Bharatanatyam Hand Gesture Recognition Using Polygon Representation

Sriparna Saha[1], Amit Konar[2], Deblina Gupta[3], Anasuya Ray[4],
Abhinaba Sarkar[5], Pritha Chatterjee[6], Ramadoss Janarthanan[7]
[1, 2] *Electronics & Telecommunication Engineering Dept.,* Jadavpur University, Kolkata, India
[3-6] *Computer Science & Engineering Dept*., Institute of Engineering and Management, Kolkata, India
[7] *Computer Science & Engineering Dept*, TJS Engineering College, India
sahasriparna@gmail.com, akonar@etce.jdvu.ac.in,
{gdeblina, ray.mishti, abhinav210990, pritha852}@gmail.com, srmjana_73@yahoo.com

***Abstract*—For automatic hand gesture recognition of 'Bharatanatyam' dance, a 4-stage system has been designed. In the first stage, texture based segmentation is done to detect the hand of the dancer from the background. Thus boundary of the hand is extracted. In the next stage, boundary of the hand is approximated using straight lines. Now at the third stage, each straight line is represented by the sides of a decagon by comparing the slopes and thus a chain code is obtained from this. In the last level, matching of an unknown chain code is done with the chain codes from the database with an accuracy rate of 89.3%. This simple yet effective code is very useful for e-learning of 'Bharatanatyam' dance. It is a fast and effective way to spread the dance form world-wide.**

Keywords—boundary extraction; chain code; polygon representation; straight line approximation; texture based segmentation

I. INTRODUCTION

'Bharatanatyam' is one of the popular dances and widely performed in world-wide. It originated from Tamil Nadu, a south Indian state. This traditional dance form signifies elegance, purity and grace. An essential element of 'Bharatanatyam' is its 'Hastas' (hand 'mudras' or movements). 'Hastas' signifies different hand symbols that a dancer use to communicate. 'Hastas' can be of two types 'Asamyukta Hastas' (single hand) and 'Samyukta Hastas' (double hand). 'Asamyukta Hastas' consists of 28 mudras and 'Samyukta Hastas' consists of 24 mudras.

Like normal hand gestures [1], 'Bharatanatyam' hand gestures are expressive and meaningful. Interaction between dancers is achieved by the hand gestures. 'Pathakam' hand gesture stands for joy, whereas 'Mrigasirsham' is used to denote fear. Various meanings are indicated by different hand gestures.

With the increasing popularity of internet, it is very much useful to identify hand gestures of the 'Bharatanatyam' dance using computers [2] [3] [4]. The objective of this paper is to develop a simple system for promoting e-learning of 'Bharatanatyam' dance [5] [6]. Thus the 'Bharatanatyam' learning becomes an interactive program between the dancer and the computer. Also e-learning is a fast and cost-effective way to spread 'Bharatanatyam' dance world-wide. Flexibility and accessibility are the other two important advantages of e-learning.

The authors in [7] have designed a system for recognizing hand gestures of 'Bharatanatyam' Dance using orientation filter [8] [9] and connectivity graph for 'Asamyukta Hastas'. There are two major problems for which the algorithm is not applied on real time images. First of all, eight hand gestures are misclassified because connectivity graphs of nearly similar hand gestures are equal. Second problem is that, typically the tips of the fingers are coated with red colors when 'Bharatanatyam' dance is performed. But in [7], skin color based segmentation is used to extract boundary of the hands. After this segmentation the tips of the fingers are removed. So boundaries of the fingers are lost by skin color based segmentation. To deal with these two problems, an algorithm is proposed in this paper. First problem is solved as the boundary of hand gesture is considered instead of the connectivity graph. To deal with the second problem, texture based segmentation is considered instead of skin color based segmentation.

In this proposed algorithm, we have designed a four level system. The main difficulty of recognising any hand gesture is to separate out the hand from the background [10] [11]. Thus at the first level, the hand is segmented out from the background using hand texture [12] [13] [14]. Then the boundary of the hand is detected using Sobel edge detection technique [15] [16] [17] [18]. At the next level, slopes of the boundaries are calculated and based on this straight line approximation [19] [20] is done. Now, the straight lines are approximated using 10-sided regular polygon (also known as decagon) [21]. Zero padding of the chain codes is done to obtain equal length chain codes for all the 28 hand gestures. At the time of matching same length chain codes are advantageous than different length chain codes [22] . At the final stage, unknown hand gestures are matched with the hand gestures from the database using the chain codes by similarity function. This algorithm is implemented to deal with translation invariant hand gestures. The average timing complexity of the proposed algorithm is 3.312 sec in an Intel

978-1-4799-2044-0/14/$31.00©2014IEEE

Pentium Dual Core processor running Matlab R011b for each static image. We have achieved 89.3% overall accuracy.

In this paper, we provide a method for hand gesture recognition of 'Bharatanatyam' dance for e-learning purpose. Section II includes the preliminary tools of the proposed algorithm with block diagram. Section III demonstrates the experimental results. Finally, section IV concludes with some idea of future work.

II. PROPOSED ALGORITHM

The steps of the proposed algorithm are explained in here in details.

A. Boundary Extraction

One of the important aspects of gesture recognition is to identify the hand of the dancer from the background. In [7], the authors are using skin-color based segmentation for separating the hand from the background. But it has two drawbacks already stated. So texture base segmentation is applied to eliminate unnecessary information about the background.

The input RGB image is first converted to grey-scale image. Here the values are in the range of 0 (for black) and 1 (for white). Then we apply texture based segmentation in this image and converted to binary image (BW) by calculating entropy

For texture based segmentation [12] [13] [14] entropy calculation is used [23]. This stage is the first pre-processing stage for hand gesture recognition. It requires finite neighbourhood for processing. First the gray-level co-occurrence matrix is obtained using a displacement vector d. Here it is specified as (1,1), which means one pixel to right and one pixel below.

$$d = (dx, dy) = (1,1) \tag{1}$$

P[i,j] is not necessarily symmetric. Each element of P[i,j] is divided by total no of pixel pairs to obtain normalize value. This normalized P[i,j] is known as probability mass function. Entropy is calculated based on P[i,j].

$$Entropy = -\sum_i \sum_j P[i,j] \log p[i,j] \tag{2}$$

Entropy values which are greater than a threshold value 0.7 are selected. The output image has 1 value for all the pixels which have greater value than 0.7 or 0 otherwise. This threshold value is taken empirically.

$$BW = \begin{cases} 1, & if \quad pixel > 0.7 \\ 0, & otherwise \end{cases} \tag{3}$$

Now the edges are smoothed and open holes are closed. Here the neighbourhood value is taken same as the value used for entropy calculation. Then the boundary of the hand is detected using Sobel edge detection technique [15] [16] [17] [18]. For Sobel edge detection technique, a 3×3 neighbourhood is taken into account as explained in Fig. 1.

a_0	a_1	a_2
a_7	[i,j]	a_3
a_6	a_5	a_4

Fig. 1. Labelling of neighbourhood pixels

Partial derivatives are calculated by the following two equations

$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \tag{4}$$

$$s_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \tag{5}$$

where c is a constant.
Magnitude, M is measured by (6)

$$M = \sqrt{s_x^2 + s_y^2} \tag{6}$$

Smoothing of the edges is done to obtain better result in the next stage. Here smoothing is done by morphological shrink [24] [25] operation to one value. The boundary of the hand gesture in some places are thicken, so to remove the unwanted pixels from the boundary and produce a single pixel bounded line, morphological shrink operation is implemented.

B. Straight Line Approximation

The boundary of the hand gesture is a curved line, but for generation of chain code by polygon representation, we need a straight line approximated boundary [19] [20]. For this reason, the two end points of the image are extracted. Then we start from one end point (which is at the left side from the other end point) and its adjoining 8 pixels are examined. This continues until the other end point is reached.

The entire boundary is represented using connected straight lines. This is done by first calculating the slopes of adjacent pixels and then detecting the point where there is a change of slope. Then these points are connected using straight lines. Thus we get a straight line representation of the image where the shape of the boundary is kept intact.

Our goal is to represent the boundary using minimum possible straight lines so as to reduce the noise by keeping the shape of the image undistorted. To smooth the image further, at first those points which are very near to each other, i.e., those points which are joined by straight lines having very small length are discarded, i.e., straight lines having length less than a particular threshold value are discarded. This value is chosen such that it discards maximum possible straight lines without affecting the shape of the boundary. In this case the threshold value is empirically chosen as 15.

Next the slope of each straight line is determined. Starting from the first straight line, its slope is compared to its adjacent connected straight lines. If the difference between the slopes is within a particular range then the latter is discarded and the next straight line is compared to the first one. Here also, the range is chosen such that it discards the maximum possible number of straight lines keeping the shape of the boundary more or less intact. In this case the range is taken to be -40^o to $+40^o$, which is also determined experimentally.

C. Polygon Representation and Chain Code Generation

Arbitrary geometric structures can be encoded by a number of ways, each having its own particular advantages and disadvantages [22]. One such method is to represent the straight lines by the edges of a regular polygon. Here we are using regular decagon to generate chain code. More will be the side of the regular polygon; complexity of the code will be more, thus increasing timing complexity. But with lesser no of sides of regular polygon, information about different slopes will be lost. The straight lines which have a large difference in slope will be approximated by same side of polygon. Thus we are taking 10-sided polygon for representing the straight line approximated boundary by a chain code.

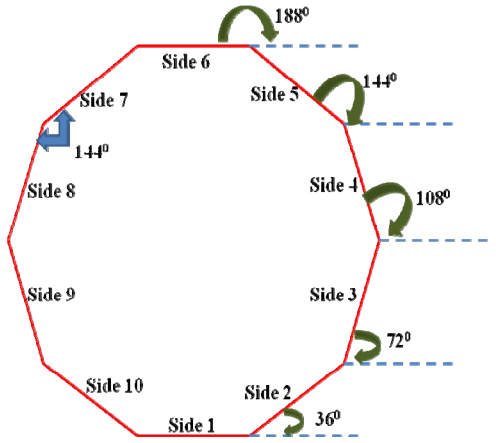

Fig. 2. Decagon showing internal and external angles

At first each straight line of the boundary of the hand gesture is divided into a number of segments. This is done so that the length of the code generated for each image is more or less uniform. Here segment length is taken as 30 pixels. Let a straight line has 62 pixels length and it is approximated using polygon edge 1, so the code is 111.

Next the slope of each segment is calculated and stored. Now a 10-sided regular polygon is considered with one side parallel to the x-axis. The slopes of its ten sides are calculated. Since we are considering a regular polygon, so two sides of the polygon have same slope. The sides 1 and 6, 2 and 7, 3 and 8, 4 and 9, 5 and 10 are parallel, having same slopes as shown in Fig. 2. All the internal angles of decagon [21] are 144^0 according to (7).

$$Interior\_angle = \frac{n-2}{n} \times 180^0 \tag{7}$$

The exterior angles of sides 1 to 5 are 36^0, 72^0, 108^0, 144^0 and 180^0 respectively as shown in fig. 6.

Now we compare the slope of each straight line of the image with the slopes of the sides of the polygon. If the slope of the straight line matches more or less with side 1 of the polygon, then its code is 1. Likewise if the slope matches more or less with side 2 of the polygon its code is 2 according to (8). The code for a specific straight line is 1 if its slope is greater than 18^0, but less than equal to 54^0. Here 54^0 is taken as it is the average of 36^0 and 72^0. In this way the codes of each individual straight line constituting the image are determined and combining them a complete code is generated for each boundary of the hand gestures.

$$Code = \begin{cases} 1 & if \quad 18 < slope \le 54 \\ 2 & if \quad 54 < slope \le 90 \\ 3 & if \quad 90 < slope \le 126 \\ 4 & if \quad 126 < slope \le 152 \\ 5 & if \quad 152 < slope \le 180 \ or \ 0 < slope \le 18 \end{cases} \tag{8}$$

The length of the code depends on the number of straight lines used to smooth the image, as well as on the number of segments into which each straight line is divided.

D. Matching of Hand Gestures

Whenever, an unknown posture is detected, its boundary is calculated and chain code using polygon approximation is produced. Now the chain code is scaled to obtain equal length code like the plots from the database. For scaling purpose, all the code lengths are made to 31. As we have found the largest code length is 31 for all the experimented images. If a code obtain is of length x, then $(31-x)$ no of zeros are added to the end of the code to form equal length chain codes. The unknown code is matched with all the 28 codes from the database. The difference of the chain codes is taking as the error function. The best matched gesture is having least error value. Now this best matched gesture is taken as the output.

$$S = \arg\min \bigvee_{p=28} \left[\sum_{q=1}^{31} \{Code_{unknown} - Code_{known}\} \right] \tag{9}$$

where S is the similarity function, which returns argument based on unknown code ($Code_{unknown}$) and known code ($Code_{known}$) for 28 hand primitives already present in the database. 31 is the maximum code length possible, which is known empirically.

So the proposed algorithm easily identifies any unknown gesture. The input hand gesture images need not to be

centralised, thus the proposed work is translation invariant. The algorithm gives 89.3% overall recognition rate.

Algorithm for hand gesture recognition

Step 0 Create an initial database with polygon represented chain codes, each having 31 numbers for the 28 basic hand gesture primitives of 'Bharatanatyam'.

BEGIN

Step 1A Apply texture based segmentation on the unknown hand gesture.

Step 1B Detect edge by Sobel edge detector.

Step 1C Smooth uneven thickness of the boundary using morphological shrink operation.

Step 2A Obtain two endpoints of the boundary.

Step 2B Start the straight line approximation code from the end point which has less column value.

Step 2C Remove the straight lines of the boundary which are less than 15 pixels, keeping the shape intact.

Step 2D Discard the straight lines which slopes are in the range -400 to +400, make them a single straight line. After this we obtain straight line approximated boundary with less no of straight lines.

Step 3A Segment the straight lines with segment length 30 pixels.

Step 3B Take regular decagon for generation of chain code.

Step 3C Calculate the interior angle of the polygon.

Step 3D Determine the polygon represented chain code for each straight line.

Step 4A Form equal length chain codes 31 length by zero-padding.

Step 4B Calculate the similarity operator for 28 hand primitives already present in the dataset.

Step 4C Best match gesture is taken as the output, i.e., the code having minimum error value.

END

III. Experimental Results

In [7], skin color segmentation is used as the pre-processing step to separate the hand gesture from the background. In normal scenario, when a 'Bharatanatyam' dancer is actually performed, tip of the fingers and middle of the palm are decorated with red color. As proposed in [7], if skin color segmentation is used as the first pre-processing step to detect hand gesture and remove unwanted information in the background. Fig. 3(ii) shows the skin color segmentation of 'Ardhachandran' hand gesture; it is shown that the portion of the hand which is coated with red color is not taken as the segmented output. Skin color segmentation is based on color, i.e., value of the pixel. If the pixel value is within a specific range, then the output is taken as the skin, otherwise those pixels are rejected as non-skin pixels. Fig. 3 (iii) depicts that after application of edge detection technique (here Sobel edge detection is performed); proper boundary of hand is not obtained. On the other side, if texture based segmentation is used with threshold value 0.7 on the same RGB image shown in Fig. 3 (i), exact boundary of the hand gesture is produced as shown in Fig. 3 (vii). Basic principle of texture based segmentation is that it is based on entropy calculation. It is the reason, texture based segmentation gives better result as compared to skin color based segmentation. The above mentioned process is elaborately explained in Fig. 3.

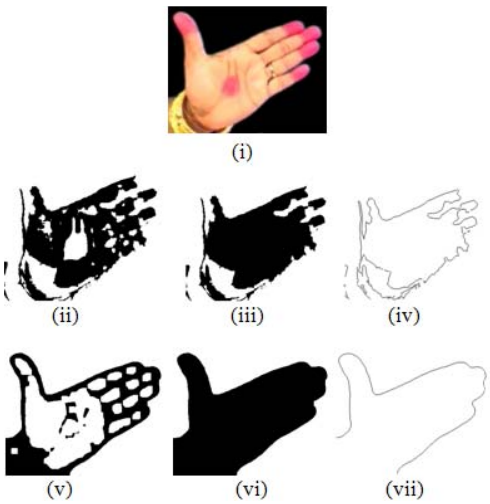

Fig. 3. 'Ardhachandran' (i) Original RGB image, (ii) Skin color segmented image, (iii) Image after filling of holes, (iv) Edge detected image, (v) texture based segmented image, (vi) Holes are removed, (vii) Smoothed boundary of hand gesture

Fig. 4 (i) shows the unknown hand primitive, where no red color decoration is done on the hand. Fig. 4 depicts how an unknown hand gesture is recognized by the proposed algorithm. Here the unknown hand gesture should be recognized as 'Hamsayam' gesture. For this case also texture based segmentation is producing proper boundary of the hand gesture shown in Fig. 4 (ii). Edge of the hand gesture detected using Sobel edge detector is presented by Fig. 4 (iv). The straight line approximated image of the boundary is described in Fig. 4 (v). The chain code for unknown hand gesture is 333 33351333335553322222. As already stated, zero padding is applied if the unknown chain code is less than 31. After adding zeros to the right side of the code, the unknown chain code becomes 33333351333335553322222000000000. The actual chain code for 'Hamsayam' hand primitive from the dataset with zero padding is 33333131143335552222220000 00000. From the two chain codes, one for the unknown image, and the other for 'Hamsayam' posture, maximum no of position is matched, which is 24. Thus the unknown hand gesture is recognized as 'Hamsayam' gesture and it proves that correct recognition is possible by the proposed algorithm.

We have prepared a dataset of 140 images. Each of the 28 single hand gestures is taken 5 times. The proposed algorithm gives accuracy rate of 89.3%, which is much greater than the recognition rate of [6], which is 81.4%. Translation invariance problem is tackled by this algorithm. The average timing complexity of the algorithm is 3.312 sec in an Intel Pentium Dual Core processor running Matlab R011b for each RGB static image. Thus the proposed algorithm is very much useful for e-learning purpose as it gives better accuracy with a

less amount of time. The dancer needs to place her hand in front of the camera, so that the camera captures the hand gesture and recognize it. To tackle the false recognition rate problem, we have introduced a threshold value (here it is taken as 15). If any hand gesture matches less than 16 code positions (taken experimentally) with the fundamental 28 hand gestures, already stored in the dataset, then that hand gesture is declared as 'Non Bharatanatyam Hand gesture'.

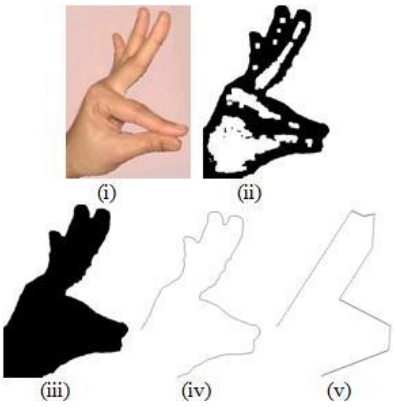

Fig. 4. 'Hamsayam': (i) Original image, (ii) Texture based segmented Image , (iii) After filling of holes, (iv) Extracted boundary, (v) Straight line approximated boundary

IV. Conclusions and Future Work

The entire procedure is cost effective as a single static camera is needed to produce the necessary input images for the proposed algorithm. Considering the complexity of the dance hand gestures, an average computation time of 3.312 sec in an Intel Pentium Dual Core processor running Matlab R011b is highly effective for each RGB input image. The accuracy rate is much better than the previous work [1], we are achieving more 7.9% accuracy than the previous one. The accuracy of our proposed work is 89.3%, which is very much appreciable for e-learning purpose.

However certain shortcomings still do exist. In the proposed algorithm, we have used simple background for hand gesture recognition and achieved recognition rate of 89.3%. However in the case of complex background the recognition rate falls, as unwanted background pixels are also considered as hand. This algorithm gives good results in simple background, but for complex background the algorithm needs to be improved. Another problem is that we are using cropped images, but in the real time scenario, whole body of the dancer is taken as the input by the camera. To overcome this shortcoming, we are using particle filtering approach for detection of hand region in a video. This approach, widely employed in face detection, consists of scanning an image at different positions and scales. Our final goal is to build a fully automatic hand posture recognition system.

In a nutshell, hand gesture recognition of 'Bharatanatyam' dance is relatively unexplored and the proposed work is an attempt to address the problem with better accuracy and scopes for further research.

References

[1] S. Mitra and T. Acharya, "Gesture recognition: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 3, pp. 311–324, 2007.

[2] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, 2001.

[3] A. Jaimes and N. Sebe, "Multimodal human–computer interaction: A survey," *Computer vision and image understanding*, vol. 108, no. 1, pp. 116–134, 2007.

[4] M. J. Rosenberg, *E-learning: Strategies for delivering knowledge in the digital age*, vol. 9. McGraw-Hill New York, 2001.

[5] R. C. Clark and R. E. Mayer, *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning*. Wiley. com, 2011.

[6] D. Hariharan, T. Acharya, and S. Mitra, "Recognizing hand gestures of a dancer," in *Pattern Recognition and Machine Intelligence*, Springer, 2011, pp. 186–192.

[7] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International Workshop on Automatic Face and Gesture Recognition*, 1995, vol. 12, pp. 296–301.

[8] H. Zhou, D. J. Lin, and T. S. Huang, "Static hand gesture recognition based on local orientation histogram feature distribution model," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, 2004, p. 161.

[9] M. Sonka, V. Hlavac, R. Boyle, and L. A. Ray, "Image processing, analysis and machine vision," *Journal of Electronic Imaging*, vol. 5, p. 423, 1996.

[10] M. Sonka, V. Hlavac, and R. Boyle, "Image processing, analysis, and machine vision," 1999.

[11] K. I. Laws, "Textured Image Segmentation.," DTIC Document, 1980.

[12] T. R. Reed and J. M. H. Dubuf, "A review of recent texture segmentation and feature extraction techniques," *CVGIP: Image understanding*, vol. 57, no. 3, pp. 359–372, 1993.

[13] H. C. Nothdurft, "Texture segmentation and pop-out from orientation contrast," *Vision research*, vol. 31, no. 6, pp. 1073–1078, 1991.

[14] T. Peli and D. Malah, "A study of edge detection algorithms," *Computer graphics and image processing*, vol. 20, no. 1, pp. 1–21, 1982.

[15] N. Senthilkumaran and R. Rajesh, "Edge detection techniques for image segmentation–a survey of soft computing approaches," *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, pp. 250–254, 2009.

[16] D. Ziou and S. Tabbone, "Edge detection techniques-an overview," *PATTERN RECOGNITION AND IMAGE ANALYSIS C/C OF RASPOZNAVANIYE OBRAZOV I ANALIZ IZOBRAZHENII*, vol. 8, pp. 537–559, 1998.

[17] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International Journal of Image Processing (IJIP)*, vol. 3, no. 1, pp. 1–11, 2009.

[18] R. Bellman and R. Roth, "Curve fitting by segmented straight lines," *Journal of the American Statistical Association*, vol. 64, no. 327, pp. 1079–1084, 1969.

[19] C. M. Williams, "Bounded straight-line approximation of digitized planar curves and lines," *Computer Graphics and Image Processing*, vol. 16, no. 4, pp. 370–381, 1981.

[20] M. Ryan, *Geometry for Dummies*. Wiley. com, 2008.

[21] S.-D. Kim, J.-H. Lee, and J.-K. Kim, "A new chain-coding algorithm for binary images using run-length codes," *Computer Vision, Graphics, and Image Processing*, vol. 41, no. 1, pp. 114–128, 1988.

[22] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*, vol. 5. McGraw-Hill New York, 1995.

[23] H. Shi and G. X. Ritter, "A new parallel binary image shrinking algorithm," *Image Processing, IEEE Transactions on*, vol. 4, no. 2, pp. 224–226, 1995.

[24] I. Sobel, "Neighborhood coding of binary images for fast contour following and general binary array processing," *Computer Graphics and Image Processing*, vol. 8, no. 1, pp. 127–135, 1978.