Hand Tracking And Guesture Recognition

Lakshminarayanan(s-ln.in)

***Abstract*—The aim of the project was to device a program that is able to detect out hands, track them in realtime and perform some guesture recognition. It is do be done with simple signal processing performed on images obtained from a regular laptop web-camera.**

I. Introduction

THIS paper elucidates a method to perform hand tracking in a mostly static environment with only the hands in the range of visiblity using OpenCV. We apply background elimination using simple techniques, and apply a set of filters to get the hand's contour, on which we create a convex hull and calculate the defect points in the hull which is used to detect the no of fingers , center and size of the palm which is a challenge.

II. Methodology

The following are the steps taken to detect and tract the hand:

A. Background Construction

Given the feed from the camera, the 1st thing to do is to remove the background. We use running average over a sequence of images to get the average image which will be the background too.

Fig. 1. The running average image used for background subtraction

So as and when we keep receiving image, we construct the new background average image as

$$CurBG[i][j] = \alpha CurBG[i][j] + (1-\alpha)CurFrame[i][j] \quad (1)$$

This equation works because of the assumption that the background is mostly static. Hence for those stationary item , those pixels arent affected by this weighted averaging and $\alpha x + (1-\alpha)x = x$. Hence those pixels that are constantly changing isnt a part of the background, hence those pixels will get weighed down. Hence the stationary pixels or the background gets more and more prominent with every iteration while those moving gets weighed out. Thus after a few iterations , you get the above average which contains only the background. In this case , even my face is part of the background as it needs to detect only my hands.

B. Background Subtraction

A simple method to start with is we can subtract the pixel values.However this will result in negative values and values greater than 255, which is the maximum value used to store an integer. Instead we use an inbuilt background subtractor based on a Gaussian Mixture-based Background/Foreground Segmentation Algorithm[1].Background subtraction involves calculating a reference image, subtracting each new frame from this image and thresholding the result which results is a binary segmentation of the image which highlights regions of non-stationary objects . This reference image is the background image constructed.

Fig. 2. Highlighted motion changes

C. Enhancing Motion Changes

Fig. 3. Original Image

The changes given by the background background subtraction is very light and is high in noise. To suppress the

noise we use a combination of erosion and dilation.These are morphological transformations. Dilation operations consists of convoluting an image with some kernel (in our case is a 3x3 grid), which can have any shape or size, usually a square or circle.The kernel has a defined anchor point, usually being the center of the kernel.As the kernel is scanned over the image, we compute the maximal pixel value overlapped by and replace the image pixel in the anchor point position with that maximal value. As you can deduce, this maximizing operation causes bright regions within an image to grow (therefore the name dilation). Take as an example the image above. Applying dilation we can get:

Fig. 4. Dilated image

The background (bright) dilates around the black regions of the letter. Then we now perform erosion on it.This operation is the sister of dilation. What this does is to compute a local minimum over the area of the kernel.As the kernel is scanned over the image, we compute the minimal pixel value overlapped by and replace the image pixel under the anchor point with that minimal value.Analagously to the example for dilation, we can apply the erosion operator to the original image (shown above). You can see in the result below that the bright areas of the image (the background, apparently), get thinner, whereas the dark zones the writing gets bigger.

Fig. 5. Eroded image

Thus by performing this all the noise and a few spots inside a region are all cleaned there by we get a much clearer motion change image.

D. Contour Extraction

Contour extraction is performed using OpenCV's inbuilt edge extraction function. It uses a canny filter to get a list of contours.It works fy convoluting a filter with the image so that gradually changing components gets cancelled out while sudden changing components like at the edge or borders are enhance. Thus the edges become visible. It does the following steps to extract the edges.

- Filter out any noise. The Gaussian filter is used for this purpose.
- Apply a pair of convolution masks (in x and y directions)
- Find the gradient strength and direction.The direction is rounded to one of four possible angles (namely 0, 45, 90 or 135)
- Non-maximum suppression is applied. This removes pixels that are not considered to be part of an edge. Hence, only thin lines (candidate edges) will remain.
- Canny does use two thresholds (upper and lower):
  - If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge
  - If a pixel gradient value is below the lower threshold, then it is rejected.
  - If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

This gives us a list of set of points , each set representing a contour. We can filter out small contours as they will be noise. So we set a threshold for the area for the contour about which we consider for the following steps.

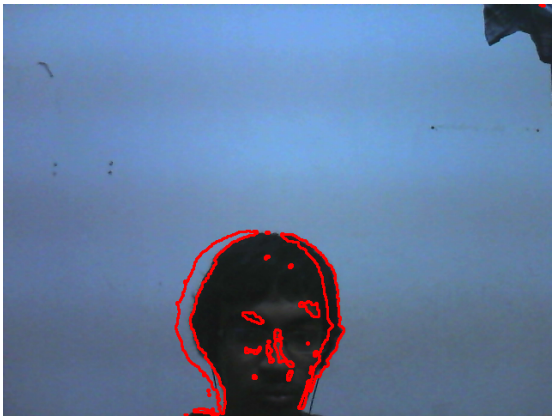

Fig. 6. Contours

E. Convex Hull and Defects

Now given the set of points for the contour, we find the smallest area convex hull that covers the contours .The Sklanskys algorithm was used to find the convex hull which has a complexity of $O(nlogn)$. The observation here is that the convex hull points are most likely to be on the fingers as they are the extremeties and hence this fact can be used to detecct no of fingers. But since our entire arm is there, there will be other points of convexity too. So we find the convex defects ie, between each arm of the hull, we try to find the deepest point of deviation on the contour.

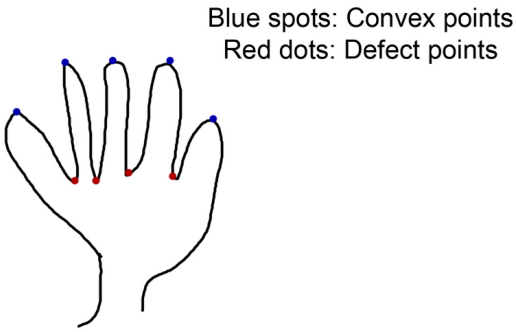

Fig. 7. Contours

F. Tracking and Finger Detection

Thus the defect points are most likely to be the center of the finger valleys as pointed out by the picture. Now we find the average of all these defects which is definitely bound to be in the center of the palm but its a very rough estimate. So we average out and find this rough palm center. Now we assume that the palm is angled in such a way that its roughly a circle. So to find the palm center , we take 3 points that closes to the rough palm center and find the circle center and radius of the circle passing though these 3 points. Thus we get the center of the palm. Due to noise , this center keeps jumping , so to stabilize it , we take an average over a few iterations. Thus the radius of the palm is a indication of the depth of the palm and we know the center of the palm . Using this we can track the position of the palm in realtime and even know the depth of the palm using the radius. The next challenge is detecting the no of fingers. We use a couple of observations to do this. For each maxima defect point which will be the finger tip, there will be 2 minimal defect points to indicate the valleys. Hence the maxima and the 2 minimal defects should form a triangle with the distance between the maxima and the minimas to be more or less same. Also the minima should be on or pretty close to the circumference of the palm. We use this factor too. Also the the ratio of the palm radius to the length of the finger trianger should be more or less same . Hence using these properties, we get the list of maximal defect points that satisfy the above conditions and thus we find the no of fingers using this. If no of fingers is 0 , it means the user is showing a fist.

III. Result

Thus we can detect the no of fingers, the location of palm and its depth. Using this we can construct systems that detect guestures. Due to the heavy noise involved in regular camers, the location of the palm and the radius are very jittery and varies continuously, but the average is the same and is suitable for use. Thus we need to average it over a few frames. Thus rapid guestures are not suited for the system that is built. Also the background needs to be stationary (slight to moderate) motion is tolerable due to the filtering techniques used). The hand needs to be facing straigt at the camera, as tilts cause the palm to obtain elliptical shape which is not well detected by the system. The system is very well suited for performing guestures like pause , play ,grab, drop, guestures based on finger location,etc.

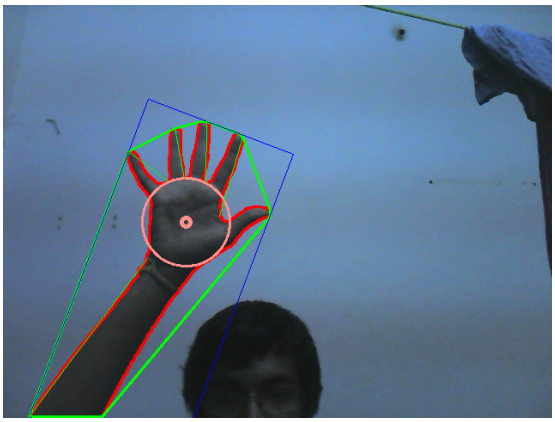

Fig. 8. Finger and Palm Detection

IV. Conclusion

Thus a program was created that was able to detect our hands, track them in realtime and perform some guesture recognition with simple signal processing performed on images obtained from a regular laptop web-camera. Reasonable accuracy and stability was obtained which can be used for steady and simple guestures to perform tasks.

References

[1] P. KaewTraKulPong and R. Bowden, *An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection*,d European Workshop on Advanced Video Based Surveillance Systems, AVBS01. Sept 2001.

[2] Aniket Tatipamula, *Hand Guesture using OpenCV*, http://anikettatipamula.blogspot.in/2012/02/hand-gesture-using-opencv.html