PERGAMON

Pattern Recognition 34 (2001) 1491–1501

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Hand gesture recognition using combined features of location, angle and velocity

Ho-Sub Yoon[a,b,*], Jung Soh[a], Younglae J. Bae[a], Hyun Seung Yang[b]

[a]*Image Processing Div. / Computer & Software Technology Lab., ETRI 161, Kajung-Dong, Yusung-Ku, Taejon, 305-350, South Korea*
[b]*Department of Computer Science / Korea Advanced Institute of Science and Technology, 373-1 Kusung-Dong, Yusung-Ku, Taejon, 305-701, South Korea*

Received 16 April 1999; received in revised form 15 June 2000; accepted 15 June 2000

Abstract

The use of hand gesture provides an attractive alternative to cumbersome interface devices for human–computer interaction (HCI). Many hand gesture recognition methods using visual analysis have been proposed: syntactical analysis, neural networks, the hidden Markov model (HMM). In our research, an HMM is proposed for various types of hand gesture recognition. In the preprocessing stage, this approach consists of three different procedures for hand localization, hand tracking and gesture spotting. The hand location procedure detects hand candidate regions on the basis of skin-color and motion. The hand tracking algorithm finds the centroids of the moving hand regions, connects them, and produces a hand trajectory. The gesture spotting algorithm divides the trajectory into real and meaningless segments. To construct a feature database, this approach uses a combined and weighted location, angle and velocity feature codes, and employs a *k*-means clustering algorithm for the HMM codebook. In our experiments, 2400 trained gestures and 2400 untrained gestures are used for training and testing, respectively. Those experimental results demonstrate that the proposed approach yields a satisfactory and higher recognition rate for user images of different hand size, shape and skew angle.

Keywords: Hand gesture recognition; Hidden markov model (HMM); Hand tracking; Spotting algorithm

1. Introduction

Hand gesture recognition using visual devices has a number of potential applications in HCI (human computer interaction), VR (virtual reality), and machine control in the industrial field [1,2]. Most conventional approaches to hand gesture recognition have employed such external devices as datagloves and color makers. For a more natural interface, however, hand gesture must be distinguishable from visual images without the aid of any external device.

Many hand gesture recognition methods using visual analysis have been proposed: syntactical analysis, neural-network based approaches, and the HMM-based recognition [3,4]. Since gesture consists of continuous motion in sequential time, an HMM must be an effective recognition tool.

Several hand gesture recognition systems have been developed using various features computed from static images or image sequences [5]. Segan [6] used an edge-based technique to extract image parameters from simple silhouettes developing a system capable of recognizing ten distinct poses in real time. Hunter [7] used Zernike moments as image features to develop a system in which sequences of hand gestures are recognized using HMMs. Starner [3] used image geometry parameters as image features and employed a five-state HMM topology for gesture classification.

* Corresponding author. Image Processing Div./Computer & Software Technology Lab., ETRI 161, Kajung-Dong, Yusung-Ku, Taejon, 305-350, South Korea. Tel.: + 82-42-860-5233; fax: + 82-42-860-4844.

E-mail address: yoonhs@etri.re.kr (H.-S. Yoon).

0031-3203/01/$20.00
PII: S0031-3203(00)00096-0

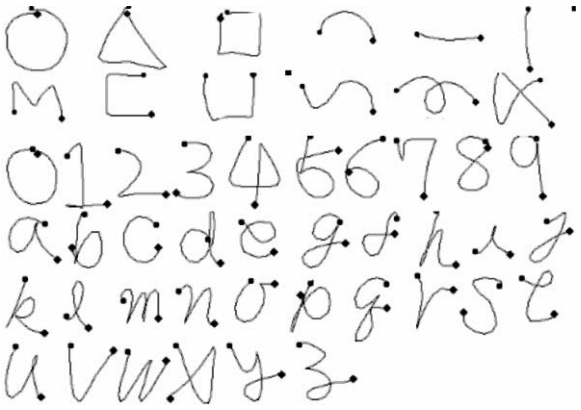


Fig. 1. Gesture shapes used in the system.

In our research, we consider planar hand gestures generated in front of the camera. Features such as "location, angle and velocity" are used as input vectors for the HMM network. We use simple context modeling on the "left-to-right" HMM model with 10 states applying it to two types of gesture consisting of 12 graphic elements and 36 alphanumeric characters.

One type of gesture is the graphic gesture consisting of six drawing elements (circle, triangle, rectangle, arc, horizontal line, and vertical line), and six edit commands (move, copy, undo, swap, remove, and close). The other type consists of the 10 Arabic numerals from 0 to 9 and the 26 characters of alphabet. Fig. 1 shows the gesture shapes used in this system.

The system's main screen is shown in Fig. 2. A camera placed in front of a monitor gives a sequence of images like those in Fig. 2. The screen shows the rectangle's gesture trace as captured by the system and the HMM's recognition result displayed in the lower-left portion.

This work explores the use of hand gestures in a realistic setting to control a graphic editor. We do not expect that the mouse will be directly replaced, but some part of the next-generation user interface will likely see the development of a graphic editor system operated by hand gesture with an image sensor.

This paper has five sections. In Section 2, we explain the preprocessing algorithm using image processing. Section 3 explains the feature extraction method. In Section 4, an HMM recognition model and the experimental results using the location, angle and velocity features are described. We present our conclusions in Section 5.

2. Preprocessing algorithm using image processing

2.1. Hand location algorithm using color analysis

Hand detection is the first step in visual image-based gesture recognition, and therefore directly influences the recognition rate. Our research exploits such a priori knowledge as skin-color, hand size, and positional information through the following steps:

Step 1: Color system conversion from RGB to YIQ.
Step 2: Estimation of similarity measures between model and input regions.
Step 3: Thresholding similarity measures.
Step 4. Noise removal and dilation.
Step 5: Detection of hand candidate regions.
Step 6: Selection of hand region.

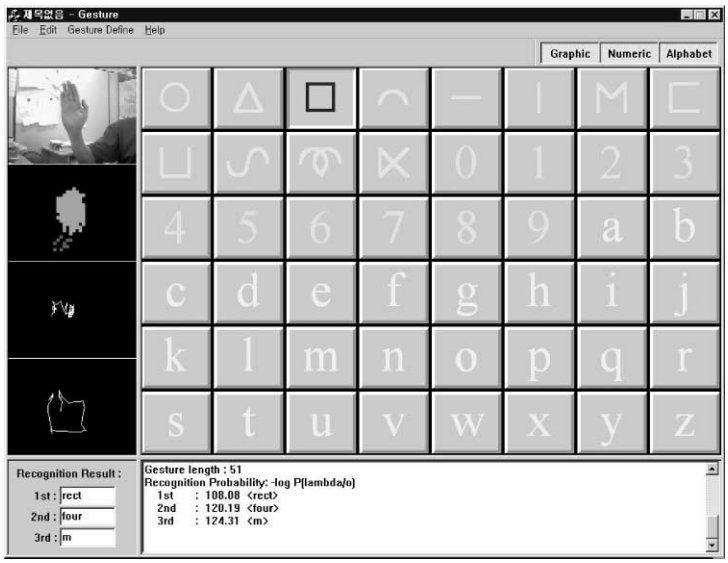


Fig. 2. Human computer interaction by hand gesture.

Color is one of the most distinctive clues in finding objects and is generally input as RGB information via a CCD or a video camera. The RGB color system is known to be sensitive to lighting conditions and has a high computing cost, since the RGB system includes the mixed information of color and intensity. Therefore, the YIQ color system, which is known to be less sensitive to lighting than the RGB, is adopted. In YIQ, Y refers to intensity, while I and Q represent color information. To reduce the effect of lighting and processing cost, only I and Q values are used to construct a skin-color model and to extract hand areas from input images by calculating the similarity between the model and the input image.

A skin-color model is constructed at the initial stage with an interactive method, and automatically updated using new skin-color information at the intermediate stages for extracting hand areas. We aim at maintaining a more robust model that can adapt to changing environmental conditions. Fig. 3 shows the skin color model constructed at the initial stage.

In order to detect hand regions from video images, the skin-color similarity regions are first observed. Since image processing by pixel unit has a high computing cost, the input image is processed on a cell unit basis, each containing 4×4 pixels. The similarity between the model histogram and the current cell is calculated using the histogram intersection method proposed by Swain and Ballard [8].

$$S_{M,I} = \frac{\sum_{i,q} \min(H^M(i,q), H^I(i,q))}{\sum_{i,q} H^M(i,q)}, \quad (1)$$

H^M indicates the i–q histogram value of the model and H^I that of the input cell. Using Eq. (1), the similarity cells are selected when they produce greater similarity values than a given threshold. Then, the selected cells are joined with the neighboring similarity cells and become candidate regions. A candidate region might contain some false similarity regions in the background and also holes whose similarity is very low. Our method observes regions usually composed of one or two cells and a few neighboring similarity regions, and then removes small and isolated regions which are unlikely to be actual skin regions. After the above processing, the dilation process is applied by increasing the similarity of cells surrounded by cells of relatively high similarity.

In the next step, the remaining candidate regions are binarized using a threshold value. Since a fixed threshold value cannot adapt to changes in lighting and background, we use a variable threshold calculated using a method proposed by Otsu [9]. Using this method has the effect of adapting the threshold value to the overall similarity level of the skin-color regions. After binarization, the connected components of the binary image are located, and each connected component then becomes a hand candidate region. In Fig. 4, the images of each processing step are shown.

Finally, the hand region must be detected from among the multiple candidate regions. The major difficulty in this step is that the face region can be one of the hand candidate regions, because it has almost the same skin color. This paper uses such a priori knowledge as the hand location of a previous video image, the location where the face is usually positioned, and the size of the hand region.

2.2. Spotting algorithm

The last preprocessing step is pure gesture detection. One gesture in our system is generated while the hand region is still on the screen. Therefore, the garbage movements that come before and after a pure gesture are included in the input gesture. To remove these garbage gestures, we use a time-varying-based spotting algorithm. This approach uses a spotting rule with user stops for a while (2 or 3 s) before the meaningful gesture starts and after it ends. Using this rule, the standstill area corresponds to the spotting area and the gesture between two spotting areas is pure gesture. In a gesture trace which consists of a sequence of (x, y) positions, the spotting area is detected by the state transition rules described in Fig. 5. From Fig. 5, we can detect a standstill area where the hand appears but does not move. Fig. 6 present an example of the gesture spotting time-varying-based spotting rule.

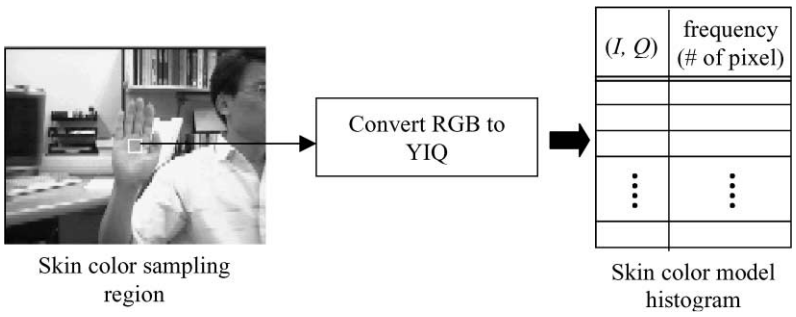


Fig. 3. Skin color model histogram on I–Q space.

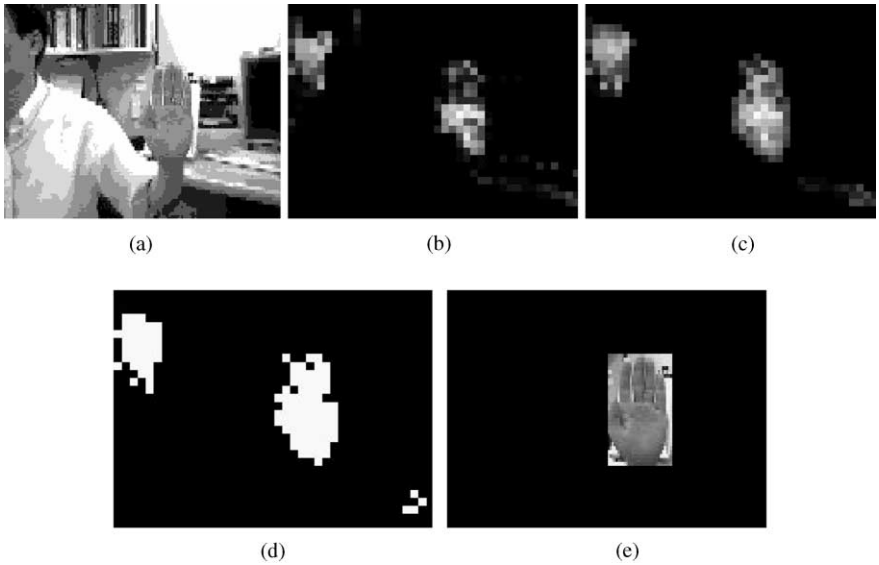
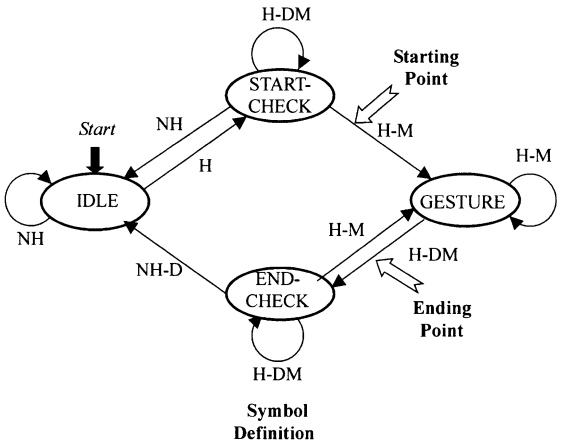


Fig. 4. Images of each processing step: (a) input image, (b) skin color similarity regions, (c) image after noise removal and dilation, (d) hand candidate regions, (e) detected hand region.



Symbol Definition

Symbol	Results of Hand Extraction	Results of Hand Tracking
H	Hand exists	Hand appears and moves
NH	No hand exists	None
H-DM	Hand exists	Hand appears but doesn't moves
H-M	Hand exists	Hand appears and moves
NH-D	No hand exists	Hand disappears

Fig. 5. States for gesture spotting.

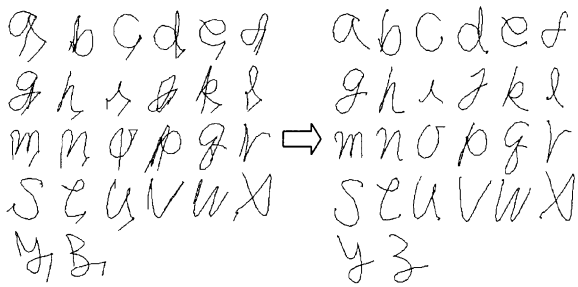


Fig. 6. The example of spotting results for alphabetical gestures.

3. Feature extraction

The performance of a general recognition system first depends on getting efficient features to represent pattern characteristics. There are several methods of representing such gesture trajectory feature: the two-dimensional edge and the time edge proposed by Seki [10], and the raw position, Cartesian velocity, and the polar and angular velocities proposed by Campbell [11]. There are many feature types such as the chain code, mesh code, momentum, and the MRF (Markov random fields), but all these features are based on only three basic features from a gesture trajectory: location, orientation and velocity, as shown in Fig. 7.

In Fig. 7, (C_x, C_y) indicates the center of gravity of the gesture, L_t the location feature at time t, V_t the velocity feature, θ_{1t} the angle feature based on (C_x, C_y), and θ_{2t} the angle features between two successive points. These features are calculated by the following formulas:

$$(C_x, C_y) = \left(\frac{1}{n}\sum_{t=1}^{n} X_t, \frac{1}{n}\sum_{t=1}^{n} Y_t\right), \tag{2}$$

$$L_t = \sqrt{(X_t - C_x)^2 + (Y_t - C_y)^2}, \tag{3}$$

$$\theta_{1t} = \tan^{-1}\left(\frac{Y_t - C_x}{X_t - C_y}\right), \tag{4}$$

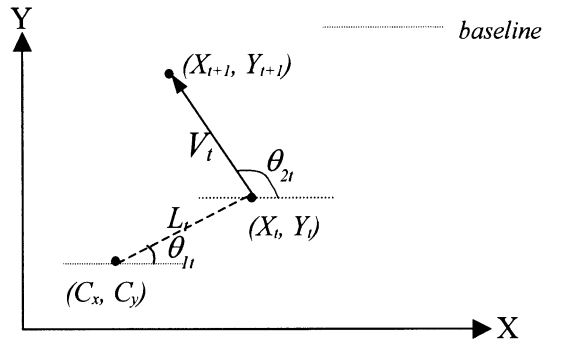


Fig. 7. Basic features between two points of a gesture sequence.

$$\theta_{2t} = \tan^{-1}\left(\frac{Y_{t+1} - Y_t}{X_{t+1} - X_t}\right), \tag{5}$$

$$V_t = \sqrt{(X_t - X_{t+1})^2 + (Y_t - Y_{t+1})^2}. \tag{6}$$

We analyze the effectiveness of these basic features contained in a gesture trajectory, and also combine them in the Cartesian and polar coordinate systems to test their recognition rate.

3.1. Location, orientation and velocity features

3.1.1. Location feature

The location feature used in this research measures the distance from the center of gravity to a point of the gesture. The reason we use the center of gravity rather than the origin is because different location features are generated for the same gesture according to the different starting gesture points. Using the center of gravity solves this problem. The location feature is similar to the velocity feature but differs in that it measures the distance from a fixed center point instead of the distance between two successive points.

The location feature varies according to the overall gesture size. Thus, we normalize the feature by using the following formula:

$$L_{\max} = \max_{t=1}^{n}(L_t), \tag{7}$$

$$l_t = \frac{L_t}{L_{\max}}. \tag{8}$$

Here, L_t is a value from Eq. (3) and $L_{\max}$ is the longest distance from a center point to any point in a gesture. The l_t value has the result of normalizing size. The l_t values obtained above are between 0.0 and 1.0, and can be converted into feature codes by a partitioning of the range 0.1–1.0. The appropriate partitioning interval can be determined by experimenting with different intervals.

3.1.2. Orientation feature

As described above, the orientation feature is represented by the angle based on the center of gravity (θ_{1t}) and the angle between two successive points (θ_{2t}). We use a chain code to convert these angles into feature codes.

A chain code is a useful coding method for converting angle values to feature code. A chain code is based on a grid and represents 4-connectivity or 8-connectivity according to the connectivity. An 8-connectivity code is assigned to each of eight possible directions between two points, where 0 is assigned to movement to the right, and the code increases by one going counterclockwise as shown in Fig. 8.

We present the chain code generation algorithm using the C language between two gesture points (X_t, Y_t) and (C_x, C_y) or (X_{t+1}, Y_{t+1}), where NumOfChain indicates an arbitrary number of chain codes.

$d_{x1} = X_t - C_x;$

$d_{y1} = Y_t - C_y;$

$d_{x2} = X_t - X_{t+1};$

$d_{y2} = Y_t - Y_{t+1};$

$\theta_1 = a\ tan\ 2(d_{y1}, d_{x1});$

$\theta_2 = a\ tan\ 2(d_{y2}, d_{x2});$

$\theta_{1t} = (\text{NumOfChain} - (\text{int})\ (\theta_1/(PI/4) + 0.5 + (d_{y1} < 0)*$ NumOfChain))%NumOfChain;
$\theta_{2t} = (\text{NumOfChain} - (\text{int})(\theta_2/(PI/4) + 0.5(d_{y2} < 0)*$ NumOfChain))%NumOfChain;

3.1.3. Velocity feature

The velocity feature is based on the fact that each gesture is made at different speeds. For example, a simple gesture, such as a circle gesture, has an almost

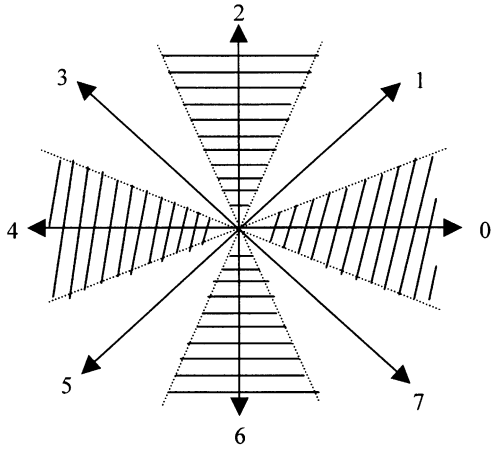


Fig. 8. The eight-directional chain code.

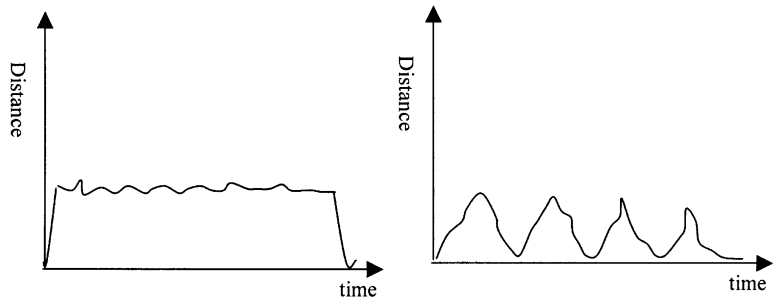


Fig. 9. Differences in velocity values from two gestures. (a) Ideal velocity values of gesture 'circle'. (b) Ideal velocity values of gesture 'rectangle'.

non-varying speed and complex gestures such as the '*q*' or '*w*' gesture have varying speeds during gesture generation. This velocity feature is measured by the distances of two successive points, such as points (X_t, Y_t) and (X_{t+1}, Y_{t+1}). The velocity feature v_t is generated by the following equation.

$$V_{\max} = \max_{t=1}^{n} (V_t), \tag{9}$$

$$v_t = \frac{V_t}{V_{\max}}. \tag{10}$$

Here, V_t is the value from Eq. (6) and V_{max} is the maximum velocity in one gesture. The v_t value has the result of normalizing size. The v_t values obtained above are between 0.0 to 1.0, and can be converted into feature codes by partitioning the range from 0.1 to 1.0. Fig. 9 shows the differences in velocity values for different gestures.

3.2. Union of three basic features

3.2.1. *x–y map feature*

The most basic gesture trajectory features are the location and angle features. Each gesture's trajectory points are composed of different *x–y* coordinates such that each point has a different location and different angle values. We use the *x–y* map feature to unit the location and angle features.

The first step in computing an *x–y* map feature is to detect of MBR(minimum bounding rectangle) which includes the whole gesture for size normalization. The second step is to divide the MBR area into a number of feature spaces. The following equation describes the normalizing algorithm for these feature values.

$$x_{\max} = \max_{t=1}^{n} (X_t), \quad x_{\min} = \min_{t=1}^{n} (X_t), \quad y_{\max} = \max_{t=1}^{n} (Y_t),$$

$$y_{\min} = \min_{t=1}^{n} (Y_t), \tag{11}$$

$$x_t = \frac{X_t - x_{\min}}{x_{\max} - x_{\min}}, \qquad y_t = \frac{Y_t - y_{\min}}{y_{\max} - y_{\min}}. \tag{12}$$

Here $x_{\max}$ and $x_{\min}$ refer to the maximum and minimum values from one gesture trajectory. x_t and y_t are then the normalized values which have values between 0.0 and 1.0.

The *k*-means algorithm [9] is used for feature clustering and is described in detail later. The proper number of feature spaces is decided based on many experimental results. Fig. 10 shows the clustered result example of the *x–y* map features using the clustering algorithm.

3.2.2. *Union of x–y–v feature*

In the Cartesian coordinate system, the two-dimensional *x–y* space implicitly includes the location and angle features. However, the *x–y* space does not include velocity information. Therefore, we propose the *x–y–v* feature map that has three basic features. A three-dimensional clustering algorithm can easily accomplish the union of these features.

3.2.3. *ρ–φ map feature*

In contrast to the *x–y* map feature which implicitly includes angle and location information, the *ρ–φ* map

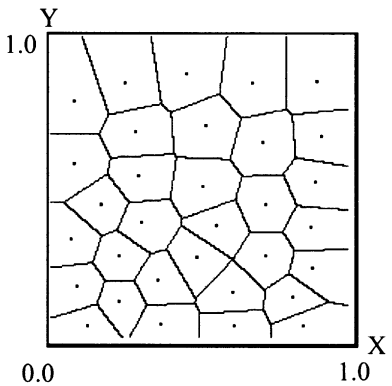


Fig. 10. Item role of the *x–y* features clustering using the *k*-means algorithm.

feature includes location and angle features separately. The ρ–φ map feature can be obtained using the location feature l_t and the angle feature θ_{1t}. This corresponds to converting the coordinate system from the x–y Cartesian to the ρ–φ polar system.

To obtain the ρ–φ feature map, the following equation is used:

$$\rho_t = l_t, \quad \varphi_t = \frac{\theta_{1t}}{2\pi}, \tag{13}$$

The factor φ_t is the number of degrees from θ_{1t} in Eq. (4) and φ_t also takes a normalized angle value between 0.0 and 1.0.

3.2.4. Union of ρ–φ–v feature

In the polar coordinate system, the two dimensional ρ–φ space is a very effective feature map for gesture recognition. But the ρ–φ space can also include velocity information. Therefore, we use the ρ–φ–v feature that has the three basic features and is independent of variance in the translation, rotation and scaling of a gesture trajectory.

Fig. 11(b) shows the trajectory of the letter 'a' gesture transformed from the x–y coordinate space into ρ–φ coordinate and (c) shows the same gesture in the ρ–φ–v coordinate. Fig. 11(c) refers to the three individual feature values according to their time sequence because this figure provides the variance of each feature.

3.3. Assigning weights to three basic features

From a number of experiments, we know that the three basic features have different degrees of importance for gesture recognition. The x–y or ρ–φ features are more stable values in the database than are the v values. This means that these features are more discriminative for gesture recognition independent of user or system variation than the v features. Experimental results show that the ρ and φ features have higher recognition rates than the v features. The main reasons are that the difference between the maximum and minimum velocity is small and the multi-tasking OS system does not guarantee the same processing speed during the preprocessing step using image processing.

Based on this observation, we propose the weighted x–y–v and ρ–φ–v features. If the three basic feature sets have weights according to the degree of their importance,

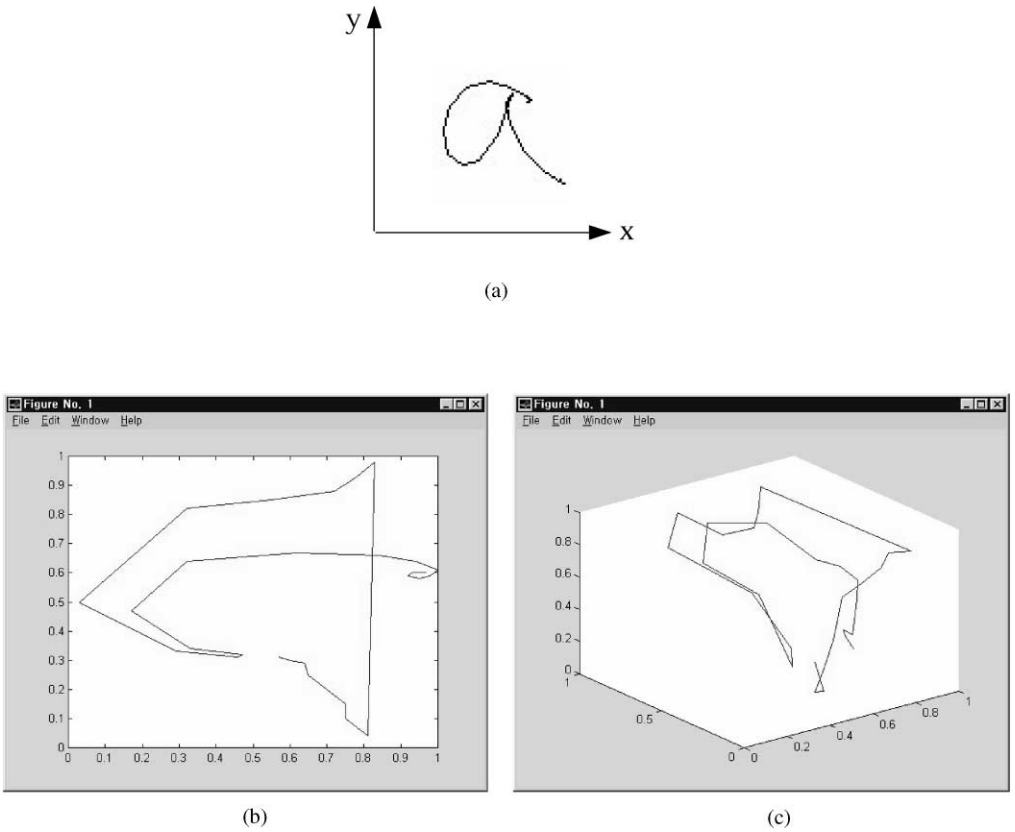


Fig. 11. The trajectory of alphabet 'a' gesture transformed from a x–y coordinate space into ρ–φ coordinate and the ρ–φ–v coordinate. (a) Alphabet 'a' gesture (b) 2D ρ–φ space of gesture 'a' (c) 3D ρ–φ–v space of gesture 'a'.

we can obtain more accurate recognition results. To assign weights, we set several weights to the three basic feature sets of a number of experiments. Basically, we set the weights of 1.0 to the x–y and ρ–φ features, which means that these features have a standard degree of importance. Next, we set the weight of the v feature with values between 0.9 to 0.1 and examined the fluctuation in the recognition rates. Finally, we obtained the best recognition rate when we set the weight of the v feature to 0.4.

4. Recognition using the HMM

4.1. Hidden Markov Model (HMM)

The HMM models are double stochastic processes as governed by an underlying Markov chain with a finite number of states, and a set of random functions each of which is associated with one state [12]. In discrete time instants, the process is in one of the states, and generates an observation symbol according to the random function corresponding to the current state. The model is hidden in the sense that all that can be seen is a sequence of observations. Quantitatively, an HMM is described as follows:

- A set of observation strings $O = \{O_1, \ldots, O_T\}$, where $t = 1, \ldots, T$
- A set of N states $\{s_1, \ldots, s_N\}$.
- A set of k discrete symbols $\{v_1, \ldots, v_k\}$.
- A state transition matrix $A = \{a_{ij}\}$, where a_{ij} is the transition probability from state s_i to s_j: $A = \{a_{ij}\} = \Pr(s_j \text{ at } t+1 \mid s_i \text{ at } t)$, $1 \leqslant i, j \leqslant N$;
- An observation probability matrix $B = \{b_{jk}\}$, where b_{jk} is the probability of generating symbol v_k from state q_j.
- An initial probability distribution for the states $\Pi = \{\pi_j\}$, $j = 1, 2, \ldots, N$; $\pi_j = \Pr(s_j \text{ at } t = 1)$.

From their definitions, a complete set of parameters of an HMM can be compactly expressed as $\lambda = (A, B, \pi)$. There are, in general, three basic problems that must be solved for the real application of the HMM: classification (evaluation), decoding, and training. The solutions to these problems are, in general, processed as the Forward–Backward algorithm, the Viterbi algorithm, and the Baum–Welch algorithm.

The generalized topology of an HMM is a fully connected structure known as an ergodic model, where any state of the model can be reached from any other state of the model. In this model, the state index transits only from the left to the right as time increases and this property is easily applied to dynamic gesture recognition. It is clear that the left–right model is just the same as a rearranged ergodic model using the transition conditions. The global structure of the HMM is constructed by parallel connection of each HMM ($\lambda_1, \lambda_2, \ldots \lambda_M$) and thus adding a new HMM or deleting an existing HMM can be easily done. Here λ indicates a constructed HMM model for each gesture and M indicates the number of gestures for recognition.

4.2. Observation sequence generation using vector quantization

The extracted features are quantized so as to obtain discrete symbols to apply to the HMM. From our gesture trajectory in the ρ–φ–v coordinate, we make discrete symbols using the k-means vector quantization algorithm. The k-means clustering algorithm [9] is adopted to classify the gesture tokens into the L clusters in the feature space. This algorithm is based on the minimum distance between the center point of each cluster and the feature points.

The codebook consists of the symbol number and the centroid coordinates of each cluster. In the experiments, the symbol observation codes can be determined using the distance between an observation and the centroid of each cluster. From the experimental results, the optimal number of cluster center points is set at 48 in our ρ–φ–v space.

Step 1: *Initialization*. Build up an initial VQ codebook (z_i, $1 \leqslant i \leqslant L$).

Step 2: Repeat the following for all train vectors.

2.1: *Classification*. Classify each element of the train vectors $\{\mathbf{x}_k\}$ into one of the C_i clusters using the minimum distance classifier ($\mathbf{x}_k \in C_i$ if $\mathrm{d}(\mathbf{x}_k, z_i) \leqslant \mathrm{d}(\mathbf{x}_k, z_j)$ for all $j \neq i$).

2.2: *Codebook updating*. Update the codebook by computing the centroid of the training vectors in each cluster.

Step 3: Termination. Terminate if the decrease in the overall difference value is below a threshold value; otherwise go to Step 2.

4.3. Training and experimental results

An experimental system using the HMM for an alphabetical hand gesture recognition system was implemented on a personal computer with an image capture board (Matrox Meteor). The input images were captured by a CCD camera with a resolution of 120×160 pixels. The processing speed is 5–7 frames/s. Since our computer can process over 5 frames/s, it is possible to use the proposed system for real-time interaction. One gesture captured by our system has 20–30 traced points. Therefore, the total processing time including recognition takes 4–6 s for one gesture.

The recognition software is implemented in Visual C++ 6.0 on Windows 98. The proposed algorithm was

applied to a database containing 4800 alphabetical gestures of 20 persons. Each person was asked to draw his/her gestures 10 times for each of the 48-gestures. There were two databases, one for training and the other for untrained data, constructed for testing the proposed algorithm. To divide the database, we saved even-numbered gestures in the training data set and odd-numbered gestures in the testing data set. Our gesture database is available by mailing a request to yoonhs@etri.re.kr. This database, however, contains the sequences of x–y coordinates representing unspotted gestures.

From the proposed system, we test the importance of the three basic features in the Cartesian and polar coordinate systems, and the observation sequence for the HMM is quantified by the k-means clustering algorithm. We have satisfactory recognition rates of 98.96% in the training data set and 93.25% in the testing data sets using weighted x–y–v features. Table 1 shows the results using 2400 training data and 2400 testing data under the various feature sets. As expected, the training data yield a higher recognition rate than does the testing data.

If we analyze these results using separated features, we see that the two angle features (θ_1 and θ_2) result in similar recognition rates (87%), and that they are better than the recognition rate using the location (l) or velocity (v) features. This means that the most effective feature among the three basic features is the angle feature. The location feature shows a lower discrimination power (47%) than that of the angle feature. Finally, the velocity feature results in the lowest recognition rate of 33% because of the effect of the multitasking OS and low discrimination power.

The test results from the unified features show that the x–y feature and the ρ–φ feature yield a higher recognition ratio than the separated individual features. In addition, the x–y–v and ρ–φ–v features, which include the velocity information, show higher recognition rates than when using the velocity information.

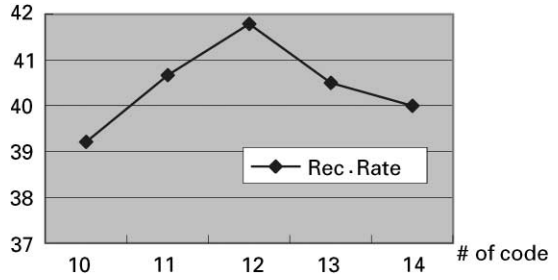


Fig. 12. Recognition rate according to the number of location feature codes.

Lastly, assigning weights to the velocity features results in better recognition than without using weights for both the x–y–v and ρ–φ–v features. This shows that using weights is helpful in improving the recognition rate.

The feature code numbers shown in Table 1 were determined in many experiments. Figs. 12–15 show the results of the experiments to determine the best number of feature codes and Fig. 16 shows the results of the experiments to determine the best weight using the x–y–v feature code.

5. Conclusion

Recent research in hand gesture recognition has aimed applications at sign language recognition, household electronic appliances control, human–computer interaction, and VR. In this research, we intend to develop a graphic editor system operated by hand gestures. We do

Table 1
Total test results

Feature class	Feature space	No. of feature code	Recognition results (%)		
			Training data	Testing data	Overall
Separated	Location (l)	12	52.13	41.79	46.96
	Angle (θ_1)	8	89.21	85.29	87.25
	Angle (θ_2)	8	93.67	81.29	87.48
	Velocity (v)	4	37.21	28.53	32.87
Combined in Cartesian system	x–y	36	97.58	91.79	94.69
	x–y–v	48	98.79	92.17	95.48
	Weighted x–y–v	48	98.96	**93.25**	96.10
Combined in polar systems	ρ–φ	36	98.29	89.88	94.08
	ρ–φ–v	48	99.08	89.63	94.35
	Weighted ρ–φ–v	48	**99.13**	92.96	96.04

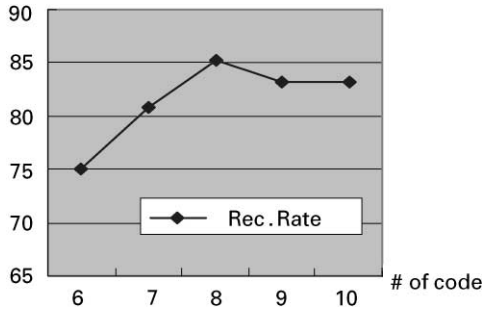


Fig. 13. Recognition rate according to the number of angle (θ_1) feature codes.

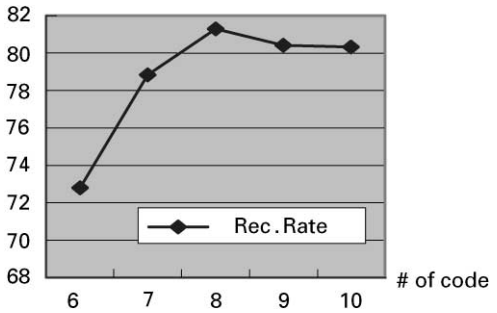


Fig. 14. Recognition rate according to the number of angle (θ_2) feature codes.

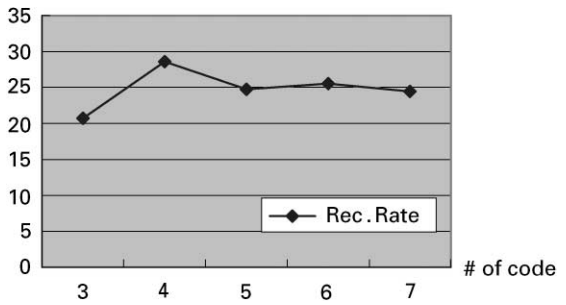


Fig. 15. Recognition rate according to the number of velocity feature codes.

not expect that the technology will directly replace the mouse, but it is likely that some part of the next-generation user interface will employ some form of hand gesture recognition using an image sensor.

For hand gesture recognition, there are four main steps. The first step is the hand location algorithm that detects the skin-colored regions in an image by using a color histogram matching technique. The second step is the spotting algorithm, which detects a spotting area using domain knowledge. The third step is the feature extraction algorithm based on three basic features. The final step are the training and testing experiments using the HMM.

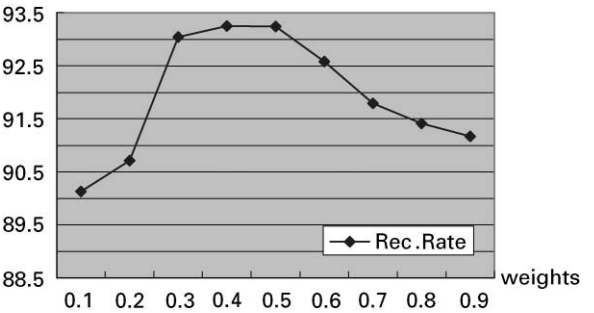


Fig. 16. Recognition rate according to the x–y–v feature weight changes.

The main topic of this paper is the examination of the discrimination capabilities of the location, angle and velocity features for gesture recognition, which are obtained from the coordinates in a gesture trajectory. We have shown that the effective combination of these features can yield reasonable recognition rates. In addition, we have shown that using weights based on feature importance can provide even better recognition capability.

The HMM is a very effective tool for hand gesture recognition in the time domain. In batch test results using file information, our method had good results with a success rate of 93% in every case. Also, the on-line test results that used direct camera input and real time HMM recognition has a success rate of more than 85%, which is very encouraging.

Future research can focus on two areas. The first would be the enhanced hand location algorithm using color analysis because color information is very sensitive to the image-capturing environment. The second one would be a new feature detection algorithm for the HMM because we do not fully use characteristics of a dynamic gesture such as optical flow information, hand posture, momentum and curvature.

References

[1] W.T. Freeman, C.D. Weissman, Television control by hand gesture, Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland, June 1995, pp. 179–183.

[2] J.S. Kim, C.S. Lee, K.J. Song, B. Min, Z. Bien, Real-time hand gesture recognition for avatar motion control, Proceedings of HCI'97, February 1997, pp. 96–101.

[3] T. Starner, A. Pentland, Visual recognition of american sign language using hidden Markov model, Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland, June 1995, pp. 189–194.

[4] J. Yang, Y. Xu, C.S. Chen, Human action learning via hidden markov model, IEEE Trans. Systems, Man, Cybernet. 27 (1) (1997) 34–44.
[5] T.S. Huang, A. Pentland, Hand gesture modeling, analysis, and synthesis, Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland, June 1995, pp. 73–79.
[6] Jakub Segan, Controlling computer with gloveless gesture, in Virtual Reality System'93, 1993, pp. 2–6.
[7] E. Hunter, J. Schlenzig, R. Jain, Posture estimation in reduced-model gesture input systems, Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, June 1995, pp. 290–295.
[8] M.J. Swain, D.H. Ballard, Color indexing, International Journal of Computer Vision 7 (1) (1991) 11–32.
[9] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1992.
[10] S. Seki, K. Takahashi, R. Oka, Gesture recognition from motion images by spotting algorithm, ACCV'93, 1993, pp. 759–762.
[11] L.W. Campbell, D.A. Becker, A. Azarbayejani, A.F. Bobick, A. Pentland, Invariant features for 3-D gesture recognition, Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, 1996, pp. 157–162.
[12] L.R. Rabiner, A tutorial on hidden Markov models and selected application in speech recognition, Proc. IEEE 77, 1989, pp. 267–293.

About the Author—YOON, HO-SUB received the B.S. and M.S. degrees in Computer Science from SoongSil University, Seoul, Korea, in 1989 and 1991, respectively. He currently works at CSTL in ETRI and is a Ph.D. student at KAIST in Taejon. His research interests are pattern recognition and computer vision.

About the Author—SOH, JUNG received his B.S. in Electrical and Computer Engineering from the University of Wisconsin at Madison in 1986, and his M.S. and Ph.D. in Computer Science from the State University of New York (SUNY) at Buffalo in 1988 and 1998, respectively. He worked at the Center of Excellence for Document Analysis and Recognition, SUNY at Buffalo, from 1988 to 1991. He is currently a senior research scientist at the Computer Software Technology Lab (CSTL) of Electronics and Telecommunications Research Institute (ETRI), Taejon, Korea. His research interests are in pattern recognition and computer vision.

About the Author—J. BAE, YOUNGLAE received the B.Sc degree in Marine Physics from Seoul National University, M.Sc degree in Computer Sciences from Hanyang University, Seoul, and Ph.D. degree in Electronic Engineering from University of Kent, England, in 1976, 1986 and 1995, respectively. From 1976 to 1979 he was a naval officer. Since 1980 he has been with Computer & Software Technology Lab/ETRI, where he is currently a principal research scientist at the Image Processing Research Department. Also as a visiting scientist, he was with the Informatik Institut, Technische Universität München, Germany, from 1988 to 1989. His major research interests include image processing, parallel processing, Fuzzy set theory and signature verification. Dr. Bae has been a regular member of the IEEE PAMI, IEEE GRSS, IEE, KITE, KISC, KSRS.

About the Author—YANG, HYUN SEUNG received the B.S. degree in Electronic Engineering from Seoul National University, Korea, in 1976, and the M.S. and Ph.D. degrees in Electrical Engineering from Purdue University, West Lafayette, Indiana, in 1983 and 1986, respectively. While he was at Purdue University, he worked at the Robot Vision Lab as a Research Associate from 1983 to 1986. From September 1986 to August 1988, he was an Assistant Professor in the Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA. He joined the faculty of KAIST in August 1988. His research interests include computer vision, artificial intelligence, neural networks, intelligent robots, multimedia and intelligent agents.