Classification of Indian Classical Dance Forms From Images

B. Jahnavi - S20160010009, G. Monica Sagar - S20160010026 and N. Srilekha - S20160010055

***Abstract*— In this project, our aim is to successfully classify Indian classical dance forms from various images. There are totally 8 classical dance forms. We have collected different poses of all the dance forms and created a training set using these images. Initially, we are extracting the features from the images. Then, we are creating an SVM model that will be trained using train set images. We are then testing it on the test set. We have extended this project for video as an input. So, when given a video, our code will predict which dance form it belongs to. Keywords: Pose recognition, Hu moments, Multi-class SVM.**

I. INTRODUCTION

India has thousands of year old culture coming from the Vedic time of fine arts, music and traditional dances. In this paper, we would be focusing on Indian classical dance forms. Indian dance forms primarily use the same 'mudras' as a sign of expression. In various sources and as told by many scholars, the total number of recognized classical dances forms ranges from eight to more. But in this paper, we would be focusing on 8 different classical Indian dance forms, namely Bharatanatyam, Kathakali, Kathak, Kuchipudi, Odissi, Sattriya, Manipuri, and Mohiniyattam. We have to obtain a pose from images and then classify them to their particular dance form. For this, we have to do several steps in series which would help us in classifying images accurately. In this paper, we will try to describe the process of extracting poses from images and classifying them to their particular dance forms.

There are three stages of building our model. They are Binarization of images, Feature extraction and Training SVM Classifier.

Before we start building our model, we have to first collect images for the training set. For each dance form, we have collected – images each containing different pose. These images form our training set.

II. PROCEDURE

A. Binarization of images

Our training set comprises of all RGB images. These RGB images are converted to gray-scale images i.e., the image consists of pixel values ranging from 0 to 255. After converting it to gray-scale image, we then binarize the images. For binarizing each image, we have to first set a threshold intensity value. We divide the image into pixels. Each pixel has an intensity value. If the intensity value of the pixel is below the threshold then we mark the pixel value as 0. If the intensity value of the pixel is above the threshold then we mark the pixel value as 1. This final image which is obtained is a binarized image. We apply binarization to all the images in the training set. In the case of testing, we do the same and then do the later steps.

B. Feature extraction

Hu moments are derived from the shape of the binarized image which helps us determine the pose. These non-orthogonal centralised moments are translation invariant and can be normalised with respect to variations in scale. Nevertheless, to facilitate invariance to the rotation we have to reformulate them. For producing rotation invariant moments, Hu moments have defined two distinct methods which are explained below. The first method is the principal axes which perceived that this approach can break down when images do not have novel principal axes. Such images are represented as being rotationally symmetric. The second method Hu moments explained the process of absolute moment invariants which is discussed and used here. Hu moments obtained these expressions from algebraic invariants which are applied to the function which is generating moment under a rotation transformation. They include a collection of nonlinear centralised moment expressions. The outcome is a set of independent orthogonal moment invariants that is rotational moment invariants, which can be used for scale, position, and rotation invariant pattern identification. These were used in a simple pattern recognition analysis to successfully recognise various transcribed letters. They are estimated from normalised centralised moments up to the third order that is shown below

$I_n$nth Hu invariant moment:

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})]$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (43)$$

These moments are of finite order. They do not constitute an entire set of image descriptors unlike centralised moments. Nevertheless, higher order invariants can be obtained. It should be recorded that this method also breaks down,

as with the method based on the principal axis for images which are rotationally symmetric as all the seven invariant Hu moments will be zero.

C. Training SVM Classifier

After the feature vector for each and every image is generated in the previous step, these feature vectors are fed to the SVM classifier. Before diving into as to how feature vectors are handled, we will brief about SVM.

A Support Vector Machine, in short, called as SVM is a discriminative classifier which generates a hyperplane that separates classes from one another. SVM is a supervised learning algorithm. When we give labelled data as a training set, SVM generates a hyperplane which classifies the classes. This hyperplane can be used to generate classes for new test examples. The representation of the SVM classifier is different in different dimensions. The SVM classifier is a line in two-dimensional space. This line divides the entire space into two sections where each side of the line has points belonging to a particular class. In case of multi-dimensional space, this line is a hyperplane which divides the entire space into two parts similar in the case of two-dimensional space. Either side of the plane has points belonging to a particular class. There might be outliers even. SVM for a simple dataset of points shown in figure 1 is shown in figure 2.

Fig. 1.

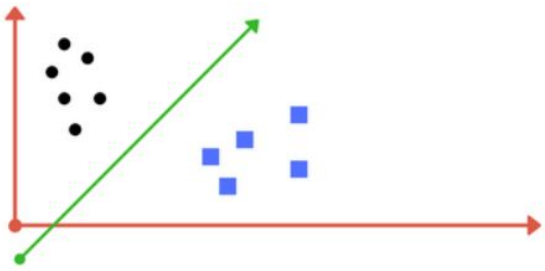

Fig. 2.

If data points are scattered in the plane as shown in figure 3, then SVM classifier is smart enough. Looking at the points one can clearly infer that a linear classifier cannot be drawn to separate the data points. The SVM classifier marks all the

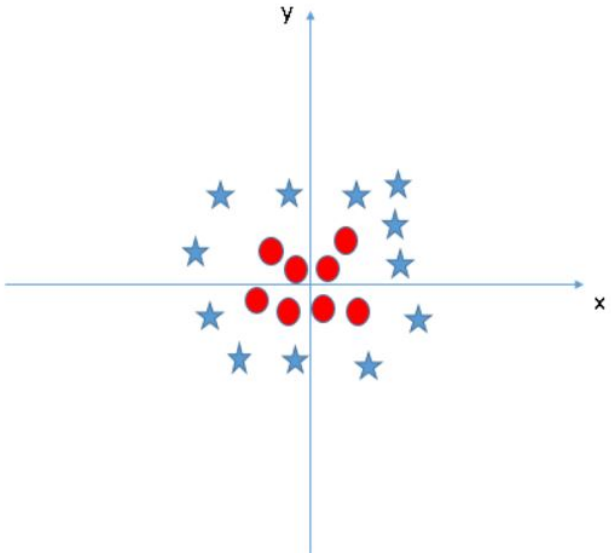

Fig. 3.

points in a higher dimension space. Here in this case it is 3 dimensional space. Figure 4 shows how the points look when plotted in 3 dimensional space.

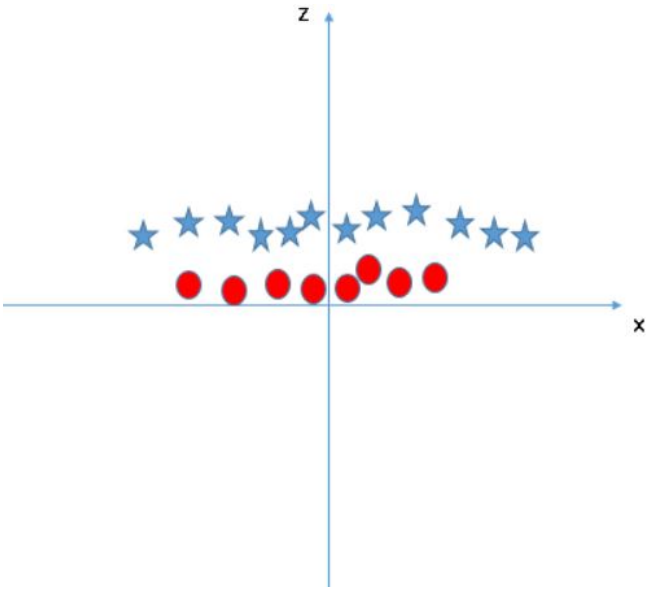

Fig. 4.

As we can clearly see in figure 4, we can draw a line that separates all data points, which indeed is a hyperplane.

The transformation of low dimensional data points to high dimensional points to obtain a classifier is called is kernel. If we transform the 3 dimensional spacial points back to 2 dimensional, we get a classifier which looks as shown in Figure 5.

If the points in the space overlaps, the svm classifier takes care of that cases even. Two possible classifiers can be drawn as shown in Figure 6 and Figure 7.

Figure 6 has some outlier points that is, it has some points

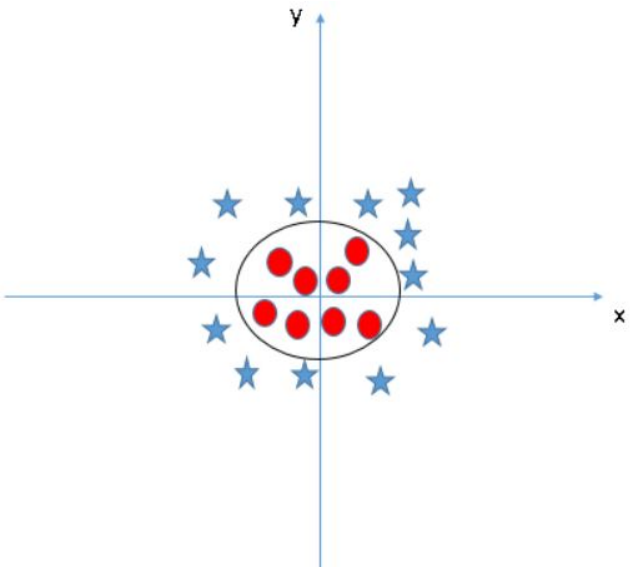

Fig. 5.

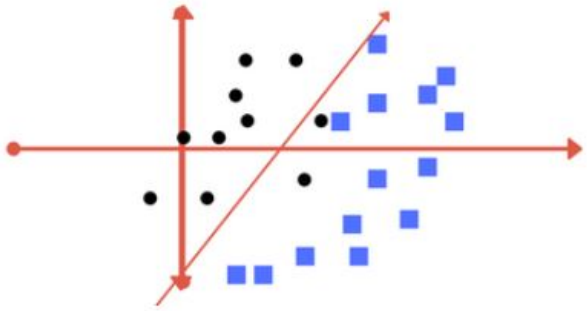

Fig. 6.

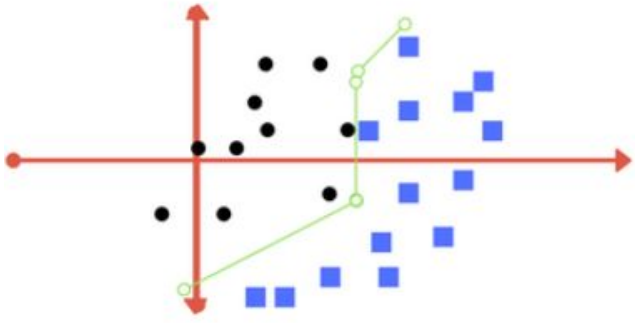

Fig. 7.

of a particular class lying in the other class where as in Figure 7 that classifier is over fitting the data. Over fitting the data means that the classifier is too perfect for the training set but might not work so perfectly for test set.

But, there is trade off. In real world application, finding perfect class for millions of training data set takes a lot of time. This is called regularization parameter. We further define two terms regularization parameter and gamma. These are tuning parameters in SVM classifier. Varying those we can achive considerable non linear classification line with more accuracy in reasonable amount of time.

One more parameter is kernel. It defines whether we want a line of linear separation.

Polynomial and exponential kernels calculates separation line in higher dimension. This is called kernel trick.

We don't have to manually calculate classifier every time. Python provides us with a library called as sklearn which has inbuilt function to execute the svm classifier.

III. METHODOLOGY

Initially, we take an image and then convert it to gray scale. That is each pixel intensity values lies between 0 and 255. Then this gray scale image is converted to binary image. For that, we have to first consider a threshold value. If the pixel value is more than that threshold value, we give the pixel value as 1 and if the pixel value is less than the threshold value we give the pixel a value of 0. After the binarized image is generated, we have to now obtain a feature vector for the shape of the person in the image. This will give us information about the posture. We then feed these feature vectors of images in the training set to the SVM classifier and train it.

For testing, we convert the test image to gray-scale image and then binarize it. Then for this binarized image we generate the feature vector. This feature vector is sent through SVM classifier which predicts the class label.

IV. RESULTS

We have trained our classifier following the steps as mentioned in methodology. Here in Figure 8, the first one is the original RGB color image. The next image is the gray scale image of the color image. The final image is the binarized image. Now, for the binarized image we obtain the feature vector Hu moments.

Fig. 8. Original, Gray-scale image, Binarized image

The values of seven Hu moments of this binarized image are shown in Figure 9. This is our feature vector.

```
array([  3.01271608,   7.09128941,  12.15183129,  10.93722971,
        22.64095196,  14.48326477, -22.62393189])
```

Fig. 9.

If we have to test this image, we have to send this to our classifier and this will predict the dance form to which it belongs to. As the Hu moments of the image closely match with the Hu moments of the Odissi dance form, hence, it is classified as Odissi dance form.

V. CONCLUSION

We have used few images in our training set. All the images are converted to gray-scale and then we have binarized them even. We have obtained feature vectors for each image and then trained our classifier using these vectors. For testing, we have repeated the procedure and then passed it to our classifier. We have obtained about 51.02% accuracy.

Though there are 8 various dance forms, there maybe some postures common among dance forms. Some times what may happen is after binarization it maybe difficult to understand which dance form it actually belongs to. So, our classifier may not classify dance forms correctly. This is one limitation to our project. There is another limitation that is when we use hazzy images classification becomes difficult.

REFERENCES

[1] Shubhangi, Tiwary U.S. (2017) Classification of Indian Classical Dance Forms. In: Basu A., Das S., Horain P., Bhattacharya S. (eds) Intelligent Human Computer Interaction. IHCI 2016. Lecture Notes in Computer Science, vol 10127. Springer, Cham

[2] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, Learning mid-level features for recognition. In CVPR, pages 2559 2566, June 2010.

[3] Niranjil Kumar A and Sureshkumar C, Background Subtraction in Dynamic Environment based on Modified Adaptive GMM with TTD for Moving Object Detection 2015, J Electr Eng Technol;10(1): 372-378

[4] Soumitra Samanta; Pulak Purkait; Bhabatosh Chanda. "Indian Classical Dance Classification by Learning Dance Pose Bases", Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, India.