Digital Image Processing

Project Report-2

Prateek Goel 201001067
Varshanjali Sayyaparaju 201030097
Aditya Singh 201002055

Problem Statement :

Goal: Given an image of a hand posture for an Indian classical dance form, identify the dance form and the mudra.

Purpose: Indian classical dance is a cultural art form which conveys meaning in every step. It is, however, not palatable to the untrained eye. To appreciate the art form, one needs to know the story behind every single movement. This project is an endeavor that will hopefully transcend into a tool that sends across the intended message clearly to anyone interested. We envision a tool that enables real time tagging of images and videos with the mudras. This project is merely a stepping stone, an initiation towards this goal.

Deliverables:

A packaged tool, complete with a graphical user interface, that lets the user upload an appropriate image illustrating any Indian classical dance form, recognizes the dance form and the mudra and provides a short description.

Minimum:

Constraints :

- The image focuses on the hands.
- The background is not distracting to the extent that it makes the hand region unrecognizable.

Maximum:

Constraints :

- The image could be a complete dance posture, in which case, the user must select the region of interest.

The story so far

Dataset:

Constraints: The dataset being considered was initially constrained to :

- Right hand Bharatnatyam mudras
- No embellishments
- No other objects (i.e plain background)

Implementation :

Sub-process 1 : Segmentation of hand

- Basic Thresholding: The first method tried worked well for the constrained dataset. However, on applying it to images that are to be targeted for the final deliverable, the results were not satisfactory.
- Skin Detection: The results obtained from skin detection using color segmentation were satisfactory even for the final dataset. However, there was scope for improvement.

Sub-process 2 : Boundary Extraction

Boundary extraction was performed by region filling the extracted binarized hand and find the external boundary using morphological operations.

Sub-process 3 : Detection of fingertips

Fingertip detection was accomplished using distance metrics. The Euclidean distance of each point on the boundary from the center of the hand was calculated and points that lay above a threshold value were classified as fingertips.

Improvements to Interim 1

Sub-process 1 : Hand Detection

Method

- Thresholding in the RGB plane
- Thresholding in the YCrCb plane
- Thresholding in the HSV plane
- At a location, if the pixel value in either of the color spaces is 255, then that location is given a pixel value of 255, else 0.

Changes in Method:

- The previous code was segmenting the image using the Cr Cb from YCrCb color model and the H plane in the HSV, but in this case we are using Cr, Cb, H,S,V, and R,G,B planes.
- In the previous code, we were stating that if the image's H, Cr, and Cb pixel values are above their respective thresholds, then the pixel is kept or else made to zero. In the new algorithm, we are thresholding in each color model seperately, and then checking if in either of the color spaces, the pixel passed the threshold.

Results:

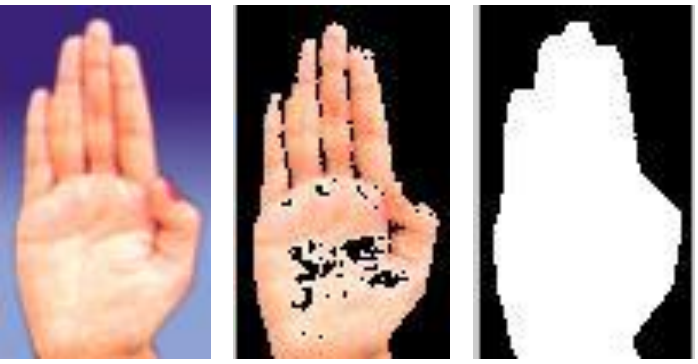


Figure 1

Figure 2

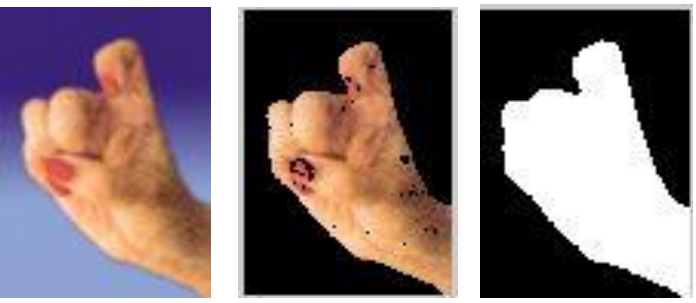


Figure 3

Figure 4

Observations:

- The unfilled regions obtained for the first three figures are lesser compared to the results obtained previously.
- Figure 4 was not segmented at all initially. In the improved algorithm, we have an output which might not look satisfactory visually, but is enough to the serve the purpose of giving us the shape of the hand.

Conclusions:

- Segmenting the RGB plane helped in skin detection. Observing that the background was shades of blue, the range allowed to pass in the b-plane was less. Such modifications can be made for images with a fixed background. Also, observing that the hand color mainly varies from pink to white, increasing the range in those planes helped give better results.
- Segmenting in the S plane of the HSV color model helps since a variety of shades of skin colour could be present, like in Figure 3. Allowing the range of saturation to vary significantly improved results.
- The idea of thresholding in the V plane was used because the amount of light falling on the hand varies. However, it needs to be separated from the light background at the bottom end of the image, which required trying several combinations of threshold values for the V plane and the H and RGB planes.

<u>Sub-process 2 : Boundary Extraction</u>

Method:

- This process of the pipelines takes a binarized image of the extracted hand as input.
- Detects edge pixels by checking if atleast one of the 8-neighbourhood pixels is zero.
- Reduces the boundary to one pixel width by using M-connectivity. The inspiration for this step was to derive an image that works well with the k-curvature algorithm for the next subprocess – fingertip detection.

Changes in Method:

- Initially morphological operations were being used to find the external boundary of the extracted hand. However, because of the change in method for fingertip detection, it became necessary to obtain a boundary that's one pixel wide, which is why neighbourhood operations were considered.

Results:

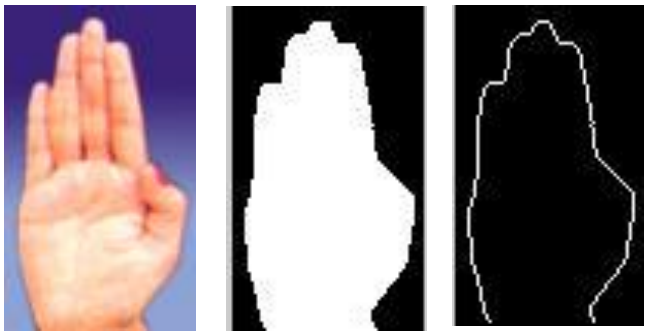


Figure 5

Figure 6

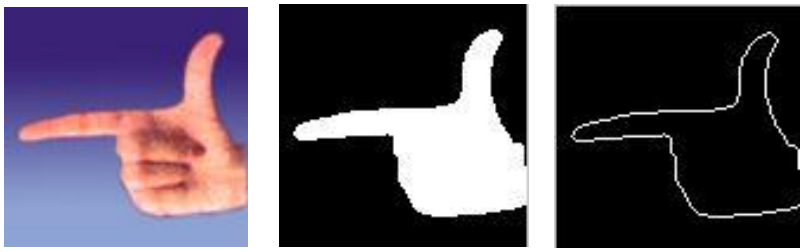


Figure 7

Figure 8

Observations:

- The cracks found in the boundary image are gone, and it is a smooth border as can be seen clearly in Figure 7 and 8. The advantage of using a 1px wide boundary is that every pixel has exactly 2 neighbours and there is only one connected component, that of the hand. This information is useful for calculating k-vectors in the algorithm for fingertip detection.

  Using morphological operations, the shape of the hand got distorted, while using neighbourhood processing the shape of the hand is preserved, as evident in Figure 6 and 7.

- The peaks and values caused by the different orientations of the hand are more prominent using neighbourhood processing, as in Figures 6 and 8.

Conclusions:

- Using morphological operations is not the best option when the shape of the hand keeps changing, because the choice of structuring element is a factor that could effect the output
- Using neighbourhood processing is guaranteed to work for any shape since we traverse the image pixel by pixel.

Subprocess 3 : Finger-tip detection

Method 1(Previous): Calculate the Euclidean distance from center of hand to each point on the boundary. When the distance exceeds a threshold, it is detected as a finger.

Results:


Figure 9

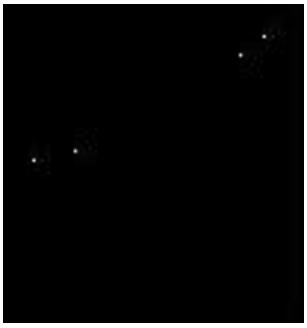



Figure 10

Observations: The method worked only for about 40% of the cases. Figure 9 shows a case where it worked and Figure 10 is one where it didn't.

Method 2: K-curvature

Method:

- Take a binarized boundary image as input.
- For each point on the boundary, calculate a vector with end point at a distance of K pixels from the current pixel. Repeat this for both neighbours (each point has only two neighbours after reduction to 1px width) to obtain two vectors for each point on the boundary.
- Calculate cos(theta) where theta is the angle between the two vectors. For points lying on the peaks and valleys of the hand, this value should be positive since the angle is likely to be acute. For other points, the value should be negative as the angle grows above 90 degrees.
- Consider the top 30 values of cos(theta) and use the corresponding points as peak and valley points.
- Reduce each cluster of points to exactly one point to obtain relevant peaks and valleys.

Results:

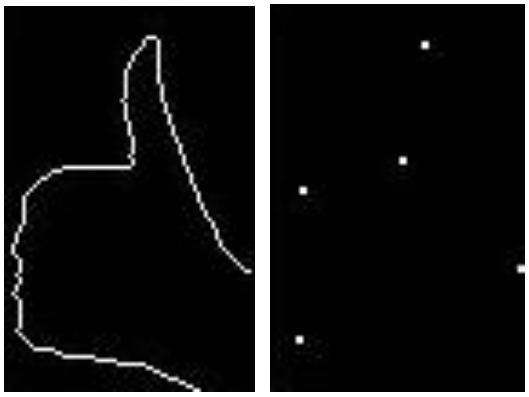


Figure 11

Figure 12

Figure 13

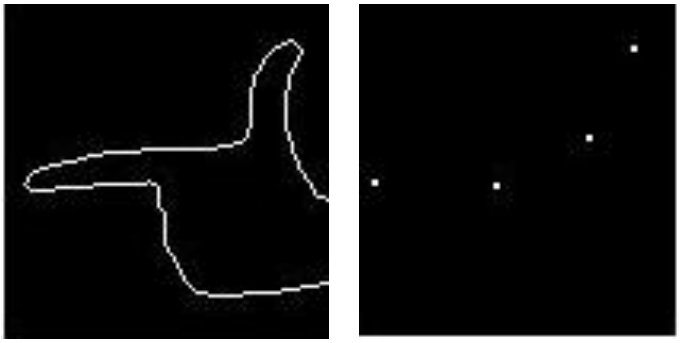


Figure 14

Observations:

- Results include not only location of fingertips but the valleys as well. This is not a problem. In fact, it only strengthens the feature to be considered for recognition.
- There are certain cases where this algorithm won't work perfectly. However, since eventually the goal is to achieve maximum correlation and not perfect correlation, this problem can be neglected without any compromise on results.
- Experimenting the value of k and the no. of points to be considered, most datasets can be very well accounted for.

Additions to Interim 1




Process Overview :

- Segment out hands
  - Skin detection
  - Edge-detection
    - Sobel
    - Prewitt
    - Gradient
  - Color based segmentation using K-means clustering
- Distinguishing between hands
  - Boundary Extraction
  - Watershed Transform

Constraints : Two hand Bharatnatyam mudras with a reasonably complex background.

Segmentation of hands :

Skin detection:

Using the same method on the new dataset :

Figure 15

Figure 16

Figure 17

Observations:

- Skin detection is working fine for cases when there isn't color similar to skin in the background. As can be seen in Figure 16, the background wall is brown, which is similar to the skin color. Therefore skin detection is better in Figure 15 where the background is less obstructive.
- In image 17, it can be seen that the color of the sari is very close to that of skin, so it is not being segmented out.

Conclusions:

- Segmentation of two hands in a complex background is difficult, because the background might have colors close to skin.
- Setting the appropriate threshold is hard, because the skin color will vary depending on various lighting conditions. However, there is some room for error since the algorithm has to work well enough so as to provide a good boundary image.

Edge-detection:

Looking at the dataset, it can be seen that the hands are the most prominent in the image. And considering that they are the area of interest in the image, the lighting conditions must have been chosen to highlight the hand. Taking this assumption, we thought if we tried edge detection, and set an appropriate threshold, we should be able to get the boundaries of the hand.

Theory:

Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction, which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.

The gradient operator is sensitive to local gray level changes and is therefore a convenient tool to detect edges. If the magnitude of the gradient image is larger than a threshold value, the pixel can be considered as on an edge.

Mathematically, for an image function, f(x,y), the gradient magnitude, g(x,y) and the gradient direction, (x,y) are computed as

$$g(x,y) \cong (\Delta x^2 + \Delta y^2)^{\frac{1}{2}}$$

$$\theta(x,y) \cong atan\left(\frac{\Delta y}{\Delta x}\right)$$

where,

$$\Delta x = f(x+n, y) - f(x-n, y) \quad and \quad \Delta y = f(x, y+n) - f(x, y-n)$$

and n is a small integer, usually unity. For example, the simplest implementation of this would be to convolve the following mask with the image data, aligning the mask with the x and y axes to compute the values of delta x and delta y.

-1	0	1

The Prewitt operator is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Prewitt operator is either the corresponding gradient vector or the norm of this vector. The Prewitt operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define as the source image, and and are two images which at each point contain the horizontal and vertical derivative approximations, the latter are computed as:

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

The Sobel operator is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high frequency variations in the image.

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define A as the source image, and Gx and Gy are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

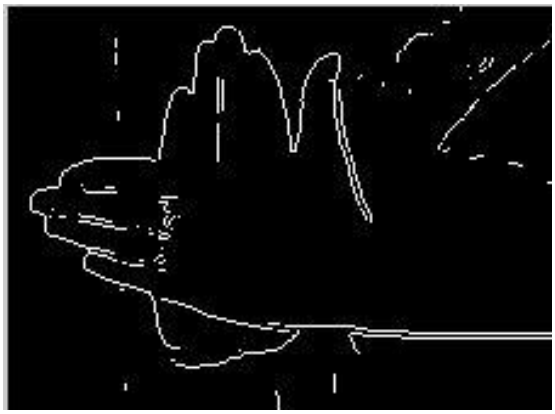
Method:

- Decompose an image into column and row vectors
- Apply filter in each direction
- Find proper threshold to eliminate intensity changes below threshold using hit and trial.

Result:

Gradient

Sobel (threshold = 0.09)



Prewitt (threshold = 0.09)



Observations:

- For each image, trying to find the edges of the hand are difficult, because threshold needed keeps changing.
- The output images for Sobel and Prewitt operators are similar.
- The gradient method is not useful, because the edges detected are not as defined as expected.
- The boundaries are not as "full" as we need them to be to carry out further operations.

Conclusions:

- Gradient method is not a good method for the given dataset.
- The hands in the image can be segmented out, but for each image a different threshold is needed, which makes it a hard problem to solve.
- The edges detected don't give the boundary required, so there is a need to try other methods.

Color based K-means clustering:

Theory:

K-means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids shoud be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we

may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point and the cluster center , is an indicator of the distance of the n data points from their respective cluster centers.

A Lab color space is a color-opponent space with dimension L for lightness and a and b for the color-opponent dimensions, based on nonlinearly compressed CIE XYZ color space coordinates.

$$\begin{aligned} L^\star &= 116 f(Y/Y_n) - 16 \\ a^\star &= 500 \left[f(X/X_n) - f(Y/Y_n) \right] \\ b^\star &= 200 \left[f(Y/Y_n) - f(Z/Z_n) \right] \end{aligned}$$

where

$$f(t) = \begin{cases} t^{1/3} & \text{if } t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases}$$

Method:

- Convert the image into the LAB color space
- Classify the colors in planes A, and B space using K-means clustering
- Label every pixel in the image using the results from the K-means
- Create Image that segments the image by color
- Identify which image contains the hand, by using skin-detection

Results:

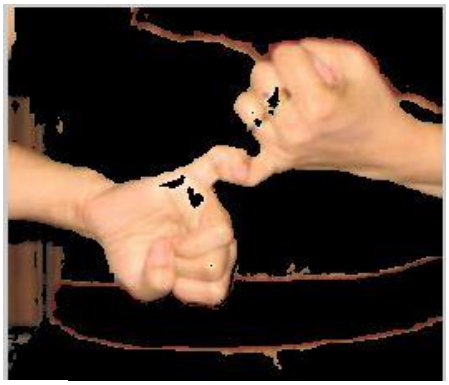



Figure 19


Figure 20

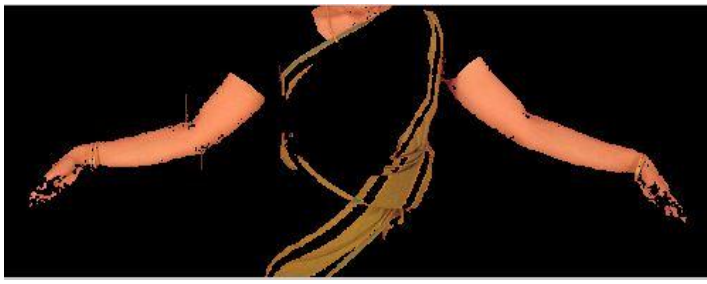



Figure 21

Observations:

- The results are pretty good, expect for the small regions in the image which have a color similar to skin.
- In Figure 19, it can be seen that the sari is not detected, as it was during skin detection.
- For the cases in which the neck is being detected, we could make use of the fact that the hand regions are likely to be the largest regions in the image. Therefore, we could erode images smaller than a particular threshold size.

Conclusion:

- K-means clustering is helpful in seperating the hands from the rest of the image, but it still has the same effect that skin detection had, which might create a problem.
- Comparitively, this gives better results than skin detection, because the sari which was detected in skin detection is not considered as skin.

Distinguishing between hands:

Boundary Extraction:

Method:

We thought we would apply boundary extraction as done for the first dataset.

Results:

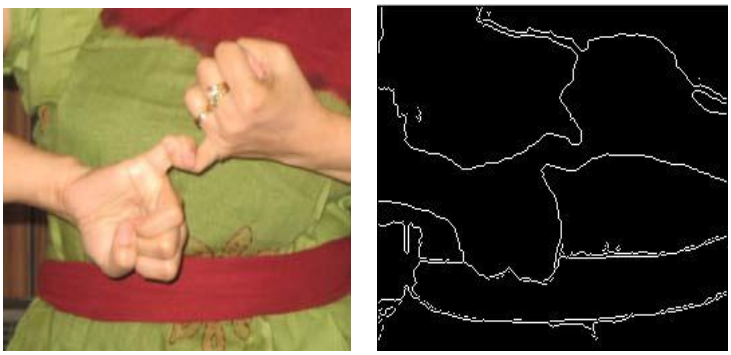


Figure 22

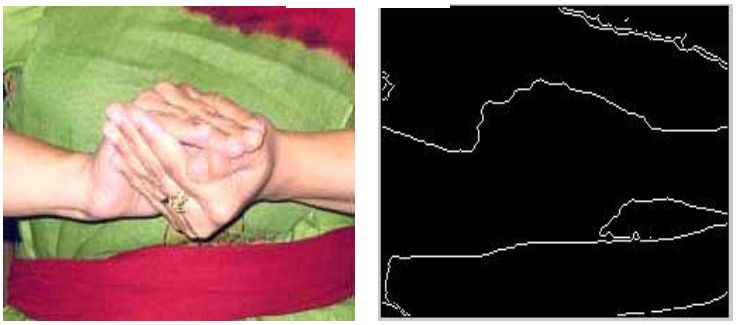


Figure 24

Figure 23

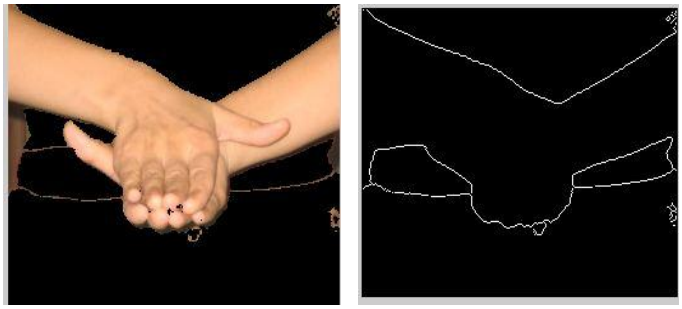


Figure 25

Observations:

- Since the image was not segmented properly, overlapping boundaries are getting mixed up, as in Figure 22, 24 and 25
- Individual hands are lost in the binarization, as in Figure 24 and 25
- Segmentation of non-overlapping hands is good, as in Figures 22 and 23.

Conclusion:

- Boundary extraction cannot work for distinguishing between the two hands, because when the hands overlap, the data of each hand is lost. Therefore, another method must be tried.

Watershed transform (To be implemented) :

Theory:

The term watershed refers to a ridge that divides areas drained by different river systems. A catchment basin is the geographical area draining into a river or reservoir.

Any greytone image can be considered as a topographic surface. If we flood this surface from its minima and, if we prevent the merging of the waters coming from different sources, we partition the image into two different sets: the catchment basins and the watershed lines. If we apply this transformation to the image gradient, the catchment basins should theoretically correspond to the homogeneous grey level regions of this image.

Method:

We had a problem with overlapping hands in the previous method, so we thought watershed transform will help us over this problem

- Find gradient of image
- Mark the foreground objects, using morphological operations
- Similarly, mark the background objects, by looking for similar pixels that are not connected.
- Compute watershed transform

Segmentation of decorated hand:

Constraints:

- Plain background
- Hand decorated with mehendi, and bangles.

Method:

- Use color segmentation to find finger tips. This step is inspired from the fact that in most dance forms, mehendi is applied only on the fingertips and is usually red in colour.
- Use morphological operations to get rid of unwanted detections.

Results:


Figure 25


Figure 26


Figure 27

Observations:

- When doing color detection for Figure 28, the mehendi in the center will get detected too, but since it is bigger in size than the fingers, it can be removed.
- The bangles that were identified were not removable because the size is same as the fingers for Figure 28. However, it does not create a problem in the other two images.
- Segmentation of overlapping fingers is not a problem as can be seen in Figure 27.

Conclusions:

- Segmentation of hands in a plain background is not difficult, unless the angle the image is taken causes problem.
- Finger positions are accurately found.

Problems faced:

- The most enduring problem we are faced with is that of a complex background which has colours similar to that of skin. If the extra objects detected form separate components, they could probably be removed. However, if the skin coloured objects intersect the hand, segmentation of the hand will be extremely difficult. The current idea is to let the user choose an area where the hand lies. This would reduce the margin for error greatly.
- Implementing the Watershed transform for the current dataset does not seem very promising. However the problem of overlapping fingers is an important one.
- Distinguishing between the two hands is another issue that will create problems in the recognition phase. This is because the relative position of fingers will have a different order for both hands.

Future Work :

- ✓ Improved segmentation for two hand mudras
- ✓ Implementation of the Watershed Transform
- ✓ Segmentation of decorated hands relaxing more constraints
- ✓ Pattern recognition