Zaidan, M. M. Salih, and M. M. bin Lakulu, ''A review on systems-based sensory gloves for sign language recognition state of the art between 2007 and 2017,'' *Sensors*, vol. 18, no. 7, p. 2208, 2018.
[19] P. Plawiak, T. Sosnicki, M. Niedzwiecki, Z. Tabor, and K. Rzecki, ''Hand body language gesture recognition based on signals from specialized glove and machine learning algorithms,'' *IEEE Trans. Ind. Informat.*, vol. 12, no. 3, pp. 1104–1113, Jun. 2016.
[20] C. K. Mummadi, F. P. P. Leo, K. D. Verma, S. Kasireddy, P. M. Scholl, J. Kempfle, and K. V. Laerhoven, ''Real-time and embedded detection of hand gestures with an IMU-based glove,'' *Informatics*, vol. 5, no. 2, p. 28, 2018.
[21] N. N. Hoang, G.-S. Lee, S.-H. Kim, and H.-J. Yang, ''Continuous hand gesture spotting and classification using 3D finger joints information,'' in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 539–543.
[22] G. Benitez-Garcia, M. Haris, Y. Tsuda, and N. Ukita, ''Continuous finger gesture spotting and recognition based on similarities between start and end frames,'' *IEEE Trans. Intell. Transp. Syst.*, early access, Jul. 28, 2020, doi: 10.1109/TITS.2020.3010306.
[23] F. Becattini, T. Uricchio, L. Seidenari, L. Ballan, and A. Del Bimbo, ''Am i done? Predicting action progress in videos,'' 2017, *arXiv:1705.01781*. [Online]. Available: http://arxiv.org/abs/1705.01781
[24] S. Kim, J. Oh, D. Jeong, W. Park, and J. Bae, ''Consistent and reproducible direct ink writing of eutectic gallium–indium for high-quality soft sensors,'' *Soft Robot.*, vol. 5, no. 5, pp. 601–612, 2018.
[25] D. P. Kingma and J. Ba, ''Adam: A method for stochastic optimization,'' 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[26] P. K. Pisharady and M. Saerbeck, ''Recent methods and databases in vision-based hand gesture recognition: A review,'' *Comput. Vis. Image Understand.*, vol. 141, pp. 152–165, Dec. 2015.
[27] S. Berman and H. Stern, ''Sensors for gesture recognition systems,'' *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 277–290, Dec. 2011, doi: 10.1109/tsmcc.2011.2161077.
[28] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden, ''Neural sign language translation,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7784–7793.
[29] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, ''Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4207–4215.
[30] D. Wu, L. Pigou, P.-J. Kindermans, N. D.-H. Le, L. Shao, J. Dambre, and J.-M. Odobez, ''Deep dynamic neural networks for multimodal gesture segmentation and recognition,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1583–1597, Aug. 2016.

MINHYUK LEE received the B.S. degree in mechanical, aerospace and nuclear engineering from the Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea, in 2016.

He is currently pursuing the master's and Ph.D. degree (combined) majoring in mechanical engineering from UNIST. His current research interests include dynamic gesture recognition, deep learning, and human–computer interaction. He received the Korean Government Minister Award from Ministry of Science, ICT and Future Planning in 2017. He was a recipient of the Global Ph.D. Fellowship from 2017 to 2021.

JOONBUM BAE (Member, IEEE) received the B.S. degree *(summa cum laude)* in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2006, and the M.S. degree in mechanical engineering, the M.A. degree in statistics, and the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2008, 2010, and 2011, respectively.

In 2012, he joined the Department of Mechanical Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea, where he is currently the Director of the Bio-Robotics and Control (BiRC) Laboratory. He was appointed as a Rising-Star Distinguished Professor with UNIST in 2018. His current research interests include modeling, design, and control of human–robot interaction systems, soft-robotics, and biologically inspired robot systems.

Prof. Bae received the Korean Government Minister Awards from the Ministry of Public Safety and Security and the Ministry of Science, ICT and Future Planning in 2016 and 2017, respectively. He also received the Young Researcher Award from the Korea Robotics Society in 2015. He founded a startup named Feel the Same, Inc. in 2017, which develops soft sensor systems. He was a recipient of a Samsung Scholarship during his Ph.D. studies.