PROJECT REPORT

Dt.20-11-2016

CS 663

Detection of faces by detecting regions of skin from color images
(Implemented in C++ - opencv2)

SURYA TEJ - 140050055
SRINIVAS NAIK BHUKYA - 140050064
CHAITANYA RAJESH BANALA - 140050073

Problem Statement :

Detection of faces by detecting regions of skin from color images.We extended this to also use
(i) webcam for live face detection
(ii) any video properly formatted
(iii) a color image

Usage is mentioned in README.

Motivation :

Detection of the human face is very useful in automatic face recognition, video surveillance, human-computer communication and large-scale face image retrieval systems. The first and foremost important step in any of these systems is the accurate detection of the presence and the position of the human faces in an image or video.

Overview:

- ➢ The algorithm uses a combination of RGB-HS-CbCr color spaces for the detection of human faces. Skin regions are extracted using a set of bounding rules based on the skin color distribution obtained from a training set.
- ➢ The model we have used utilizes the additional hue and chrominance information of the image on top of standard RGB properties to properly differentiate between skin pixels and non-skin pixels.
- ➢ Two region properties Face Detection phase – box ratio and eccentricity were used to classify the shape of each skin region after getting skin detected part of the image to report if it is face region or not.
- ➢ This project is done in C++ using opencv libraries.We did not use any direct existing libraries for face detection.

Procedure:

The procedure involves two main steps :
(i) Skin Detection
(ii) Face Detection using the skin detected intermediate result obtained from step (i)

The procedure is explained using a sample image and applying operations on it.The basic algorithm we used is same as the one explained in research paper (mentioned in references section).But In the report we tried to explain the algo using the code we have written.

Let us take a sample image

Skin Detection :

- The given image is in RGB color space, first step is to get two more color spaces YCrCb and HSV of the image. We use these color-spaces to get the intensities of pixels in all the color spaces and mark the pixels as skin pixel using those values.
  We do this by using the functions :
  cvtColor(input,ycrcbimg,cv::COLOR_BGR2YCrCb)
  ` cvtColor(input,hsvimg,CV_BGR2HSV)

YCrCb color space

HSV color space

- Now filter out the skin pixels using the values of RGB , YCrCb , HSV using the max and min boundary values for each of intensity values.These values were found using the training dataset which gave good results.
  For e.g. (R>220) && (G>210) && (B>170) && (abs(R-G)<=15) && (R>B) && (G>B) and
  (CR>=133) && (CR<=173) etc.
  which are mentioned in the research paper referenced in the References section of the report.

➢ After applying all the filters using combination of colorspaces we get the following binary skin image.

Skin detected image of given image

Face Detection :

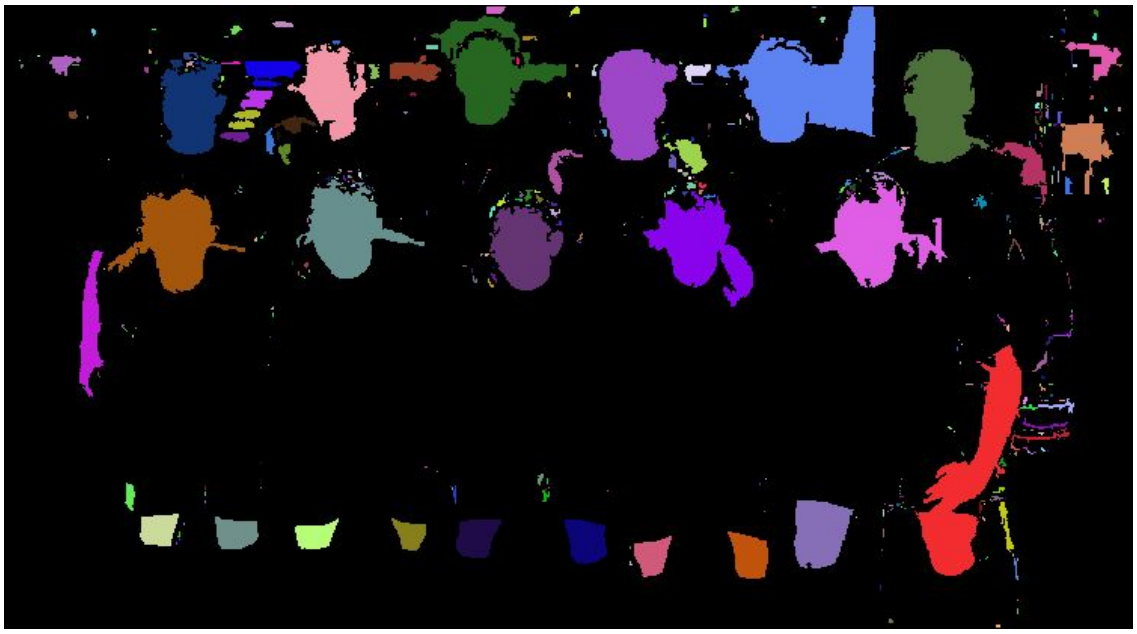
➢ First step is to fill all the holes in a connected set of pixels which we call connected components.This is done using **imfill** function written in the code.After filling the holes we get the following image.

Image after filling holes

Notice that all the holes in faces (eg. eyes) are now filled and the image is divided into separate components.

➢ Next step of the Face Detection is to divide the above obtained image into separate connected components in the function "**getconnectedcomponents**" which primarily calls

"**findconnectedblobs**" that separates each connected components and stores them into the "blobs" vector which has the list of all connected pixels as a vector.Each blob is then randomly colored to show that they have been indeed separated into separate connected components.



Each connected component colored randomly

- Now iterate through every connected component to detect faces.We use box ratio and eccentricity to filter out faces which are stored in **faceblobs** in the function **getface.** Ratio and eccentricity (mentioned in the research paper)are calculated by first creating a bounding box to every component and finding values xmin,xmax,ymin,ymax(Full logic is in the function "getallfaces").We then use the values to filter out blobs which might be faces :

  blobs[i].size() > 1500 && ratio >=0.4 && ratio <= 1.8 && ecc >= 0.25 && ecc<=0.97

- After obtaining the "faceblobs" we now draw a rectangle in the function drawrectangle which takes in original image and faceblobs and draws rectangle over every faceblob with red.

  Mat drawrectangle(Mat img,vector<vector<Point2i> >faceblobs)

  output.at<cv::Vec3b>(y,x)[0] = 0;
  output.at<cv::Vec3b>(y,x)[1] = 0;
  output.at<cv::Vec3b>(y,x)[2] = 255;

Final image with detected faces :

Experimental Results:

The model was evaluated on test data set of 20 images and 2 videos, the images and videos were randomly selected from the Internet, each comprising of some near-frontal faces. The test images and videos consists of various illuminations and poses.

Metrics calculated for each image:

- False Detection Count(FDC) = $\frac{\text{no. of false detections}}{\text{total number of detections}} \times 100\%$

- Detection Success Count (DSC) = $\frac{\text{no. of correctly detected faces}}{\text{total number of faces}} \times 100\%$

*(where the number of correctly detected faces is equivalent to the number of faces minus the number of false dismissals)

We seek to have results with high DSC and low FDC values to judge performance of the model used.

Results for Training Images:

Figure		FDC(%)		DSC(%)
1		0		100
2		0		83.4
3		20		80
4		16.7		100
5		33.3		100
6		0		100
7		0		100
8		0		100
9		20		100
10		0		66.7
11		7.2		100
12		0		78
13		16.7		100
14		33.3		100
15		40		100
16		0		100
17		0		100
18		37.5		100
19		33.3		75
20		16.7		100
video 1.mp4		0		100
video 2.mp4		10		80

All the images and videos used for testing are included in the submission folder.

Observations:

- Experimental results tells us that the model achieves good detection success rates for near-frontal faces of varying orientations, skin color and background.

- ➢ In skin detection phase, We first tried to implement detection based on only two color spaces like RGB and HSV or RGB and YCbCr space values but the use of all 3 color spaces showed very good face detection accuracy.
- ➢ Images with smaller faces and very low quality illuminations are not detected properly.

Summary and Challenges Faced:

The main challenges encountered in face detection is to cope with a wide variety of variations in the human face such as posture and scale, face orientation, facial expression, ethnicity and skin color. External factors such as occlusion, complex backgrounds inconsistent illumination conditions and quality of the image may also contribute significantly to the overall problem.

The resulting segmented skin color regions have three common issues:
a) Regions are fragmented and often contain holes and gaps.
b) Occluded faces or multiple faces of close proximity may result in erroneous labeling.
c) Extracted skin color regions may not necessarily be face regions. There are possibilities that certain skin regions may belong to exposed limbs (arms and legs) and also foreground and background objects that have a high degree of similarity to skin color.

Contribution:

140050064:

Code upto skin detection phase.
Testing and report.

140050073:

Code for dividing the image into connected components, labelling them and facedetection,.
Testing and report.

140050055:

Code for dividing the image into connected components, facedetection.
Testing and report.

References:

(1) Skin Detection(https://www.cs.rutgers.edu/~elgammal/pub/skin.pdf) - a Short Tutorial-Ahmed Elgammal, Crystal Muang and Dunxu Hu Department of Computer Science, Rutgers University, Piscataway, NJ, 08902, USA

(2) Face Detection Using Skin Tone Segmentation(http://www.softcomputing.net/wict11_4.pdf)-Sayantan Thakur , Sayantanu Paul , Ankur Mondal

(3) Documentation of opencv2(http://docs.opencv.org/2.4.0/)