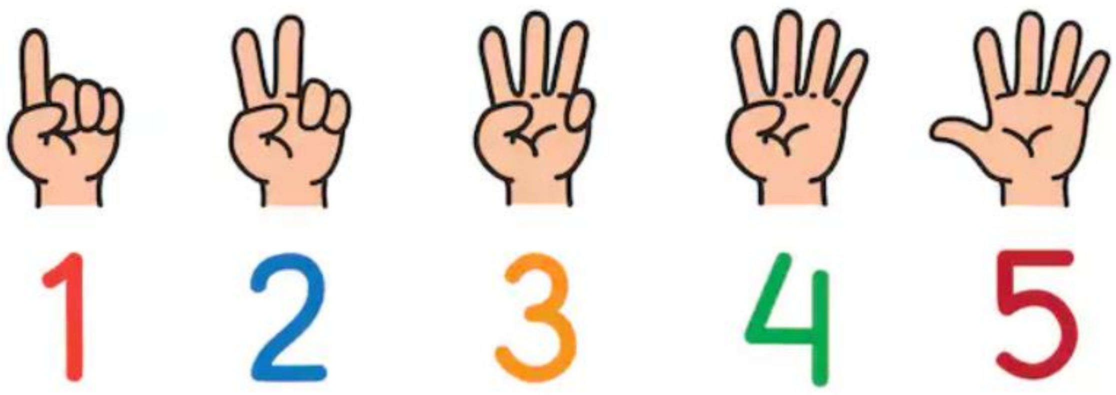
Image Processing (UCS615) Project Report

(B.E 3rd Year May 2018)



Fingers Count

Submitted by:

Abhishek Sharma – 101503008

Group: COE-1

Submitted to:

Dr. Jhilik Bhattacharya

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Index

Introduction:

Technology has always developed in the direction to simplify our day to day life. Today we are developing towards creating technology even more easy to use. Human Computer Interface (HCI) is the term used to refer the technologies that have been developed for interacting with machines. Till date, a number for hand gesture base HCI have been introduced. Yet here, we want to just add a bit of ease by bringing up an easy to use algorithm that uses computer vision to count fingers of our hands. In an intermediate step we also calculate orientation of our hand which itself can be used for interacting with computer.

Method:

I can sub-divide the whole procedure into small manageable fragments described as under.

1. Hand Extraction

2. Scan for lowest valley

3. Create separate images by finding a split line at lowest valley point

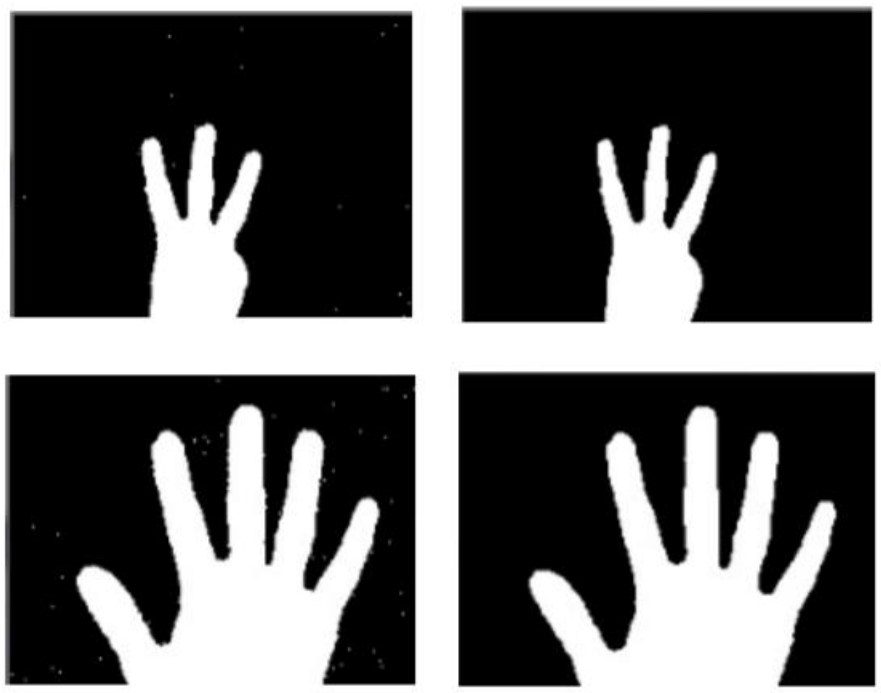
4. Recursively find centroid and orientation of each images and again scan for lowest valley

5. Under no valley condition check if it is a finger

1. Hand Extraction

Standard image from the camera consists of coloured pixels in usually RGB format, each of which has a value ranging from 0 to 255 for each of red, blue and green values. This data is stored in a matrix form where each element of the matrix represents a pixel or picture element. Hand extraction is done by background elimination. It is assumed that the first fame contains only static background. There after the subsequent frames are checked for changes with respect to the first background frame. We use Gaussian blur of 5x5 on the background so as to remove bit of noise and detect sharp changes. We construct a binary image (black & white image) out of the hand that comes over the background. Hand is represented in white on the otherwise black background. We use linear blur of 3x3 and threshold the image such that at least 7 pixels of the concerned 9 pixels have to be white to consider the central pixel to be white. This is a strong way to ensure that no noise speckles arise unless the image is badly affected by noise due to significant movement of camera or in background. This also makes the valleys more prominent and thus they are easier to locate.

The matrix so formed is further reduced by a factor of 3 by spatial sampling in both coordinates to form a much smaller matrix which gives similar image with reduced dimensions. The idea is to replace the bulky matrix with a smaller one so that the calculations can be done in a much faster way involving fewer iterations. The factor by which the matrix is reduced is a point of trade between speed of the algorithm and the distance of the hand from the camera till where it can detect individual fingers correctly.



2. **Scan for the lowest valley**

We start scanning the image upwards from the base line. we find a valley point. Here, we define valley point as spatial mean of the black pixels lying between two white regions. We scan the image along the slope of the base line and for scanning the next upper line we change the y intercept of the line. Thus, we scan the image along a non-conventional axis rather than conventional x-y axis.

The minimum width of the white regions can be fixed according to the expected maximum distance of the hand from the camera. Also, the minimum width of the black region can also be fixed in order to take the minimum spread of fingers in consideration. If minimum width of the black and white regions is not satisfied, then it is not considered as a valley point and scanning for valley point is continued. While scanning for a valley point, we turn the scanned pixels to black till we meet the first valley point.

3. Finding split line

Upon getting a valid valley point, we must have two disjoint matrices of white pixels. Here, we wish to draw a line such that the two disjoint matrices lie on either of the line. We refer this line as split line, which is a straight line passing through the valley point and has no white pixels on its locus. We find such a line by scanning along the line through valley point starting with the angle 180º and decreasing it gradually down to 0º. We stop scanning as soon as we find a split line. If we do not find any split line upon full scanning, then we reject that valley point and again start scanning for new valley point.

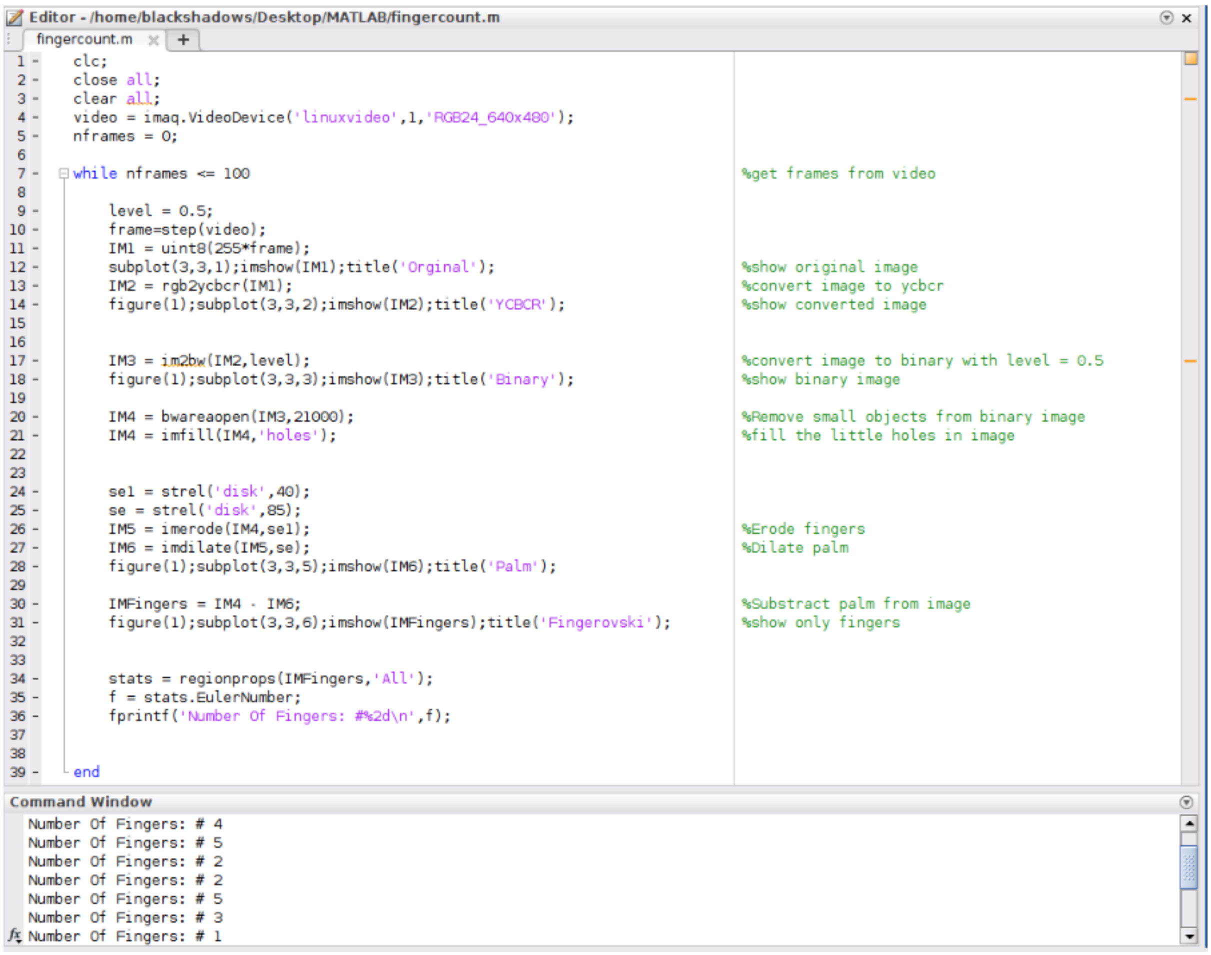
4. Going through recursion

Once we get the two separate images, we find the centroids of each of the images. Then we calculate new base line for each of the images. For calculating the new base line, we use the base line of the parent image passing through the valley point that produced the separate images and we choose a line slightly higher to it by changing its y-intercept. Along this line we find centroid (Cn). The line passing through the valley point and perpendicular to the line joining the centroid of the image and Cn becomes the base line for the new image. Now we scan for valley points in these images. We stop scanning once we cannot find any more valley point.

5. Identifying Fingers

Once we get an image in which no more valley point is present, we need to check if that image represents a single finger or not. At first we reject the images that do not have the certain threshold number of pixels. This threshold number is calculated by squaring the value that was used for minimum number of white pixels in the white region for finding the valley point. Of the remaining images, we find the number of white pixels lying on the locus of the line parallel to the base line and passing through the centroid. Let the number of pixels found be 'W'. Next we find the number of white pixels lying on the locus of the line perpendicular to the base line and passing through the centroid. Let the number of pixels found be 'H'. We found that we got good results for H/W ratio of more than 1.3.

Code:


Editor - /home/blackshadows/Desktop/MATLAB/fingercount.m

fingercount.m

```
clc;
close all;
clear all;
video = imaq.VideoDevice('linuxvideo',1,'RGB24_640x480');
nframes = 0;

while nframes <= 100                                                          %get frames from video

    level = 0.5;
    frame=step(video);
    IM1 = uint8(255*frame);
    subplot(3,3,1);imshow(IM1);title('Orginal');                              %show original image
    IM2 = rgb2ycbcr(IM1);                                                     %convert image to ycbcr
    figure(1);subplot(3,3,2);imshow(IM2);title('YCBCR');                      %show converted image


    IM3 = im2bw(IM2,level);                                                   %convert image to binary with level = 0.5
    figure(1);subplot(3,3,3);imshow(IM3);title('Binary');                     %show binary image

    IM4 = bwareaopen(IM3,21000);                                              %Remove small objects from binary image
    IM4 = imfill(IM4,'holes');                                                %fill the little holes in image


    se1 = strel('disk',40);
    se = strel('disk',85);
    IM5 = imerode(IM4,se1);                                                   %Erode fingers
    IM6 = imdilate(IM5,se);                                                   %Dilate palm
    figure(1);subplot(3,3,5);imshow(IM6);title('Palm');

    IMFingers = IM4 - IM6;                                                    %Substract palm from image
    figure(1);subplot(3,3,6);imshow(IMFingers);title('Fingerovski');          %show only fingers


    stats = regionprops(IMFingers,'All');
    f = stats.EulerNumber;
    fprintf('Number Of Fingers: #%2d\n',f);


end
```

Command Window

```
Number Of Fingers: # 4
Number Of Fingers: # 5
Number Of Fingers: # 2
Number Of Fingers: # 2
Number Of Fingers: # 5
Number Of Fingers: # 3
Number Of Fingers: # 1
```

Conclusions:

I intended to develop a very easy algorithm and I was successful in my endeavour. The algorithm has great amount of flexibility and thus can be used for numerous applications with accurate results.

I proceeded with few assumptions. Firstly, I assume that the hand is shown from bottom of the image such that fingers lie on relatively upper part of the image with respect to the rest of the hand. Secondly, I assume that whenever the hand comes into view, it shows fingers and not a closed fist.

I successfully used the algorithm to count up to ten fingers under the resolution of 640x480. If not constrained by the resolution of the camera we can have any number of hands for finger counting.

It can primarily be used for Human-Computer Interaction. For example, controlling mouse cursor, slide shows and can even be used for controlling a robot. By using our two hands we can trigger 10 unique commands and by changing the orientation of our hands we can provide even more commands like changing the direction of movement of a robot or speed control of robot.

Acknowledgement:

I would like to thank Dr. Jhilik Bhattacharya for her immense support and encouragement. I profoundly grateful to her for her constant guidance and support whenever I required and without whom this project would never be possible. I would like to thank my parents and teachers whose very existence mattered significantly for my success.

References:

[1] Ankit Gupta and Kumar Ashis Pati, A Project on Finger Tips Detection and Gesture Recognition, Indian Institute of Technology, Kanpur. November 12, 2009

[2] Gary Bradski and Adrian Kaehler, September 2008, Learning OpenCV, O'Reilly, Shroff publishers and distributers.

[3] Shaikh Shabnam Ahmed, Shah Aqueel Ahmed and Sayyad Farook Bashir, (2013) International Journal of Engineering Research & Technology (IJERT), Vol. 2, Issue 10, October 2013

[4] V. Bansal, S. Singh, H. Arora and S. Tiwari, Project on Simple Cloth Modeling and Control using Finger Count, Jaypee Institute of Information Technology, 2012

[6] Daeho Lee and SeungGwan Lee, Vision-Based Finger Action Recognition by Angle Detection and Contour Analysis, ETRI Journal, Vol. 33, Number 3, June 2011

[7] Mokhtar M. Hasan and Pramod K. Mishra, Novel Algorithm for Multi Hand Detection and Geometric Features Extraction and Recognition, International Journal of Scientific & Engineering Research Vol 3, Issue 5, May 2012