Hand Tracking and Gesture Recognition

A Project Synopsis Submitted
In Partial Fulfilment of the Requirements
For the Degree of
Bachelor of Technology
In
Computer Science & Engineering
By

Pranshul Agarwal (20114038)
Aditya Mohan (20114026)
Sagar Upadhyay (20113034)
Sandeep Kumar (20114103)

ENGINEERING PROJECT

GROUP : CS09

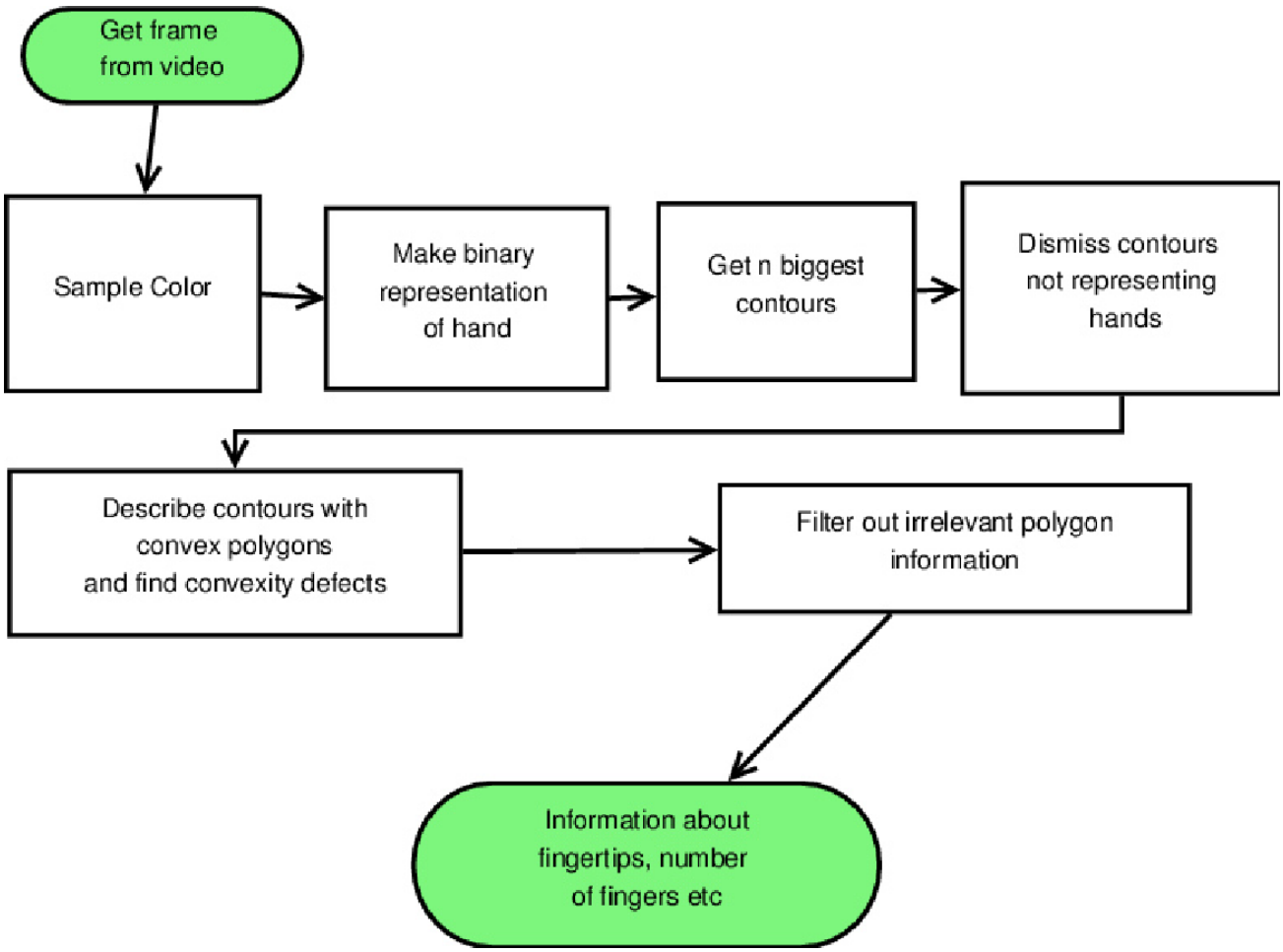
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY
ALLAHABAD

Objective of the Project

The aim of the project is to device an application that is able to detect and track hands in real time and locate some important features of hand like fingers and centre and size of palm etc. On identification of traits of the hand, like the number of fingers pointed, we can instruct the system to perform functions like initiation of programs or system calls, etc. The project will make use of digital image processing techniques performed on images obtained from a regular laptop web-camera.

Architecture



Design and Implementation

This project provides a method to perform hand tracking using webcam with only the hands in the range of visibility using OpenCV. Background is eliminated using simple techniques, and a set of filters are applied to get the hands contour. A convex hull is created on the contour. We calculate the defect points in the hull which is used to detect the no of fingers, centre and size of the palm.

The various steps followed in our project are –

A. Input Raw Video feed

B. Background Construction

C. Background Subtraction

D. Enhancing Motion Changes

E. Contour Extraction

F. Convex Hull

G. Finding convexity defects

H. Palm tracking

I. Tracking and Finger Detection

J. Performing Operations

A. Input raw video feed

The video feed input in taken directly through the system's webcam handled by in-built libraries of the operating system in real time.

B. 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. Our assumption is that the background is mostly static. Hence for those stationary items, those pixels aren't affected by this weighted averaging and those pixels that are constantly changing aren't a part of the background, hence those pixels will get weighed down. 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 the face is part of the background as it needs to detect only my hands.

C. 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. 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.

D. Enhancing Motion Changes

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. As you can deduce, this maximizing operation causes bright regions within an image to grow (therefore the name dilation). Then we now perform erosion on it. 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. The bright areas of the image (the background, apparently), get thinner, whereas the dark zones the writing gets bigger. 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.

E. Contour Extraction

Contour extraction is performed using OpenCVs inbuilt edge extraction function. It uses a canny filter to get a list of contours. It works by 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. We can filter out small contours as they will be noise.

F. Convex Hull

Now given the set of points for the contour, we and the smallest area convex hull that covers the contours .The Sklanskys algorithm was used to and 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 extremities and hence this fact can be used to detect no of fingers. But since our entire arm is there, there
will be other points of convexity too. So we and the convex defects i.e., between each arm of the hull, we try to and the deepest point of deviation on the contour.

G. Finding convexity defects

We calculate the convex hull of the fore-arm contour in order to get the convexity defect of the contour. The observation here is that the convex hull points are most likely to be on the fingers as they are the

extremities and hence this fact can be used to detect no of fingers.

Since the fingertips connect the convex hull edge, it would be easy to create a very small convexity defect if there is a concave contour. We should regardless such convexity defects and take the convexity defects with large area as what we want. The large convexity defects also have long depth. So we need to check the depth of all the convexity defects. If the depth is longer than a certain threshold, we draw the depth point of it. If it's not longer than a certain threshold, the defect will be ignored.

H. Palm tracking

Thus the defect points are most likely to be the centre of the finger valleys as pointed out by the picture. Now we and the average of all these defects which is definitely bound to be in the centre of the palm but its a very rough estimate. So we average out and find this rough palm centre. Now we assume that the palm is angled in such a way that its roughly a circle. So to find the palm centre , we take 3 points that closes to the rough palm centre and find the circle centre and radius of the circle passing though these 3 points. Thus we get the centre of the palm. Using this we can track the position of the palm in real time and even know the depth of the palm using the radius.

I. Tracking and Finger Detection

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 minima 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 ratio of the palm radius to the length of the finger triangle 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 and the no of fingers using this.

J. Performing Operations

If no of fingers is 0 , it means the user is showing a first Operations. Based on the number of fingers detected, various commands are being execute using Linux system call commands.
On detecting 1 finger, VLC media player is being opened.
On detecting 2 or 3 fingers, Mozilla Firefox is being opened.
On detecting 4 fingers, Gnome Terminal is being opened.

User manual / Help

References:

[1] OpenCV API Reference
Available from http://docs.opencv.org/2.4.1/modules/refman.html

[2] CMake and Make commands
Available from http://stackoverow.com/questions/6914524/how-to-generate-cmakelists-txt

[3] Introduction to OpenCV
Articles present on Wikipedia.
Available from http://en.wikipedia.org/wiki/OpenCV

[4] OpenCV Tutorials
Available from http://www.pages.drexel.edu/ nk752/tutorials.html

[5] Palm anatomy and exclusive attributes
Available from http://goo.gl/sJgDis

[6] Image Processing Morphological Operations
Available from http://goo.gl/TW8Xfj

Instructions to run program :-

- ➔ sudo apt-get install libopencv-dev
- ➔ pkg-config --modversion opencv
- ➔ g++ -o opencv main.cpp `pkg-config --cflags --libs opencv` -lX11 -lXtst -lXext
- ➔ ./opencv