Real Time Hand Tracking and Gesture Recognition

Digital Image and Processing,
MONSOON 2011,

Team Members:
Shrenik Lad, Shaik Allauddin
Introduction

- **Aim:**
  To track hands of a person standing in front of a webcam and recognise hand gestures through which he can interact with the machine.

- **Application:**
  Human Computer Interaction, Immersive Technologies, Interactive applications

The above diagram shows a game which is played through human gestures.
Basic Gestures

START

MOVE

LEFT

RIGHT
Basic Gestures

- Back
- Front
- Stop
- No Hand
The steps involved in the entire process are as follows:

1. Skin color Learning
2. Segmentation
3. Morphological Operations
4. Connected Components and Hand Detection
5. Modeling the Hand
6. Gesture Recognition
In this stage, we learn the hand colour of person standing in front of the webcam. The person places his hand in the learning box on the screen, and the pixels in learning box are captured.

The pixels are converted to YCrCb space and using the (Cr, Cb) values in the learning box, the mean and variance values of Cr and Cb are calculated.
Segmentation

- Using the Mean 'M' and Variance 'V' values calculated above, we perform segmentation on entire frame.

A probability image is generated, in white a white pixels means a strong probability of being hand-pixel.

Other skin coloured regions are also identified as hand-regions.
Morphological Operations

- To enhance the output of segmentation process, we apply morphological operations on the image to obtain blobs/regions.

We apply DILATION on the probability image with a 9 x 9 rectangular mask.

After that, EROSION is applied with a smalled rectangular mask of size 5 x 5.

RESULT: - Separate Blobs
Labeling Connected Components

- We apply Connected components labelling on the above image to label the various blobs present in the image.
- Once the blobs are labelled, we find the blob which represents the hand.

A bounding box is drawn around the hand blob.
We are able to track the hand as a blob in every frame. Now, we need to model the hand and find some parameters which will help us to recognise the state of hand.

Parameters:

- Centroid: The centroid \((p_x, p_y)\) of the blob (hand)
- Angle: The orientation/angle made by the blob (hand) with the horizontal.
- Depth: A measure of closeness of hand with the camera.

\[ m : \text{minor axis length} \]
\[ M : \text{major axis length} \]
\[ (p_x, p_y) : \text{position of hand} \]
\[ \alpha : \text{orientation of hand} \]
Centroid, Angle and Area

- Centroid of the hand is simply the centre of mass of the hand – blob. (mean of all [x,y] in the blob)

- Area of the blob can be taken as number of pixels in the hand – blob.

- For calculating Angle, we need to first find the contour of the hand. Once the Contour is obtained, there are several ways to find angle
  - Find the point P on the perimeter of contour which is farthest from the centroid C. The angle made by the line joining C and P, with horizontal is the required angle.
  - Find the expectation (average) of the vectors joining points on the perimeter of contour and the centroid. The angle made by this vector with horizontal is required angle.
Contour and angle of Open Hand

Angle is drawn in green colour and Contour is drawn in Blue colour
Tracking Hand between Frames
Hand Tracking – (Centroid based)

- We need to track the hand between frames when multiple hands are present or background changes occur.
- We use a constant velocity model in which,
  - $P(t+1) - P(t) \approx P(t) - P(t-1)$
  - The distance covered by hand between 2 frames would not vary much.
- In any frame, if there are $N$ blobs in segmentation output ($b_1, b_2, b_3, \ldots, b_N$), we chose the blob whose centroid is nearest to previous frame’s centroid.

Results:

Here, we can see 2 hands in the image. The second hand has greater area but the first one is detected.
Gesture Recognition
Gesture State Machine
Finding the hand's contour and its convex hull can help us…

- From the contour and its convex hull, we can find contour points between 2 consecutive convex hull vertices.
- This sequence forms the convexity defect.
- It is possible to find depth of $i^{th}$ convexity defect

\[
\bar{d} = \frac{1}{n} \sum_{i=0}^{n} d_i
\]
Finding Convex Hull

- How to find Convex Hull of the Contour Polygon?
- **MELKMAN Algorithm**
  - Constructs Convex hull of a simple Polyline.
  - Given N points of the Contour Polygon, the Convex Hull can be obtained
  - An O(N) Algorithm

- Result:

  The convex hull of hand’s contour is shown in white color
The Average Depth of Convexity defects in START gesture is much more than in MOVE Gesture.

\[ \text{Avg}_\text{depth} (\text{START}) = 3 \times \text{Avg}_\text{depth} (\text{MOVE}) \approx \]

Thresholding the difference in average depth recognizes the Start \( \rightarrow \) Move transition

Result:

![The START Gesture]
As we can see, the depth of convexity defects is much greater in START gesture than in MOVE gesture.
The MOVE Gesture

- Once the MOVE Gesture is recognized, we take average of Hand parameters over N frames.
- These parameters become the REFERENCE parameters.
  - Reference Centroid (px, py)
  - Reference Area ‘A’
  - Reference Angle / ZERO Angle

- Using these Reference Parameters, FRONT, BACK, LEFT, RIGHT, NEXT, PREVIOUS Gestures can be recognized.

The Hand parameters in this state become the REFERENCE parameters
FRONT and BACK Gesture

- We recognize FRONT and BACK gestures by comparing the Area of Hand with the reference area ‘A’.
- If (current area – ‘A’ > 0.3 * A), FRONT gesture
- If (‘A’ – current area > 0.3 * A), BACK gesture
- Results:

The Area is greater than reference in FRONT whereas smaller in BACK.
LEFT and RIGHT Gesture

- We recognize LEFT and RIGHT gestures by comparing the Angle of Hand with the reference angle ZERO.
- If (current angle – ‘ZERO’ > 30°), RIGHT gesture
- If (‘ZERO’ – current angle > 30°), LEFT gesture
- Results:

![LEFT gesture example](image1)

![RIGHT gesture example](image2)
NEXT and PREVIOUS Gesture

- We recognize NEXT and PREVIOUS gestures by comparing the Centroid of Hand with the reference Centroid \((px, py)\).
- If \((\text{centroid}_x - px > \text{threshold})\), NEXT gesture
- If \((px - \text{centroid}_x > \text{threshold})\), PREVIOUS gesture
- Threshold can be taken as \(W/4\), where \(W\) is image width
- Results:

PREVIOUS

NEXT
STOP Gesture

- STOP Gesture can be done only after MOVE gesture.
- We use the HU moments to recognize STOP gesture.
- Results:
Constraints

- **Illumination**: The light conditions in the background will affect the segmentation process and may not detect hands correctly. When shadow falls on hand regions, those hand regions get clipped. Hence, sufficient lighting must be present.

Because of shadow falling on finger regions, the fingers get clipped and segmentation output is not good.
Constraints

- **Red Colored Objects in background:** Since human skin color is very near to red color, it is difficult to learn the distribution perfectly. This leads to errors in segmentation and in case of a user wearing a red shirt, it is very difficult to detect the hand.

It is clear that red objects are also detected in segmentation. When objects are bigger, hand cannot be detected.
Constraints

- **FPS of the algorithm**: Since the images are received in real time, the fps given by the algorithm determines the performance of hand tracking. We have optimized the algorithm so that time taken to process one frame is minimum. Small lag can be seen between user’s action and the response generated on screen.

  (Possible Solution: Parallel programming …)

- Most of the steps used in the algorithm are $O(N)$ where $N$ is the number of pixels in image.

- Connected components labelling involves BFS in the image, and hence may take some time.

- Profiling the code and then optimizing the CPU intensive steps can be done.

- Required FPS around :- 30 fps, **Persistence of Vision**
A small application ……

- Consider an image viewer controlled through hand gestures

- Define operations on a image for particular Gestures :
  - **START** gesture : starts the image viewer
  - **LEFT** gesture : left rotates the image with some angle theta
  - **RIGHT** gesture : right rotates the image with some angle theta
  - **FRONT** gesture : ZOOM IN the image upto particular level
  - **BACK** gesture : ZOOM OUT the image upto particular level
  - **ONE** gesture : show first image
  - **TWO** gesture : show second image
    and so on……
Conclusion

- We have presented a real-time system to track and recognise hand gestures for human-computer interaction.
- The algorithm is based on hand segmentation, hand tracking and gesture recognition from extracted hand features.
- This low cost interface can be used by users as a back-end for several applications. Some common applications include,
  - Controlling mouse through hand gestures.
  - Gesture based Image viewer.
  - Interactive Games.
Reference Papers

THANK YOU