Indian Institute Of Technology Bombay

Department of Computer Science & Engineering

Interacting with Software using Hand Gestures

Group:

Tamojit Chatterjee 153050002

Abhishek Sethi 153050023

Kush Goyal 153050045

Lakshya Kumar 153050051

28 April 2016

Contents

1 Introduction 2

2 Problem Statement 2

3 Requirements 2

4 Implementation 2

4.1 Method for recognizing gestures 2

4.2 Haar Classifier 3

4.3 Cascade Haar classifier 4

4.4 AdaBoost 4

4.5 Using OpenCV to Train and Test Haar classifier 5

5 Processing Captured Image to get Finger Count 6

5.1 Generating 9 rectangles and finding the medians 6

5.2 Binarizing Images 7

5.3 Sum & Blur Method 8

6 Convexity Defects, Contours and Convex Hull 8

7 Playing/Stopping music using finger count 14

8 Future Work 14

9 Conclusion 15

10 References 15

1 Introduction

Machine Learning is being used recently in various human computer interaction jobs. For e.g, Motion Sensing input device by Microsoft i.e Kinect , enables users to control and interact with their console/computer without the need for a game controller. Another software is Sony Vaio's music player that uses hand gestures to change songs and control volume.
These features make user experience more interactive, real time and enjoyable. This is the motivation behind our project i.e interacting with software using hand gestures.

2 Problem Statement

Our project aims to interact with software using hand gestures. We will be using static hand gestures and try to map those gestures to software events.

3 Requirements

Following are the software and hardware requirements.
A) Hardware Requirements

- WebCam

B) Software Requirements

- OpenCV 2.4.9
- Python 2.7.6
- Numpy

4 Implementation

4.1 Method for recognizing gestures

We are using **Haar** classifier for hand recognition, i.e., when the user brings his hand in front of the camera, we use the classifier to recognize the hand in the captured image.

After successful identification of the area containing the hand we use that sub-image for further classification, i.e., we have several fixed gestures mapping to several fixed software actions and we classify the coming input image on the basis of finger count) and take the action corresponding to the gesture, e.g., play and stop music track.

4.2 Haar Classifier

It is a famous classification algorithm for identifying particular objects in images in an efficient manner.

Haar Classifier considers adjacent rectangular regions at specific locations present in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. The detection window can be moved throughout the image as the object can be located anywhere. This difference is then used to categorize subsections of an image. In our case the Haar features try to identify the intensity difference between the background and the region of our hand. These differences follow certain pattern because even though the possibilities are large but our hand still has predictable shape and size.
The following equation represents a particular Haar like feature.

$$\text{Sum} = I(C) + I(A) - I(B) - I(D)$$

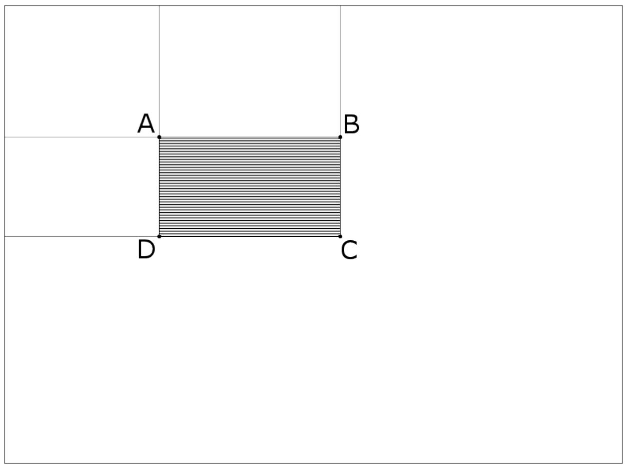


Figure 1: Using Windows to calculate Intensity diffences

The Haar classifier uses something known as the integral image, where each element of the integral image contains the sum of all pixels located on the upleft region of the original image (in relation to the element's position). This allows to compute sum of rectangular areas in the image, at any position or scale, using only four lookups.

4.3 Cascade Haar classifier

The number of Haar classifiers turn out to be very large even for small images and hence if detection is required in real time this results in efficiency barrier. So instead of applying the whole classifier at once, it is broken down into a series of weak classifiers and they are tested sequentially. Earlier stages contain less number of features and hence can be used to prune the area that does not have the object(hand). So further classifiers need not to be run on the windows failing the previous classifier.

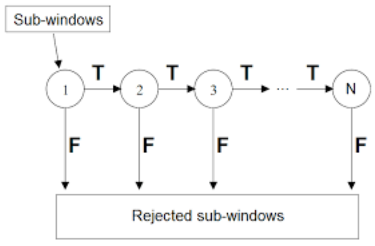


Figure 2: Series of Weak classifiers

4.4 AdaBoost

AdaBoost, short for "Adaptive Boosting", is a machine learning meta-algorithm. It can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances mis-classified by previous classifiers. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (e.g., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner.

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{1}$$

Each f_t is a weak learner that takes an object x as input and returns a real valued result indicating the class of the object.

4.5 Using OpenCV to Train and Test Haar classifier

We have used the OpenCV library to train the Haar cascade classifier for detecting hands. It is explained in the below stages:-

- **Collecting negative and positive images**:

  We have used the opencv_createsamples command to build a set of positive examples by taking a single image of a hand and several background images and inserting the hand image into the background image at random locations, angles and scales. And we kept the background images as the negative images. These background images were obtained from internet as well as from our local surroundings. We scaled every background image to a specific height and width and ensured that enough variations were present.
  Command:- opencv_createsamples -img positive.jpg -bg bg.txt -info info/info.lst -pngoutput info -maxxangle 0.5 -maxyangle -0.5 maxzangle 0.5 -num 180 . This commad outputs and info.lst file that contains the bound of the rectangle containing the object of interest.

- **Vectorizing the positive images**:

  We have used below command to vectorize the image, i.e, converting them into a .vec file, which is a binary feature file for the images.

  Command: opencv_createsamples -info info/info.lst -num 180 -w 50 -h 50 -vec positive.vec

- **Training the Haar classifier**: In this step we have used opencv_traincascade command to train the Haar classifier mentioning number of stages in the process and number of negative and positive images used in the training.

  Command: opencv_traincascade -data data -vec positive.vec -bg bg.txt -numPos 180 -numNeg 180 -numStages 5 -w 50 -h 50

  The last step is combining the .XML files generated as a result of training the Haar classifier and combining them into a single .XML file representing the whole decision tree. We have used the final .XML file to detect hands in images captured using the webcam.

5 Processing Captured Image to get Finger Count

5.1 Generating 9 rectangles and finding the medians

In this step we generate 9 rectangles in the center of the bounding box of hand. We get the median color intensities of each of the 9 rectangles.

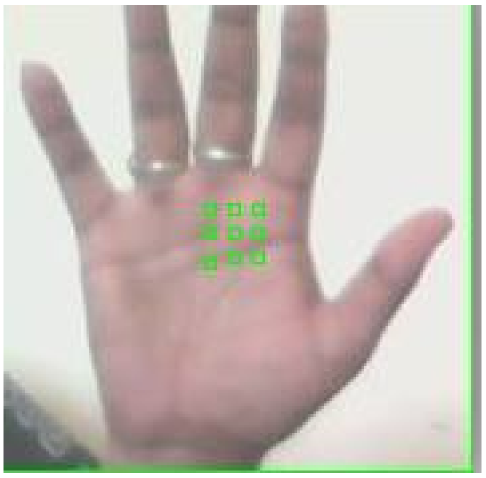


Figure 3: 9 rectangles in the center of bounding box

5.2 Binarizing Images

We use the 9 median intensities of the previous step to binarize our image and generate 9 binary images.

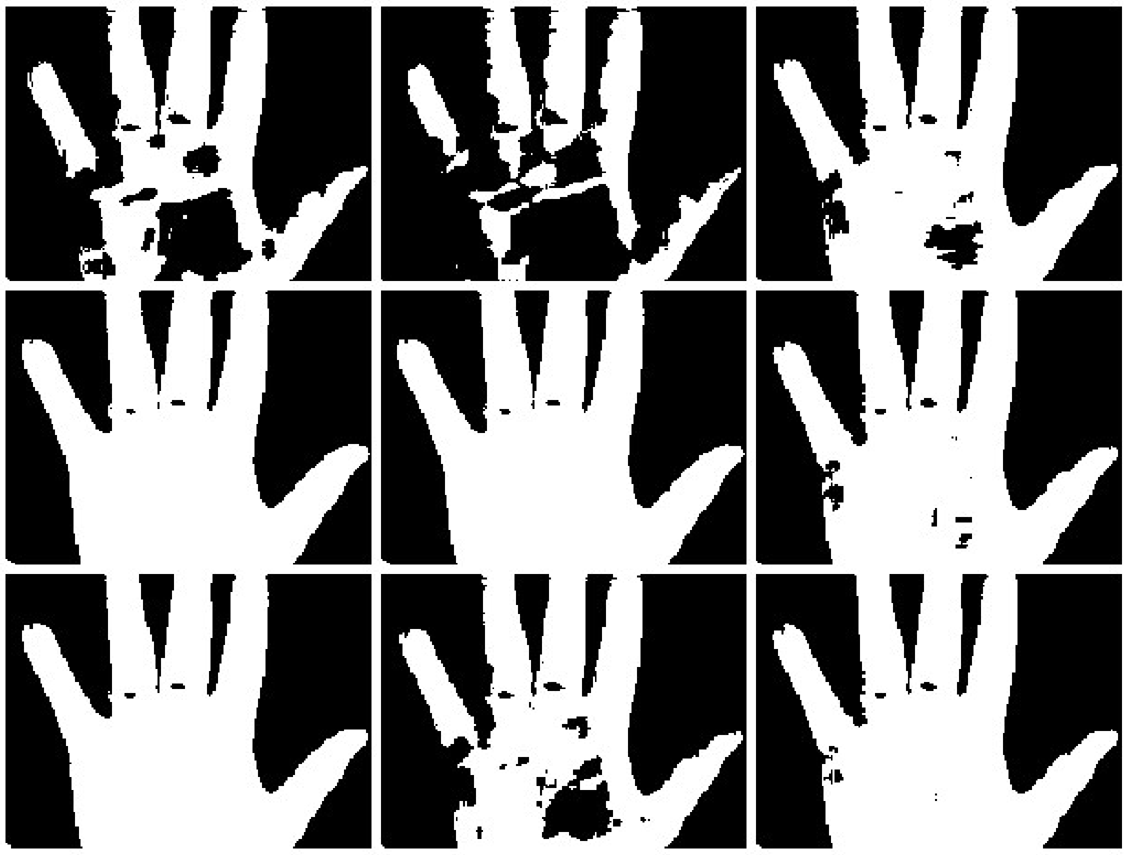


Figure 4: Binary Images according to each of the 9 intensities

5.3 Sum & Blur Method

This method gets 9 binary images as input and generates the final image by majority method. By majority method we mean that a pixel is painted white if it is white in majority of the 9 images otherwise it is painted black.

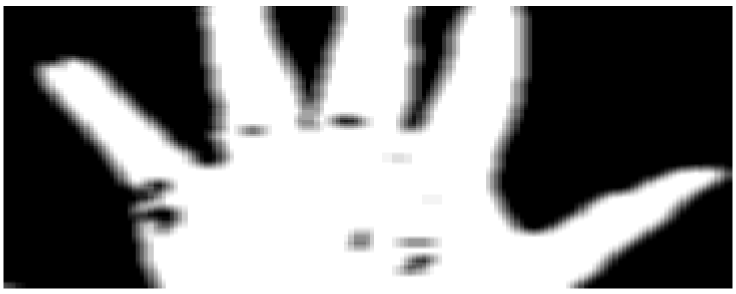


Figure 5: Converting 3 channel image to a single channel image

6 Convexity Defects, Contours and Convex Hull

To get the finger count from our image the following steps are followed:-

- **Generating Contours**: Contours are curves joining all the continuous points (along the boundary), having same color or intensity. cv2.findContours() function is used to get the contours in an image. This function returns a python list of contours. Out of this list we extract the contour with maximum number of points as the contour around our hand will be the largest one.

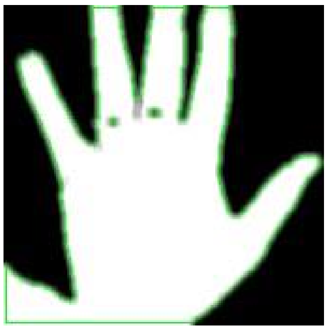
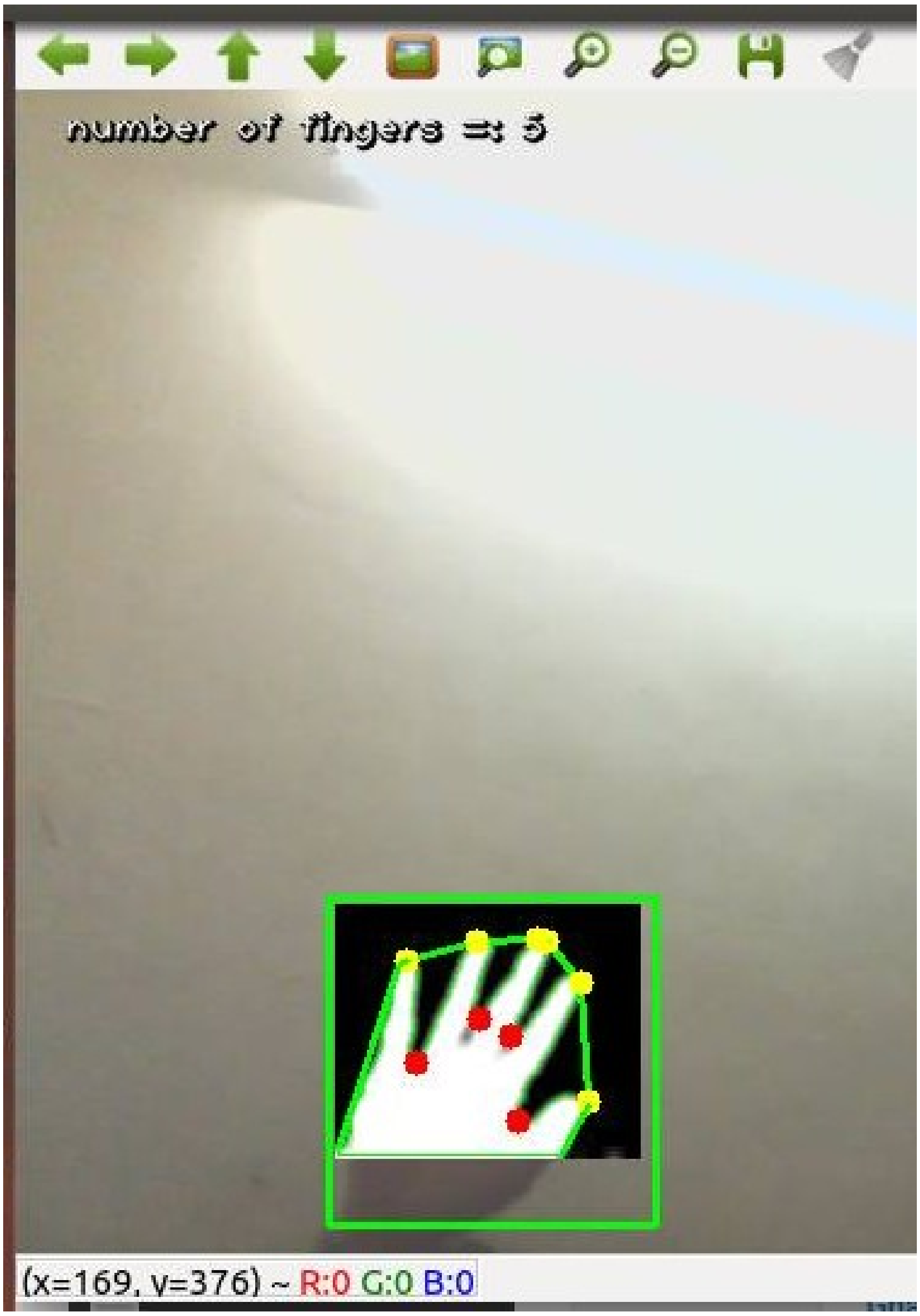


Figure 6: Drawing the Largest Contour

- **Finding Convex Hull and Convexity Defects**: Convex Hull is similar to contour approximation, cv2.convexHull() function checks a curve for convexity defects and corrects it. Convex curves are the curves which are always bulged out, or at-least flat. And if it is bulged inside, it is called convexity defects. cv2.convexityDefects() uses the output of cv2.convexHull() and cv2.findContours() and returns an array where each row is a defect having the folowing attributes, start point, end point, farthest point, and aproximate distance to farthest point. The folowing figures show the corrected convexity defects. In our method the number of fingers= number of convexity defects + 1. The red circles mark the convexity defects (farthest point from the convex hull line) and the yellow circles denote the start and end point of a convexity defect.


number of fingers =: 5

(x=169, y=376) ~ R:0 G:0 B:0

Figure 7: Example-1 Finger Count-5

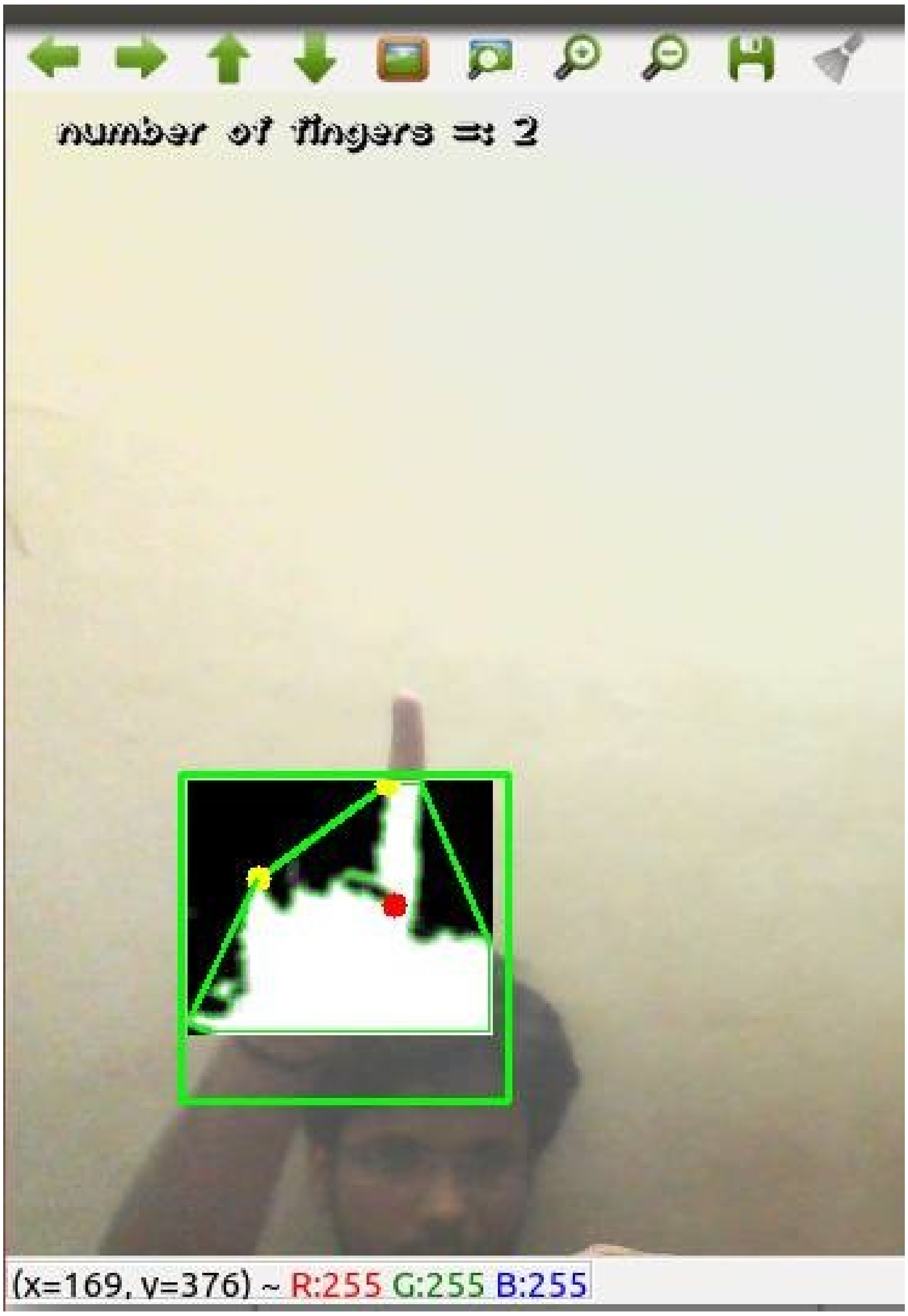


Figure 8: Example-2 Finger Count-1

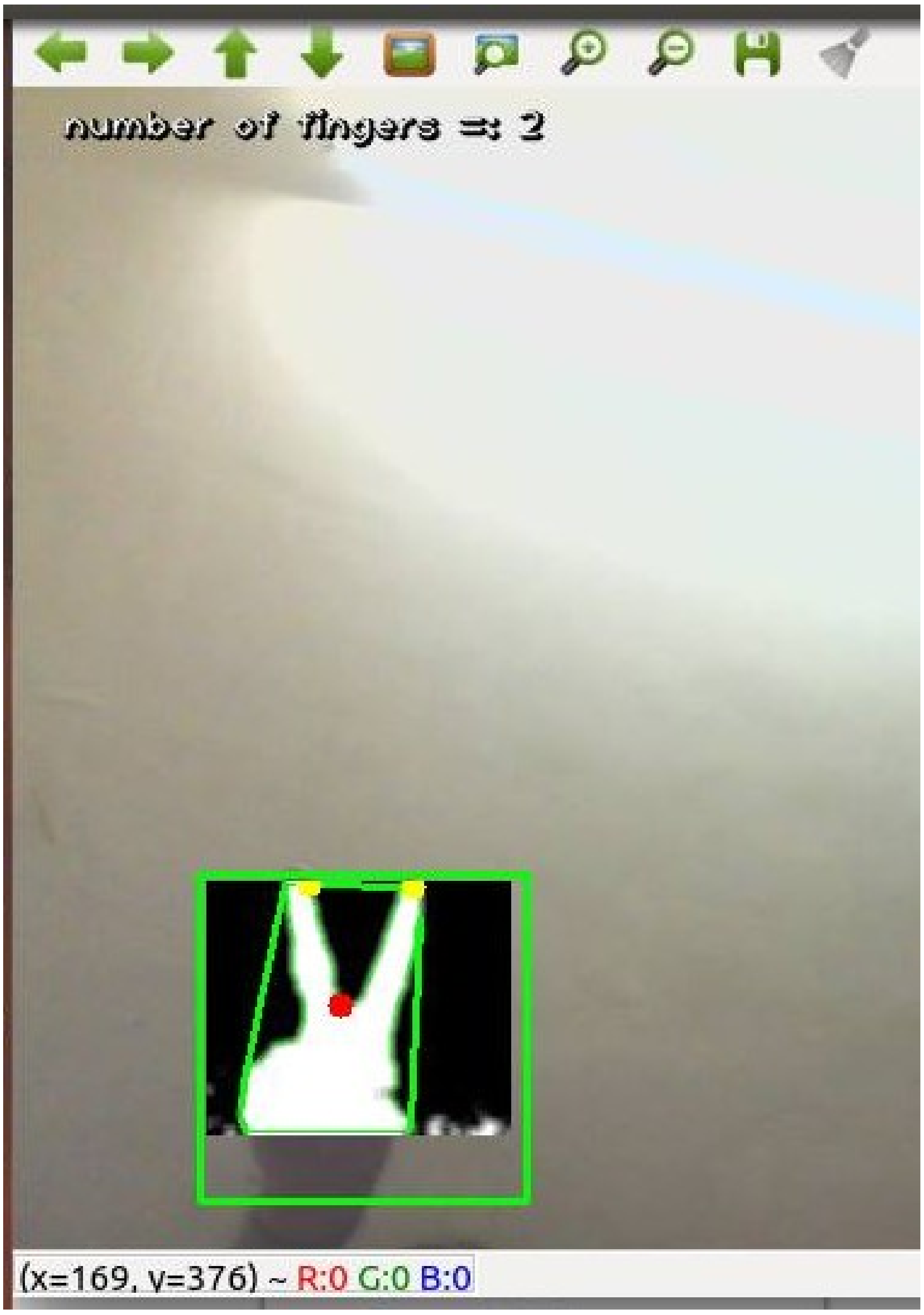
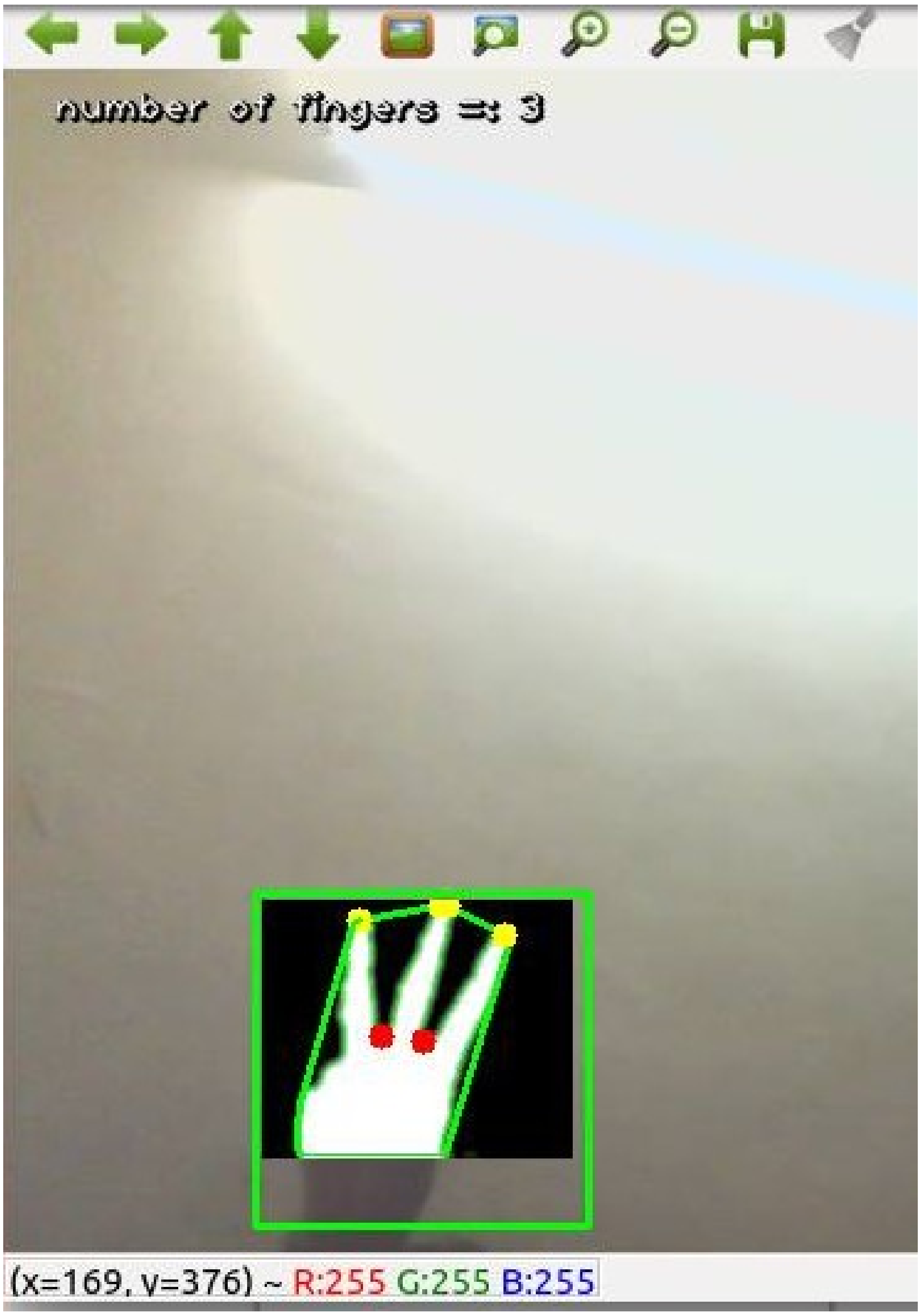


Figure 9: Example-3 Finger Count-2

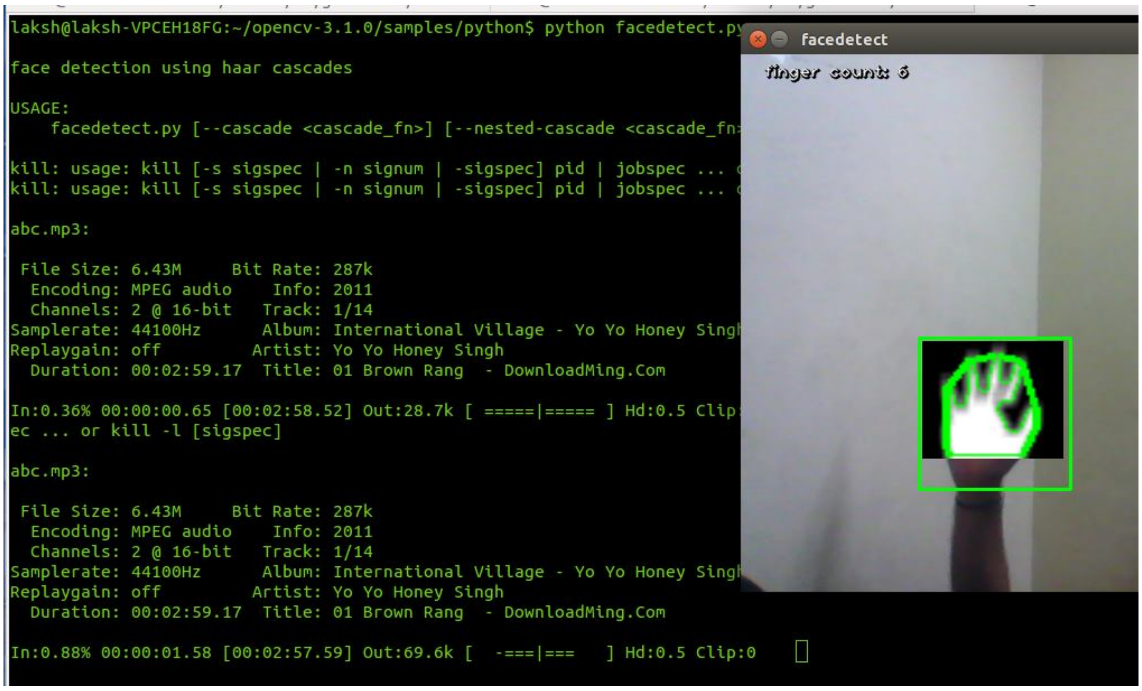

number of fingers =: 3

(x=169, y=376) ~ R:255 G:255 B:255

Figure 10: Example-4 Finger Count-3

7 Playing/Stopping music using finger count

Using the number of fingers we are playing/stopping music, opening you tube, facebook etc. Lot of more things can be done using finger count.



```
laksh@laksh-VPCEH18FG:~/opencv-3.1.0/samples/python$ python facedetect.py

face detection using haar cascades

USAGE:
    facedetect.py [--cascade <cascade_fn>] [--nested-cascade <cascade_fn

kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ...
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ...

abc.mp3:

 File Size: 6.43M     Bit Rate: 287k
  Encoding: MPEG audio    Info: 2011
  Channels: 2 @ 16-bit   Track: 1/14
Samplerate: 44100Hz      Album: International Village - Yo Yo Honey Singh
Replaygain: off         Artist: Yo Yo Honey Singh
  Duration: 00:02:59.17  Title: 01 Brown Rang  - DownloadMing.Com

In:0.36% 00:00:00.65 [00:02:58.52] Out:28.7k [ =====|===== ] Hd:0.5 Clip
ec ... or kill -l [sigspec]

abc.mp3:

 File Size: 6.43M     Bit Rate: 287k
  Encoding: MPEG audio    Info: 2011
  Channels: 2 @ 16-bit   Track: 1/14
Samplerate: 44100Hz      Album: International Village - Yo Yo Honey Singh
Replaygain: off         Artist: Yo Yo Honey Singh
  Duration: 00:02:59.17  Title: 01 Brown Rang  - DownloadMing.Com

In:0.88% 00:00:01.58 [00:02:57.59] Out:69.6k [  -===|===   ] Hd:0.5 Clip:0
```

Figure 11: Demonstration

8 Future Work

Due to limitations of the computational resources available to us, we could only make the Haar Classifier moderately robust i.e it still gives significant number of false positive and sometimes in noisy background fails to detect the hand. We can also explore various other methods of classification like LBP(local binary parameters) .
To improve the Haar classifier we have to train it on a large dataset and that will require a significantly powerful computer. So we can try to train the classifier using GPU.
We have only considered static gestures and hence we are limited in the way we interact with the software. We can explore dynamic gestures like hand waving, which can be thought of as a series of static gestures but will require various other intricacies.

9 Conclusion

Our work involves first finding out the ROI(hand) in the live image captured by the video feed and then process it using various computer vision and image processing techniques. So, we can say that our project lies on the intersection of computer vision and machine learning.

10 References

- http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/
- http://opencv.org/documentation/opencv-2-4-9.html
- http://ds.cs.ut.ee/Members/artjom85/2014dss-course-media/ObjectdetectionusingHaar-final.pdf
- https://en.wikipedia.org/wiki/AdaBoost
- https://en.wikipedia.org/wiki/Haar-like_features