Chapter 6

A Hierarchical Classification Algorithm for Sattriya Dance Single-Hand Gestures Recognition

In this chapter, a hierarchical classification algorithm for single-hand gesture (Asamyukta Hastas) recognition of Sattriya dance is proposed. The algorithm attempts to narrow down the search space at each level of the hierarchy till a hasta is completely recognized at a leaf node. An entropy based similarity measure is also introduced to measure the correctness of performing an Asamyukta hastas. The similarity measure provides a similarity score between the input Asamyukta hasta image and hasta images performed by experts in the database. In this approach, initially entropy of all classes are calculated, where a class comprises of hasta images of an Asamyukta hasta performed by experts. The similarity of a new hasta image with a class is computed by considering the change of entropy value after adding this hasta image to that class. The overall approach is tested on the Medial Axis Transformation (MAT) image dataset. The algorithm uses vision-based structural (shape-based) features of single-hand gestures. These vision-based features represent the shape of hand gesture images. These features are extracted from the MAT images. The algorithms for extraction of these features are also presented.

The chapter is organized as related work of features extraction are summarized in the next section. The proposed vision-based structural features are discussed briefly in Section 6.2. Section 6.3 describes the proposed hierarchical classification approach for single-hand gestures of Sattriya dance and computation of similarity score. We conclude this chapter with some future research directions in Section 6.4.

6.1 Related Works

Shape is an important vision-based features used to describe the image contents. Extraction of shape feature from 2-dimensional images of 3-dimensional objects is a difficult task due to the information loss incurred in projecting an object from 3-dimension to 2-dimension. The task becomes more complicated when the images are corrupted with noise, distortion and occlusion. The various features like moments, curvature, spectral features can be used to describe the shape of an object. Many shape based features are available in the literature. These shape based features represent the whole image or sometimes boundary image. If the features represent the whole image then they are contour based features and otherwise they are region based features. Again the contour based features are sub categorized into global features and structural features. The global features are represented by segments or sections and structural features are represented as a whole. The global shape descriptors are area, perimeter, eccentricity, major axis length, minor axis length [84], convexity, principle axis, circular variance and elliptic variance [9]. These global features are describe the boundary shape. Again, the structural feature extraction methods such as chain code, polygon decomposition, smooth curve decomposition, scale space methods and syntactic analysis [84] are capable of partial matching and unable to capture the global information. Since, these global features are only describe the boundary shape and structural features are unable to capture global information. So, these features are not sufficient to classify our Sattriya dance single-hand gestures dataset. In the next section, some of the vision-based shape features from the MAT image dataset.

6.2 Proposed Features

In this section, eight vision-based structural features used in the proposed method are described. The extraction of MAT image dataset from the gray image dataset was discussed in the previous Chapter 5. Vision-based features represent those features which can represent the shape of objects (hands) in a scene and can be visualized by normal eye. The features used in the proposed method which are invariant with respect to rotation and scaling are Euler number, angle between fingers, fingers_tips distance, number of edges from cycle, number of edges from border, edges from border merge to a cycle, edge length and number of branch points. These vision-based features are briefly described below:

6.2.1 Euler Number

Euler number represents the topological structure of an image. It is used to find number of holes (connecting cycle) present in the image. This feature is invariant to translations, rotation and scaling. Mathematically, Euler number can be calculated as $E = N - H$, where E= Euler Number, N= Number of region and H= Connecting Cycle. Examples of Euler number of some MAT images are shown in the Figure 6-1.

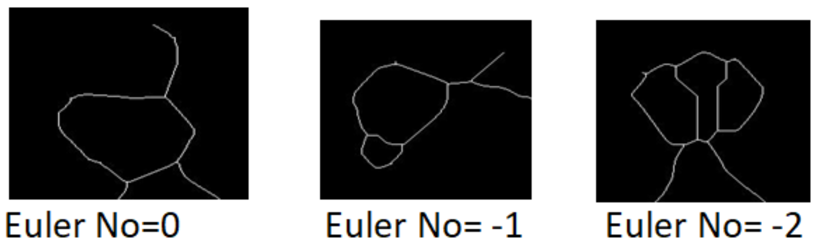


Figure 6-1: Example of Euler Number Features

6.2.2 Angle Between Fingers and Finger_tips distance

The feature 'Angle between fingers' is used to measure the gap exists between fingers. Let x, y are the length of two fingers of a MAT image of a hasta as shown in the second image of Figure 6-2. Also let p_1 and p_2 are the end-points of first finger, p_1 and p_3 are the end points of second finger. Here, p_2 and p_3 are the tips of the both fingers. Let z is the distance between the finger tips. The angle A made by the two fingers at p_1 is given by the equation 6.1

$$z^2 = x^2 + y^2 - 2xycosA \tag{6.1}$$

$$\cos A = \frac{x^2 + y^2 - z^2}{2xy} \tag{6.2}$$

$$A = \cos^{-1}(\frac{x^2 + y^2 - z^2}{2xy}) \tag{6.3}$$

And, the distance between the finger tips, i.e., z, can be approximated by the length of the arc between p_2 and p_3 made by the circle centered at p_1 with radius x(or y) as given in the following expression:

$$Finger\_tipsdistance = \frac{3.14 * angle * radius}{180^0} \tag{6.4}$$

and x (or y)=$\sqrt{(p1 - p2)^2}$

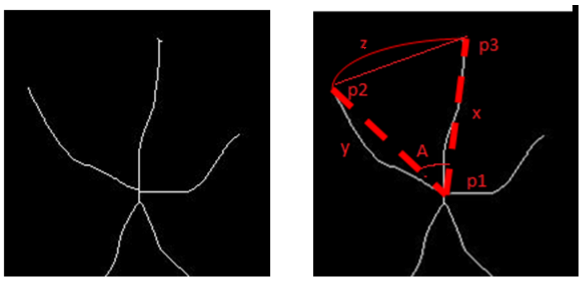


Figure 6-2: Angle Between Fingers and *Finger_Tips* Distance

6.2.3 Number of Edges from Cycle

The feature 'number of edges from cycle' is used to represent the number of edges touch in the connecting cycle. The Algorithm 3 is used to find out this feature from the images. An example showing the steps of this algorithm

Algorithm 3: Number of Edges from Cycle

```
Input: MAT Image
Output: $x1$
1. Fill all holes present in the input MAT image.
2. Subtract the original MAT image from the filled holes image.
3. Convert gray scale image to binary image.
4. Create a disk-shaped morphological structuring element with radius two
5. $morph \leftarrow$ Dilate the binary image using the structuring element.
6. for $i = 1 to rowstep 1$ do
       for $j = 1 to columnstep 1$ do
           if $morph(i, j) = 1$ then
               $image(i, j) \leftarrow 0$
7. $image2 \leftarrow$Convert the image1 to binary image;
8. $component \leftarrow$Find the connected component in the image2;
9. $x1 \leftarrow$Number of objects in the component;
10. return x1
```

are shown in Figure 6-3

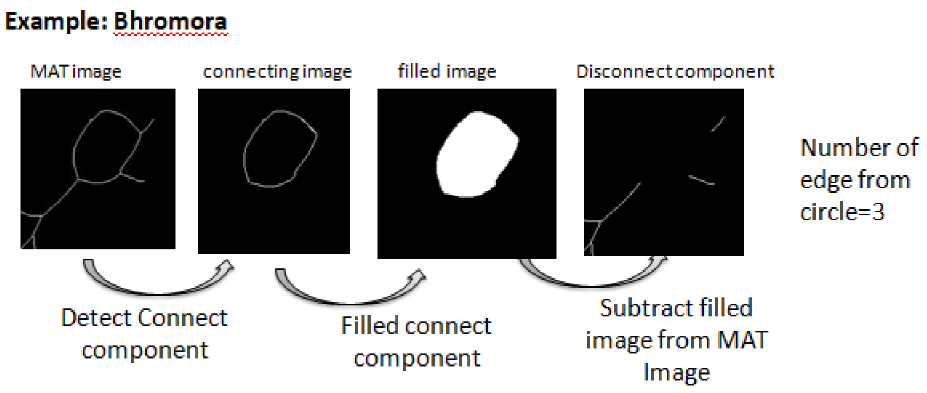


Figure 6-3: Steps for Finding Number of Edges from Cycle

6.2.4 Number of Edges from Border

To feature represent the number of edges touching the boundary in a MAT image. In the binary image, when continuous 1 present then it represent one edge (line). So, to find out the number of line come out from the border, we check the first column and first raw of the image

Then, similar procedure is applied after reversing the image matrix. The algorithm to extract the feature 'number of edge from border' is explained in Algorithm 4. In addition, the dependent algorithm for border_check and

Algorithm 4: Number of Edges from Border

```
Input: MAT Image
Output: $y_1$
1. Read MAT image matrix
2 Set count= 0;
3. Check first row and first column using Algorithm 5
4. If consecutive 1 present from the border
6. count=count+1
7. Reverse the image matrix using Algorithm 6
8. Repeat step 3-6
9. $y_1$= Count
10. Return $y_1$
11.end
```

reverse_matrix are explained in Algorithm 5 and Algorithm 6 respectively. The Figure 6-4 shows example of number of edge from border features.

6.2.5 Edges from Border Merge to a Cycle

The feature 'edge from border merge to a cycle' represents the edge coming out from the border and directly merging to a cycle (connecting hole). This

Algorithm 5: Check_border

```
Input: MAT Image
Output: Count edge come out from border
1. Initialize $i \leftarrow 1$, $j \leftarrow 1$, $count \leftarrow 0$
2. while $i < row$ do
    if $a(i, j) = 0$ then
        $i \leftarrow i + 1$
    else
        if $a(i - 1, j + 1) = 1$ then
            $i \leftarrow i - 1$
            $j \leftarrow j + 1$
    else
        if $a(i, j + 1) = 1$ then
            $j \leftarrow j + 1$
    else
        $a(i + 1, j + 1) = 1$
    $i \leftarrow i + 1$ $j \leftarrow j + 1$
    for $i = 1 : end(row)$ do
        for $j = 1 : end(column)$ do
            if $a(i, j) = 1$ and $BW(i, j) = 1$ then
                $Count \leftarrow Count + 1$
3. return count;
```

Algorithm 6: Reverse_Matrix

```
Input: MAT Image
Output: Reverse_Matrix
1. Read image matrix
2. Initialize $K \leftarrow 1$
for $i = end : -1$ do
    $m \leftarrow 1$ for $j = end : -1$ do
        $a1(k, m) = a(i, j)$
        $m \leftarrow m + 1$
    $k \leftarrow k + 1$
3. Initialize $p \leftarrow 1$
for $s = end : -1$ do
    $q \leftarrow 1$
    for $t = endto - 1$ do
        $BW1(p, q) \leftarrow BW(s, t)$ $q \leftarrow q + 1$
    $p \leftarrow p + 1$
4. return BW1
```

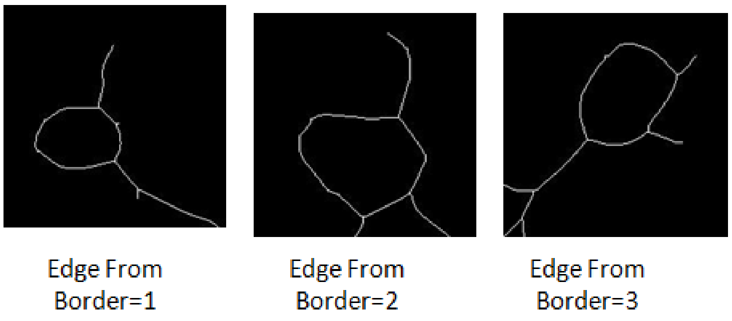


Figure 6-4: Example of Number of Edges from Border

feature provides binary output. If the edge from border directly merge to a cycle then output returns the value 1 otherwise returns 0. The algorithm 7 extracts the features value edge from border merge to a cycle. The example

Algorithm 7: Edges from Border Merge to a Cycle

```
Input: MAT Image
Output: $z1$
1. Fill all holes present in the image (I) and store in BW
2. Find difference of I & BW and store in BW1
3. Detect the edges in binary image BW1
4. Find the edges those originate from border area using Algorithm 5
5. Set count=0;
6.for Each detected edge do
    Traverse from the border until junction point is reached
    if the junction points is a part of hole(BW1) then
        count++;
7. if $count > 0$ then
    $z1 \leftarrow 1$
else
    $z1 \leftarrow 0$
8. Return z1
```

of edge from border directly merge to a cycle is shown in Figure 6-5

6.2.6 Edge Length

Edge length feature is used to find length of fingers. To find edge length, first we find the length of each edge. Then, taking the average length of the edges (finger), and one threshold value is set. If the average length of the edge is shorter than the threshold value, then the edge length is considered as shorter edge length other wise it is considered long edge length. Here, the output of the feature value is binary where 0 denotes short edge length and

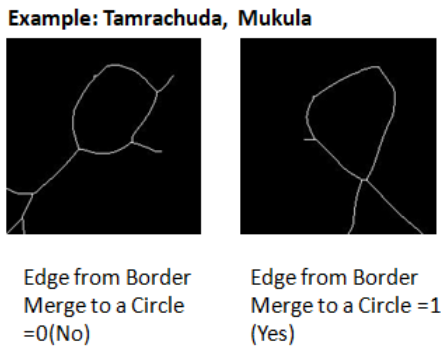

Example: Tamrachuda, Mukula
Edge from Border Merge to a Circle =0(No)
Edge from Border Merge to a Circle =1 (Yes)

Figure 6-5: Example of Edges from Border Directly Merge to a Cycle

1 for long edge length. Example of two hastas are shown in Figure 6-6. The steps for finding edge length feature is shown in Algorithm 8

Algorithm 8: Edge Length

```
Input: MAT Image
Output: $BinaryNumber$
1. Detect all the edges of the image using Sobel edge detection method.
2. $Edges$ ←Store all the edges;
3. Find the length of all edges
4. Find the number of edges
5. $AvgEdgeLength = lengthOfEdges/numberOfEdges$;
6. if $AvgEdgeLength > 10$ then
    $disp('LongEdge')$;
    Return 1;
else
    $disp('ShortEdge')$;
    Return 0;
```

6.2.7 Number of Branch Points

Number of branch point feature represents number of junction point. The steps to find out the branch points from the images are shown in Algorithm 9. Figure 6-7 shows an example of branch point.

6.3 Proposed Method

The steps involved in the proposed hierarchical classification approach are pre-processing, medial axis transformation, features extraction, classification and finally similarity score computation. The work flow diagram as shown

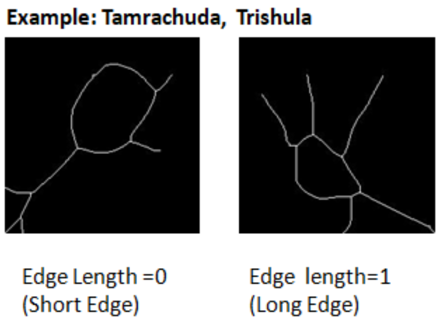


Figure 6-6: Example of Edge Length

Algorithm 9: Branch Point Location

```
Input: MAT Image
Output: $BranchPoint$
1. Read input image;
2. $bp \leftarrow bwmorph(BinaryImg,' branchpoints')$;
3. $[rowcolumn] \leftarrow find(bp)$;
4. $branchpoints \leftarrow [rowcolumn]$;
5. $U \leftarrow numel(row)$;
6. $U \leftarrow$Number of Branch Points in the image;
7. return U
```

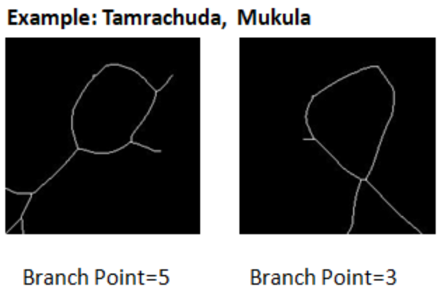


Figure 6-7: Example of Branch Point

in Figure 6-8 depicts the basic framework of the whole system. Each steps of this flow figure has been explained briefly in the next subsection.

6.3.1 Preprocessing

During preprocessing the RGB image is converted to binary image as shown in Figure 3.3.5.3.

6.3.2 Medial Axis Transformation (MAT)

The Medial Axis Transformation (MAT) reported in Algorithm 1 in Chapter 5 is used to find out the skeletal of the images.

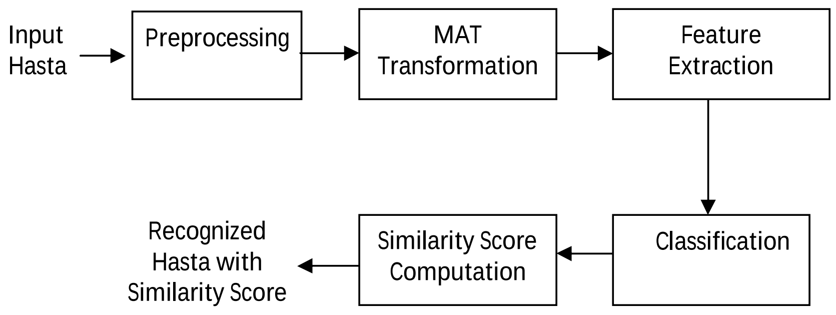


Figure 6-8: Schematic Diagram for Hierarchical Classification Approach

6.3.3 Feature Extraction

The features are mainly used to distinguish the images into different classes. In this chapter shape based features are extracted to classify the images. Euler number (EN), edge length (EL), finger_tips distance (TD) and angle between fingers (AF), number of edges from cycle (EC), number of edges from border (EB), edge from border merge to a cycle (EBMC) and branch point (BP) features are explored from the MAT image dataset. The methods of extraction are described briefly in Section 6.2 .

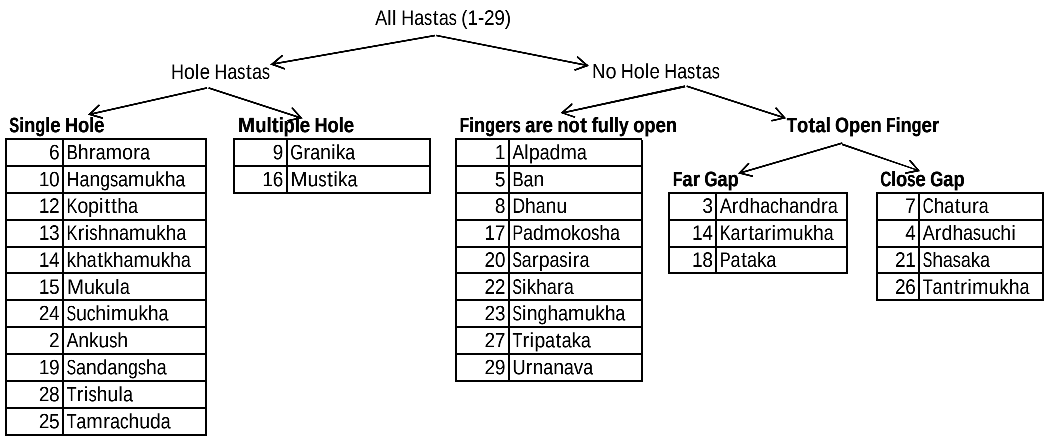
6.3.4 Classification

The proposed hierarchical classification algorithm can be explained by the tree structure shown in Figure 6-9. In this figure, internal nodes of the tree represent the group of hastas and a leaf node of the tree represent a hasta. The features used in this algorithm are categorized into high level features and low level features. The high level features are Euler number (EN), edge length (EL), finger_tips distance (TD) and angle between fingers (AF) used for group recognition. The low-level features are number of edge from cycle (EC), number of edge from border (EB), edge from border merge to a cycle (EBMC) and number of branch point (BP) used for individual hasta recognition within the hasta group. The high level features are used at leaf nodes to classify a hasta image into a hasta group while low level features are used at the leaves of hierarchy to recognize the hasta image within the hasta groups represented by the predecessor nodes.

The hierarchical classification algorithm is presented in Algorithm 10. At

Algorithm 10: Hierarchical Classification for Asamyukta Hastas (Single-Hand Gestures) Recognition

```
Input: RGB Image
Output: Recognized Hasta Name
1. $image \leftarrow ConvertRGBtoMAT(imgRGB)$;
2. $EN \leftarrow EulerNumber$;
3. if $EN > 0$ then
    Print('No hole exist')
    if $OpenFingerGroup(image)$ then
        if $GapExist(image)$ then
            $WideOpenGroup(image)$;
        else
            $SlightOpenGroup(image)$;
    else
        $CloseFingerGroup(image)$;
else
    Print('Hole exist')
    if $EN = 0$ then
        Print('Single Hole exist')
        $SingleHoleGroup(image)$;
    else
        Print'Multltiple hole exist'
        if $EN = -1$ then
            Print'Granika Hasta'
        if $EN = -2$ then
            Print'Mustika Hasta'
        else
            Print('Match not found')
```



Single Hole	
6	Bhramora
10	Hangsamukha
12	Kopittha
13	Krishnamukha
14	khatkhamukha
15	Mukula
24	Suchimukha
2	Ankush
19	Sandangsha
28	Trishula
25	Tamrachuda

Multiple Hole	
9	Granika
16	Mustika

Fingers are not fully open	
1	Alpadma
5	Ban
8	Dhanu
17	Padmokosha
20	Sarpasira
22	Sikhara
23	Singhamukha
27	Tripataka
29	Urnanava

Far Gap	
3	Ardhachandra
14	Kartarimukha
18	Pataka

Close Gap	
7	Chatura
4	Ardhasuchi
21	Shasaka
26	Tantrimukha

Figure 6-9: Conceptual Diagram of Hierarchical Classification Algorithm

the begining, the preprocessed input image of hand gestures (hasta) is converted to a MAT image using *ConvertRGBtoMAT* function presented in Algorithm 1.

6.3.4.1 Hasta Group Recognition

Hasta group recognition at the initial level is based on Euler number (EN) computed from the MAT image as follows:

- if EN>0, no-hole group
- if EN=0, single-hole group
- if EN<0, multiple-hole group

Under the no-hole group, hastas are classified into either open-finger group or close-finger group base on the attribute Edge Length (EL). If the avarage EL of input hasta is greater than a threshold (selected from the observed value) EL, then EL is considered as long edge and categorized into open-finger group otherwise close-finger group. Again, under the open-finger group a hasta is classified as either as wide-open or slight-open based on the features 'angle between fingers' (AF) and *finger_tips* distance (TD). If the angle between adjacent finger other than the thumb finger is greater than 45° and *finger_tips* distance is greater than threshold (experimentally determined from a range of possible value) then the input hasta is categorized into wide-open group otherwise close-finger group

6.3.4.2 Hasta Recognition within Hasta Group

After the input hasta is classified into one of hasta group as described in the previous subsection, the low-level features are used to recognize the hasta at the leaf node.

1. Single hole group: The hastas within single-hole group are recognized using features number of edge from cycle (EC), number of edge from border (EB), edges from border merge to a cycle (EBMC) and edge length (EL). The feature values of single-hole group hastas are shown in Table 6.1. Algorithm for single-hole group is presented in Algorithm 11

Table 6.1: Features Values of Hand Gestures Having Single-Hole Group

Gesture-Name	**Number of Edge From cycle**	**Number of Edge from Border**	**Edge From Border Merge to a Cycle**	**Edge Length**
Tamrachur	2	2	0	0
Krishnasarmukh	2	2	0	1
Sandagsha	3	1	0	-
Hangsamukha	3	2	0	0
Khatkhamukh	3	2	0	1
Mukula	3	2	1	-
Bhramora	3	3	0	-
Kopitha	4	0	0	-
Suchimukha	4	2	0	-
Ankush	4	2	1	-
Trishul	5	2	0	-

2. Close-finger group: The hastas within Close-finger group are recognized using the features number of branch points (BP) and number of edges from border (EB). The values of these features of close-finger group hastas are shown in Table 6.1. Algorithm for close-finger group is presented in Algorithm 12

3. Wide-open group: The hastas within wide-open group are recognized by counting number of fingers. A simple Algorithm 13 is used to count the number of open fingers from the upper half of the MAT images

   Since these hastas are fully open, so it can be possible to horizontally divide the hastas into two half. Then ignore the lower part and count the number of straight fingers.

Algorithm 11: Single Hole Group Recognition

```
Input: MAT Image
Output: Recognized Hasta Name
$x1 \leftarrow NoOfEdgeFromCircle(image)$;
$y1 \leftarrow NoOfEdgeFromBorder(image)$;
$z1 \leftarrow EdgeFromBorderMergeToACircle(image)$;
$u1 \leftarrow EdgeLength(image)$;
print Hand Gesture Recognized is:
if $x1 = 2$ AND $y1 = 2$ AND $z1 = 0$ AND $u1 = 0$ then
    print'Tamrachur'
if $x1 = 2$ AND $y1 = 2$ AND $z1 = 0$ AND $u1 = 1$ then
    print'Krishnamukha'
if $x1 = 3$ AND $y1 = 1$ AND $z1 = 0$ then
    print'Sandangsha'
if $x1 = 3$ AND $y1 = 2$ AND $z1 = 0$ AND $u1 = 0$ then
    print'Hangsamukha'
if $x1 = 3$ AND $y1 = 2$ AND $z1 = 0$ AND $u1 = 1$ then
    print'Khatkhamukha'
if $x1 = 3$ AND $y1 = 2$ AND $z1 = 1$ then
    print'Mukula'
if $x1 = 3$ AND $y1 = 3$ AND $z1 = 0$ then
    print'Bhramora'
if $x1 = 4$ AND $y1 = 0$ AND $z1 = 0$ then
    print'Kopittha'
if $x1 = 4$ AND $y1 = 2$ AND $z1 = 0$ then
    print'Suchimukha'
if $x1 = 4$ AND $y1 = 2$ AND $z1 = 1$ then
    print'Ankusha'
if x1=5 AND y1=2 AND z1=0 then
    print'Trishula'
```

Algorithm 12: Close-Finger Group Recognition

```
**Input:** MAT Image
**Output:** Recognized Hasta Name
for $i := 1 to size(image, 1) step 1$ do
    for $forj := 1 to size(image, 2) step 1$ do
        if $image(i, j) < size(image)$ then
            $image(i, j) \leftarrow 0$
$bp \leftarrow CountBranchPoints$;
$[rowcolumn] \leftarrow find(bp)$;
$u \leftarrow Numberofelement(row)$;
$v \leftarrow EdgeLength(image)$;
Print Hand Gesture Recognized is:
if $u = 5$ AND $v = 3$ then
    print'Ban'
if $u = 4$ AND $v = 3$ then
    print'Dhanu'
if $u = 3$ AND $v = 2$ then
    print'Padmokosha'
if $u = 3$ AND $v = 0$ then
    print 'Sarpasira'
if $u = 5$ AND $v = 1$ then
    print'Sikhara'
if $u = 6$ AND $v = 2$ then
    print'Singhamukha'
if $u = 6$ AND $v = 0$ then
print'Tripataka'
if $u = 5$ AND $v = 2$ then
print'Uranava'
if $u = 4$ AND $v = 1$ then
print'Alapadama'
```

Table 6.2: Feature Value of Hand Gestures Having Close-Finger Group

Gesture Name	No of Branch Points	No of Edge from Border
Ban	5	3
Dhanu	4	3
Padmokosh	8	2
Sarpasir	3	0
Sikhar	5	1
Singhamukh	6	2
Tripataka	6	0
Uranava	6	2
Alapadama	4	1

4. Slight-open group: The recognition of hastas of slight-open group is almost same as wide-open hastas recognition, however the hasta images to MAT images conversion are different, Since there a little gaps between the finger of this hastas, a different threshold is used to make the fingers more distinguishable. The threshold value is experimentally determined from a range of possible values. In these experiment, the best result is observed by taking the value is $1.9 \times graythreshold$. The algorithm slight-open group classification is presented in Algorithm 14.

Output steps of conversion from gray to MAT for hasta of wide-open and slight open groups are shown in Figure 6-10 and Figure 6-11 respectively.

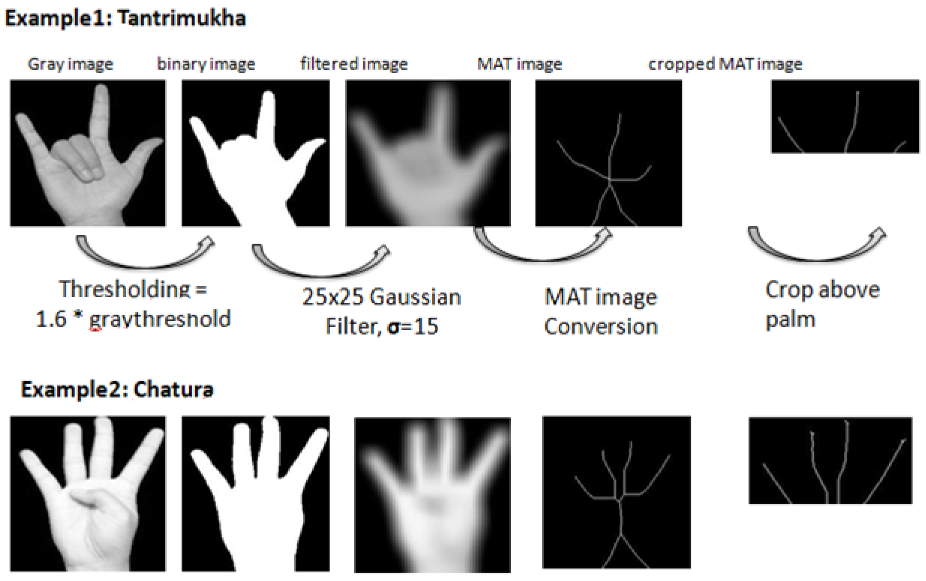


Figure 6-10: Output Steps of Conversion from Gray to MAT Image for Wide-Open Hasta Group

Algorithm 13: Wide-Open Group Recognition

```
Input: MAT Image
Output: Recognized Hasta Name
Divide the input image in to two half horizontally;
$image \leftarrow Upperhalf$;
$[rowcolumn] \leftarrow size(image)$;
Initialize $i \leftarrow 90$ $count \leftarrow 0$
for j=1 to column do
    if $image(i,j) = 1$ then
        $count++$
switch count do
    case 1 do
        'Ardhasuchi'
    case 2 do
        'Shasaka'
    case 3 do
        'Tantrimukha'
    case 4 do
        'Chatura'
    otherwise do
        Match not Found
```

Algorithm 14: Slight-Open Group Recognition

```
Input: MAT Image
Output: Recognized Hasta Name
Divide the input image in to two half horizontally; $image \leftarrow Upperhalf$;
$[rowcolumn] \leftarrow size(image)$;
Initialize $i \leftarrow 90$ $count \leftarrow 0$
for j=1 to column step1 do
    if $image(i,j) = 1$ then
        $count++$
if count=2 then
    Print'kartarimukha'
if count=3 then
    Print'Sangdangsha'
if count=4 then
    Print'Pataka'
```

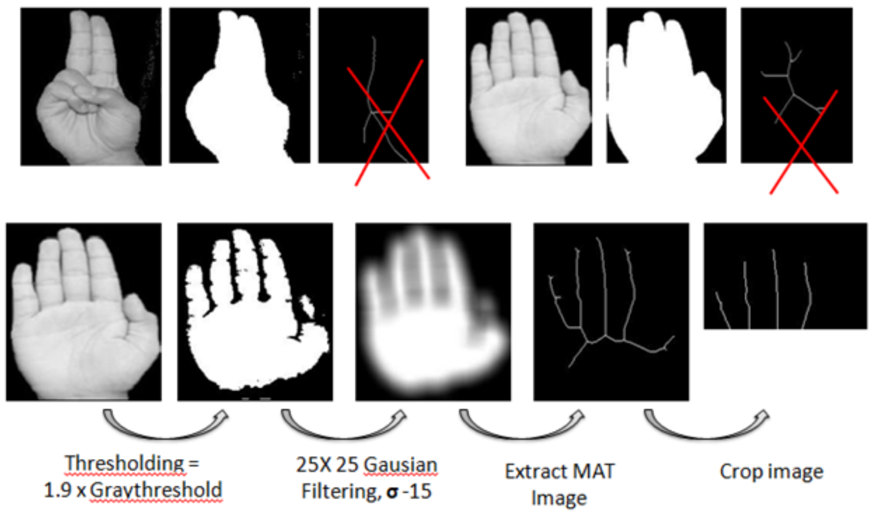

Thresholding = 1.9 x Graythreshold
25X 25 Gausian Filtering, σ -15
Extract MAT Image
Crop image

Figure 6-11: Output Steps of Conversion from Gray to MAT Image for Slight-Open Hasta Group

6.3.4.3 Complexity Analysis

The time complexity of different steps of the Hierarchical classification Algorithm 10 is presented as follows:

1. RGB image to MAT image conversion needs a linear time complexity i.e.,O(n).
2. To matching features value in if and else condition, it requires to compare some constant values i.e., complexity is O(1).
3. $WideOpenGroup(image)$ involves complexity O(N), where N is the number of column in an image.
4. $SlightOpenGroup()$ involves same complexity as previous function $WideOpenGroup(image)$.
5. $CloseFingerGroup(image)$ involves complexity $O(N^2)$, where N is the number of raw and column consider a square matrix of the image.
6. $SingleHoleGroup(image)$ needs a complexity of $O(N^2)$, where N is the number of raw and column consider a square matrix of the image.
7. If multiple hole present then constant amount of time required. So, complexity is O(1).

Among these steps, since large amount of time is needed if fingers are not fully open or single hole present. In that case we need to check the whole image i.e., complexity $O(N^2)$ dominates. Hence, the worst case complexity of the proposed method is $O(N^2)$.

6.3.5 Similarity Score Computation

A methods for computation of similarity score to measure the correctness of a hasta performed by a dancer. The method uses Shanon's entropy measure to compare a hasta with a database of hastas performed by an expert. Initially, the entropies of 29 hasta classes are computed where a hasta class consists of hasta images performed by experts of Sattriya dance. In the step2, the input hastas (feature represents) are classified by considering entropy based similarity measure. For each input instance, it computes the new entropy for each class by including the instance. So, it gives a set (cardinality) of new entropy values. Next, we compare these values with the older values. And, we assign the instance to that class for which the entropy variation is minimum. we summarize the steps as follows

1. For each class of training instances
   (a) Compute entropy using Shannon's entropy measure.
   (b) Store entropy values in an array, Arrayold[]
2. For each input test instance
   (a) Include in a class of training instances.
   (b) Compute entropy for the class.
   (c) Repeat for all other classes and store entropy values in Arraynew[].
   (d) Compare entries for each class between Array old and Array new and store their value in Array diff.
   (e) Calculate similarity score =(1-Array diff)*100

The entropy for the class is given by

$$Entropy(C_i) = \frac{1}{L}\sum_{r=1}^{L} Entropy(C_i^r) \tag{6.5}$$

where L is the number of samples for each class.

6.4 Experimental Results

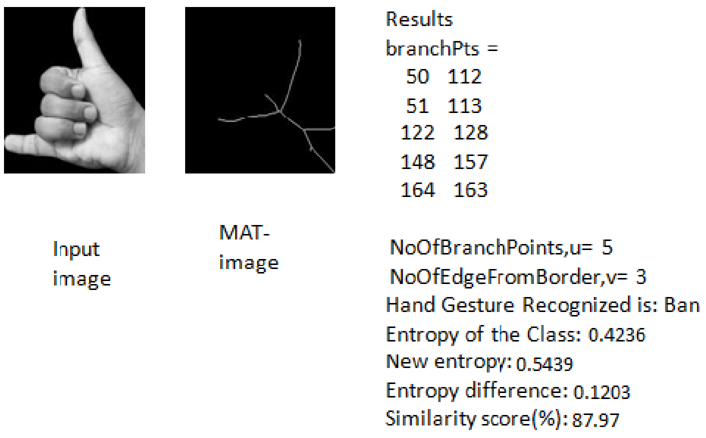
The experimental description and the results obtained for the proposed method on Sattriya dance single-hand gestures dataset are discussed in the following subsections.

6.4.1 Dataset Description

In this chapter, we use SSHG-MAT image dataset which contains 1450 images. From the MAT image dataset, eight vision-based features were extracted. This extracted features values were used throughout this experiment.

6.4.2 Classification Results

The overall classification accuracy achieved for asamyukta hastas of Sattriya dance classification is 86.52%. The 1450 instances are divided into 90% (1309 images) as training data and 10%(141 image) as testing data. Out of this 141 testing images 122 images are correctly classified. After recognition, the entropy of all the individual classes are calculated for continuing research work in the next phase for finding the similarity score of each individual hasta. The snapshot of Matlab results of Ban hasta recognition is shown in Figure 6-12 and corresponding ROC graph in Figure 6-13.



```
Results
branchPts =
   50   112
   51   113
  122   128
  148   157
  164   163

NoOfBranchPoints,u= 5
NoOfEdgeFromBorder,v= 3
Hand Gesture Recognized is: Ban
Entropy of the Class: 0.4236
New entropy: 0.5439
Entropy difference: 0.1203
Similarity score(%): 87.97
```

Figure 6-12: Example of Ban Hasta Recognition with Similarity Score

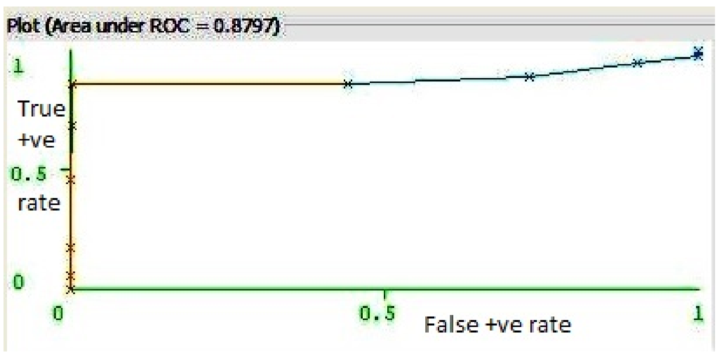


Figure 6-13: ROC graph of Ban Hasta Recognition with Similarity Score

6.5 Discussion

In this chapter, a hierarchical classification algorithm is presented for single-hand gestures of Sattriya Dance. In addition, a similarity measure to support effective classification also has been reported. Moreover, eight vision-based feature have been proposed in this chapter.

It has not been possible to compare the proposed classification method with other dataset because of unavailability of similar datasets. One dataset for single-hand gestures of Bharatnatyam classical dance was used in [19], however it is not publically availablle. Also, no response was received authors in this regard. Furthermore, out of the 28 hand gestures of Bharatnatyam dance, 8 hand gestures are not similar as used Sattriya dance. Also, the method reported in [19] for single-hand gestures of Bharatnatyam dance misclassified 4 hand gestures in their own dataset. Among these 4 misclassified hand gestures, 2 are also used in Sattriya dance which are correctly classified by our proposed method.

Our future research will focus on extending the dataset for other classical dance and use deep learning method to improve recognition accuracy.