Proc. VIIth Digital Image Computing: Techniques and Applications, Sun C., Talbot H., Ourselin S. and Adriaansen T. (Eds.), 10-12 Dec. 2003, Sydney

Gesture Classification Using Hidden Markov Models and Viterbi Path Counting

Nianjun Liu, Brian C. Lovell

Intelligent Real-Time Imaging and Sensing (IRIS) Group
School of Information Technology and Electrical Engineering
The University of Queensland, Brisbane, Australia 4072
{nianjunl,lovell}@itee.uq.edu.au

Abstract. Human-Machine interfaces play a role of growing importance as computer technology continues to evolve. Motivated by the desire to provide users with an intuitive gesture input system, our work presented in this paper describes a Hidden Markov Model (HMM) based framework for hand gesture detection and recognition. The gesture is modeled as a hidden Markov model. The observation sequence used to characterize the states of the HMM are obtained from the features extracted from the segmented hand image by Vector Quantization. In the recognition system, we try several different HMM models and training algorithms to find the algorithms with high recognition rate and low computational complexity.

1 Introduction

A large number of potential applications in advanced gesture interfaces for Human Computer Interaction (HCI) have been designed in last decade[1,7]. The goal of gesture interpretation is to push the envelope of advanced human-machine communication, that is, to bring human-machine interaction closer to human-human interaction. Just as the mouse has not replaced the keyboard, a gesture recognition system is not intended to replace the mouse. Instead, gesture recognition will enhance existing applications and make possible new applications such as sign language translation, which naturally benefit from the addition of gesture recognition, and can be used to help physically impaired people communicate more easily with each other.
Previous attempts [2,11] to develop a hand gesture recognition system include geometric feature-based methods, template-based methods, and more recently active contour and active statistical models. Given the success of HMMs in speech recognition and character recognition, HMM has begun to be applied in spatio-temporal pattern recognition and computer vision [4,7,8] .
In the recognition system, the hand gesture video is first captured by a digital camera attached to a standard Intel Pentium III PC computer. Each video is composed of 25 frames. Skin colour segmentation based on HSV colour space is applied. Pre-processing (Morphological) operations are used to smooth the image and remove the

noise. An improved Camshift algorithm has achieved a good performance in tracking the hand. After getting the binary image of the hand, we apply moment algorithm to calculate the features of the hand, normalize these features, and input them to Vector Quantizer. The pre-trained Vector Quantizater outputs the Codeword sequence with the length of 25, each frame corresponding to one codeword. The Codeword sequence representing a discrete observation sequence is input to a Hidden Markov Model. We trained 8 different HMMs based on the model type, model structure and number of states. Each model corresponds to one of six predefined gestures. The experimental results are shown in the latter section. A flow chart of the system is shown in Figure 1. Once the gesture is recognized, our application software plays a song from a list according to the recognized gesture. As the number of recognizable gestures increases, the software can play more songs or implement more complex applications.

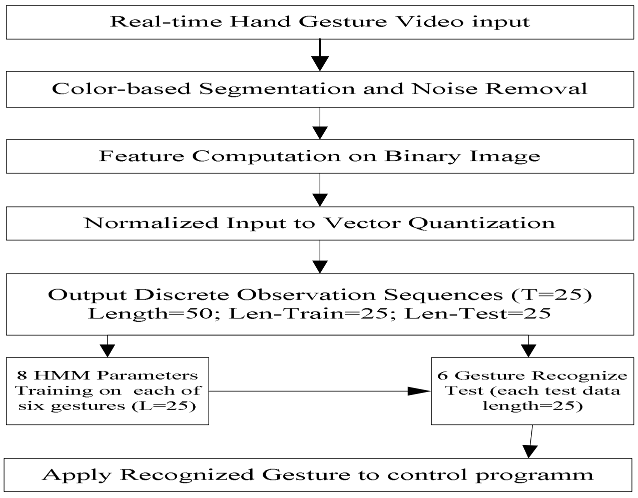
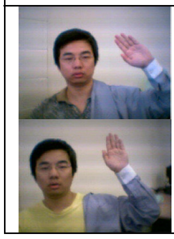

Real-time Hand Gesture Video input
Color-based Segmentation and Noise Removal
Feature Computation on Binary Image
Normalized Input to Vector Quantization
Output Discrete Observation Sequences (T=25)
Length=50; Len-Train=25; Len-Test=25
8 HMM Parameters Training on each of six gestures (L=25)
6 Gesture Recognize Test (each test data length=25)
Apply Recognized Gesture to control programm

Figure 1 System Flowing Chart

2 Three Poses and Six Gestures

For use by our system, we have designed 3 poses and 6 gestures [5], shown below:

Pose Pictures	Description
	**Pose 1** is the open palm, what- ever the shaking (tilted) angle.

Video	Gesture Description
	Gesture 1 Shaking the palm from left to right and return, it including pose1.
	Gesture 2 Quickly rotate the palm from front to vertical and repeat, including pose1 and pose 2

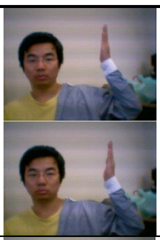
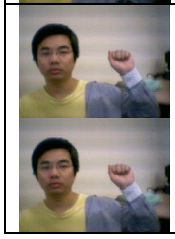
	**Pose 2** is Hand edge (Vertical Palm likes a knife)		**Gesture 3** Repeat opening the palm and clenching it, including pose1 and pose 3
			Gesture 4 From the front palm to vertical one and clenching the fist, repeating, from pose1 to pose 2 to pose 3, Looping.
	**Pose 3** is Clench-ing the fist		**Gesture 5** Vertical Palm to fist, looping. From pose 2 to pose 3 ,loop
			Gesture 6 Shaking the palm for a while and then clenching the fist to the end. Shaking Pose1 and then pose 3, loop

Table 1 Three Poses and Six gestures

3 Tracking and Segmenting of the Hand

Skin colour-based segmentation is a reliable method for segmenting the hand in an unrestricted environment. The first step is to convert the image from RGB colour space to HSV colour space. The H value of each pixel is analysed and determined to be either skin or non-skin by a predefined lookup table. This is done after the colour image has been mapped into a binary image of ones and zeros representing skin and non-skin regions. Pre-processing refinement then follows, to remove the noise and smooth the binary image. During the colour segmentation, it is common to return values that are close to skin but are non-skin, or other skin-like coloured regions that is not part of the body. These noisy erroneous values are generally isolated pixels or group of pixels that are dramatically smaller than the connected hand region in the binary image. We apply the morphological operation, using opening (eroding first and then dilating) and then closing (dilating first and then eroding) in order to remove the spurious noises which are generally much smaller than the hand region itself, without changing the shape of main object (such as hand and face). After this, Matlab®'s imfill function is used to fill the holes inside the hand. The next step is to automatically locate the hand region and remove all other "skin-like" regions. Finally, we select the region of maximum size and remove all other parts, as the hands are always the biggest connected regions in our search window.

An improved Continuously Adaptive Mean-shift algorithm (Camshift) [9] is applied to determine a region of interest (ROI) surrounding the hand in each successive frame, in order to track the hand and reduce the region of calculation. The Camshift Algorithm adjusts the search window size in the course of its operation. For each video frame, the colour probability distribution image is analysed to determine the centre and size of the ROI. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image. The process is then repeated for continuous tracking.

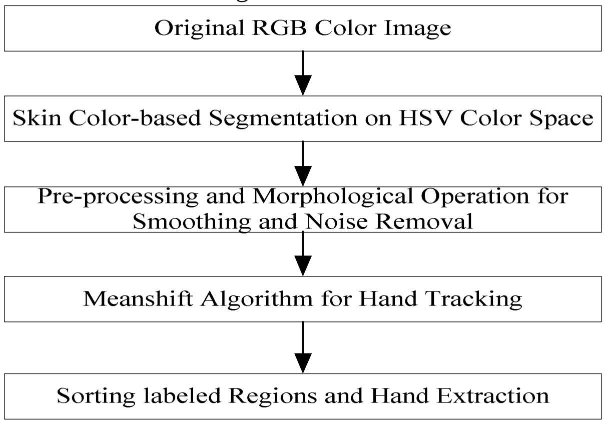

Original RGB Color Image
Skin Color-based Segmentation on HSV Color Space
Pre-processing and Morphological Operation for Smoothing and Noise Removal
Meanshift Algorithm for Hand Tracking
Sorting labeled Regions and Hand Extraction

Figure 2 Hand Segmentation

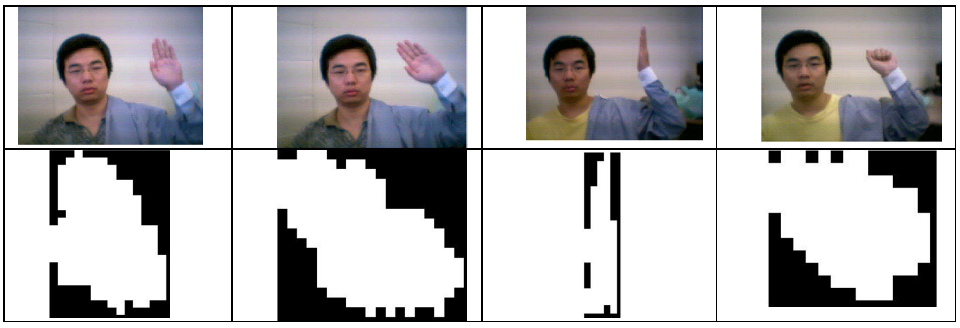


Figure 3 Skin Color-based Segmenting Output.

4 Features Computation and Blob Ellipse Model

After getting the binary image, a moments algorithm is applied to calculate the features of the hand. The centroid of the hand depends on the zero^th and first moments of the binary image. In the following equations, I(x,y) is the pixel value at position (x,y) in the image, and x and y range over the size of binary image. The following Equations illustrate how the rotational angle is computed using the second moments.

$$\theta = \frac{\arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)}\right)}{2} \qquad \textbf{(1)}$$

The first two eigenvalues (major length and width) of the hand block may be expressed in the following closed form:

$$a = \frac{M_{20}}{M_{00}} - x_c^2 , b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right), c = \frac{M_{02}}{M_{00}} - y_c^2 \qquad \textbf{(2)}$$

Then the length *l* and width *w* from the centroid are

$$l = \sqrt{\frac{(a+c) + \sqrt{\left(b^2 + (a-c)^2\right)}}{2}} ; w = \sqrt{\frac{(a+c) - \sqrt{\left(b^2 + (a-c)^2\right)}}{2}} \qquad \textbf{(3)}$$

Based on the features extracted from the image, we design the hand 'blob'--an ellipse model to represent the hand. Figure 4 is the ellipse model and matching the hand.

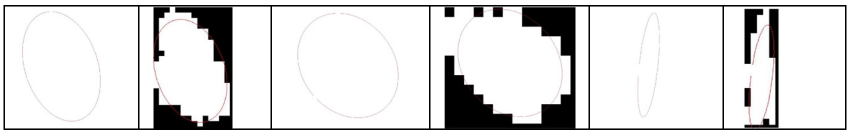


Figure 4 Hand Ellipse model and image

5 Vector Quantization for Discrete Output

Vector Quantization is widely used in applications such as image and voice compression, voice recognition, and statistical pattern recognition [3]. In our system, it is a good way to determine the cluster of frames to which a given frame is most similar. Because each gesture is composed of three basic poses, we allocate the gesture frames to some predefined clusters. For pose 2 and pose 3, they are almost static and uniform, frames for pose2 and frames for pose 3 could be treated as one cluster each. For pose 1, the shaking palm, the rotation angle is different. In the experiment, we tried 1 cluster, 2 clusters and 3 clusters for pose 1, and checked the cluster output and distribution. We found that 2 clusters for pose1 gives the best results. Adding the other two poses' clusters gives a total of 4 clusters in the Vector Quantizer.

The feature data extracted from the binary are in different scales, and so we begin by normalizing the feature data in the range [0, 1], and then output them to Vector Quantizer. The data set for code vector calculation includes six gestures, each of which has 50 videos, and each video has 25 frames (total is 6x50x25). We apply the well known K-means Vector Quantization algorithm [3] to calculate the code vector coordinates in 4 dimensions. After getting the code vector position, when a normalized feature of a new frame is input, we calculate the Euclidean distance between the input vector and each code vector, choosing the code vector with the minimum distance as the output. Each gesture video includes 25 frames, and each frame corresponds to one codeword, so the output sequence length is 25, which acts as the discrete observation sequence to be input to the Hidden Markov model.

Figure 5 depicts the Vector Quantization process and table 2 shows the input and output data of the Vector Quantization.

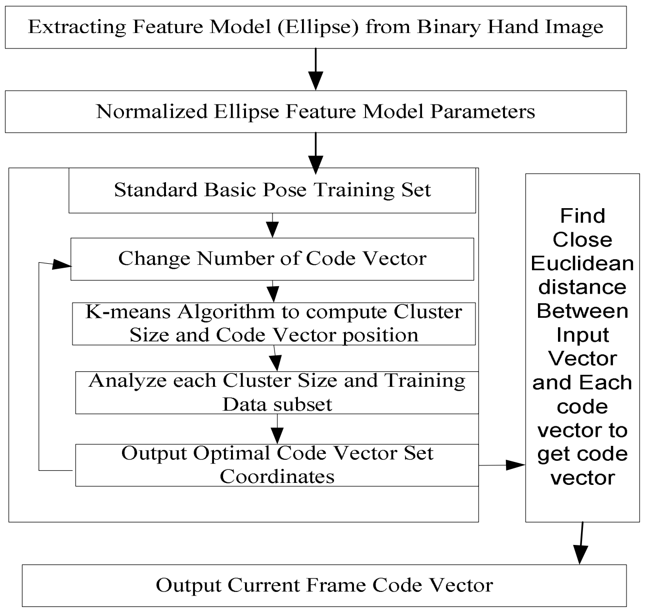

Extracting Feature Model (Ellipse) from Binary Hand Image
Normalized Ellipse Feature Model Parameters
Standard Basic Pose Training Set
Change Number of Code Vector
K-means Algorithm to compute Cluster Size and Code Vector position
Analyze each Cluster Size and Training Data subset
Output Optimal Code Vector Set Coordinates
Find Close Euclidean distance Between Input Vector and Each code vector to get code vector
Output Current Frame Code Vector

Figure 5 Vector Quantization Method

Table 2. Hand features Extraction and Vector Quantization In and Output

	Length	width	Angle	ratio(L/W)
1	11.389	6.267	-18.336	1.817
2	9.780	6.301	-18.649	1.552
	~~			
25	9.972	5.803	-34.558	1.719
	Normalized Data			
1	0.766	0.796	0.461	0.149
	~~			
25	0.551	0.707	0.180	0.123
Output VecterCode Sequence(Length=25)				
From 1 to 25	3 3 3 4 3 4 3 3 4 3 4 3 4 4 4 3 4 3 4 3 4 3 4 3 4 3			

6 Hidden Markov Model in Gesture Recognition

Hidden Markov Models (HMMs) are a set of statistical models used to characterize the statistical properties of a signal [10].We use an HMM to define a gesture in the recognition system. We define a gesture to be a sequence of static hand poses, and the static pose information contained within a frame is the observation symbol. Each gesture corresponds to one HMM. We designed six gestures and eight different types of Models of various model structures, learning algorithms, and state numbers.

6.1 Baum-Welch algorithm

HMM parameters $\lambda = (A, B, \pi)$ can be improved to $\tilde{\lambda} = (\tilde{A}, \tilde{B}, \tilde{\pi})$ using the follow re-estimation formulas for multiple observation sequences, in our system, each gesture has multiple observation Sequences (Length=25). Now consider the case where k observation sequences are known to be generated by the same HMM and the objective is to determine the HMM parameters that yield high probability of generating all K observed sequences. Rabiner and Juang [10] proposed such a multi-sequence training algorithm using the K observation sequences at each stage of the Baum-Welch re-estimation to iteratively update a single HMM parameter set. The re-estimation formulas for this type of iterative method are as follows:

$$\tilde{a}_{ij} = \frac{\Sigma_k W_k \Sigma_{t=1}^{T_k} \alpha_i^k a_{ij} b_j (O_{t+1}^{(k)}) \beta_{t+1}^{(k)}(j)}{\Sigma_k W_k \Sigma_{t=1}^{T_k} \alpha_i^k (i) \beta_t^{(k)}(i)} \qquad \tilde{b}_{ij} = \frac{\Sigma_k W_k \Sigma_{O_t(k)=v_j} \alpha_t^k (i) \beta_t^{(k)}(i)}{\Sigma_k W_k \Sigma_{t=1}^{T_k} \alpha_i^k (i) \beta_t^{(k)}(i)} \tag{4}$$

Where $W_k = 1/P_k$, $k \in [1\ K]$ is the inverse of the probability of the current model estimate generating training sequence k, evaluated using the forward algorithm.

6.2 Viterbi Path Counting Algorithm

The Viterbi-Path Counting algorithm method (VPC) is a simplification of the Best First method of Stolke and Omohundro [12] that has exhibited very good performance on simulated HMM data. VPC based on the Viterbi optimal path method[10]. For a vector of training sequences, $O_i(t)$, for time $t \in [1\ T]$ (T is the length of the sequence and ***i*** is the sequence index) and specified HMM model structure we perform the following training procedure:

1. Create a random HMM with the specified structure (eg. Full connection, Left-Right), with parameters A, B, π .
2. Create counter matrices A_c, B_c, π_c by setting their elements to be 0 if the corresponding element in A, B, π is zero, otherwise set them to 1.
3. Repeat steps 4 to 8 as many times as desired to achieve good training
4. Repeat steps 5-8 over all training sequences $O_{i.}$
5. Set the current estimate A', B', π' of the model parameters to be normalized counter matrices.
6. Use the Viterbi algorithm with current model parameter estimates A', B', π' to estimate the most likely path p(t) for the current observation sequence O_i
7. Add 1 to $\pi_c(p(1))$ to increment the counter for π at the first node p(1) in the most likely path.
8. Loop through the observation sequence, adding 1 to the values in A_c for every transition from p(t), and adding 1 to B_c for every emission in p(t).

9. Set the final estimate A_n, B_n, π_n of the model parameters to be the normalized counter matrices.

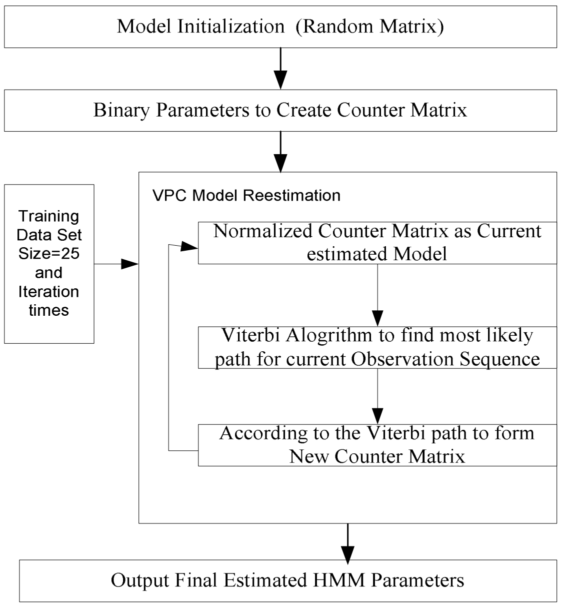

Model Initialization (Random Matrix)
Binary Parameters to Create Counter Matrix
VPC Model Reestimation
Training Data Set Size=25 and Iteration times
Normalized Counter Matrix as Current estimated Model
Viterbi Alogrithm to find most likely path for current Observation Sequence
According to the Viterbi path to form New Counter Matrix
Output Final Estimated HMM Parameters

Figure 6 HMM training Method TWO--VPC

This new method simplifies the training process because it simply counts states and transitions, and estimates the model parameters from those total counts. The hard work of finding the best is hidden with the call to the Viterbi algorithm. In addition, the dependence upon the initial model is minimized because the model starts out a maximum entropy state, and the random seed only sets down the very first choice. In contrast, Rabiner's method relies on complex re-estimation formulae which make it difficult to find globally optimal solutions and there is a high degree of dependence upon the initial seed value. Note the new method is not a standard EM-type training algorithm because the update rule depends upon local counts of the number of modifications of each branch of the model. In contrast, EM algorithm operates upon the probabilities from the previous iteration alone.

7 Experiment Output and Discussion:

7.1 Analysis and Discussion

In the recognition phase, a set of 300 gesture videos are used to determine the recognition performances of the system. For each gesture we use 50 videos (25 for the training set, 25 for testing). We have eight kinds of HMM models, with various training methods, model structures, and numbers of states. After extracting the observation sequence from the gesture video, the probability of the observation sequence on each HMM is calculated. The model with highest likelihood is recognized as the corresponding gesture. $P(O|\lambda_k)) = \max_n (O|\lambda_n)$

Table 3 Recognition results, BW=Baum Welch; VPC=Viterbi Path Counting; FC=Model Structure Full connection; LR=Left and Right structure; 4S=Number of states is 4; 5S=Number of States is 5; Six Gestures(1—6)

Correct Ratio(%)	1	2	3	4	5	6
BW-FC-4S	100	92	88	96	92	92
BW-FC-5S	100	96	88	100	96	96
BW-LR-4S	100	92	100	100	92	100
BW-LR-5S	100	96	100	100	100	100
VPC-FC-4S	100	100	100	96	96	100
VPC-FC-5S	100	100	100	96	96	100
VPC-LR-4S	100	92	88	100	96	100
VPC-LR-5S	100	96	88	100	96	96

7.2 Analysis and Discussion

From the table of experimental results, we see the following:

1) Baum Welch is the traditional algorithm for HMM learning, while Viterbi path accounting is the new method (VPC). VPC performed steadily in comparison with Baum Welch, and is demonstrated to be a reliable method.
2) Model Structure variation. Structure doesn't greatly affect the recognition ratio. For Baum Welch, Left-Right is a bit better than full connection, while in Viterbi path accounting, Full connection has a slightly higher correct ratio.
3) The number of states doesn't affect the correct ratio very much. The effect on Baum Welch is greater than on Viterbi path accounting.

At present, we are testing 26 other gestures, and the early results look very encouraging, with an overall correct rate of approximately 90 percent. Later, we will try to use other discrete symbols and coupled Hidden Markov Models, as well as 2D VPC training algorithms in an attempt to improve the performance.

From the results of experiments on the six gestures and its extended 26 gestures[13], we find Left-Right models perform similarly or even better than Full-Connection models in these trials. We believe the reason is that they make it easier for the training algorithm to extract the most information out of the data. Full connection models tend to lose structure information because there is too much freedom for the algorithm to manage, whilst restricting the model to Left-Right form means that there is a greater chance that the training algorithm will be able to match the data to the model in a useful way. VPC is better-suited to more restrictive models, but in those cases, it produces higher-quality models because there is less emphasis on structure learning and more emphasis on gaining accurate estimates of statistics. This suggests that separating the problems of structure learning and statistics collection may make it easier to obtain better-quality models.

8 Conclusion

This paper presents a hand gesture detection and recognition system using an HMM approach. An efficient method for extracting the observation sequence using the feature model and Vector Quantization has been presented. Compared to the classic template-based methods, the HMM-based approach offers a more flexible framework for recognition. We compared two training algorithm, the traditional Baum-Welch, and the newer Viterbi Path Counting method, varying the model structure and number of states. Both of them achieved a reasonable performance, and so we have demonstrated that the Viterbi Path Counting is a reliable algorithm for training HMMs.

References

1. V. Pavlovic, Visual Interpretation of Hand Gestures for Human-Computer Interaction, IEEE Transactions on Pattern Analysis and Machine Intelligence July 1997 (Vol. 19, No. 7) pp. 677-695
2. Hyeon-Kyu Lee and Jin H. Kim, An HMM-Based Threshold Model Approach for Gesture Recognition IEEE Transactions on Pattern Analysis and Machine Intelligence October 1999 (Vol. 21, No. 10). pp. 961-973
3. A. Gersho and R.M.Gray. Vector Quantization and Signal Compression, Kluwer Academic Press,1991, ISBN 0-7923-9181-0.16
4. J. Yang, Y. Xu, and C.S. Chen, "Gesture Interface: Modeling and Learning," IEEE International Conference on Robotics and Automation, Vol. 2, 1994, pp. 1747-1752.
5. Junji Yamato, Jun Ohya, Kenichiro Ishii: "Recognizing Human Action in Time-Sequential Images using Hidden Markov Model", In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 1992, pp 379--385
6. Davis, Richard I. A. and Lovell, Brian C. and Caelli, Terry Improved Estimation of Hidden Markov Model Parameters from Multiple Observation Sequences. In Proceedings International Conference on Pattern Recognition, August 11-14,2002. II, pages 168-171, Quebec City, Canada.
7. T.Starner and A.Pentland, Real-time American Sign Language Recognition", IEEE Trans, On Pattern Analysis and Machine Intelligence, Vol 20.pp 1371-1375, Dec.1998.
8. Ara V. Nefian and Monson H. Hayes III. "An embedded HMM based approach for face detection and recognition", IEEE International Conference on Acoustics, Speech and Signal Processing, March 1999, p.3553-3556, vol VI.
9. Intel Open Source Computer Vision Library Reference Manual. Aug,2003, Available on line at: www.intel.com/research/mrl/research/opencv
10. LR Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proc. of the IEEE, Vol.77, No.2, pp.257--286, 1989.
11. D.Heckenberg and B. C. Lovell, "MIME: A Gesture-Driven Computer Interface", Proceedings of Visual Communications and Image Processing, SPIE, V 4067, pp 261-268, Perth 20-23 June, 2000
12. A. Stolcke and S. Omohundro. Hidden Markov Model induction by Bayesian model merging, Advances in Neural Information Processing Systems, pp. 11-18, Morgan Kaufmann, San Mateo, United States of America, CA1993.
13. N. Liu, B.C. Lovell, P.J. Kootsookos, "Evaluation of HMM training Algorithm for Letter Hand Gesture Recognition" .IEEE International Symposium on Signal Processing and Information Technology. (ISSPIT 2003). Paper Accepted.