sensors

MDPI

Article

IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces

Minwoo Kim [1], Jaechan Cho [1], Seongjoo Lee [2] and Yunho Jung [1,*]

[1] School of Electronics and Information Engineering, Korea Aerospace University, Goyang-si 10540, Korea; minwoo@kau.kr (M.K.); jccho@kau.kr (J.C.)

[2] Department of Information and Communication Engineering, Sejong University, Seoul 143-747, Korea; seongjoo@sejong.ac.kr

* Correspondence: yjung@kau.ac.kr; Tel.: +82-2-300-0133

Received: 12 August 2019; Accepted: 2 September 2019; Published: 4 September 2019

Abstract: We propose an efficient hand gesture recognition (HGR) algorithm, which can cope with time-dependent data from an inertial measurement unit (IMU) sensor and support real-time learning for various human-machine interface (HMI) applications. Although the data extracted from IMU sensors are time-dependent, most existing HGR algorithms do not consider this characteristic, which results in the degradation of recognition performance. Because the dynamic time warping (DTW) technique considers the time-dependent characteristic of IMU sensor data, the recognition performance of DTW-based algorithms is better than that of others. However, the DTW technique requires a very complex learning algorithm, which makes it difficult to support real-time learning. To solve this issue, the proposed HGR algorithm is based on a restricted column energy (RCE) neural network, which has a very simple learning scheme in which neurons are activated when necessary. By replacing the metric calculation of the RCE neural network with DTW distance, the proposed algorithm exhibits superior recognition performance for time-dependent sensor data while supporting real-time learning. Our verification results on a field-programmable gate array (FPGA)-based test platform show that the proposed HGR algorithm can achieve a recognition accuracy of 98.6% and supports real-time learning and recognition at an operating frequency of 150 MHz.

Keywords: dynamic time warping (DTW); hand gesture recognition (HGR); inertial measurement unit (IMU); machine learning; real-time learning; restricted coulomb energy (RCE) neural network

1. Introduction

A human-machine interface (HMI) presents information to a user regarding the state of a process, accepts commands, and operates associated devices [1]. HMIs designed for human convenience are aimed at allowing users to freely control devices via simple operations without requiring the user's full attention [2]. Therefore, hand gesture recognition (HGR) is an essential feature of HMIs because it allows users to efficiently control devices with simple hand gestures. HGR gets rid of the limitations of controlling devices, and has been widely used in various fields, such as switching TV channels by drawing numbers in a smart home or turning on the air conditioner with a simple gesture while driving.

HGR can be categorized into vision-based gesture recognition (VGR) and sensor-based gesture recognition (SGR) [3]. VGR is a method of recognizing gestures using camera images, and various technologies have been proposed [4–6]. However, VGR accuracy degrades in light-sensitive application scenarios because camera images are affected by lighting conditions [7]. On the other hand, SGR methods have relatively few limitations because they use various sensors that are not affected

by lighting conditions such as inertial measurement unit (IMU) sensors, electromyography (EMG) sensors, brain wave sensors, electrocardiograph sensors, and radar sensors [8,9]. Among them, IMU sensors, which are generally compact, are widely used for SGR owing to their low cost and low power characteristics [10]. In addition, because IMU sensors can be directly attached to the user's body, they can obtain relatively accurate hand gesture data.

Dynamic time warping (DTW) [11–14], multilayer perceptrons (MLPs) [7,15] and convolutional neural networks (CNNs) [16,17] are widely used to recognize hand gestures with IMU sensors. DTW-based recognition algorithms output the most similar hand gestures by measuring DTW distance between the input data and the representative data of each hand gesture, called templates. Before calculating the distance, DTW is used to align two datapoints via warping with a nonlinear method to make them match each other. Because of this alignment process, these algorithms can cope with deformations in time-dependent data caused by different speeds [18]. DTW-based recognition algorithms exhibit good performance for HGR. However, recognition performance greatly depends on the quality of the selected templates. Therefore, because these algorithms require finding optimal templates from a prepared dataset, it is hard to perform real-time learning with high accuracy, which is one of the most important features for HGR because users want devices to learn their hand gestures immediately. MLPs calculate the weighted sum of input features and are used in many studies because of their high recognition performance [7,15]. The learning algorithm of MLPs uses the gradient descent method to find the optimal weights that minimize recognition errors. These algorithms have a high computational complexity, which makes it difficult to train them in real time. As MLPs become deeper to achieve higher performance, the number of weights increases exponentially, and their computational complexity becomes much higher. In addition, MLPs are vulnerable to the deformation of the input data. To cope with this problem, researchers have come up with three ideas involving CNNs: local receptive fields, shared weights, and spatial subsampling [19]. In CNNs, raw sensor data are used without extracting any features, which represents an advantage in that no data are lost through feature extraction processing [16]. Some studies have demonstrated superior performance in HGR using CNNs, but it is hard to train such network in real time because their learning algorithm is as computationally complex as those of MLPs.

On the contrary, restricted coulomb energy (RCE) neural networks employ a relatively simple training method [20] and can thus perform real-time learning [21]. An RCE neural network consists of neurons, and each neuron has a center point and a radius, which forms an activation region. The training process involves determining the center point of each neuron and changing the size of its radius. In the recognition process, distance information between the center point of each neuron and the input feature data are used to determine whether each neuron is activated. Then, the label of the activated neuron that best matches the input data becomes the recognition result. However, RCE neural networks do not consider the characteristics of time-dependent data when calculating distance information. Therefore, it is hard for an existing RCE neural network to provide high performance in IMU sensor-based HGR.

In this paper, we address the following question: *how can we make a high-performance HGR algorithm that supports real-time learning?* We propose an efficient HGR algorithm that uses a simple learning algorithm based on an RCE neural network and employ the distance measurement method of DTW. To achieve our goal and deliver high-performance gesture recognition, the distance measurement method used in RCE neural networks is removed and that of DTW is employed. A test platform was constructed to verify the real-time operation of the training and recognition algorithms. This platform consisted of an IMU sensor, an Arduino module for data preprocessing, and a field-programmable gate array (FPGA), on which the proposed HGR algorithm was implemented. Performance evaluation results show that the proposed system can achieve an accuracy of 98.6% with a training time of 0.423 ms and a recognition time of 0.426 ms.

The rest of this paper is organized as follows: Section 2 describes the background of RCE neural networks and DTW. Section 3 presents the proposed HGR algorithm, and Section 4 presents the test

platform along with its hardware structure. In Section 5, we present the experimental results of the proposed HGR algorithm. Finally, Section 6 concludes the paper.

2. Backgrounds

2.1. RCE Neural Network

An RCE neural network consists of three layers: an input layer, a hidden layer, and an output layer, as shown in Figure 1 [22]. The input layer contains feature data and is connected in parallel to each neuron in the hidden layer. Each neuron has a center point and a radius, which forms an activation region similar to the shape of a sphere. In the feature space, the activation region creates a decision boundary to distinguish the data. As the data enter the input layer, each neuron calculates the distance between the input data and its center point. The distance is compared with the radius of each neuron to determine if that neuron is activated, and the results are passed to the output layer. The output layer uses the received information to output the label of the neuron that best matches the input data.

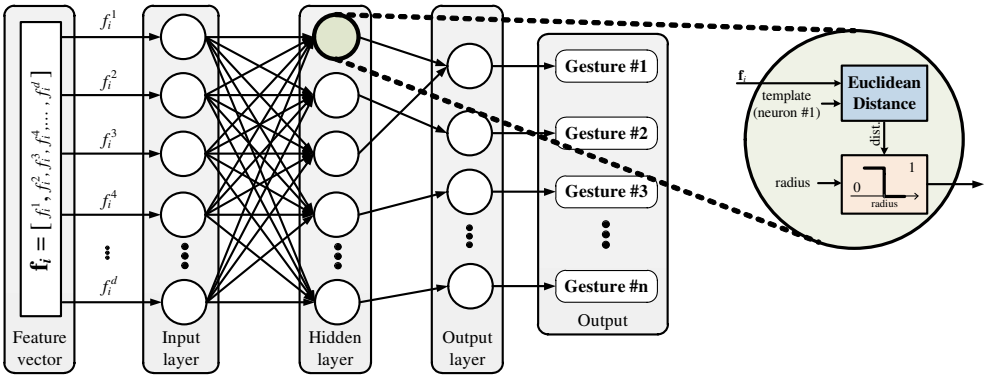


Figure 1. Structure of an RCE neural network.

The hidden layer consists of a set of neurons:

$$\mathbf{N} = [\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_j, \ldots, \mathbf{n}_q], \tag{1}$$

where q represents the total number of neurons generated during the training process. Each neuron can be described by:

$$\mathbf{n}_j = [\mathbf{c}_j, r_j], \tag{2}$$

$$\mathbf{c}_j = [c_j^1, c_j^2, \ldots, c_j^d], \tag{3}$$

where $j \in \{1, 2, 3, \ldots, q\}$ is the index of each neuron, $\mathbf{c}_j$ denotes a center point with d dimensions, and r_j represents the radius of the neuron. If the number of d-dimensional feature vectors used in the training process is m, the feature vector set can be expressed as follows:

$$\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_i, \ldots, \mathbf{f}_m]. \tag{4}$$

The i-th feature vector $\mathbf{f}_i$ is represented as follows:

$$\mathbf{f}_i = [f_i^1, f_i^2, \ldots, f_i^d]. \tag{5}$$

The feature vector is sequentially given as input to each neuron $\mathbf{n}_j$, and the Euclidean distance $ED(\mathbf{f}_i, \mathbf{c}_j)$ between the feature vector and the neuron center point can be obtained as follows:

$$ED(\mathbf{f}_i, \mathbf{c}_j) = \sqrt{(f_i^1 - c_j^1)^2 + (f_i^2 - c_j^2)^2 + \ldots + (f_i^d - c_j^d)^2}. \tag{6}$$

Afterwards, the activation of each neuron $\mathbf{n}_j$ is determined by:

$$ED(\mathbf{f}_i, \mathbf{c}_j) \leq r_j. \tag{7}$$

If no neurons activated for an input vector $\mathbf{f}_i$, a new neuron $\mathbf{n}_{q+1}$ with a center point $\mathbf{f}_i$ and an initial radius is generated, and the total number of neurons q is increased by one. On the other hand, if there is an activated neuron, there are two possible cases: the label of the activated neuron is the same as the label of the input feature or not. If the two labels are the same, no new neurons are generated. If not, the radius of the activated neuron is reduced to $ED(\mathbf{f}_i, \mathbf{c}_j)$ and a new neuron with a radius of $ED(\mathbf{f}_i, \mathbf{c}_j)$ and a center point equal to the input feature vector $\mathbf{f}_i$ is created.

2.2. DTW

DTW is a technique for aligning two data sequences by warping them in a nonlinear fashion to match them each other [18], as shown in Figure 2, and then finding distance between the data. This method is widely used in areas that deal with data that changes over time because it results in better performance than employing Euclidean distance on time-dependent data [23].

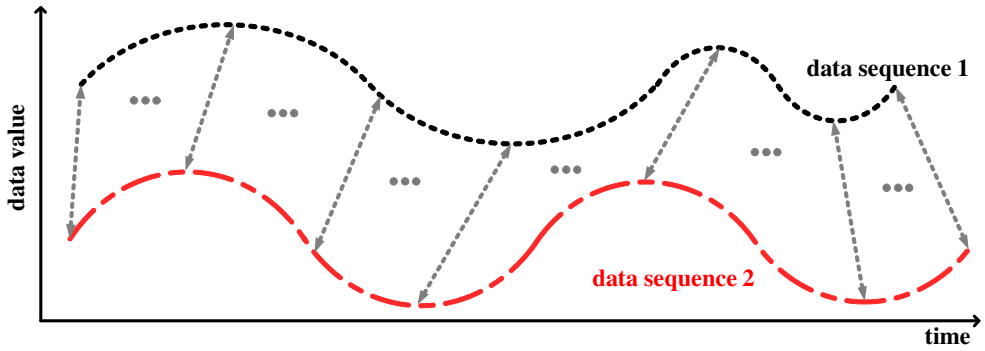


Figure 2. Time alignment of two data sequences; the aligned points are denoted by the arrows.

DTW distance is calculated by finding the optimal alignment between two data sequences, calculating the difference between aligned points, and accumulating the difference values [24]. The problem of finding the optimal alignment can be solved by using an accumulated cost matrix. To obtain this matrix, we need to find the cost matrix in advance. Consider two data sequences:

$$\mathbf{x} = (x_1, x_2, \ldots, x_{d_1}),\ \mathbf{y} = (y_1, y_2, \ldots, y_{d_2}). \tag{8}$$

The lengths of data sequences $\mathbf{x}$ and $\mathbf{y}$ are d_1 and d_2, respectively. We can find a cost matrix $\mathbf{C}(n, m) \in \mathbf{R}^{d_1 \times d_2}$, where $1 \leq n \leq d_1, 1 \leq m \leq d_2$, as shown in Figure 3, by using the following equation:

$$\mathbf{C}(n, m) = |x_n - y_m|. \tag{9}$$

With this cost matrix, our goal is to find the optimal alignment, which has the minimal overall cost. The optimal alignment is determined by following a small value in the cost matrix. The accumulated cost matrix can be obtained by accumulating the minimum value of the cost matrix under three conditions: the boundary condition, the monotonicity condition, and the step size condition [25]. The optimal path p with length L can be defined as:

$$p = (p_1, \ldots, p_l, \ldots, p_L), \tag{10}$$

$$p_l = (n_l, m_l) \in \mathbf{C}(n, m) \text{ for } l \in [1, L]. \tag{11}$$

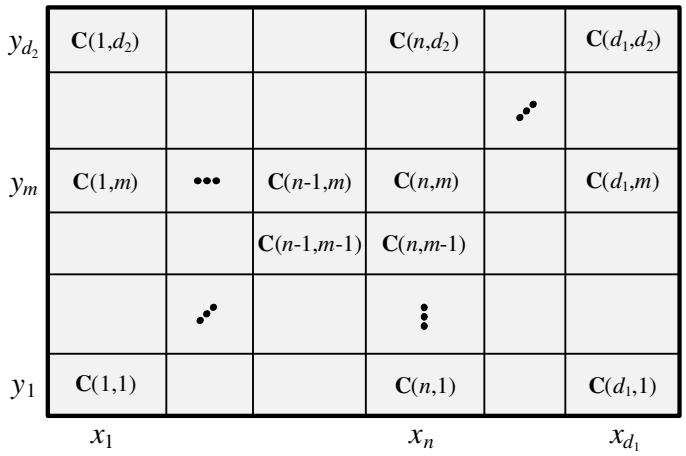


Figure 3. Cost matrix.

For the optimal path p, the following three conditions are defined:

1. Boundary condition
   In the optimal path, the starting point and ending point are defined as:

$$p_1 = (1,1) \text{ and } p_L = (d_1, d_2). \tag{12}$$

2. Monotonicity condition
   In the optimal path, the index value must be equal to or greater than previous index value:

$$n_1 \leq n_2 \leq \ldots \leq n_{L-1} \leq n_L \text{ and } m_1 \leq m_2 \leq \ldots \leq m_{L-1} \leq m_L. \tag{13}$$

3. Step size condition
   In the optimal path, the difference between neighboring values has a step size, which can be expressed as the following condition:

$$p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\} \text{ for } l \in [1 : L-1]. \tag{14}$$

The formula for the accumulated cost matrix $\mathbf{A}(n,m) \in \mathbf{R}^{d_1 \times d_2}$, where $1 \leq n \leq d_1, 1 \leq m \leq d_2$ can be expressed as:

$$\mathbf{A}(n,m) = \begin{cases} \mathbf{C}(n,m), & \text{if } n = 1 \text{ and } m = 1, \\ \mathbf{A}(n-1,m) + \mathbf{C}(n,m), & \text{if } n > 2 \text{ and } m = 1, \\ \mathbf{A}(n,m-1) + \mathbf{C}(n,m), & \text{if } n = 1 \text{ and } m > 2, \\ \min\{\mathbf{A}(n-1,m-1), \mathbf{A}(n-1,m), \mathbf{A}(n,m-1)\} + \mathbf{C}(n,m), & \text{if } n > 2 \text{ and } m > 2. \end{cases} \tag{15}$$

After constructing the accumulated cost matrix, we can find an optimal path by following a small value from $\mathbf{A}(d_1,d_2)$ to $\mathbf{A}(1,1)$, as shown by the dark blue line in Figure 4. The set of index values of the optimal path corresponds to the aligned points shown in Figure 2, and the DTW distance can be expressed as:

$$DTW(\mathbf{x},\mathbf{y}) = \mathbf{A}(d_1,d_2). \tag{16}$$

DTW-based recognition algorithms measure this distance between the input data and the template of each label and output the most similar label.

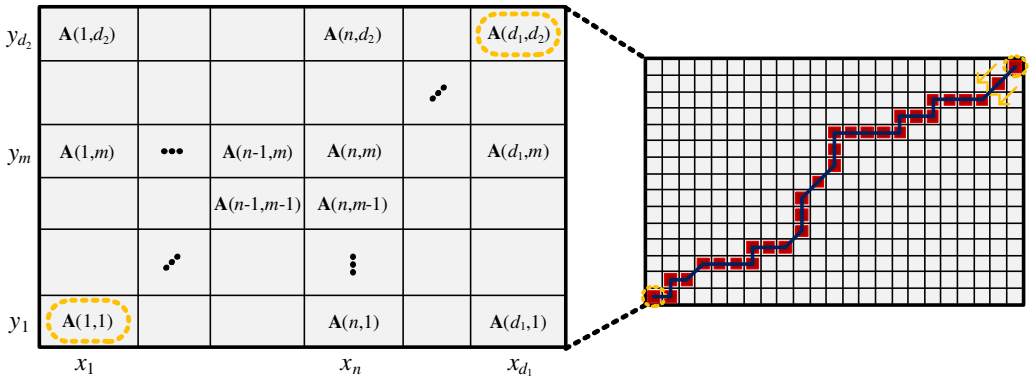


Figure 4. Accumulated cost matrix.

3. Proposed HGR Algorithm

DTW is a nonlinear method for obtaining the distance between two data sequences. The calculation of these distances involves an alignment process that extends or shrinks one data sequence along the time axis to match the other sequence. Thanks to this alignment process, the DTW distance between the two IMU sensor data sequences shown in Figure 5 is more accurate than the Euclidean distance.

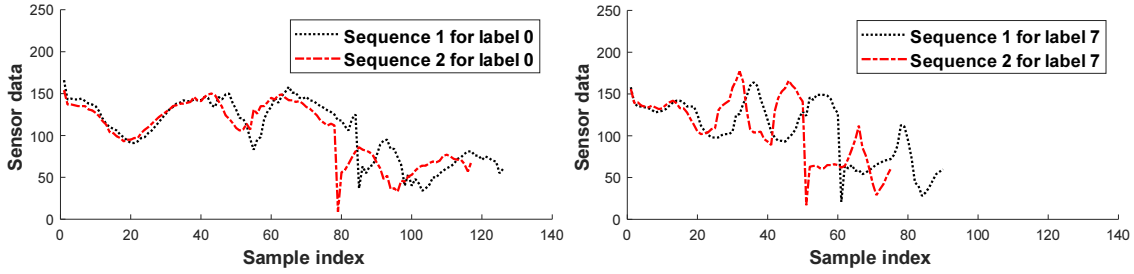


Figure 5. Examples of the data sequences: (**a**) two sequences for label 0; (**b**) two sequences for label 7.

This more accurate distance has led to DTW being used in various fields, and, when employed for HGR, it has exhibited superior performance according to many previous studies [11–14]. However, because DTW-based HGR involves a complex learning algorithm, its applications are limited by the number of recognizable hand gestures. Unlike DTW-based recognition algorithms, RCE neural networks have simple learning algorithms, which allows for real-time training. In other words, they can be used to train the network with the required hand gestures immediately. However, it is hard to use conventional RCE neural networks because they employ the Euclidean distance in their distance measurements during the learning and recognition processes. Therefore, we propose employing DTW in an RCE neural network to create a high-performance HGR algorithm that supports real-time learning. The structure of the proposed HGR algorithm is shown in Figure 6.

Algorithm 1 describes the proposed learning algorithm. When the input feature vector $\mathbf{f}_i$ is sequentially inputted to each neuron, the distance between the center point of the neuron and the input features is obtained via DTW. Afterwards, whether neuron $\mathbf{n}_j$ is activated or not is determined by comparing the distance $DTW(\mathbf{f}_i, \mathbf{c}_j)$ with the radius r_j of each neuron. The rest of the learning algorithm is the same as that used in existing RCE neural networks. Algorithm 2 describes the recognition process. DTW is applied to obtain the distance between the centers of neurons and the feature data extracted from the hand gesture, and the label of the neuron with the minimum distance becomes the recognition result.

Algorithm 1: Learning algorithm.

Input : The feature vector $\mathbf{f}_i$, the initial radius R, the number of learned neurons q, and the label l
Output: The set of neurons **N** and the number of learned neurons q

for $j = 1$ *to* q **do**
  $distance(j) \leftarrow DTW(\mathbf{f}_i, \mathbf{c}_j)$
end
find activated neurons
if *no learned neurons or no neurons are activated* **then**
  $q \leftarrow q + 1$
  $r_q \leftarrow R$
  for $k = 1$ *to* d **do**
    $c_q^k \leftarrow f_i^k$
  end
  $\mathbf{n}_q \leftarrow [c_q^1, c_q^2, \ldots, c_q^d, r_q]$
else if *neurons are activated* **then**
  if *activated neuron's label* $== l$ **then**
    do not generate a new neuron
  else
    decrease the activated neuron distance
    $q \leftarrow q + 1$
    $r_q \leftarrow$ decreased distance
    for $k = 1$ *to* d **do**
      $c_q^k \leftarrow f_i^k$
    end
    $\mathbf{n}_q \leftarrow [c_q^1, c_q^2, \ldots, c_q^d, r_q]$
  end
end

Algorithm 2: Recognition algorithm.

Input : The feature vector $\mathbf{f}$, the number of learned neurons q
Output: The label l

for $k = 1$ *to* q **do**
  $distance(j) \leftarrow DTW(\mathbf{f}, \mathbf{c}_j)$
end
$[value, index] = min(distnace)$
$l \leftarrow$ label of neuron $index$

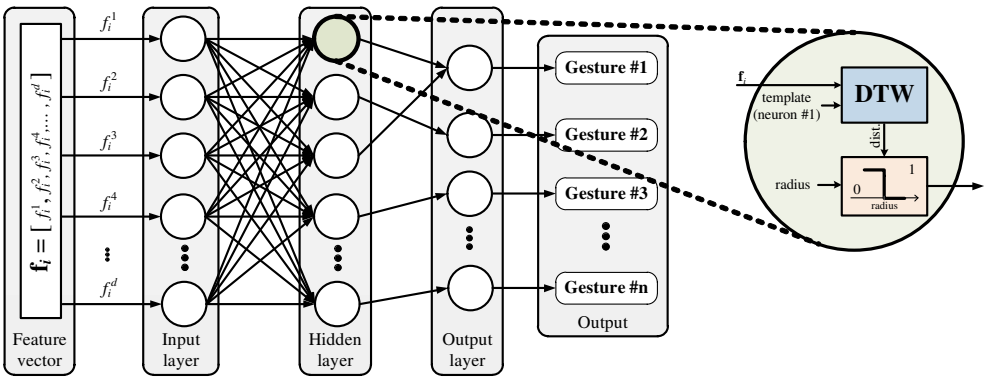
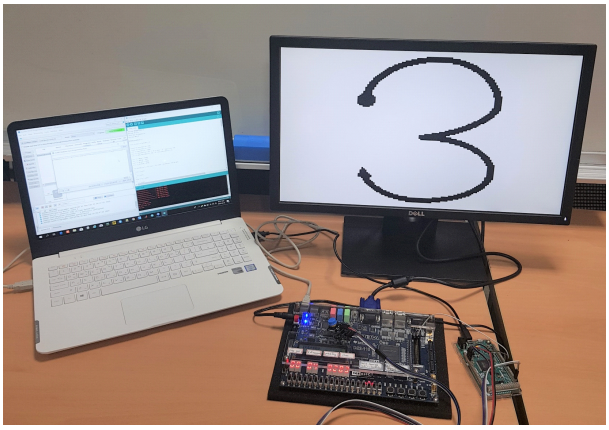


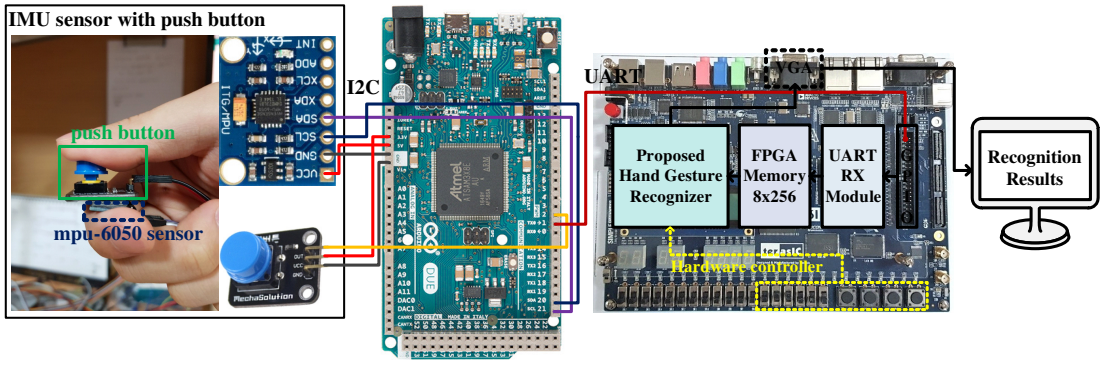
Figure 6. Structure of the proposed HGR algorithm.

4. Test Platform

We constructed a test platform to evaluate the proposed HGR algorithm, as shown in Figure 7. The platform consisted of an IMU sensor for obtaining hand gesture data, an Arduino module for preprocessing, and an FPGA for the proposed HGR implementation. We used an MPU-6050 [26] as the IMU sensor module, which can measure three-axis acceleration and three-axis gyroscope values. Among these values, we only used three-axis acceleration data for HGR, whereas the gyroscope data were reserved for future use. Hand gesture data from the IMU sensor are transferred to the Arduino through an inter-integrated circuit (I2C) interface by pressing a button on the IMU sensor module. The Arduino module preprocesses the sensor data to make them a fixed length, and a DUE model [27] was chosen considering the memory size and processing speed required for real-time operation. The length of the data depends on the speed and shape of the gesture, and thus data sequences with various lengths need to be processed to make them have a fixed length. We determined that the shortest data length was 33 and the longest one was 243 through experimental tests, as shown in Figure 8. Therefore, we set the fixed data length to 252, which is greater than 243, so that users can freely perform hand movements. Data sequences shorter than 252 are zero padded to avoid affecting the calculation of the distance between data sequences. Preprocessed data are transferred to the FPGA through universal asynchronous receiver/transmitter (UART) communication. The hand gesture recognizer implemented on the FPGA performs the training and recognition functions, and the recognition result is displayed on a monitor.



(**a**)



(**b**)

Figure 7. HGR test platform: (**a**) photograph of the test platform; (**b**) configuration of the test platform.

The proposed hand gesture recognizer was designed using Verilog hardware description language (HDL) and implemented in an Intel-Altera Cyclone IV FPGA device to verify that real-time learning and recognition was possible. Figure 9 shows a block diagram of the proposed hand gesture recognizer,

which consisted of a neural network, a network control unit (NCU), and an activated neuron detection unit (ANDU). Each neuron consisted of a neuron memory for storing the center point, a DTW unit for calculating the distance, and a neuron management unit (NMU) for managing the state and the operation of the neuron. During the training process, each neuron stores the input features in the memory, and the distance calculation is performed by the DTW unit. The calculated distance and the label information are sent to the ANDU, which finds activated neurons and sends the state of neurons and the minimum DTW distance to the NCU. Using this information, the NCU controls the neural network and modifies the network structure by creating new neurons and modifying the activation regions. During the recognizing process, all the learned neurons compute the DTW distance between the input features and the center point stored in the memory of the learned neurons. Then, the activated neurons transmit their distance to the ANDU, which finds the minimum distance and sends it to the NCU. The NCU then uses the minimum distance to generate a control signal so that the neural network can output the recognition result.

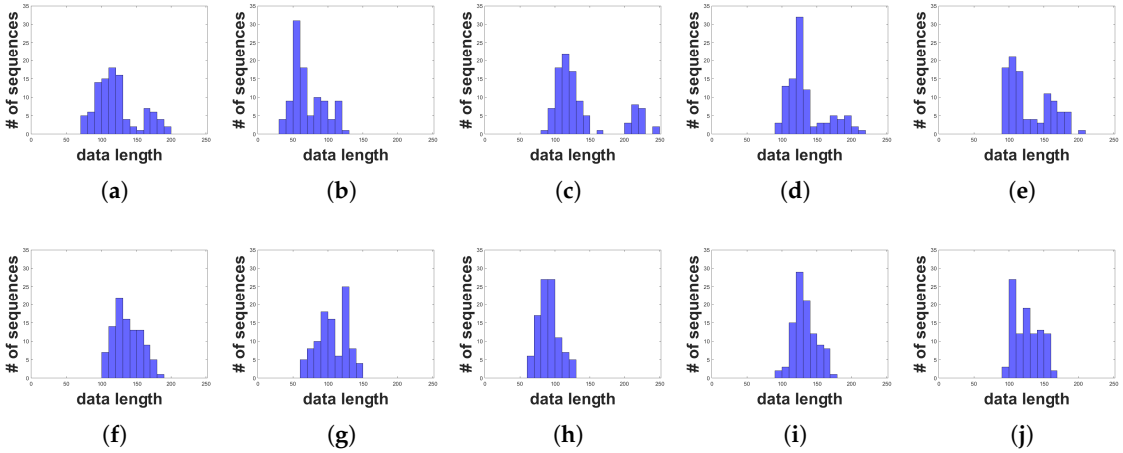


Figure 8. Histogram of the dataset: (**a**) label 0; (**b**) label 1; (**c**) label 2; (**d**) label 3; (**e**) label 4; (**f**) label 5; (**g**) label 6; (**h**) label 7; (**i**) label 8; (**j**) label 9.

Our FPGA implementation results show that the proposed hand gesture recognizer can be implemented with 30.54 K logic elements and 274.31 Kbits memory as shown in Table 1. In addition, we confirmed that the real-time training and recognition were possible because the proposed hand gesture recognizer required only 0.423 ms for training and 0.426 ms for recognition, at an operating frequency of 150 MHz.

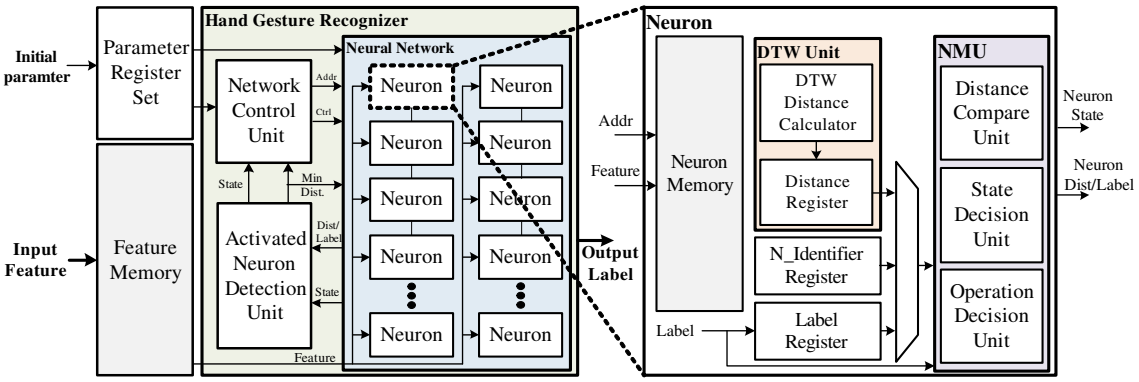


Figure 9. Block diagram of the proposed hand gesture recognizer.

Table 1. Implementation results of the proposed hand gesture recognizer.

Block	Neural Network	NCU	ANDU	Total
FPGA Logic Elements (/114,480)	25,765	2540	2970	31,275 (27.32%)
Memory [bists] (/3,981,312)	280,896	0	0	280,896 (7.06%)

5. Experimental Results

We constructed a 3D number dataset similar to the one used in a previous study [11]. Five participants, consisting of four men and one woman, were asked to hold the IMU sensor to write ten digits in the air. The dataset was generated by extracting the accelerometer values at a sampling rate of 20 Hz, and each hand gesture performed from a starting point by following the direction of the arrows shown in Figure 10. Each participant wrote each digit 20 times, and data corresponding to a total of 1000 hand gestures were collected.

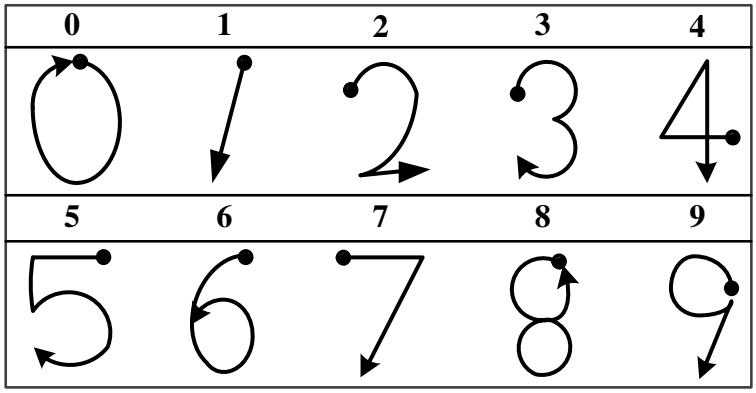


Figure 10. 3D number dataset.

Performance evaluation was performed using the collected dataset; accuracy was measured by performing 5-fold cross-validation for each person. Table 2 shows the confusion matrix of the proposed HGR algorithm, and the average recognition accuracy was 98.6%.

Table 2. Confusion matrix of the proposed algorithm.

Answer	Prediction									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	96%	0%	3%	0%	0%	0%	0%	0%	1%	0%
1	0%	100%	0%	0%	0%	0%	0%	1%	0%	0%
2	0%	0%	97%	0%	0%	0%	0%	0%	0%	0%
3	0%	0%	0%	99%	0%	1%	0%	0%	0%	0%
4	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
5	0%	0%	0%	0%	0%	97%	0%	0%	0%	0%
6	2%	0%	0%	1%	0%	2%	100%	0%	0%	0%
7	2%	0%	0%	0%	0%	0%	0%	99%	0%	0%
8	0%	0%	0%	0%	0%	1%	0%	0%	99%	1%
9	0%	0%	0%	0%	0%	1%	0%	0%	0%	99%
Total	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 3 shows the comparison results in terms of recognition accuracy between existing HGR algorithms and the proposed approach. The RCE neural network in Table 3 performs recognition

based on Euclidean distance, and MLP calculates the weighted sum of input features to recognize hand gestures. The DTW-based HGR proposed in [11] recognizes hand gestures based on the DTW distance measurement method as described in Section 2.2. We employed the hand gestures recorded from five participants, and the evaluation results were obtained by 5-fold cross-validation. As shown in this table, the proposed HGR algorithm outperformed the others for all users because it uses the DTW distance measurement method suitable for a time-dependent data sequence such as a hand gesture from an IMU sensor.

Table 3. Recognition performance of the proposed algorithm and others.

Algorithm	User1	User2	User3	User4	User5	Average
RCE neural network	82.5%	88.0%	81.0%	92.5%	83.0%	85.4%
MLP	81.5%	91.5%	86.5%	91.0%	89.5%	88.0%
DTW-based HGR	94.6%	94.6%	94.6%	94.6%	94.6%	94.6%
Proposed	99.5%	97.0%	97.5%	99.5%	99.5%	98.6%

6. Conclusions

Hand gesture recognition requires the classification of different hand motions depending on the user's preference or the type of application being considered. However, the application of existing HGR algorithms such as DTW, MLP and CNN is limited because they can only recognize predetermined gestures through preliminary training due to the very complex learning process. In this paper, we proposed an efficient HGR algorithm that can be used in various applications owing to the real-time learning. In order to enable real-time learning with high accuracy, the proposed HGR algorithm uses the learning method of RCE neural networks and distance measurement scheme of DTW. We constructed a test platform with an IMU sensor to verify that real-time learning and recognition was possible using the proposed algorithm. In addition, a 3D number dataset was constructed using the test platform, which could generate three-axis acceleration data samples at 20 Hz. We carried out a performance evaluation using 5-fold cross-validation on the constructed dataset and found that the proposed HGR algorithm could achieve a recognition accuracy of 98.6%, which is 13.2%, 10.6%, and 4% higher than that of RCE neural networks, MLPs, and DTW-based HGR algorithms, respectively. The proposed algorithm was designed and verified in hardware, which could support real-time learning and recognition at an operating frequency of 150 MHz.

With the rapid increase of wearable devices applying HGR technology, the simplified HGR algorithm is required to implement low-cost and low-power very large scale integrated circuits (VLSI). Therefore, our future research is focused on the simplification of the proposed HGR algorithm and its VLSI implementation.

Author Contributions: M.K. designed the algorithm, performed the simulation and experiment, and wrote the paper. J.C. and S.L. implemented the evaluation platform and performed the experiment. Y.J. conceived and led the research, analyzed the experimental results, and wrote the paper.

Funding: This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00528, 2019-0-00056) and CAD tools were supported by IC Design Education Center (IDEC).

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

References

1. Patil, S.; Bidari, I.; Sunag, B.; Gulahosour, S.; Shettar, P. Application of HMI Technology in Automotive Sector. In Proceedings of the International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), Mysuru, India, 9–10 December 2016.

2. Parimalam, P.; Shanmugam, A.; Raj, A.S.; Murali, N.; Murty S.A.V.S. Convenient and Elegant HCI features of PFBR Operator Consoles for Safe operation. In Proceedings of the 4th International Conference on Intelligent Human Computer Interaction (IHCI), Kharagpur, India, 27–29 December 2012.
3. Gupta, H.P.; Chudgar, H.S.; Mukherjee, S.; Dutta, T.; Sharma, K. A continuous hand gestures recognition technique for human-machine interaction using accelerometer and gyroscope sensors. *IEEE Sens. J.* **2016**, *16*, 6425–6432. [CrossRef]
4. Cheng, H.; Yang, L.; Liu, Z. Survey on 3D Hand Gesture Recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 1659–1673. [CrossRef]
5. Bao, P.; Maqueda A.I.; Del-Blanco C.R.; Garcia, N. Tiny hand gesture recognition without localization via a deep convolutional network. *IEEE Trans. Consum. Electron.* **2017**, *63*, 251–257. [CrossRef]
6. Ohn-Bar, E.; Trivedi, M.M. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 2368–2377. [CrossRef]
7. Xie, R.; Cao, J. Accelerometer-based hand gesture recognition by neural network and similarity matching. *IEEE Sens. J.* **2016**, *11*, 4537–4545. [CrossRef]
8. Jiang, S.; Lv, B.; Guo, W.; Zhang, C.; Wang, H.; Sheng, X.; Shull, P.B. Feasibility of wrist-worn, real-time hand, and surface gesture recognition via sEMG and IMU sensing. *IEEE Trans. Ind. Inf.* **2018**, *14*, 3376–3385. [CrossRef]
9. Zhang, Z.; Tian, Z.; Zhou, M. Latern: Dynamic continuous hand gesture recognition using FMCW radar sensor. *IEEE Sens. J.* **2018**, *18*, 3278–3289. [CrossRef]
10. Ahmad, N.; Ghazilla, R.A.R.; Khairi, N.M.; Kasi, V. Reviews on various inertial measurement unit (IMU) sensor applications. *Int. J. Signal Proc. Syst.* **2013**, *1*, 256–262. [CrossRef]
11. Hsu, Y.L.; Chu, C.L.; Tsai, Y.J.; Wang, J.S. An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition. *IEEE Sens. J.* **2016**, *16*, 154–163.
12. Srivastava, R.; Sinha, P. Hand movements and gestures characterization using quaternion dynamic time warping technology. *IEEE Sens. J.* **2015**, *16*, 1333–1341. [CrossRef]
13. Ji, Z.; Li, Z.-Y.; Li, P.; An, M. A new effective wearable hand gesture recognition algorithm with 3-axis accelerometer. In Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, 15–17 August 2015.
14. Patil, S.; Chintalapalli, H.R.; Kim, D.; Chai, Y. Inertial Sensor-Based Touch and Shake Metaphor for Expressive Control of 3D Virtual Avatars. *Sensors* **2015**, *15*, 14435–14457. [CrossRef] [PubMed]
15. Arhan, A.; Tora, H.; Uslu, B. Hand gesture classification using inertial based sensors via a neural network. In Proceedings of the 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Batumi, Georgia, 5–8 December 2017.
16. Kwon, M.C.; Park, G.; Choi, S. Smartwatch User Interface Implementation Using CNN-Based Gesture Pattern Recognition. *Sensors* **2018**, *18*, 2997. [CrossRef] [PubMed]
17. Ma, Y.; Liu, Y.; Jin, R.; Yuan, X.; Sekha, R.; Wilson, S.; Vaidyanathan, R. Hand gesture recognition with convolutional neural networks for the multimodal UAV control. In Proceedings of the Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linkoping, Sweden, 3–5 October 2017.
18. Müller, M. Dynamic time warping. In *Information Retrieval for Music and Motion*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 69–84.
19. Lawrence, S.; Giles, C.L.; Tsoi, A.C.; Back, A.D. Face recognition: A convolutional neural-network approach. *IEEE Trans. Neural Netw.* **1997**, *8*, 98–113. [CrossRef] [PubMed]
20. Guo, D.; Ming, X. Color clustering and learning for image segmentation based on neural networks. *IEEE Trans. Neural Netw.* **2005**, *16*, 925–936.
21. Cho, J.; Jung, Y.; Lee, S.; Jung, Y. VLSI Implementation of Restricted Coulomb Energy Neural Network with Improved Learning Scheme. *Electronics* **2019**, *8*, 563. [CrossRef]
22. Yin, X.M.; Guo, D.; Xie, M. Hand image segmentation using color and RCE neural network. *Robot. Addit. Auton. Syst.* **2001**, *4*, 235–250. [CrossRef]
23. Keogh, E.J.; Pazzani, M.J. Derivative Dynamic Time Warping. *SDM* **2001**, *1*, 5–7.
24. Berndt, D.J.; Clifford, J. Using Dynamic Time Warping to Find Patterns in Time Series. *KDD Workshop* **1994**, *10*, 359–370.

25. Barth, J.; Oberndorfer, C.; Pasluosta, C.; Schülein, S.; Gassner, H.; Reinfelder, S.; Kugler, P.; Schuldhaus, D.; Winkler, J.; Klucken, J.; et al. Stride segmentation during free walk movements using multi-dimensional subsequence dynamic time warping on inertial sensor data. *Sensors* **2015**, *15*, 6419–6440. [CrossRef] [PubMed]
26. InvenSense Inc. MPU-6000 and MPU-6050 Product Specification Revision 3.4. Available online: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf (accessed on 20 July 2019).
27. Arduino. Arduino DUE Board. Available online: https://store.arduino.cc/usa/due (accessed on 20 July 2019).