Face Detection for Mobile Device

18-551 Spring 2005 Group 2

Dason Cheung, Anuj Batra, Jenney Zhang
2nd May 2005

1. INTRODUCTION

PROJECT DESCRIPTION

Bio-metric authentication has been a hot topic in recent years in awake to the lack of security in conventional username and password authentication method. Passwords may be forgotten or stolen by unauthorized personals. Bio-metric signatures such as finger-print, face or voice offer higher level of security and convenience as they are always on people and are unique to each person. Face recognition is a popular choice among these Bio-metric authentication methods as it is very easy to obtain face image. Face detection is needed before recognition could be performed, thus face detection is a vital step correctly recognize a face from an image. As most modern mobile phones have built in camera, they could be candidates for portable face recognition devices. We propose to develop a face detection system targeted at detecting faces in colored digital images with sized between 1.3 – 2.0 mega pixels which could be taken from current mid-high end camera enabled mobile phones.

We propose to implement such a system using the TI C67EVM evaluation module. Since mobile phones have limited processing power and memory, they often include a Digital Signal Processor (DSP) for signal processing task such as image processing. The C67EVM is a piece of dedicated signal processing hardware with limited memory which is similar to real world conditions for mobile phones. There are many on going research in face detection, majority of them follow a two step approach starting with face localization (finding candidate face regions in an image) and then a rejection analysis that rejects non-face objects from the candidate pool. There are different

methods in performing face localization namely skin color differentiation and hair color differentiation which detects skin region and hair region respectively. We will follow a skin color differentiation scheme that is used by **Diedrick Marius's** group at Stanford where they use skin tone as a parameter to create a binary image to obtain area of interest (faces). After obtaining the binary image, we will use our own blobbing labeling algorithm to generate a list of blobs in the binary image. Our rejection scheme will incorporate several blob analysis namely shape rejection, size rejection and feature rejection (eye, noise, mouth).

We will use headcounts as a measurement of the accuracy of our system. We will compare our system with **Diedrick Marius's** group system which has an 95% accuracy in detecting faces in an color image with single skin tone and under uniformed outdoor lighting conditions. Under same outdoor lightning conditions, we are looking for equal detection accuracy 95% for single skin tones. And we will aim for a 90% accuracy with multiple skin tones and a 75% accuracy under indoor lighting conditions.

SOLUTION

Basically a simple 3 step face detection system was used to process the jpeg image to detect faces , that uses skin tone segmentation, morphological processing and blob detection. It does not require extensive statistical analysis to detect faces as previous projects suggest. The Morphological processes include first opening the image and then closing it with appropriate structuring elements. The Algorithm is extensively discussed below. So to implement the system we built a Win32 console application that connects to

the C67 EVM. The Win32 console feeds a 1024*720 input image to EVM and retrieves detected face from EVM.

PRIOR WORK AND DIFFERENCE

In Spring 2002, group 2 in 18-551 conduct a project to complete a similar system titled “Face Detection for Surveillance”, where their goals was to develop a face detection system for wireless surveillance cameras. Their project uses a similar approach to our system but differs at the final face detection step. Below is a comparison chart between our projects.

	Our Project	Face Detection for Surveillance
Image	Color Image	Color Image with down sampling to optimize wireless bandwidth use
Skin segmentation	CbCr thresholding	Skin transform using red to green ratio
Morphological processing	Sequences of erosion and dialation	Sequences of erosion and dialation
Blob Labeling	4 way connected	Data not avaiable
Face detection	Own rejection scheme	Schneiderman & Kanade Algorithm

Other than the above differences in our approach, “Face Dectection for Surveillance” project also performed all their morphological processing in PC where as our system perform all of the processing inside the EVM.

We have also reviewed other projects in the Sixth IEEE International Conferece on Automatic Face and Gesture Recognition. We noticed most of the projects performed skin segmentation as a pre-processing step to calculate the binary image. Many methods have been used to detect skin regions with the majority using either YCbCr (YUV) or

HSV colorpsace, only a small group uses RGB colorspace. As compare to our simple single max and min skin segmentation threshold, many projects uses a skin transform matrix with eigen values to refine the skin region to a confined clustered which yields better results, the draw back is that a database of at least 600 images is required to locate the cluster region and the results is only applicable to image with the same condition as the database.

As for the face detection algorithm, we uses a series of blob rejection schemes mostly shape, aspect ration rejection as we find it is relatively inexpensive to compute in the EVM and yields reasonable results. There are other rejects schemes such as the template matching used by **Diedrick Marius's** group, an IEEE conference research group uses a spatial statistical model for blob rejection. Each reject schemes works under certain constraints, a combination of different rejection schemes is required to achieve good results in images taken under different conditions.

2. DATABASE

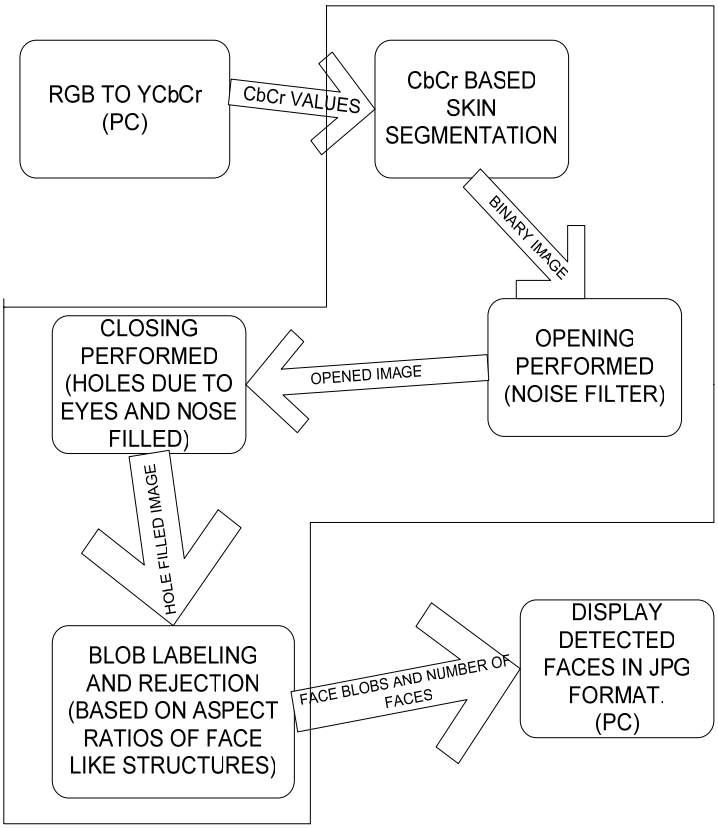
Initially, we tried to get image database from the "Face Detection Group" a group lead by Dr. Henry Schneiderman, a research scientist at Carnegie Mellon University in the field of computer vision for object detection and recognition. His group has an extensive face database which consists of 1.4GB of data with more than 1000 images. Their database mainly consists of group photos taken under a set number of pre-defined conditions. The advantage of using such a database is it can act as a benchmark test for our system. Unfortunately, the database only holds greyscale images will not work with our skin segmentation algorithm. Thus we turned our direction to online search. We

obtained a total 30 images from various online sites, the size of which are between 800*600 and 1600*1200. Later on of the project, we decided to use images above 1024*720, which can be simply taken from a digital camera. Any image sized below that threshold cannot be used because they need to be resized and stretched which will destroy the aspect ratio of faces. For images above the minimum requirement, the pc will read the first 1024*720 pixels and process that part of the image. We divided the images into 5 sets depending on their skin tone: white, yellow, brown, black and mixed. Each set contains 4 outdoor and 2 indoor pictures for later comparison. The number of people in each image is constrained from 10 to 50. Thus the head size of each person we used in the filtering process is 20*25-100*150.

3. IMPLEMENTATION:

All our image processing is done on the EVM, we first converts a RGB jpg image into YCbCr colorpsace on the PC using the Image Magick C library. Then we discard the Y component and send the CbCr components from the PC to the EVM through HPI transfer. The EVM will perform all the image processing and send back binary images of various image processing stages to the PC for debug and demo use. The last step is to send labeled blob coordinated and accepted face blobs to the PC for face framing which is perform at the PC side with the Image Magick C library.

Data flow between PC/EVM:



NOTE: AREA INSIDE BOX INDICATES EVM PROCESSING

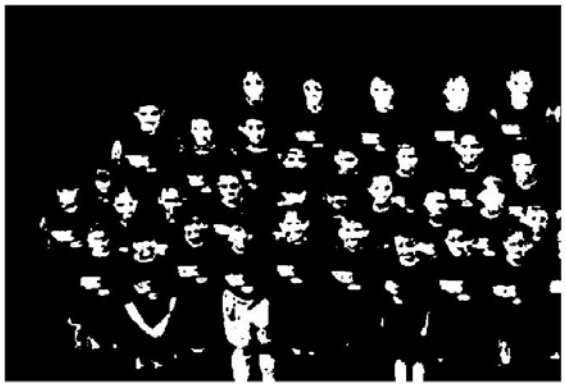
4. ALGORITHM

IMAGE THRESHOLDING

This was probably one of the most important aspects and the basis of this project. It was required that the threshold values be such that all skin can be detected under most conditions (different lightening, background), as well as various skin tones. We did a lot of research in this field online by looking into research projects and found that the most accurate way of achieving our goal was to covert the RGB image to YCbCr colorspace.

There were lot of issues associated with this, but by training a set of 30 images we found the best threshold values in the CbCr colorspace to be the following. We denote Rcr and Rcb as the respective ranges of Cr and Cb values that correspond to skin color, which subsequently define our skin-color reference map. The ranges that we found to be the most suitable for all the input images that we have tested are RCr [140,165] and RCb [100,133].

Color Image



Threshold Image

BINARY MORPHOLOGY

A) EROSION

An image has a point set A which defines the pixels that are separate from the background. Every point p in the image will be tested. The structuring image will consist of a point set B. The eroded image will result in those points p for which all possible p+B are in A. This process will be the same for all three structuring elements (rectangular, square, and circular).

B) DILATION

An image has a set point A which defines the pixels that are separate from the background, and the structuring element will consist of the points in B. The dilation of A by b is the point set of all possible vector additions of pairs of elements, one from each of the set A and B.

C) OPENING

In order to perform the morphological property of opening on an image, we will first dilate the image with the structuring image. We will then erode the image by the structuring image. The process of dilation followed by erosion is opening. Opening is used for reducing the noise in the binary image, by basically removing objects smaller than 11*11 pixels. The same structuring element has to be used to erode as well as dilate the binary image. The structuring element used was an 11*11 pixel square.

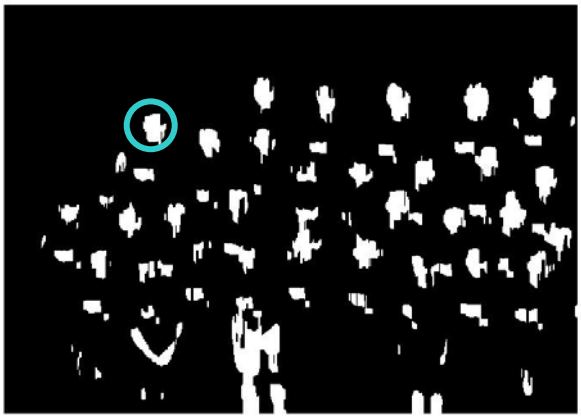
Binary Image before open

Opened Binary Image

D) CLOSING

Closing is defined as erosion of an image followed by dilation of the image. We will first erode the image by the structuring image, and follow this by dilating the image by the structuring image. Closing is used to fill the holes that account for the eyes and the nose in a face in the binary image (see image below). We obtained best results by a square structuring element of 7*7 pixels. As mentioned before, the same structuring element has to be used to erode as well as dilate the binary image.

Binary Image before close



Closed Binary Image

BLOB DETECTION (BLOB-Binary large Object)

A) BLOB LABELING AND STORAGE

Our scheme for blobbing images is to scan the image to detect a "1", and then labeling pixels according to their already labeled neighbors. Once all the connected neighbors of a single blob are labeled with the same label, another "1" is searched, then a new label is used. The labels are usually integers stored in a new 'image' or data structure, which I will call the 'blob image', and are used go associated the corresponding pixels in the original image. For keeping things simple, I have just used minimum x and y, and

maximum x and y coordinates to a blob and assigned a size to the blob -giving the number of pixels in that particular blob.

B) BLOB REJECTION SCHEME:

There are a series of steps that are used before Accepting/Rejecting a particular blob as a face.

I. The minimum width of the blob has to be 20 pixels and the minimum length 25 pixels. Also, the maximum allowed width is 80 pixels and maximum length is 150 pixels.

II. Then aspect ratio of the blob is considered: If the width of the blob is greater than the length, then most probably it's not a face. Also if the length of the blob is much larger than the width then also it's probably not a face. These kinds of long blobs were generally found to be either hands of a person that were not covered by clothes or barks of trees or some other noise of similar features.

III. The final step that we used was a density check. This checked the number of pixels in the blob divided by the number of possible pixels in the bounding rectangle. We found that the best value for a blob to be a face was that its density had to be greater than 50%.

Geometric MOMENTS

Even though we did not use moments in our final program, we still did research on the algorithm and it is worth to mention. Image moments and various moment-based invariants play an important role in image recognition and shape analysis. In our project, we were to use moments as a blob rejection method, which could determine if the blob is an oval shape (a head) or non-oval. The *(p+q)* order geometric moment *Mpq* can be calculated as below:

$$m_{pq} = \sum_{i=1}^{N} \sum_{j=1}^{N} i^p j^q f_{ij}$$

Where *N* is the size of the pixel group and *fij* in this case is either 0 or 1 because our process images are binary images.

However, in the end, we decided not to implement moments as our final step because most of the heads generated from blobbing were not perfect oval due to their hair color or background issue.

5. RESULT AND ANALYSIS

Performance

Below is a statistical table showing the result of all the images we tested on. Noise detected means the number of non-faces objects such as hands, clothing or background that are not able to be filtered out by blob rejection scheme. See Appendix for more processed images.

image	# of Faces	Face Detected	Accuracy	Noise Detected	Error

1	18	7	0.38888889	3	0.16666667
2	12	10	0.83333333	1	0.08333333
3	14	12	0.85714286	2	0.14285714
4	21	14	0.66666667	3	0.14285714
5	11	11	1	4	0.36363636
6	33	26	0.78787879	6	0.18181818
7	21	18	0.85714286	3	0.14285714
8	18	11	0.61111111	7	0.38888889
9	17	7	0.41176471	4	0.23529412
10	17	6	0.35294118	2	0.11764706
Avg.	182	122	**0.67032967**	35	**0.19230769**

$$\text{S Accuracy} = \frac{FaceDetected}{\#ofFaces}, \text{ Error} = \frac{NoiseDetected}{\#ofFaces}$$

As we can see, most of the accuracies fall into 70%-90%. But some of the pictures have exceptionally low accuracy due to the lighting condition, shadows or skin tone background, which drag down the total accuracy. But overall, we are able to detect 7 out of 10 faces correctly. But the 20% error rate shows the need of improvement of our thresholding and blob rejection methods. However, we realize there are always a tradeoff between accuracy and error. We noticed that our threshold works relatively well in most conditions but have trouble at differentiating skin under shadow from non-uniform lighting condition and it performs poorly with a special kind of pinkish red color which is very close to reddish faces skin color. In such scenario, a lot of noises will be detected which affects our accuracy rate.

Memory & Speed

There is no input or output data rate because we use static images as our input. All the data memories are allocated by malloc which are stored in SDRAM0.

Image Data – 1024*720*2 = 1478880 bytes

Filter Kernels – 10*10 + 15*15 = 325 bytes

Blob data – Varies, depends on the number of blobs produced in each image

The total code size is 824 bytes stored in on-chip memory.

Skin Segmentation – 292 bytes

Morphological (144 bytes): Opening – 132 bytes

Closing – 136 bytes

Blobbing – 264 bytes

Below is the profile table indicating the number of cycles for each of the four processing stages and the transfer rate between PC and EVM under different optimization level.

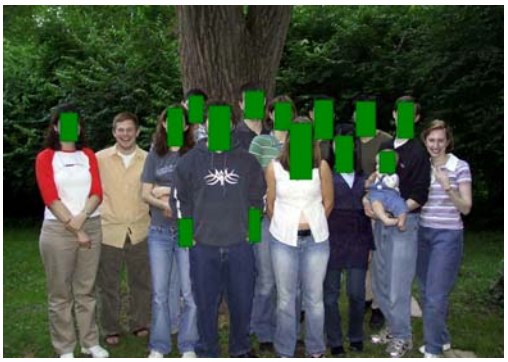
Image	Skin (Cycles)	Open(Cycles)	Close(Cycles)	Blob(Cycles)	Total(Cycles)
o1 no print statements	94411362	656479363	907302241	74538530	1732731496
o3 with print Statements	102848899	628018666	872047018	84737700	1687652283
o3 no print statements	94078657	627167904	866912854	84720524	1672879939

Avg Transfer Rate = 4.413333 MB/s (Obtained by dividing total transferred data with total transfer time)

The profile stats show that level 3 optimization with no printf statements has the best performance results for our system in the EVM.

6. DEMO

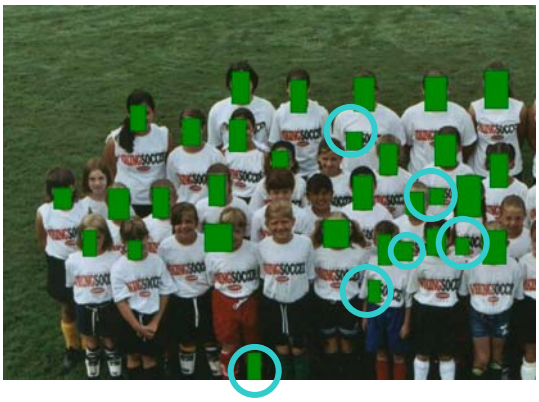
The demo consists of showing a series of 5 pre-selected processed images and 1 random image that has never been processed before through its 3 steps of skin segmentation, morphological processing and blob rejection. The final output image will be the original color image with green rectangles covering detected face blobs.



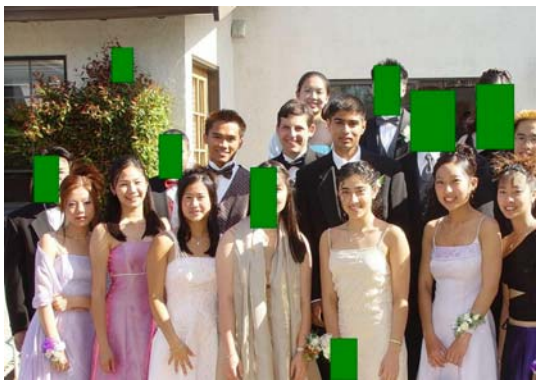
Outdoor Image

Indoor Image

As mention earlier in the paper, the skin segmentation range we chose has poor performance on a specific pinkish red and skin under shadow. Our system also has problem detecting faces from people wearing skin color clothing. These drawbacks are shown in the following two processed images.



Pinkish red color



Shadowed skin + Skin color clothing

The pinkish red color corresponds to certain skin tone group's skin color and could not be discounted if we want to use a universal threshold to include all skin tone. We have isolated the Cr Cb values for that red color which corresponds to Cr[155,160], Cb[110,120].

When we incorporate the new threshold limit to filter out the red color, the system correctly filter noise in images with the red color but it also filters out skin areas from images with people having reddish skin color. After going through our 10 test images the results were disappointing, the tradeoff for incorporating the red color filter scheme is too high to be used, and thus we decided to stay with our original threshold.

For skin color clothing, a suggestion has been made to use center of blob intensity (average index of the white pixels in a blob) to help identify a blob consist of a face connected to a skin color dress as a single face. In our implementation of the center of blob intensity rejection test, which rejects blobs with center that deviate from the blob frame center (center point of the max,min index of blob) for more than 30%, results were inconsistent in different types of images. For images with small to medium size blobs, the center of intensity filter out a lot of unwanted noises and reduced false detection. But it did no work the same for our intended large blobs (face with skin color clothing), these

blobs were still being rejected discounting the face. Further investigation is needed to find the appropriate deviation before the center of intensity rejection scheme could be used.

People standing next to each other or rows of people create addition problem for face detection. If a face is next to another face or an arm, both the face and the arm would be detected as a single blob and will be rejected due to its awkward aspect ration. There is currently no counter measure to such problems that could be implemented with the C67 EVM. A possible solution is to perform a template matching with every single blob with a standard face template, this could be done using correlation but would requires additional memory and computation cycles which is not feasible with our current system setup.

An error is also detected in our morphological processing algorithms where we use preset structural elements to erode and dilate the binary image. We perform kernel filtering on the whole image at once and there are situation where faces that are not connected in the initial threshold image become connected after opening and closing operations. A possible solution is to change the image processing sequence to perform blobbing first then conduct morphological processing on each blob individually. The could potentially eliminate the problem of connecting skin at close proximity to a single large blob.

7. CONCLUSION

The face detection system we have developed for this project met our goals of creating a face detection system that runs entirely on the signal processor. It also

performs relatively fast 12 seconds compared to 26 seconds for **Diedrick Marius's** group. Our system is able to maintain consistent accuracy in both outdoor and indoor images and in all skin tone segments except for brown skin tone color where it has trouble during skin segmentation stage.

Future expansion of our system includes further development in the initial binary image generation, including a more refined skin segmentation threshold and using a different colorspace such as HSV or incorporating the Y component into the threshold parameters. Development in the area of template matching or other methods to separate connected faces could be considered as this will allow the system to function with images with a close cluster of people. Other than improving accuracy, the next step is to improve performance to real-time processing with the possibility of creating a face tracking system for surveillance or child monitoring. The face detection techniques discussed in this paper could also be used in other image processing projects such as object detection for plants and other object with a characteristic color cluster at a certain colorspace region that could be identified with color threshold.

8. REFERENCE

Threshold Values/ Skin Segmentation:

http://www.ee.cuhk.edu.hk/~knngan/TCSVT_v9_n4_p551-564.pdf

Pg 555 2nd paragraph

http://www.computer.org/cspress/CATALOG/p2122.htm (IEEE conference)

The above website was used to determine the Threshold values for Cb Cr components.

We basically fine tuned / Trained the System according to the skin color values of images in our database to improve accuracy.

Erosion/Dilation - Opening/Closing

http://www.personal.psu.edu/users/y/z/yzz100/pj/Morphol.c

The C Code for Erosion and Dilation had been implemented as an example with reference to all the images given in the link below.

http://www.personal.psu.edu/users/y/z/yzz100/pj/pj3.html

The only issue that I encountered was that the structuring element that this code provided was using structuring elements of a size as small as 4*4 pixels, but for our purposes we needed a flexible structuring element which was created easily as the code provides a structure of the element itself which can be extended to any size or shape.

The TI website was also searched for this purpose, and as discussed in class we found the above code more useful and flexible.

Blobbing :

The C Code for this process was written entirely by us, without referring to any code online as we wanted to keep it simple, short and optimized. A recursive function was written to optimize the 4Way connect algorithm and then the coordinates of each blob were stored in a separate blob types and processed thereafter for rejections.

Appendix


JUL 29 2007


www.shootoo.com