Real Time Gesture Recognition System with Gesture Spotting Function

Yuta Ichikawa
Department of Electrical and Electronic Engineering, Kansai University, Suita Osaka 564-8680, Japan
Email: k423998@kansai-u.ac.jp

Shuji Tashiro
Department of Electrical and Electronic Engineering, Kansai University, Suita Osaka 564-8680, Japan
Email: k375553@kansai-u.ac.jp

Hidetaka Ito
Depatrment of Electrical and Electronic Engineering, Kansai University, Suita Osaka 564-8680, Japan
Email: h.ito@kansai-u.ac.jp

Hiroomi Hikawa
Depatrment of Electrical and Electronic Engineering, Kansai University, Suita Osaka 564-8680, Japan
Email: hikawa@kansai-u.ac.jp

***Abstract*—A real-time dynamic hand gesture recognition system with gesture spotting is discussed in this paper. The gesture spotting detects the start and the end of the gesture frames. The system consists of preprocessing, posture sequence generation, and SOM-Hebb network. Feature vectors are computed by the preprocessing from video frames taken by a USB camera in real time, and these feature vectors are fed to the posture sequence generation and a vector that represents the sequence of postures, called a posture sequence vector, is generated. Then, gesture classification and the gesture spotting are performed in the SOM-Hebb network. Our gesture spotting function detects the end of the gesture by using vector distance between the posture sequence vector and the winner neuron's weight vector. The gesture recognition algorithm is implemented on a PC. A USB camera was used to acquire live images. The real time experimental results show that the system recognizes nine gestures with the accuracy of 96.22%.**

I. Introduction

Hand gesture is an attractive alternative to cumbersome interface devices for human-computer interaction (HCI). Since hand gestures are one of the most important communication methods frequently used in daily human life, it can be used in the same way as words. Thus, advances of using hands in HCI would be a great benefit to users. In recent years, a number of video-based hand gesture recognition have been proposed [1]. Hand gestures are divided into two kinds, i.e., hand postures or dynamic gestures of hand. Hand postures are static hand poses without any movements [2][3], and gestures are defined as dynamic movement, which is a sequence of hand postures [4][5].

Generally, gesture spotting that segments meaningful gestures from the continuous sequence of hand motion, is required for the hand gesture recognition. Therefore, the dynamic gesture recognition system must detect the start or end of the gesture. This paper proposes a real time dynamic gesture recognition system that can perform the gesture spotting.

The proposed system consists of two self-organizing maps (SOMs) [6] and a Hebbian learning network. The SOM is an unsupervised neural network which has been used in pattern recognition, data analysis and visualization, due to its clustering properties. A main characteristic of SOM is making a feature map, in which vectors having a similar feature are mapped to same neuron automatically. With this property, the SOM performs vector quantization.

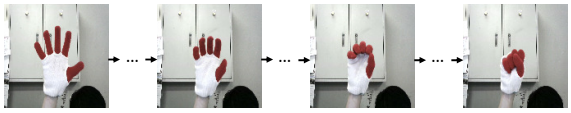

Fig. 1. Gesture.

In the proposed system, feature vectors are computed from RGB video frames that are taken by a USB camera in real time. Then a posture sequence vector that represents the current gesture, is generated using the SOM. The SOM performs posture classification in each video frame, and its results are stored to a shift register to form the posture sequence vector, which represents a dynamic gesture. A SOM-Hebb network [7] classifies and identifies the given gesture. End of gesture frame is detected by observing vector distance between input and weight vector of winner neuron since it is minimized when the posture sequence vector completes at the end of gesture. In the proposed system, the bottom in the vector distance transition is detected as the end of the gesture. The recognized gesture is outputted when the end of the gesture is detected. This recognition algorithm has already been tested by simulations [8]. This paper describes a real time dynamic gesture recognition system with this algorithm. Recognition accuracy of the proposed system was examined by experiments, in which nine dynamic gestures were recognized in real time.

This paper is organized as follows: The gesture recognition system is described in section II. The gesture spotting function is described in section III. The proposed system was tested by real time experiments and its results are given in section IV. Finally this paper is concluded in section V.

II. Gesture Recognition System

As shown in Figure 1, a dynamic hand gesture is defined as a sequence of static hand postures in video frames. Therefore, the dynamic gesture recognition is carried out in two steps, i.e., posture recognition and posture sequence recognition. Fig. 2

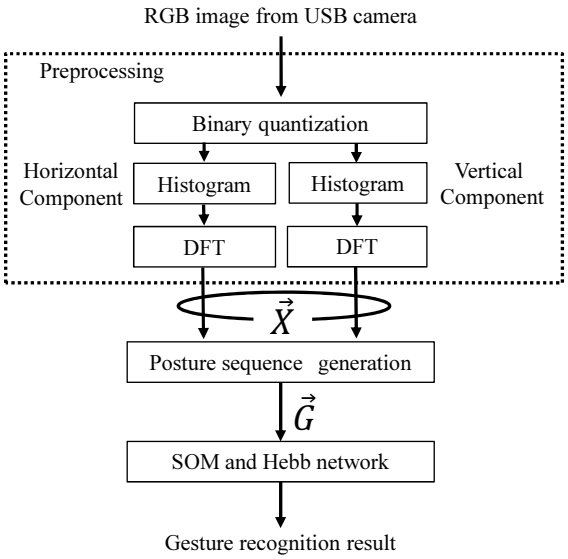

Fig. 2. Flow of gesture recognition.

Fig. 3. Binary image.

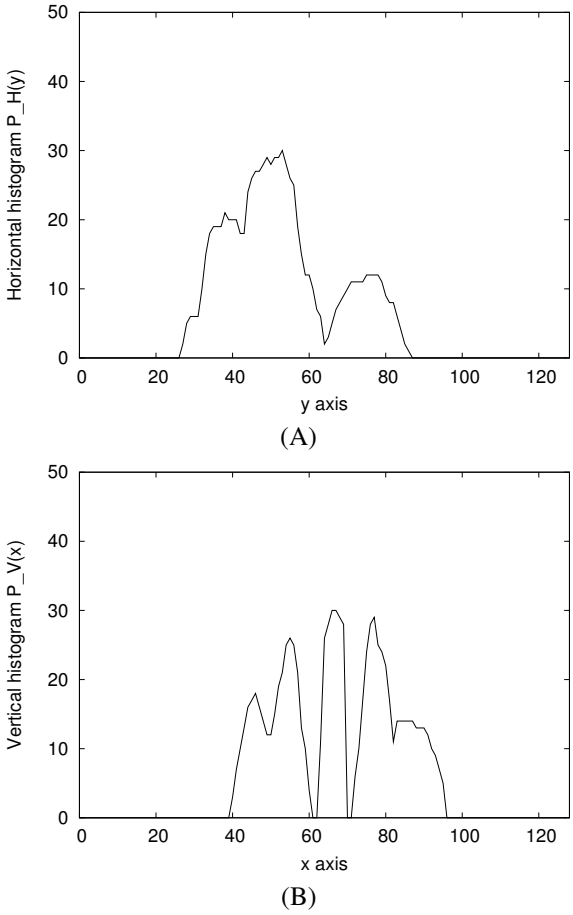

Fig. 4. Histogram of (A) horizontal, (B) vertical projection.

shows algorithm employed in the proposed gesture recognition system. The system is made of preprocessing, posture sequence generation, followed by a SOM-Hebb network.

The input is a video footage taken from a USB camera in real time. In this the system, a single gesture is defined by F video frames. Since each frame contains different postures, each dynamic gesture can be classified by using examining postures in the F video frames.

The preprocessing computes the feature vector. Then, the posture sequence generation module generates a posture sequence vector. The subsequent SOM-Hebb network identifies the gesture class from the posture sequence vector. The gesture spotting function is implemented in this network.

A. Preprocessing

Each video frame from USB camera is $P \times Q$ pixels in RGB color format. Each video frame is fed to the preprocessing, which generates D dimensional feature vector $\vec{X}$. A binary quantization, two horizontal and vertical projection histogram calculations, and two discrete fourier transforms (DFTs) constitute the preprocessing.

1) Binary Quantization: First, the input color image taken by the USB camera is converted to a binary image as shown in Fig. 3. The system requires its users to wear a red glove as shown in Fig. 1 so that the removal of the background image including the arm as well as the extraction of finger segments are made easier. The extraction and binary quantization of the image are carried out by the following equation.

$$I(x,y) = g(R(x,y), G(x,y) + B(x,y)) \cdot g(R(x,y), \rho) \quad (1)$$

$$g(x,\theta) = \begin{cases} 1 & (x \geq \theta) \\ 0 & (otherwise) \end{cases} \quad (2)$$

where, $I(x,y)$ is a binary pixel value at (x,y) coordinate, while $R(x,y)$, $G(x,y)$, $B(x,y)$ are red, green and blue color levels at (x,y) coordinate, respectively. ρ is a threshold parameter and $g(\cdot)$ is a threshold function.

2) Histogram calculations: Then horizontal and vertical histograms, $P_H(y)$ and $P_V(x)$ of $I(x,y)$ are calculated using the following equations.

$$P_H(y) = \sum_{x=0}^{P-1} I(x,y) \quad (3)$$

$$P_V(x) = \sum_{y=0}^{Q-1} I(x,y) \quad (4)$$

Figs. 4 (A) and (B) show examples of the horizontal and vertical projection histogram, respectively.

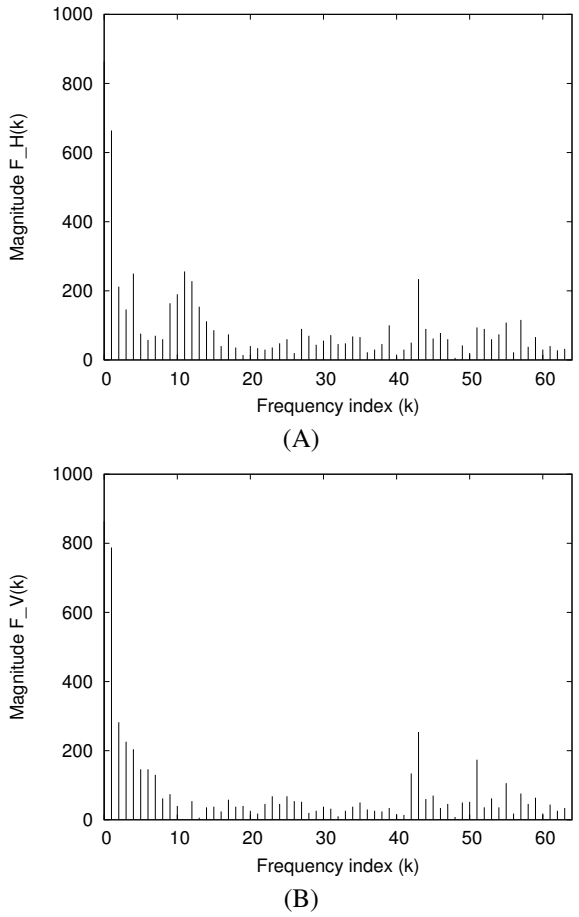

Fig. 5. Frequency spectrum, (A) horizontal, (B) vertical projection.

3) Discrete Fourier Transforms (DFT): After the histogram calculations, DFTs are carried out on the projection histograms.

$$A_H(k) = \sum_{n=0}^{Q-1} P_H(n) \cdot \cos(\frac{2\pi nk}{Q}) \quad (5)$$

$$B_H(k) = \sum_{n=0}^{Q-1} P_H(n) \cdot \sin(\frac{2\pi nk}{Q}) \quad (6)$$

$$A_V(k) = \sum_{n=0}^{P-1} P_V(n) \cdot \cos(\frac{2\pi nk}{P}) \quad (7)$$

$$B_V(k) = \sum_{n=0}^{P-1} P_V(n) \cdot \sin(\frac{2\pi nk}{P}) \quad (8)$$

$A_H(k)$, $B_H(k)$ and $A_V(k)$, $B_V(k)$ are real part and imaginary parts of DFT in horizontal and vertical projection, respectively. Then, $F_H(n)$ and $F_V(n)$, i.e., the magnitude spectra of $P_H(y)$ and $P_V(x)$ are computed as follows,

$$F_H(k) = \frac{\sqrt{A_H^2(k) + B_H^2(k)}}{Q} \quad (9)$$

$$F_V(k) = \frac{\sqrt{A_V^2(k) + B_V^2(k)}}{P} \quad (10)$$

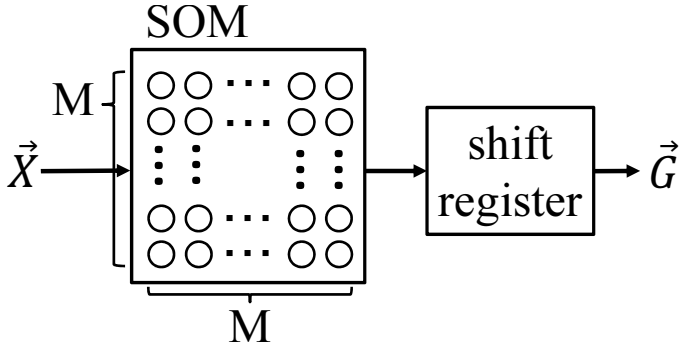

Fig. 6. Posture sequence generation.

$F_H(n)$ and $F_V(n)$ of the same hand posture image placed in different positions are identical because they are the magnitude spectrum lacking the phase information related to the hand posture position. As a result, the hand posture identification is very robust against the position change of the hand posture. Figs. 5 (A) and (B) show magnitude spectra of $P_H(y)$ and $P_V(x)$ that are computed from the histograms shown in Figs. 4 (A) and (B), respectively. As shown in the these figures, lower frequency components represent the majority of image's feature information, thus these lower frequency components are used as a feature vector of the input image. A D dimensional feature vector $\overrightarrow{X}$ is defined by the following equation.

$$\overrightarrow{X}_i = \begin{cases} F_H(i+1) & (0 \le i \le \frac{D}{2}) \\ F_V(i+1-\frac{D}{2}) & (\frac{D}{2} \le i \le D) \end{cases} \quad (11)$$

The feature vector $\overrightarrow{X}$ is fed to SOM in the posture sequence generation process.

B. Posture Sequence Generation

Fig. 6 shows the posture sequence generation process, which is made of SOM and a shift register. The SOM quantizes the input feature vectors, and quantized results are given by winner neuron's coordinates. The coordinates of the winner neuron are stored sequentially in the shift register. The contents of the shift register is used as the posture sequence vector, which is fed to the next SOM-Hebb network.

The SOM includes $M \times M$ neurons. A D-dimensional vector, $\overrightarrow{m}_i$ called weight vector, is assigned to each neuron.

$$\overrightarrow{m}_i = \{\mu_{i1}, \mu_{i2}, \cdots, \mu_{iD}\} \in \Re^D \quad (12)$$

The operation of SOM is divided into two phases, the learning phase and the recall phase. The weight map is trained with a set of input vectors in the learning phase. After that, the map is used in the recall phase. In the learning phase, the distances between the input vector and all weight vectors are calculated by the following equation.

$$V(i) = \vec{X} - \vec{m}_i \quad (13)$$

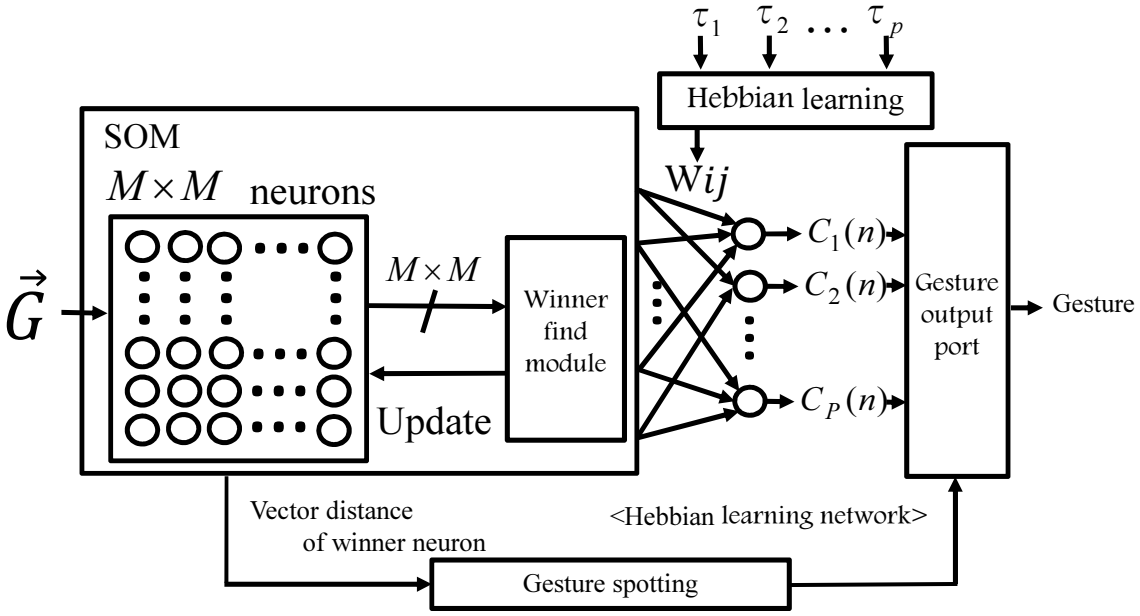

Fig. 7. SOM-Hebb network with gesture spotting.

Then the winner neuron$c$ that has the weight vector which is the nearest to the input vector is determined.

$$c = \arg\min_i \parallel V(i) \parallel \tag{14}$$

The weight vectors of the winner neuron and its neighborhood neurons are adapted toward the input vector with the following equation.

$$\vec{m}_i(t+1) = \vec{m}_i(t) + h_{ci}(t) \cdot [\vec{X(t)} - \vec{m}_i(t)] \tag{15}$$

where, t is time index. A neighborhood function h_{ci}, is defined as

$$h_{ci} = \alpha(t) \exp\left(-\frac{\|\vec{r}_c - \vec{r}_i\|}{2\sigma^2(t)} \right) \tag{16}$$

where, $\alpha(t)$ is a learning coefficient ($0 < \alpha(t) < 1$). $\vec{r_c}$ and $\vec{r_i}$ are the coordinate vectors of the winning neuron c and neuron i. $\sigma(t)$ represents the neighborhood radius, and weight vectors within the radius from the winner neuron are updated.

After the learning phase, the weights of the map remain unchanged and the map is used in the recall phase. In the recall phase, winner neuron is searched, but the weight adaptation is not carried out. Then, the coordinates of the winner neuron for each input frame are saved in the shift register sequentially. Contents of the shift register is the posture sequence vector $\vec{G}$ that is a $2F$-dimensional vector. ξ is defined as follows:

$$\vec{G} = \{\xi_0, \xi_1, \cdots, \xi_{2F-1}\} \tag{17}$$

$$\xi_i = \begin{cases} W_x(f-i) & (0 \le i < F) \\ W_y(f-i+F) & (F \le i < 2F) \end{cases} \tag{18}$$

where F is the number of frames, included in a single gesture. $W_x(n)$ and $W_y(n)$ are X and Y coordinates of the winner neuron at time n. When all F postures belonging to a gesture are processed, the posture sequence vector $\vec{G}$ can be treated as a sequence vector that represents the current input gesture. As mentioned before, since $F = 10$ in our system, vector dimension of $\vec{G}$ is 20. The vector is fed into SOM-Hebb network described in next section as the input vector.

C. SOM-Hebb Network for Gesture Classification

From the gesture sequence vector $\vec{G}$, the SOM-Hebb network identities the given gesture. The network is depicted in Fig. 7, which includes the gesture spotting function. The gesture spotting is implemented in this network. This SOM is trained in the same way as the SOM used in the previous module.

In the SOM, the winner neuron for each input vector $\vec{G}$ is determined, and from the winner neuron, the class to which the input vector belongs can be identified. The Hebb network converts the winner neuron number to the corresponding gesture number, which is the recognition result. The different neurons may become winner by the same dynamic gesture inputs because of variation of hand postures in the given gestures. Therefore, all those neurons belonging to the same class must be associated with that class. This selection is done by the Hebb network that is a simple layer feedforward network.

Training vectors and teaching data, τ_1,τ_2,$\cdots$,τ_P, that indicate the class of the given vectors are sequentially fed to the network during the learning phase. If a training vector belongs to j, then only $\tau_j = 1$ and the other $\tau's$ are zero. P represents the number of the class to recognize. If a strong correlation between the training vectors in class j and a winner neuron is found, that neuron is assigned to the class τ_j. During the training phase, simultaneous activations of the neurons i and τ_j are counted. Then neuron i is associated to class j if the

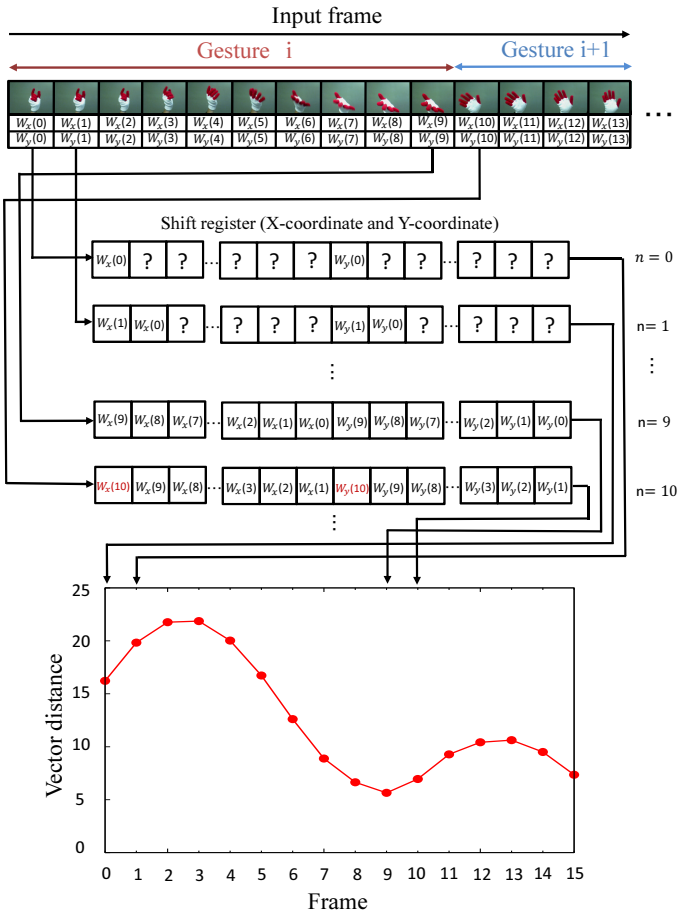

Fig. 8. The number of input frames and vector distance.

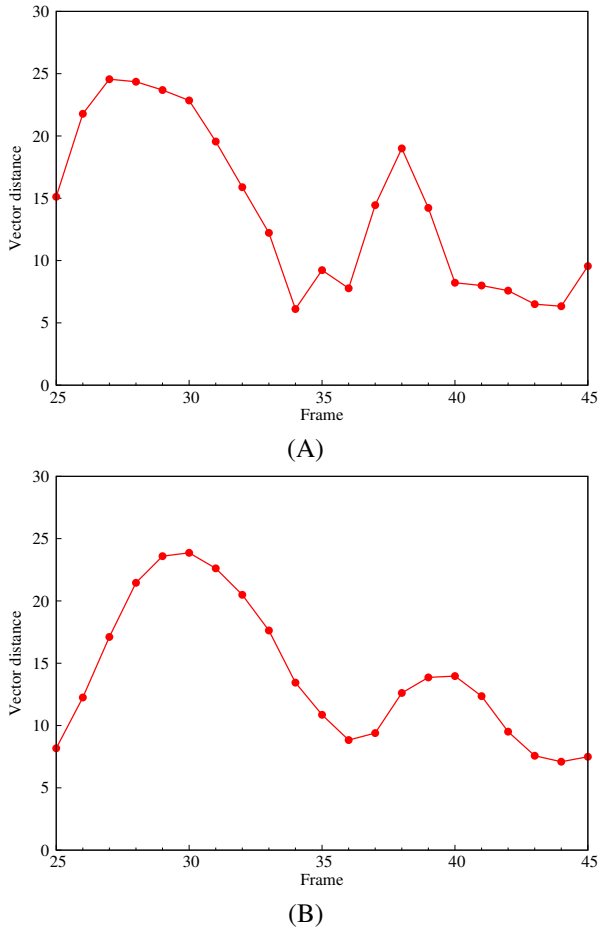

Fig. 9. Transition of vector distance, (A) actual vector distance, (B) moving average of vector distance.

count exceeds a threshold T_{Hebb}. By doing this, input gesture is recognized as class j gesture if neuron i wins in the recall phase. Due to the variation of the training vectors, multiple neurons may be associated to a single gesture class.

In this way, the Hebb network finds the association between neurons and gesture classes. However, there are some neurons which may have no connections to any gesture class. Obviously, the selection of these neurons as the winner in the recognition phase causes false recognition. In order to avoid situations where such neurons are selected, these neurons are culled. The culling replaces weight vectors of those neurons with a huge vectors so that they will not win.

III. Gesture Spotting by Using Vector Distance

Gesture spotting is one of the most important functions in gesture recognition, which detects the end of the posture. As Fig. 7 shows, the gesture spotting is implemented in this network. This network performs the recognitions for every input frames, but output is given only if the end of gesture is detected. In the proposed system, the gesture spotting is based on sequence of the vector distance between the input vector and the winner neuron's weight vector. Fig. 8 shows transitions of shift registers containing the posture sequence vector $\overrightarrow{G}$ and corresponding vector distances to winner neuron's weight vectors. f is the frame number, and the question marks indicate the information of the previous gesture. As the gesture proceeds toward its end, the vector distance gets shorter. Since each gesture is made of 10 frames, the vector distance is minimized when $f = 9$, at which the shift register is filled with all vector elements belonging to the current gesture. After that, distance increases because new vector elements belonging to next gesture are stored in the shift register. At some point the distance decreases again and it will hit another bottom at the end of next gesture. Therefore the end of the gesture can be detected by searching the bottom of the distance transition. The gesture spotting module detects the bottom of transition when the vector distance transition is "dip $\rightarrow$ dip $\rightarrow$ dip $\rightarrow$ rise $\rightarrow$ rise $\rightarrow$ rise". When the end of the posture, i.e., the bottom of the distance transition is detected, the gesture spotting module outputs current recognition result of the SOM-Hebb network as system's recognition result.

However, the actual distance transition which resulted in the SOM-Hebb network is not as smooth as shown in Fig. 8. Fig. 9 (A) shows actual vector distance in the real time gesture recognition. In this example, the vector distance takes the

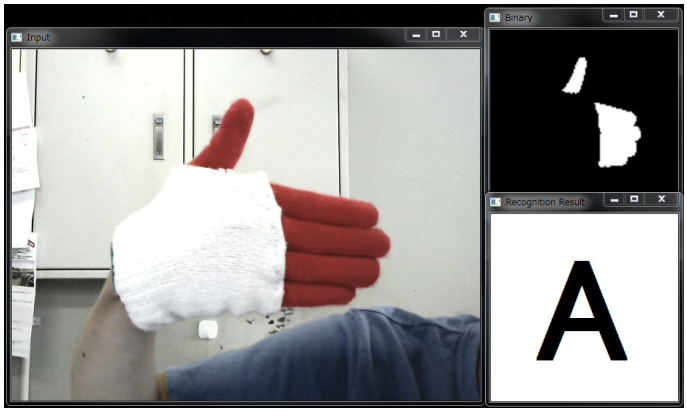

Fig. 10. Real time gesture recognition system.

minimum value at $Frame = 34$, which must be the end of the gesture. However, the distance transition after that is not "rise $\rightarrow$ rise $\rightarrow$ rise", and this posture end is not properly detected. As shown in this example, the transition goes up and down frequently, which makes it difficult to find the bottom and causes false detections of the bottom. In order to solve this problem, moving average of the vector distance is employed. The moving average $V_{ma}(n)$ is computed as,

$$V_{ma}(n) = \frac{1}{N_r}\sum_{i=1}^{N_r} V(n-i+1) \tag{19}$$

where, $V(n)$ is the vector distance between the input vector $\overrightarrow{G}$ and the winner neuron's weight vector at time index n. N_r is the number of samples which is used to compute the average of vector distance.

Fig. 9 (B) shows the moving averaged vector distance transition computed from Fig. 9 (A), where $N_r = 4$. The transition in Fig. 9 (B) is smoother than Fig. 9 (A). Thus, by using the moving averaged vector distance transition, it becomes easier for the system to find the bottoms accurately. Due to the moving average, the bottom is two frames delayed. The bottom in Fig. 9 (A) is given at $Frame = 34$, while the bottom of Fig. 9 (B) is at $Frame = 36$ Therefore, the recognition result obtained at two frames before the bottom is adopted.

IV. Experiment

Real time gesture recognition experiment was conducted to examine the performance of the system. The recognition system described in the previous section is implemented in software that runs on a PC equipped with a USB camera for the live image acquisition, and the system performs the real time recognition. Fig. 10 is one of the captured images of the real time recognition system. Training of the SOMs and Hebbian learning network were performed off-line by using pre-captured gesture images. Then after the training, the recognition system classified dynamic gestures in real-time by using the weight vectors obtained by the training. During the recognition process, SOMs were in recall mode.

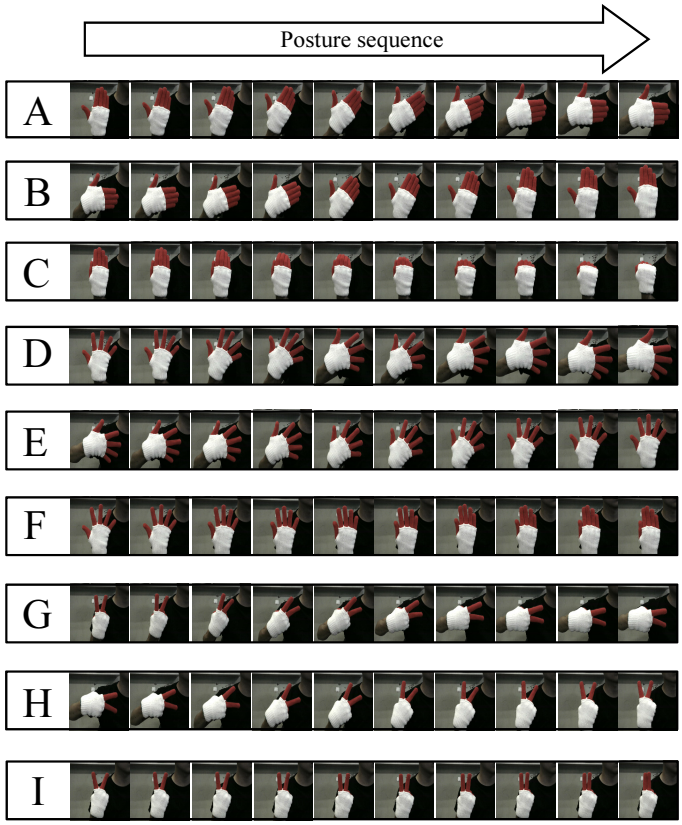

Fig. 11. Gestures used in the experiment.

The real time experiment was performed using the same gestures defined in Cambridge-Hand-Gesture Data Set [9]. Nine gestures used in the experiment are shown in Fig. 11. Note that A and B, D and E, G and H are gestures of reverse order, respectively. System parameters were set as follows:

- The size of frame was $P \times Q = 128 \times 128$.
- The feature vector dimension was D = 32.
- The dimension of the posture sequence vector was 20 ($F = 10$).
- The number of neurons in SOM used in the posture recognition and gesture recognition were both $M \times M = 16 \times 16 = 256$.
- For the moving average in Equ.(19), $N_r = 4$.
- The number of classes was $H = 9$.

Table I shows the recognition accuracies of 9 dynamic gestures by the proposed system. The experiment was carried out at 100 times for each gesture by the same person who provides the training data set. The number of training data for the off-line training was 50 gestures for each class. The average recognition rate of the proposed method for this 9 gestures was 96.22%.

In order to show the advantage of the proposed algo-

TABLE I
RECOGNITION ACCURACIES.

Gesture	Accuracy
A	86.00%
B	93.00%
C	100.00%
D	96.00%
E	96.00%
F	100.00%
G	99.00%
H	96.00%
I	100.00%
Average	96.22%

TABLE II
ACCURACY COMPARISON.

Method	Accuracy
proposed	96.22%
TCCA	86%
CCA	69%
pLSA	71%
MGO/RVM	44%
MGO/SVM	30%
NN-ED	29.44%
NN-NC	29.03%
SVM	41.25%

rithm. Tab. II shows recognition performance of other 8 algorithms[9]. The algorithms in the table are, tensor canonical correlation analysis (TCAA) method [10], simple canonical correlation analysis (CAA) method [11], probabilistic latent semantic analysis (pLSA) with space-time descriptors [12], motion gradient orientation with relevance vector machine (MGO/RVM) or support vector machine (MGO/SVM) [13], nearest neighbor (NN) classifier in the sense of Euclidean distance (NN-ED) or normalized correlation (NN-NC) of video vectors, and SVM of the video vectors.

It seems that the comparison in this table reveals the advantage of the proposed system in terms of recognition accuracy. However it should be noted that experimental setups were different. Our system and the algorithms in the table used the same gestures but different data set. The proposed system was trained and tested by video frames taken in the same illumination setting, while training and test data used in the algorithms in the table were different in their illumination settings. The difference between the training and test data makes the recognition harder. Therefore, we assess actual performance advantage of our system is not as much shown in the table. In addition, the proposed system requires users to wear the red glove to segment hand portion from background. Even though the contribution of the red globe to the system's performance is fairly large, it is annoying for users. Thus, our recognition system needs to be modified so that no glove is required.

V. CONCLUSION

This paper has proposed a real-time dynamic hand gesture recognition system with the gesture spotting function using vector distance. Using the sequential vector distances for gesture classification, this system can perform the gesture spotting that provides realistic recognition to the system. The feasibility of the proposed gesture recognition system has been tested by real time experiments. The experimental results show that the system can recognize 9 gestures with the accuracy of 96.22%, which is better than other recognition algorithms. However our system uses a red glove to extract the feature vector from the hand image easily, which may have improved the recognition accuracy compared to other systems.

Our future research objective is the development of a hardware gesture recognition system that can provide much faster recognition speed with smaller hardware compared to the PC used in this paper. Development of efficient hardware implementation and a new feature extraction method without the red glove, are left for future research.

REFERENCES

[1] V. I. Pavlovic, R. Sharma, T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.19, pp.677-695, 1997.

[2] J. Triesch, C. von der Malsburg, "A System for Person-Independent Hand Posture Recognition against Complex Backgrounds," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.23, no.12, pp.1449-1453, 2001.

[3] K. Hoshino, T. Tanimoto, "Realtime Hand Posture Estimation with Self-Organizing Map for Stable Robot Control," *IEICE Trans. on Information and Systems*, Vol.E89-D, no.6, pp.1813-1819, 2006.

[4] A. F. Bobick, A. D. Wilson, "A state-based approach to the representtion and recognition of gesture," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.19, no.12, pp.1325-1337, 1997.

[5] H-H. Yang, N. Ahuja, and M. Tabb, "Extraction of 2D motion trajectories and its application to hand gesture recogntion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.24, no.8, pp.1061-1074, 2002.

[6] T. Kohonen, *Self-Organizing Maps* (Information Sciences), vol. 30, 3rd ed. New York, USA: Springer-Verlag, 2001.

[7] H. Hikawa, K. Kaida, "Novel FPGA Implementation of Hand Sign Recognition System With SOM-Hebb Classifier," *IEEE Trans. Circuits and Systems for Video Technology*, Vol.25, no.1, pp.153-166, 2015.1.

[8] Y. Ichikawa, S. Tashiro, H. Hikawa, "Gesture Spotting by Using Vector Distance of Self-Organizing Map," *International Conference on Neural Information Processing (ICONIP)* 2016, to appear.

[9] "Cambridge Hand Gesture Data set," http://www.iis.ee.ic.ac.uk, (cited June 2016).

[10] T-K. Kim and R. Cipolla, "Canonical Correlation Analysis of Video Volume Tensors for Action Categorization and Detection, " *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(8):1415-1428, 2009.

[11] T.-K. Kim, J. Kittler, and R. Cipolla, "Discriminative Learning and Recognition of Image Set Classes Using Canonical Correlations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1005-1018, June 2007.

[12] J.C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words," *Proc. British Machine Vision Conf.*, 2006.

[13] S.-F. Wong and R. Cipolla, "Real-Time Interpretation of Hand Motions Using a Sparse Bayesian Classifier on Motion Gradient Orientation Images," *Proc. British Machine Vision Conf.*, pp. 379-388, 2005.