A Hierarchical Temporal Memory Based Hand Posture Recognition Method

Yea-Shuan Huang and Yun-Jiun Wang, *Member, IAENG*

***Abstract*—In various pattern recognition applications, angle variation is always a main challenging factor for producing reliable recognition. To increase the endurance ability on angle variation, this paper adopts a Hierarchical Temporal Memory (HTM) algorithm which applies temporal information to organize time-sequence change of image features, and constructs invariant features so that the influence of angle variation can be effectively learnt and overcome. The proposed multi-angle HTM-based posture recognition method consists of two main modules of Hand Posture Image Pre-processing (HPIP) and Hand Posture Recognition (HPR). In HPIP, each input image is first processed individually by skin color detection, foreground segmentation and edge detection. Then, the three processed results are further combined linearly to locate a hand posture region. If a forearm exists in the located hand posture region, a forearm segmentation process will be executed to keep only the part of palm. In HPR, the normalized image is forwarded to a HTM model for learning and recognizing of different kinds of hand postures. Experiment results show that when using the same continuous unconstrained hand posture database, the proposed method can achieve an 89.1% high recognition rate for discriminating three kinds of hand postures, which are scissors, stone and paper, and outperforms both Adaboost (78.1%) and SVM (79.9%).**

***Index Terms*—Hierarchical Temporal Memory, Multi-angle hand posture recognition.**

I. INTRODUCTION

In recent years, the interaction of Human Computer Interface (HCI) has become a popular research domain. Many latest types of HCI has come out gradually such as LED-based multi-touch sensor for LCD screens[1], audio modeling system[2], and moving object sensory detection system[3]. Comparing to other methods, moving object sensory detection system requires only a commonly-used camera to interact with users, which is the most nature and convenient way for HCI and therefore has gained much research attention in recent years. In light of this, this paper focuses on hand posture recognition because it plays an important role in a moving object sensory detection system.

In general, a posture recognition system mainly contains two processing modules: posture detection and posture recognition. For posture detection, it could be divided into two types of processing methods. The first type is skin color detection which locates the posture area by either using one set or multiple sets of threshold values to determine the range of skin color in a certain color space or applying the method of probability learning model to compute the probability that a certain color belongs to skin color. The second type is the moving object detection which in general considers only a single moving hand in the image so that the location of the hand area can be known easily when a moving object was detected. There are three common ways in detecting moving objects, which are background subtraction[4], temporal differencing[5], and optical flow[6]. The first two apply subtraction of different images and determine the changed part of image as foreground and the non-changed part of image as background. While optical flow utilizes the displacement of moving objects from multiple images to count the variation of gray-scale value of images at a time for each pixel point, and construct the speed field formed by speed vector. As for posture recognition, Adaboost, Support Vector Machine (SVM), and Neural Network are the most commonly used recognition algorithms. Three methods have each pros and cons on recognition accuracy and tolerance limits. For example, the recognition rate of SVM is better than those of the other two. However, when it comes to the context of image preprocessing of training the data base, Adaboost has a higher correct rate than others. Furthermore, if only the execution speed of recognition is considered, Adaboost is the fastest in general.

While developing a hand posture recognition system, image noise, light, and angle may influence the recognition accuracy considerably. Therefore, the purpose of this study is to figure out how to derive stable features from posture images under the environment of varying light and angles, and to construct a robust hand posture recognizer. To this end, we propose a multi-angle HTM-based posture recognition method (MA-HTM-PR) which includes a useful hand posture region locating step and a forearm excluding step to derive the proper hand posture region, and adopts a HTM algorithm to induce the continuous changing images and form invariant features so that the constructed classifier is insensitive to various noise such as angle variation.

This paper in total contains six sections. Sec. 2 presents the overall framework of the proposed MA-HTM-PR method, Sec.3 and Sec. 4 describes the proposed HPIP and HPR respectively, and Sec. 5 specifies the experiments and their results by using different recognition methods. Finally, Sec. 6 makes a conclusion and points out the future research direction.

II. MODULAR ARCHITECTURE

This section presents the overall framework of the proposed MA-HTM-PR method which mainly contains two modules of Hand Posture Image Pre-processing (HPIP) and Hand Posture Recognition (HPR) as shown in Fig. 1. At the HPIP module, each input image is first processed by skin

Manuscript received April. 08, 2013; revised May. 06, 2013. This work was supported by National Science Council, Republic of China, under grants NSC 101-2221-E-216-037-MY2.

Yea-Shuan Huang is with the CSIE, Chung-Hwa University, Taiwan, TEL: 886-3-5186408; e-mail: yeashuan@chu.edu.tw).

Yun-Jiun Wang has finished his master program from the CSIE of Chung-Hwa University in 2012.

color detection, foreground segmentation, and edge detection individually. Then, the three processed results are combined with a linear weighting method to acquire the hand posture region. Further, a simple mechanism is to check whether the hand posture region contain a forearm or not. If a forearm exists, a forearm segmentation process will be executed to exclude the forearm so that the hand posture region contains only the parts of palm and fingers. Then, the image of the processed hand posture region will be normalized to a fixed size. At the HPR module, the normalized hand posture image is forwarded to a HTM model for learning and recognizing of different kinds of hand postures.

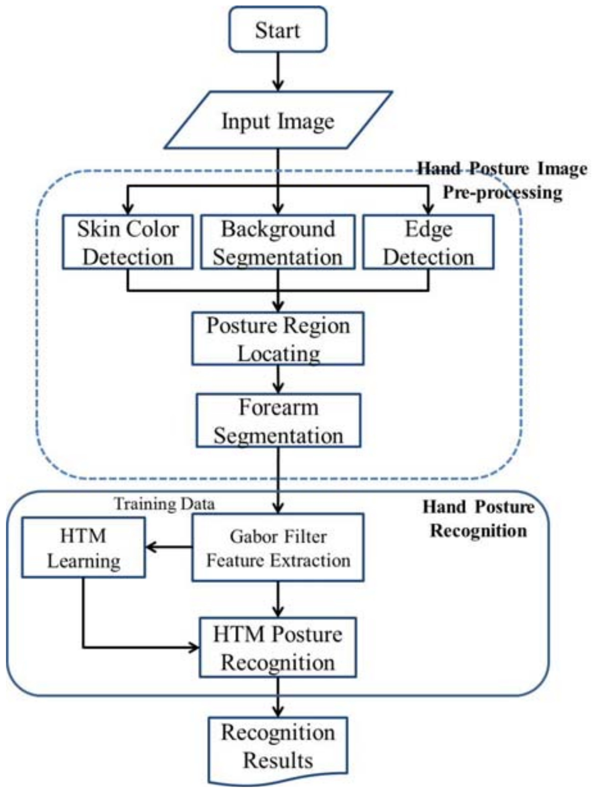

Fig.1. Flow chart of hand posture recognition algorithm.

III. Hand Posture Image Pre-processing

The purpose of this module is to locate the valid hand posture region from the input image. To this end, skin color detection, foreground segmentation, and edge detection are performed on the input image, and their processed results are further combined with a linear weighting method. Then a forearm segmentation process will check and derive the valid hand posture region. Here, a Codebook algorithm [7] is used for foreground segmentation, and a 3*3 Sobel operator is used to detect the edge pixels.

A. Skin Color Detection

Since most pixels of hands belong to skin color, skin color detection becomes essential in locating hands from images. We adopt a Bayesian Classifier [8] with YC_bC_r color space to detect the skin color pixels. By manually identifying skin and non-skin color pixels on images, a Bayesian skin-color detector based on C_bC_r information can be trained. Let H_S and H_n be individually the histograms of skin and non-skin color pixels, $(C_{b,}C_r)$ denote a color vector, $s[(C_{b,}C_r)]$ and $n[(C_{b,}C_r)]$ be the pixel numbers of color $(C_{b,}C_r)$ in H_S and H_n respectively, C_s and C_n represent the total pixel numbers of H_S and H_n, $P((C_{b,}C_r)|skin)$ and $P((C_{b,}C_r)|nonskin)$ be the probability that a skin pixel appears a specific $(C_{b,}C_r)$ value, $P((C_{b,}C_r)|nonskin)$ be the probability that a non-skin pixel appears a specific color vector $(C_{b,}C_r)$, $P(skin)$ and $P(nonskin)$ be the proportion of skin and non-skin pixels, and $P(skin|(C_{b,}C_r))$ be the probability that a pixel with color $(C_{b,}C_r)$ is indeed a skin pixel. When a pixel with color vector $(C_{b,}C_r)$, then the probability that it is a skin pixel becomes

$$P(skin|(C_{b,}C_r)) = \frac{P((C_{b,}C_r)|skin)P(skin)}{P((C_{b,}C_r)|skin)P(skin) + P((C_{b,}C_r)|nonskin)P(nonskin)}$$

$$P((C_{b,}C_r)|skin) = \frac{s[(C_{b,}C_r)]}{C_s}$$

$$P((C_{b,}C_r)|nonskin) = \frac{n[(C_{b,}C_r)]}{C_n}$$

$$P(skin) = \frac{C_s}{C_s + C_n},\ P(nonskin) = \frac{C_n}{C_s + C_n}$$

We defined a threshold θ, when $P(skin|(C_{b,}C_r)) > \theta$, then a pixel with color vector $(C_{b,}C_r)$ is determined to be a skin pixel; otherwise, it is determined to a non-skin pixel. Experiment results revealed that when θ is set to 0.06, the performance was best.

B. Foreground segmentation

In the foreground segmentation step, we adopt the Codebook background model which quantizes and clusters the variation patterns of background of each pixel. To construct the Codebook background model [7], according to a sequence of images each pixel will create its own codebook which contains a number of codeword. Assume that $C = \{c_1, c_2, \ldots, c_L\}$ represents the codebook of one pixel which has L codeword, each codeword c_i consists of a RGB color vector $v_i = \{\overline{R_i}, \overline{G_i}, \overline{B_i}\}$ and six parameters $aux_i = (\check{I}_I, \hat{I}_i, f_i, \lambda_i, p_i, q_i)$, where $\check{I}$ and $\hat{I}$ represent the maximum and minimum luminance of this codeword, f is the number of success match, λ is the value of Maximum Negative Run-Length(MNRL) which is the longest time that this codeword has not been updated, p is the first time of this codeword to appear, and q is the last time of this codeword to appear. In the training step, let $= \{x_1, x_2, \ldots, x_N\}$ be the N RGB color vectors at a certain pixel position in N continuous images. For each x_t, if it is similar to an existed codeword c_m, then c_m will be updated; however if there isn't any existed codeword similar to x_t, a new codeword should be constructed. The closeness of two colors is examined by two measurements: color distance and luminance distance. Only the two measurements are small enough, the two colors are considered to be close. The Codebook construction algorithm is described as follows.

Step 1. Initialize L to 0, t to 1, and C to an empty set ϕ;
Step 2. For each pixel x and t = 1 to N, repeat
(a) Calculate the luminance value I of x_t

$$I = \sqrt{R_t^2 + G_t^2 + B_t^2}$$

(b) For each codeword c_i, compute the color distance of this codeword and x_t, and its luminance bounds

$$colordist(x_t, v_t) = \|x_t\|\,|sin\theta|$$

$$I_{low} = \alpha\hat{I},\ I_{high} = min\left\{\beta\hat{I}, \frac{\check{I}}{\alpha}\right\}$$

where α and β are two constants with $\alpha < 1$ and $\beta > 1$, the value of α is usually set to be between 0.4 to 0.7, and the value of β is between 1.1 to 1.5.

(c) find a codeword c_m which satisfies the following two conditions

1. Color closeness

$$colordist(x_t, v_m) \le \varepsilon$$

2. Luminance closeness

$$brightness(I, \langle \check{I_m}, \hat{I_m} \rangle) = true$$

here

$$brightness(I, \langle \check{I}, \hat{I} \rangle) = \begin{cases} true & , \text{if } I_{low} \le \|x_t\| \le I_{high}; \\ false & , \text{otherwise.} \end{cases}$$

(d) If C is an empty set ϕ or there is no codeword which matches the two conditions, let $L = L + 1$ and create a new codeword c_L.

$$v_L = \{R_t, G_t, B_t\};$$

$$aux_L = (I, I, 1, t-1, t, t);$$

(e) If we find a matched codeword c_m, then c_m is updated.

$$v_m = \left(\frac{f_m\overline{R_m}+R}{f_m+1}, \frac{f_m\overline{G_m}+G}{f_m+1}, \frac{f_m\overline{B_m}+B}{f_m+1}\right)$$

$$aux_m = (\min\{I, \check{I_m}\}, \max\{I, \hat{I_m}\}, f_m + 1, \max\{\lambda_m, t - q_m\}, p_m, t)$$

Step 3. Re-calculate the $\lambda_i (i = 1, \ldots, L)$ of the whole codeword as

$$\lambda_i = max\{\lambda_i, (N - q_i + p_i - 1)\}$$

After processing all pixels with the above algorithm, the codebook background model is constructed, but some codeword might still correspond to noise or foreground information. In order to filter out such codeword, a MNRL mechanism is designed. Basically, not only static scene and objects but also swinging objects such as shacking branches, grass and water can be truly background. Therefore, if the MNRL parameter λ representing the longest non-update period of one codeword is larger than a threshold T, the corresponding codeword is regarded as either noise or foreground color and will be deleted. After finishing training the codebook background model, the trained codebook model is used to determine whether a pixel is foreground or not by the following process.

For each pixel $x = (R, G, B)$ of an input image,

Step 1. Calculate its luminance $I = \sqrt{R^2 + G^2 + B^2}$

Step 2. Compare x to every codeword c_i of this pixel sequentially, if both of their color distance and luminance distance are smaller than their individual thresholds, x is determined to be a background pixel and the matched codeword is updated accordingly; otherwise x is determined to be a foreground pixel.

C. Hand Region Locating and Forearm Segmentation

After performing foreground segmentation, edge detection, and skin color detection on each input image, three binary images (0 and 255) are generated. With a linear weighting method, the three binary images are combined together by

$$Result(x) = \begin{cases} 255, \text{ if } \begin{bmatrix} \alpha * Res_{Skin}(x) \\ +\beta * Res_{Edge}(x) \\ +\gamma * Res_{Codebook}(x) \end{bmatrix} > 128 \\ 0 \text{ , otherwise} \end{cases}$$

where $Result(x)$ is the integrated binary value of x, $Res_{Skin}(x)$, $Res_{Edge}(x)$ and $Res_{Codebook}(x)$ are the individual binary result of skin color detection, edge detection and foreground segmentation, and α, β and γ are three weight values representing the importance of the three kinds of responses. By applying morphology operation and connected component analysis on the integrated binary image, the valid hand region can be easily obtained. Fig.2 is one sample of hand posture region locating process, Fig.2(a) is the original image, Fig.2(b) is the result of skin color detection, Fig.2(c) is the result of edge detection, Fig.2(d) is the result of foreground segmentation, and Fig.2(e) is the combined result.

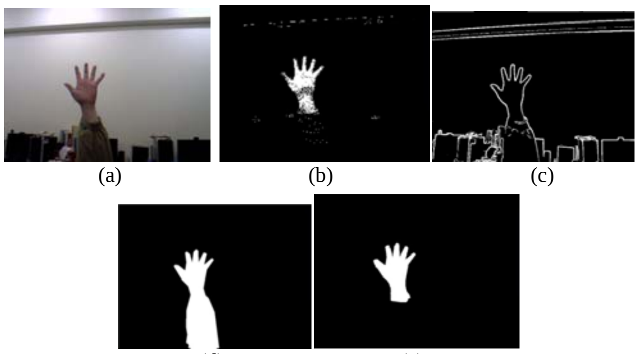

Fig.2. Example of posture region location, (a)original image, (b)result of skin color detection, (c)result of edge detection,(d)result of foreground segmentation, (e) combined result.

For hand posture recognition, both palm and fingers are important, but the forearm part is irrelative and may even degrade the recognition rate. Therefore, it is necessary to remove the forearm information effectively. For this purpose, a forearm removal mechanism is designed. The main idea of this mechanism is deduced from the facts that palm and fingers in general contains much more edge information than the forearm part, and the gravity center (GC) of the detected edge points in a valid hand region generally is located at the upper part of palm. Therefore, if a GC is estimated, then the proper hand region could be approximately defined. With this realization, we compute the gravity center from the detected edge points of the extracted valid hand region first, then the four distances (x_1, x_2, y_1, and y_2) between this GC and the left, right, top, and bottom boundaries of the valid hand region are calculated individually. Afterwards, the left, right, and bottom boundaries are updated independently if some of the following situations are matched.

$$\begin{cases} \text{right boundary} = GC.x + 1.1 \times x_1 & \text{, if } x_2 > 1.1 \times x_1; \\ \text{left boundary} = GC.x - 1.1 \times x_2 & \text{, if } x_1 > 1.1 \times x_2; \\ \text{bottom boundary} = GC.y + 1.2 \times y_1 & \text{, if } y_2 > 1.2 \times y_1; \end{cases}$$

After that, each resultant hand image is normalized into 128*128 pixels, and this normalized image becomes the input data to the successive HTM model for training and recognizing different kinds of hand postures.

IV. HAND POSTURE RECOGNITION

Hierarchical Temporal Memory (HTM) [9] is a biometric engine model based on the memory-prediction theory of brain function described by Jeff Hawkins in his book On Intelligence[10]. HTM is able to discover and infer the high-level causes of observed input patterns and sequences, so that it can build an increasingly complex model of the world. By learning the sequence of continuous images, HTM can get invariant features of various kinds of objects so that it is able to get robust recognition on unseen patterns. Each HTM region learns by identifying and memorizing spatial patterns - combinations of input bits that often occur at the same time. It then identifies temporal sequences of spatial patterns that are likely to occur one after another. After an HTM has learned the patterns in its world, it can perform inference robustly on novel inputs [11][12].

A. Structure of HTM Network

A typical HTM network is a tree-shaped hierarchy of levels that are composed of smaller elements called nodes. A single level in the hierarchy is also called a region. Higher hierarchy levels can reuse patterns learned at the lower levels by combining them to memorize more complex patterns. HTM uses the time clue to perform an unsupervised learning, summarize the training data, and produce the invariant feature. Fig. 3 shows the basic computation structure of HTM which mainly contains four kinds of processing nodes:

(1) Sensor Node: it locates at the bottom layer of a HTM network, receives the sensed data (such as images, sound and pressure measurements, etc.) and transports the data to the one-layer above HTM nodes.

(2) HTM node: it is the main computation node of HTM and can be formed into several levels according to the complexity of the dealing problem. Each HTM node has the same basic functionality which performs unsupervised learning to infer the input data and produce time and space similar categories. In the learning stage, nodes in lower level can derive smaller concepts which are more restricted in time and space, and nodes of upper level can derive more general concepts. Input data comes in from the lower level nodes and the HTM nodes output the generated concepts. The top level usually has a single node that stores the most genera concepts. When in inference mode, a node in each level interprets information coming in from its child nodes in the lower level as probabilities of the concepts it has memorized.

(3) Category Node: during the training stage, it performs a supervised learning algorithm (such as SVM) to construct a robust classifier based on the invariant feature inferred by the HTM top node. Then, during the recognition stage, it produces the recognition result for each input pattern.

Briefly speaking, the sensor node at the bottom of network gets the data to learn and train the model; it will produce the inference belief and transport to the HTM node in next level. In Fig.3, the inference belief is transported from sensor node2 to HTM node3, and then further transported from HTM node3 to HTM top node4. The HTM top node4 will then transport the summarized invariant features to the category node5, and the category node5 uses the classifier to recognize the input data and output the final recognition result.

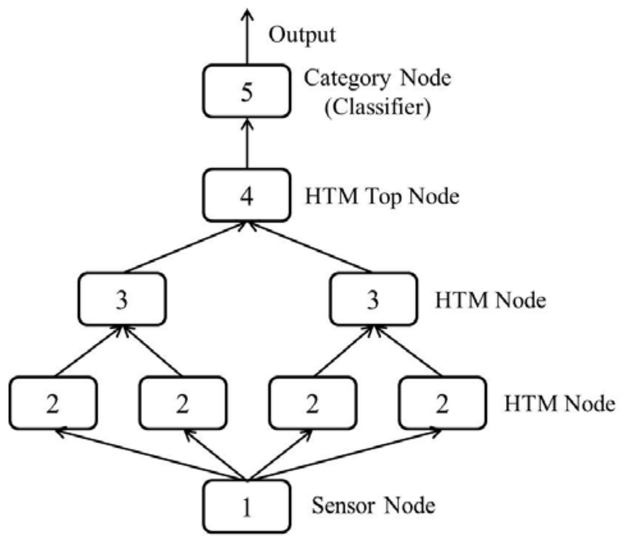

Fig.3. Structure of HTM network.

B. HTM Algorithm

HTM adopts a Bayesian network, spatial and temporal clustering algorithms with a tree-shaped hierarchy of neural nodes. Each HTM node contains a Spatial Pooler (SP) and a Temporal Pooler (TP) which performs two stages of processes (learning stage and inference stage). During learning, a node receives a temporal sequence of spatial patterns as its input. SP identifies frequently observed patterns and memorizes them as coincidences. Patterns that are considerably similar to each other are treated as the same coincidence, so a large amount of possible input patterns are reduced to a few number of known coincidences. TP partitions coincidences that are likely to follow each other in the training sequence into temporal groups. Each group of patterns represents a "cause" of the input pattern. During inference (recognition), the node calculates the set probabilities that a pattern belongs to each known coincidence. Then it calculates the probabilities that the input represents each temporal group. The set of probabilities assigned to the groups is called a node's "belief" about the input pattern. This belief is the result of the inference that is passed to one or more "parent" nodes in the next higher level of the hierarchy.

Fig.4 shows a schematic example of the learning process results in SP and TP. Fig.4(a) displays that there are in total 6 coincidences (#1~#6) stored in a SP. Fig.4(b) shows that the 6 spatial coincidences are clustered into 3 temporal groups (G1, G2 and G3) because coincidence #1 and #2, coincidence #3 and #4, and coincidence #5 and #6 have large correlations and often appear successively in time sequence. Basically, each temporal group takes one bin to represent, so there is a 3-bin output of TP. Fig.4(c) shows that after training SP and TP, a specific input pattern is taken into the HTM and it matches coincidence #3 in SP and group 2 in TP. Through both spatial and temporal clustering mechanisms, resolution in space and time is reduced considerably. Therefore, the derived information from an input pattern is rather insensitive to both spatial and temporal variations, and can be regarded as an invariant feature for representing this input pattern to further recognition.

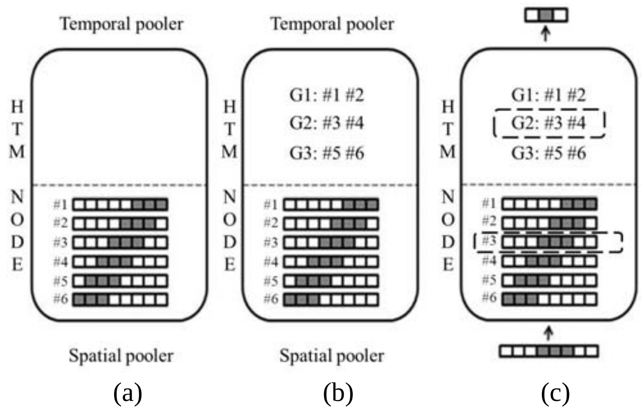

Fig.4. HTM learning process, (a) SP learns the training data, (b) TP groups the data with time inference, (c) invariant feature inference on unknown data.

(B.1) *Spatial Pooler Clustering*

The most important task of SP is to quantify the infinite input data and transform it into a limited number of coincidences. Each input pattern is first transformed into a fixed-size vector, and then this vector is compared with every coincidence recorded in SP. If there is no similar coincidence to this vector, a new coincidence should be constructed by taking this vector as its value and setting its occurrence number to be 1; otherwise, the occurrence number of the similar coincidence should be increased by one. For measuring the similarity between an input pattern *x* and a coincidence *w*, the Euclidean distance is calculated

$$d(x,w) = \sqrt{\sum_{j=1}^{N}(x_j - w_j)^2}$$

where N is the vector dimension of x and w. Let D be a threshold. If $d(x,w)$ is smaller than D, w and x are regarded to be similar to each other, otherwise they are not similar. If D is set too high, the number of coincidences will be small, and this will cause the inference ability of a HTM node degenerated. However, if D is set too low, the number of coincidences will be rather large, and this will not only take longer computation time in the inference stage but also decrease the generalization ability of this HTM node.

(B.2) *Temporal Pooler Grouping*

After SP has constructed all coincidences, TP then proceeds to perform the temporal clustering and construct the frequently occurred temporal groups. Each temporal group may contain several coincidences which are highly related to occur in time sequence with at least one other coincidence in the group. To do this, the HTM algorithm builds a time-adjacency matrix to record the occurrence numbers of every two coincidences by sequentially sending all the training data into the trained SP. Table 1 shows one example of time-adjacency matrix, where (i,j) represents the situation that coincidence i appears right after coincidence j, and the value of (i,j) is the occurrence number of situation (i,j). For example, the number in (6,8) is 12, it means that there are in total 12 times that coincidence 6 appears after coincidence 12 in the training sequence. After sending all training data to a SP, a $N * N$ time-adjacency matrix is obtained and N is the total number of coincidences in this SP. Then, the time-adjacency matrix is further transformed into a Markova graph, and this graph records the transformation probabilities of all coincidence pair as shown in Fig. 5. These transformation probabilities indeed become the fundamental information for the following TP clustering process.

TABLE 1. Temporal adjacency matrix

	#1	#2	#3	#4	#5	#6	#7	#8
#1	4	2	13	1	8	2	3	1
#2	0	1	6	9	2	3	14	2
#3	8	7	2	2	5	0	4	11
#4	0	8	4	3	1	3	2	9
#5	8	1	4	1	3	0	2	1
#6	1	4	1	2	1	5	0	12
#7	1	9	4	0	1	3	4	2
#8	3	1	5	11	0	8	3	1

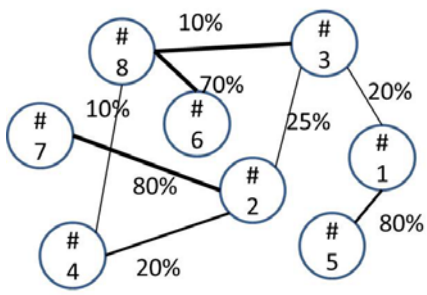

Fig.5. Markov chart of Table 1.

According to the Markov graph transformed from the time-adjacency matrix, the temporal pooler grouping by Agglomerative Hierarchical Clustering (AHC)[19] is performed. AHC sequentially clusters the coincidences having high correlation into the the same group and then removes all members of this group from the Markov graph. The clustering process is iterated until every coincidence has belonged to one group. Taking Fig. 6 as an illustration example, the first group contains coincidences #2, #4 and #7, the second group contains coincidences #3, #6 and #8, and finally the third group contains coincidences #1 and #5.

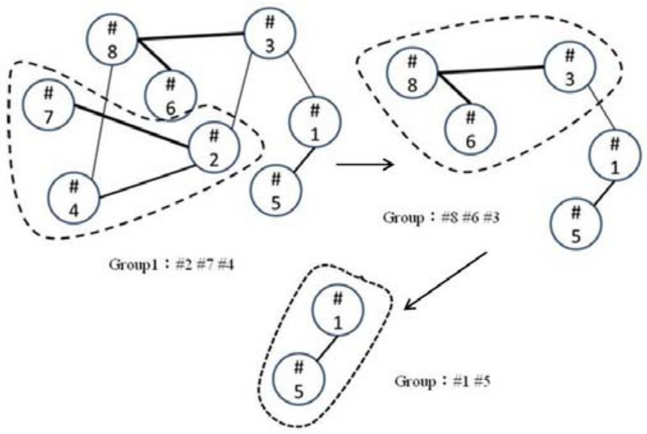

Fig.6. Example of Agglomerative Hierarchical Clustering.

V. Result and Experiment

In order to evaluate the performance of the proposed method, a self-constructed database with various angles of three kinds of hand postures (rock, paper and scissor) is developed. This database involves four persons and eight different environments including one simple background scene and seven complex background scenes (as shown in Fig. 7). The image resolution is 640*480 pixels, and the range of angle variation is $\pm 60^0$ from left to right (as shown in Fig.8(a)), $\pm 45^0$ from top to bottom (as shown in Fig.8(b)), and $\pm 30^0$ with a horizontal self-spin (as shown in Fig.8(c)). In fact, hand gesture images of this database were taken in a quite free and unconstrained way. In total, there are 3841 rock samples, 4097 paper samples, and 4242 scissor samples. Among them, only 150 images of each kind of hand gestures are used for training HTM and the remaining images are used for testing the trained HTM. In comparison with other databases, this database possesses considerably large variations in both illumination and orientations, and this makes the derived performance more representative to the practical applications.

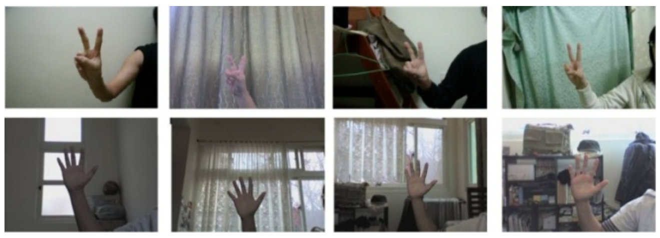
Fig.7. various hand postures in different backgrounds

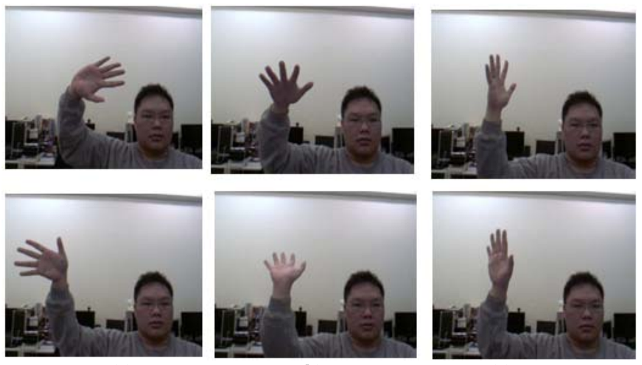
(a) (b) (c)

Fig.8. Examples of orientation variation, (a) ± 60 degrees from left to right, (b) ± 45 degrees from top to bottom, and (c) ± 30 degrees in a horizontal self-spin.

Then, for each background scene 200 images of each kind of hand postures were randomly selected as testing samples, and therefore in total there are 1600 testing images for each kind of hand postures. In Table 2, the corresponding recognition results are listed with respect to different hand postures and background scenes. Also, as shown in Fig.9, the red block contains an entire arm which obviously is not suitable for recognition, the green block is the hand posture region after forearm being removed which is much easy to be recognized, and the recognition results are shown in the upper-right part of each image. According to Table 2, the recognition rate reaches to 95% for the plain-background taken images and an average 89.1% for all the eight complex-background taken images. Table 3 is the corresponding confusion matrix which shows the correlation between input classes and recognized classes, and from this table it reveals that scissor and paper are much easier to be misrecognized compared with other pairs of hand gestures. There are two possible reasons for this phenomenon. First, when the angle of a hand gesture varies too large, its image would result in considerable fingers overlapping, and thus lead to a recognition failure. Second, because of the uneven illumination distribution, a binary hand image could become fractured and consequently leads to misrecognition.

TABLE 2. Results of recognizing multi-angle hand postures

Types of background	Hand postures	Recognition rate	average
Singles background	Scissor	95.5%	95%
	Stone	96%	
	Paper	93.5%	
Complex background (1)	Scissor	87.5%	87.3%
	Stone	93%	
	Paper	81.5%	
Complex background (2)	Scissor	91.5%	91.8%
	Stone	95.5%	
	Paper	89.5%	
Complex background (3)	Scissor	92.5%	91%
	Stone	94.5%	
	Paper	86%	
Complex background (4)	Scissor	80%	81.7%
	Stone	88.5%	
	Paper	76.5%	
Complex background (5)	Scissor	90.5%	89.2%
	Stone	92.5%	
	Paper	84.5%	
Complex background (6)	Scissor	82%	84.8%
	Stone	88.5%	
	Paper	84%	
Complex background (7)	Scissor	92%	91.5%
	Stone	95%	
	Paper	87.5%	
Average of recognition rates	Scissor	88.9%	89.1%
	Stone	92.9%	
	Paper	85.3%	

TABLE 3. Confusion matrix of recognizing multi-angle hand postures

Input \ Recognition	Scissor	Stone	Paper
Scissor	88.9%	3.3%	7.8%
Stone	5.5%	92.9%	1.6%
Paper	11%	3.7%	85.3%

We also performed an experiment by using the same 450 images (150 for each kind of hand gestures) to train a SVM-based hand gesture classifier and the same 4800 testing images to test the trained classifier. The goal of this experiment is simply to compare the recognition abilities of both HTM and SVM, therefore their hand gesture preprocessing procedure on the training and testing images is the same. The detected hand gesture block images then are inputted to train and test the classifier. Table 4 lists the recognition result with an average recognition rate of 79.9% which is worse than that of HTM with 89.1%. This experiment reveals clearly that HTM has much superior recognition ability than SVM. This superiority mostly comes from HTM has the ability to extract the spatially and temporally invariant feature from images.

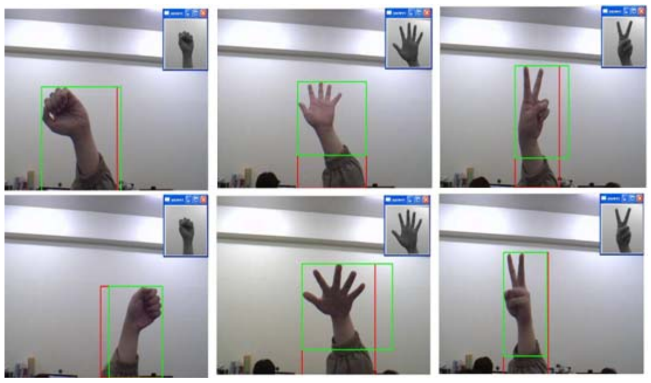
Fig.9. Examples of different hand posture recognition

TABLE 4. Result of recognition with algorithm of SVM

	Scissor	Stone	Paper
Training images	150	150	150
Testing images	1600	1600	1600
Recognition rate	78.3%	86.1%	75.3%
Average	79.9%		

VI. Conclusion

This paper proposes an effective multi-angle hand posture recognition method. A useful hand posture region locating method is designed which involves skin color detection, foreground segmentation, edge detection, and forearm removal. As a result, the image of hand posture region is derived correctly and stably. Next, use HTM algorithm to process and generalize the continuous changing images to form invariant feature and to overcome the impact of angle changing. Finally, construct a high-efficient hand posture recognizer. Under the same test of multi-angle images, the proposed method performs better than both Adaboost and SVM algorithm.

Although the proposed method has already improved the impact of angle changing on hand postures recognition, the hand posture pre-processing and forearm segmentation still has room to improve, especially for illumination variation. The severe variation in illumination will cause the change of the skin color, therefore, it can’t be detected correctly. Besides, the present study only can detect postures. If there are other objects with the skin color(e.g. face), the result of recognition will be inappropriate because the proposed method can’t eliminate them. In the future, we are going to indemnify the light measure in order to stabilize the derived image on the skin color and further analyze the features of wrists in order to figure out a more accurate palm image for recognize. In addition, we will investigate and develop an efficient hand posture detector to make the system be utilized in every environment.

References

[1] F. Echtler et al., 2010, “An LED-Based Multitouch Sensor for LCD Screens,” in *Proceedings of the ACM Tangible and Embedded Interaction Conference*, pp. 227-230.

[2] L. Xie and Z. Liu, Apr. 2007, “Realistic Mouth-Synching for Speech-Driven Talking Face Using Articulatory Modelling,” in *Proceedings of the IEEE Transactions on Multimedia*, Vol. 9, No. 3, pp.500-510.

[3] R. Cucchiara et al., Oct. 2003, “Detecting Moving Objects, Ghosts, and Shadows in Video Streams,” in *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 10, pp. 1337-1342.

[4] Alan M. McIvor, Nov. 2000, “Background Subtraction Techniques,” in *Proceedings of the Image and Vision Computing*, Auckland, New Zealand.

[5] S. Murali and R. Girisha, 2009, “Segmentation of Motion Objects from Surveillance Video Sequences using Temporal Differencing Combined with Multiple Correlation,” in *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 472-477.

[6] S. Sun et al., 2000, “Motion Estimation Based on Optical Flow with Adaptive Gradients,” in *Proceedings of the IEEE International Conference on Image Processing*, Vol. 1, pp. 852-855.

[7] K. Kim et al., Jun. 2005, “Real-Time Foreground-Foreground segmentation using Codebook Model,” in *Proceedings of the Real-Time Imaging*, Vol. 11, No. 3, pp. 172-185.

[8] D. Chai and A. Bouzerdoum, 2000, “A Bayesian Approach to Skin Color Classification inYCbCr Color Space,” in *Proceedings of the IEEE Region Ten Conference*, Kuala Lumpur, Malaysia, Vol.2 , pp 421-424.

[9] J. Hawkins and D. George, 2006, “Hierarchical Temporal Memory Concepts, Theory, and Terminology,” *Numenta*, <http://www.numenta.com/htm-overview/education/Numenta_HTM_Concepts.pdf>.

[10] J. Hawkins and S. Balkeslee, 2004, “On Intelligence”, New York : Owl Books.

[11] Christopher N. Vutsinas et al., Apr. 2008, “A Neocortex Model Implementation on Reconfigurable Logic with Streaming Memory,” in *Proceedings of the IEEE International Symposium Parallel and Distributed Processing*, pp. 1-8.

[12] Stephen C. Johnson, Sept. 1967 , “Hierarchical Clustering Schemes,” in *Proceedings of the Springer journal on Pyschometrika* , Vol. 32, No. 3, pp. 241-254.

[13] Yea-Shuan Huang and Yun-Jiun Wang, “Multi-Angle Hand Posture Recognition Based on Hierarchical Temporal Memory,” in *Proceedings of The International MultiConference of Engineers and Computer Scientists* 2013, 13-15 March, 2013, Hong Kong, pp. 70-75.

Yea-Shuan Huang was born in Taiwan in 1960. He received the Bachelor and Master degrees from Chung Kung University n 1982 and 1984, respectively. In 1995, he received the Ph.D. degree from the Computer Science Department of Concordia University. His main research areas involve Image Processing, Pattern Recognition, Computer Vision and Video Surveillance.

From 1984 to 2006, he worked as a senior researcher in Industrial Technology and Research Technology (ITRI), and he developed the optical character recognition (OCR) systems for both printed Chinese characters and hand written Chinese characters. Since 2006, he joined the Computer Science and Information Engineering Department of Chung Hua University, and becomes an associate professor. He has obtained about 40 patents in the Computer Vision area and published more than 100 papers.