

# **INTERACTIVE FACE RECOGNITION**

**By**  
**Nishanth Vincent**  
**nishanth.fairfield@gmail.com**

**A Thesis submitted to the Graduate Faculty of Fairfield  
University in partial fulfillment of the requirements for the  
degree of A Master of Science in the Electrical and Computer  
Engineering program**

**Advisor: Professor Douglas A. Lyon, Ph.D.**

Electrical and Computer Engineering Department  
Fairfield University, Fairfield CT 06430

## **Acknowledgement**

I would like to express my deep and sincere gratitude to my advisor, Professor, and Chair of the Electrical and Computer Engineering Dr. Douglas A. Lyon, PhD. His wide knowledge and his logical way of thinking have been of great value to me. His understanding, encouraging and personal guidance have provided a good basis for the present thesis.

I am deeply grateful to my Dean, Dr.Hadjimichael PhD., for his detailed and constructive comments, and for his important support throughout this work.

I owe my loving thanks to my Parents. They have lost a lot due to my research abroad. Without their encouragement and understanding it would have been impossible for me to finish this work. My special gratitude is due to my brothers for their loving support.

## Table Of Contents

Abstract-----	4
1.Introduction-----	4
1.1 Problem Definition-----	4
1.2 Motivation-----	5
1.3 Approach-----	5
1.4 Ethics and Societal Implications-----	6
2. Literature Survey-----	7
3. Hardware and Software-----	10
4. Experiments On Images-----	12
4.0 Camera-----	12
4.1 Face detection-----	13
4.2 YIQ color model-----	14
4.3 YCbCr color model-----	15
4.4 Binary Image Processing -----	17
4.5 Blob Detection-----	18
4.6 Face Recognition-----	19
4.7 Analysis-----	20
5. Conclusion -----	25
6. Literature Cited-----	26
7. Appendix-----	30

## Abstract

This paper describes the design and construction of a test-bed for prototyping of embedded face detection and recognition algorithms. The test-bed is called the PITS (Portable Interactive Terrorist Identification System), it makes use of a hand-held device called the *Sharp Zaurus*. The embedded device has a processor, camera, color display, and wireless networking. This system is different from existing systems because of its embedded information technologies. The embedded device performs both detection and recognition. We present a skin color approach in the YCbCr color space for fast and accurate skin detection. We then process this image using a combination of morphological operators and elliptical shape of faces to segment faces from the other skin colored regions in an image. An eigenface algorithm processes the segmented faces and matches the face to a face database.

## 1. Introduction

Face detection locates and segments face regions in cluttered images. It has numerous applications in areas like surveillance and security control systems, content-based image retrieval, video conferencing and intelligent human computer interfaces. Some of the current face-recognition systems assume that faces are isolated in a scene. We do not make that assumption. The system segments faces in cluttered images [2].

With a portable system, we can ask the user to pose for the face identification task. This can simplify the face-detection algorithm. In addition to creating a more cooperative target, we can interact with the system in order to improve and monitor its detection. The task of face detection is seemingly trivial for the human brain, yet it remains a challenging and difficult problem to enable a computer /mobile phone/PDA to do face detection. The human face changes with respect to internal factors like facial expression, beard, mustache, glasses, etc. is sensitive to external factors like scale, lighting conditions, and contrast between face, background and orientation of face. Thus, face detection remains an open problem. Many researchers have proposed different methods for addressing the problem of face detection. Face detection is classified into feature-based and image-based techniques. The feature-based techniques use edge information, skin color, motion, symmetry, feature analysis, snakes, deformable templates and point distribution. Image-based techniques include neural networks, linear subspace methods, like eigen faces [1], fisher faces etc. The problem of face detection in still images is more challenging and difficult when compared to the problem of face detection in video, since motion information can lead to probable regions where faces could be located.

### 1.1 Problem definition

We are given an input scene and a suspect database, the goal is to find a set of possible candidates. We are subject to the constraint that we are able to match the faces from the scene in an interactive time and that our algorithm is able to run

on the given embedded hardware.

## 1.2 Motivation

Face detection plays an important role in today's world. It has many real-world applications like human/computer interface, surveillance, authentication and video indexing. However research in this field is still young. Face recognition depends heavily on the particular choice of features used by the classifier. One usually starts with a given set of features and then attempts to derive a optimal subset (under some criteria) of features leading to high classification performance with the expectation that similar performance can also be displayed on future trials using novel (unseen) test data.

Interactive Face Recognition (IFR) can benefit the areas of: Law Enforcement, Airport Security, Access Control, Driver's Licenses & Passports, Homeland Defense, Customs & Immigration and Scene Analysis. The following paragraphs detail each of these topics, in turn

*Law Enforcement:* Today's law enforcement agencies are looking for innovative technologies to help them stay one step ahead of the world's ever-advancing terrorists.

*Airport Security:* IFR can enhance security efforts already underway at most airports and other major transportation hubs (seaports, train stations, etc.). This includes the identification of known terrorists before they get onto an airplane or into a secure location.

*Access Control:* IFR can enhance security efforts considerably. Biometric identification ensures that a person is who they claim to be, eliminating any worry of someone using illicitly obtained keys or access cards.

*Driver's Licenses & Passports:* IFR can leverage the existing identification infrastructure. This includes, using existing photo databases and the existing enrollment technology (e.g. cameras and capture stations); and integrate with terrorist watch lists, including regional, national, and international "most-wanted" databases.

*Homeland Defense:* IFR can help in the war on terrorism, enhancing security efforts. This includes scanning passengers at ports of entry; integrating with CCTV cameras for "out-of-the-ordinary" surveillance of buildings and facilities; and more.

*Customs & Immigration:* New laws require advanced submission of manifests from planes and ships arriving from abroad; this should enable the system to assist in identification of individuals who should, and should not be there.

## 1.3 Approach

The basic algorithm starts with a pre-processing step, consisting of digitization and segmentation. The next step is called face segmentation. We define the face segmentation problem as: given a scene that may contain one or more faces, create sub-images that crop out individual faces. After face segmentation, the device enters into

the *face identification mode*, as shown.

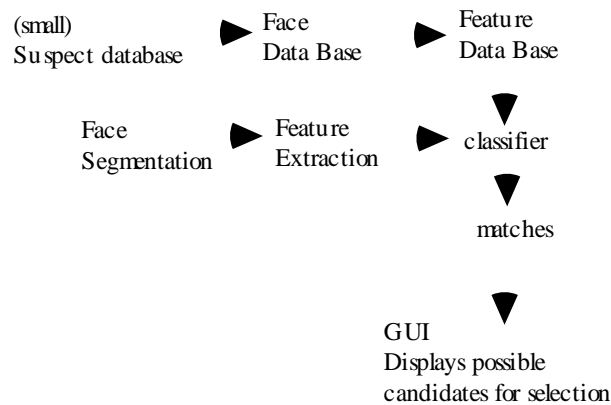


Fig 1.3 Face Identification System

Human skin is relatively easy to detect in controlled environments, but detection in uncontrolled settings is still an open problem [6.]. Many approaches to face detection are only applicable to static images assumed to contain a single face in a particular part of the image. Additional assumptions are placed on pose, lighting, and facial expression. When confronted with a scene containing an unknown number of faces, at unknown locations, they are prone to high false detection rates and computational inefficiency. Real-world images have many sources of corruption (noise, background activity, and lighting variation) where objects of interest, such as people, may only appear at low resolution. The problem of reliably and efficiently detecting human faces is attracting considerable interest. An earlier generation of such a system has already been used for the purpose of flower identification by [7, 8].

## 1.4 Ethics and Societal Implications

Face detection is the fastest growing biometric technology today [2]. Despite their lingering questions regarding the practical usefulness of facial identification technology, law enforcement and military facial identification systems have been in place for several years without arousing too much controversy. According to industry insiders, this is because these applications have proven quite successful in carrying out specific objectives and the public is often unaware of these uses. After September 11, many of the face recognition companies redoubled its efforts to create reliable facial recognition equipment. According to study, the industry still has a lot of work to do.

Even though there are lots of advantages in a face recognition system. Some people still feel that face recognition system invades privacy of a citizen. Also the accuracy of the

systems is of concern. Even if a subject's face is stored in the database, a disguise or even a minor change in appearance, like wearing sunglasses or wearing or growing a mustache can often fool the system. Even an unusual facial expression can confuse the software. Facial identifiers often cannot distinguish twins. Other factors affecting the reliability of the images are changes in the lighting and the angle at which the photos are taken. The systems often have difficulty recognizing the effects of aging [35].

## **2 Literature Survey**

Face detection is a process that determines whether or not there are any faces in an image. Face detection is not an easy process as it is governed by lot of external and internal factors which affect the detection. Even if a subject's face is stored in the database, a disguise or even a minor change in appearance, like wearing sunglasses or wearing or growing a mustache can often fool the system. Even an unusual facial expression can confuse the software. Facial identifiers often cannot distinguish twins. Different illuminations deform faces significantly. There are several algorithms available in the literature that can solve this problem. A survey on face detection with more than 150 references appears in [29]. There are two categories of algorithms in Face detection

- Feature based approach[13]
- Image based approach[4]

Feature-based approach requires prior information of the face. It makes an explicit use of facial features which includes color, shape and component features.

Image-based approach does direct classification without any face knowledge derivation and analysis. It incorporates facial features implicitly into the system through training. Some of the feature based and image based algorithms are

### **Feature based algorithms**

#### **Color segmentation algorithms**

There are several color-segmentation algorithms available which are effective for face detection. Some of them are listed below.

A detailed experimental study of face detection algorithms based on “Skin Color” was read. Three color spaces, RGB, YCbCr and HSI are of main concern. They compared the algorithms based on these color spaces and have combined them to get a new skin-color based face-detection algorithm that improves accuracy. Experimental results show that the proposed algorithm is good enough to localize a human face in an image with an accuracy of 95.18% [2].

Another face detection algorithm uses color images in the presence of varying lighting

conditions as well as complex backgrounds. The method detects skin regions over the entire image, and then generates face candidates based on the spatial arrangement of these skin patches. The algorithm constructs eye, mouth, and boundary by using a transfer of color space from RGB to YCbCr maps for verifying each face candidate [13].

### Edge-detection algorithms

Edge detection detects outlines of an object and boundaries between objects and the background in the image.

The Roberts' Cross algorithm performs is an edge detection algorithm that performs a two dimensional spatial gradient convolution on the image. The idea is to bring out the horizontal and vertical edges individually and then to put them together for the resulting edge detection [19].

The Sobel edge detector is similar to that of the Roberts' Cross algorithm. Both former and the latter use two kernels to determine edges running in different directions. The main difference is the kernels that each of these operator uses to obtain these initial images. Roberts' Cross kernels are designed to detect edges that run along the vertical axis of 45 degrees and the axis of 135 degrees whereas the Sobel kernels are more apt to detect edges along the horizontal axis and vertical axis [19]

### Template matching algorithms

Cross correlation is a template matching algorithm that estimates the correlation between two shapes that have a similar orientation and scale. Consider two series  $x(i)$  and  $y(i)$  where  $i=0,1,2,...N-1$ . The cross correlation  $r$  at delay  $d$  is defined as

$$r = \frac{\sum_1 [(x(i) - m_x) * (y(i-d) - m_y)]}{\sqrt{\sum_1 (x(i) - m_x)^2} \sqrt{\sum_1 (y(i-d) - m_y)^2}}$$

Where  $m_x$  and  $m_y$  are the means of the corresponding series. If the above is computed for all delays  $d=0, 1, 2,.. N-1$  then it results in a cross correlation series of twice the length as the original series.

$$r(d) = \frac{\sum_1 [(x(i) - m_x) * (y(i-d) - m_y)]}{\sqrt{\sum_1 (x(i) - m_x)^2} \sqrt{\sum_1 (y(i-d) - m_y)^2}}$$

There is the issue of what to do when the index into the series is less than 0 or greater than or equal to the number of points. ( $i-d < 0$  or  $i-d \geq N$ ) The most common approaches are to



either ignore these points or assuming the series  $x$  and  $y$  are zero for  $i < 0$  and  $i \geq N$ . In many signal processing applications the series is assumed to be circular in which case the out of range indexes are "wrapped" back within range, i.e.:  $x(-1) = x(N-1)$ ,  $x(N+5) = x(5)$  etc

The range of delays  $d$  and thus the length of the cross correlation series can be less than  $N$ , for example the aim may be to test correlation at short delays only. The denominator in the expression above serves to normalize the correlation coefficients such that  $-1 \leq r(d) \leq 1$ , the bounds indicating maximum correlation and 0 indicating no correlation. A high negative correlation indicates a high correlation but of the inverse of one of the series but of the inverse of one of the series. It is quite robust to noise, and can be normalized to allow pattern matching independently of brightness and offset in the images [3].

We find the cross-correlation algorithm to be of limited utility because of its assumption on geometric scale and orientation of the templates.

### **Gray-scale algorithms**

This gray-scale algorithm was suggested by Yang and Huang [33], who observed that when the resolution of a face image is reduced gradually either by sub sampling or averaging, macroscopic features of the face will disappear and that at low resolution, face region will become uniform.

### **Image based algorithms**

- Statistical approach
- Neural networks[4]

Many commercial applications of face recognition are also available such as security system, criminal identification, and film processing. Like face detection face recognition can also be categorized into tree types. They are

- Feature-based approach,
- Holistic approach and
- Hybrid approach.

### **Feature based Approach**

In feature based methods, local features such as eyes, nose,

and lips are segmented which is then used as an input data for structural classifier. Hidden Markov model and dynamic link architecture fall under this category.

### **Holistic Approach**

In holistic methods, the face as a whole is taken as input data. One of the main algorithms that fall under this category is the eigenface method

Eigenface method is based on the implementation of Principal Component Analysis (PCA) over images. In this method, the features of the studied images are obtained by looking for the maximum deviation of each image from the mean image. This variance is obtained by getting the eigenvectors of the covariance matrix of all the images. The eigenface space is obtained by applying the eigenface method to the training images. Later, the training images are projected into the eigenface space. Next, the test image is projected into this new space and the distance of the projected test image to the training images is used to classify the test image [1]. Other examples of holistic methods are fisherfaces and support vector machines [1] [16] [17].

### **Hybrid Approach**

The idea of this method comes from how human vision system sees both face and local features (includes nose, lips and eyes). Some of the examples in hybrid approach are modular eigenfaces and component-based methods [6].

Even though there are wide range of algorithms available for both face detection and recognition. Tuning these algorithms on to our embedded system will be a real challenge [5].

## **3. Hardware and Software**

The IFR system is a stand-alone GUI implementation on the Sharp Zaurus SL-6000L. The Zaurus is provided with a 400MHz processor, 64 MB RAM, and Compact Flash and Serial Device ports. It is equipped with a Sharp CE-AG06 camera attachment which is inserted into the Compact Flash port. The operating system is embedded Linux with Personal Java support. All code was written to Personal Java specifications. The code was migrated from a laptop to the Zaurus. In addition to that, the embedded device is provided with color display, wireless networking card and a QWERTY key board.



Fig 3.1 Sharp Zaurus sl-6000L



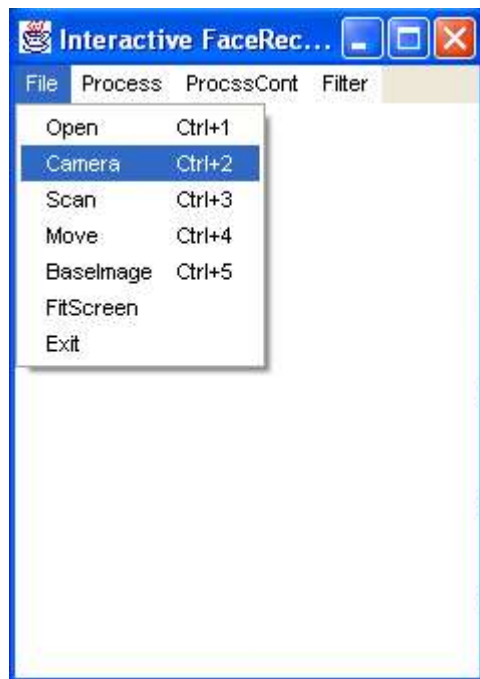
Fig 3.2 Camera

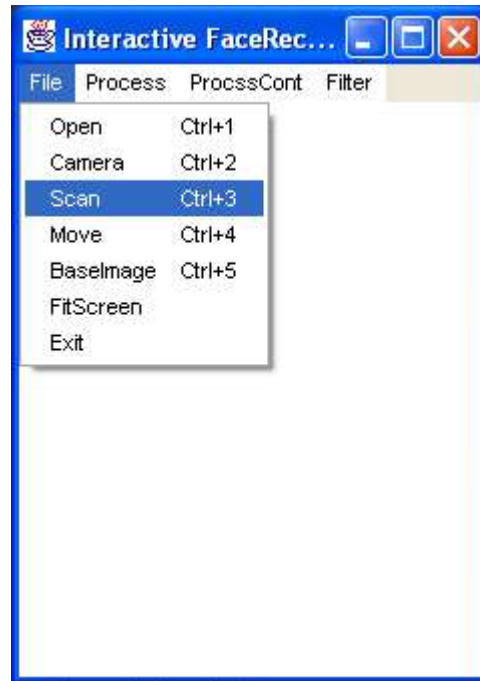


Fig 3.3 GUI for Zaurus

#### 4.0 Camera

The Sharp Zaurus SL-6000L is provided with CE-AG06 camera attachment is inserted into the Compact Flash port which allows direct capture of images. The source code for the camera is in C so we call the executable at runtime using java. There is a scan option in our GUI menu which will load the image that is being recently captured. The source code for calling the executable is given below.





```
private void camera() {
    try {

        Runtime.getRuntime ().
            exec
            ("/home/QtPalmtop/bin/./sq_camera");
    } catch (IOException ioe) {
        ioe.printStackTrace ();
    }
}
```

## 4.1 Face Detection

The first stage in face detection is to perform skin detection. Skin detection can be performed in a number of color models. To name a few are RGB, YCbCr, HSV, YIQ, YUV, CIE, XYZ, etc. An efficient skin detection algorithm is one which should be able to cover all the skin colors like black, brown, white, etc. and should account for varying lighting conditions. Experiments were performed in YIQ and YCbCr color models to find out the robust skin color model.

## 4.2 YIQ Color Model

YIQ color model belongs to the family of television transmission color models. This color model is defined by the National Television Systems Committee (NTSC). This color space is used in televisions in the United States. The main advantages of this format is that grayscale information is separated from color data, so the same signal can be used for both color and black and white sets. In this color model, image data consists of three components: luminance (Y) which represents grayscale information, while hue (I) and saturation (Q) represents the color information. The following conversion is used to segment the RGB image into Y, I and Q components. Fig 4.2.1 shows the conversion of a RGB color model in to a YIQ color model and Fig 4.2.2 shows the skin threshold in the YIQ color model.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.212 & -0.523 & -0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(1)



Fig 4.2.1 RGB  $\rightarrow$  YIQ



Fig 4.2.2 Skin threshold in YIQ

### Threshold

The thresholds are selected by repeated experimentations. We have arrived at the following code:

```
if (      (Y[x][y] < 223) &&
          (Y[x][y] > 44) &&
          (I[x][y] < 0) &&
          (I[x][y] > 64)
      )
    setPixel(x, y, 255);
    else
        setPixel(x, y, 0);
}
```

### 4.3 YCbCr Color Model

YCbCr color model also belongs to the family of television transmission color models. In this color model, the luminance component is separated from the color components. Component (Y) represents luminance, and chrominance information is stored as two color-difference components. Color component

Cb represent the difference between the blue component and a reference value and the color component Cr represents the difference between the red component and a reference value. The following conversion is used to segment the RGB image into Y, Cb and Cr components:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (2)$$

Among all the color models found, YCbCr seems to be better for skin detection since the Colors in YCbCr are specified in terms of luminance (Y channel) and chrominance (Cb and Cr channels).The main advantage of converting the image from RGB color model to the YCbCr color model is the influence of luminance can be removed during our image processing. We deal only with Cb and Cr components to perform skin detection. From analysis, we found that the Cb and Cr components give a good indication on whether a pixel is part of the skin or not. . Fig 4.3.1 shows the conversion of a RGB color model in to a YCbCr color model. Fig 4.3.2 shows the skin threshold in the YCbCr color model



Fig 4.3.1 RGB → YCbCr



Fig 4.3.2 Skin threshold in YCbCr



## Threshold

The thresholds are selected by repeated experimentations. We have arrived at the following code

```
if (      (Cb[x][y] < 173) &&
          (Cb[x][y] > 133) &&
          (Cr[x][y] < 127) &&
          (Cr[x][y] > 77)
    )
    setPixel(x, y, 255);
    else
        setPixel(x, y, 0);
}
```

From the above figures, we found that YCbCr color model is more efficient than YIQ color model for skin detection.

## 4.4 Binary Image Processing

Depending on the Cb and Cr threshold values a binary image is obtained with the skin regions masked in white and the non skin regions masked in black. This mask is further refined through morphological operators. The two basic morphological operators used are erosion and dilation.

Erosion is defined as a morphological operator which is usually applied to binary images. It is used to erode away the boundaries of regions of foreground pixels. Thus the areas of foreground pixels shrink in size, and holes within those areas become larger [18] [22]. Equation 3 defines the basic morphological operator erosion on sets A and B

$$\text{erosion: } A \ominus B = \bigcap_{b \in B} T_{-b}(A) \quad (3)$$

Dilation is defined as a morphological operator, which is usually applied to binary images. The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels [18] [22]. Equation 4 defines the basic morphological operator dilation on sets A and B

$$\text{dilation: } A \oplus B = \bigcup_{b \in B} T_{-b}(A) \quad (4)$$

The image is first eroded to eliminate small background objects and separate individual faces.

This eroded image is then dilated to refill gaps within the faces. The blobs that are

elliptical in shape are termed as faces while the other blobs are rejected.



Fig 4.4.1 Eroded Image

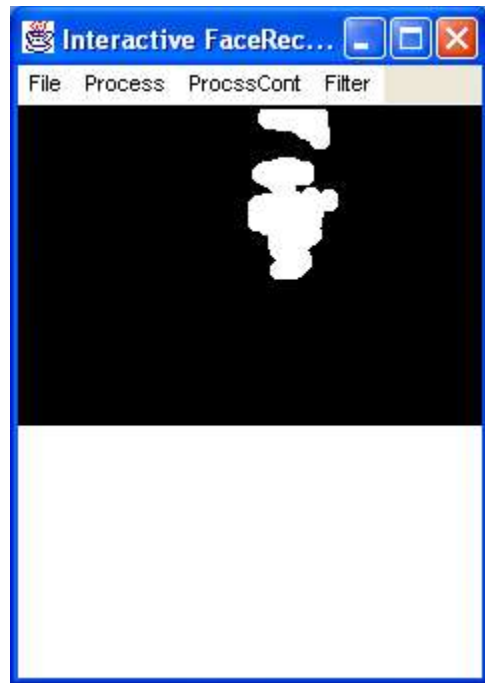


Fig 4.4.2 Dilated Image

## 4.5 Blob detection

We used an open GL blob detection library. This library designed for finding 'blobs' in an image, i.e. areas whose luminosity is above or below a particular value. In our case it is just a binary image (black and white). It computes their edges and their bounding box. This library does not perform blob tracking; it only tries to find all blobs in each frame it was fed with.

Blobs in the image which are elliptical in shape are detected as faces. The blob detection algorithm draws a rectangle around those blobs by calculating information such as position and center.

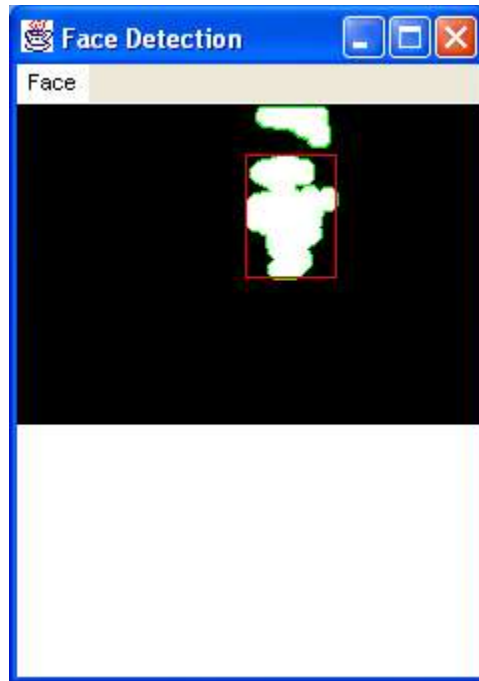


Fig 4.5.1 Blob detected Image

## 4.6 Face Recognition

Face recognition is performed by an algorithm called principal component analysis (PCA) also known as Karhunen-Loeve's transform [1]. It is a method of identifying patterns in data. It is mainly useful in expressing the data in such a way which will highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analyzing data.

It is essential to know the following elementary background mathematical skills to understand the process of Principal Components Analysis.

- Standard deviation,
- Variance,
- Covariance,
- Eigen vectors and
- Eigen values

A small database is created with images. Each of these images are  $m$  pixels high and  $n$  pixels wide For each image in the database an image vector is created and are put in a matrix form which gives a start point for PCA. Covariance is found from the matrix of images and from the covariance the eigen vectors

are found for the original set of images. The way this algorithm works is by treating face recognition as a "two-dimensional recognition problem, taking advantage of the fact that faces are normally upright and thus may be described by a small set of 2-D characteristics views. Face images are projected onto a feature space ('face space') that best encodes the variation among known face images. The face space is defined by the eigenfaces which are the eigenvectors of the set of faces; they do not necessarily correspond to isolated features such as eyes, ears, and noses. So when a new image is passed from the blob detected image, the algorithm measures the difference between the new image and the original images, not along the original axes, but along the new axes derived from the PCA analysis. It proves out that these axes works much better for recognizing faces, because the PCA analysis has given us the original images in terms of the differences and similarities between them. The PCA analysis has identified the statistical patterns in the data [18] [36]

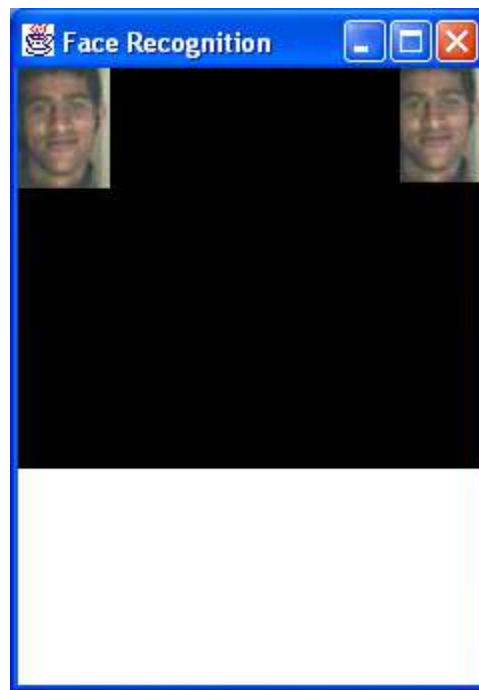


Fig 4.6.1 Face Recognized Image

## 4.7 Analysis

Face detection was tried on different complex images. The algorithm works fairly well in detecting faces. The performance of the algorithm in detecting faces is above 85%. Few of the images that were tried are shown below. Fig 4.7.1, fig 4.7.4 and fig 4.7.9 show the input images fed in to the GUI. Fig 4.7.8 shows the database used for face recognition. Fig 4.7.2, fig 4.7.5 and fig 4.7.10 show's the image after skin segmentation and binary image processing. Fig 4.7.3, fig 4.7.6 and fig 4.7.11 show's the face detected image. Fig 4.7.7 and fig 4.7.12 show's the face recognized image. The performance of recognizing 50x60 face images using PCA is approximately 76%.

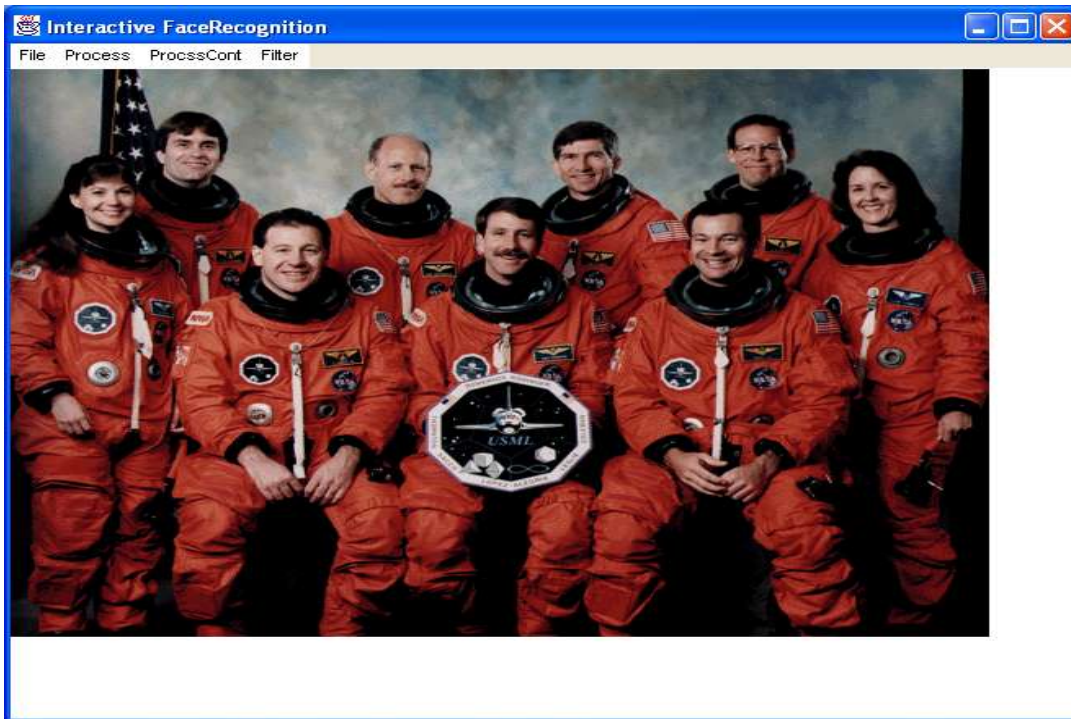


Fig 4.7.1 Input image

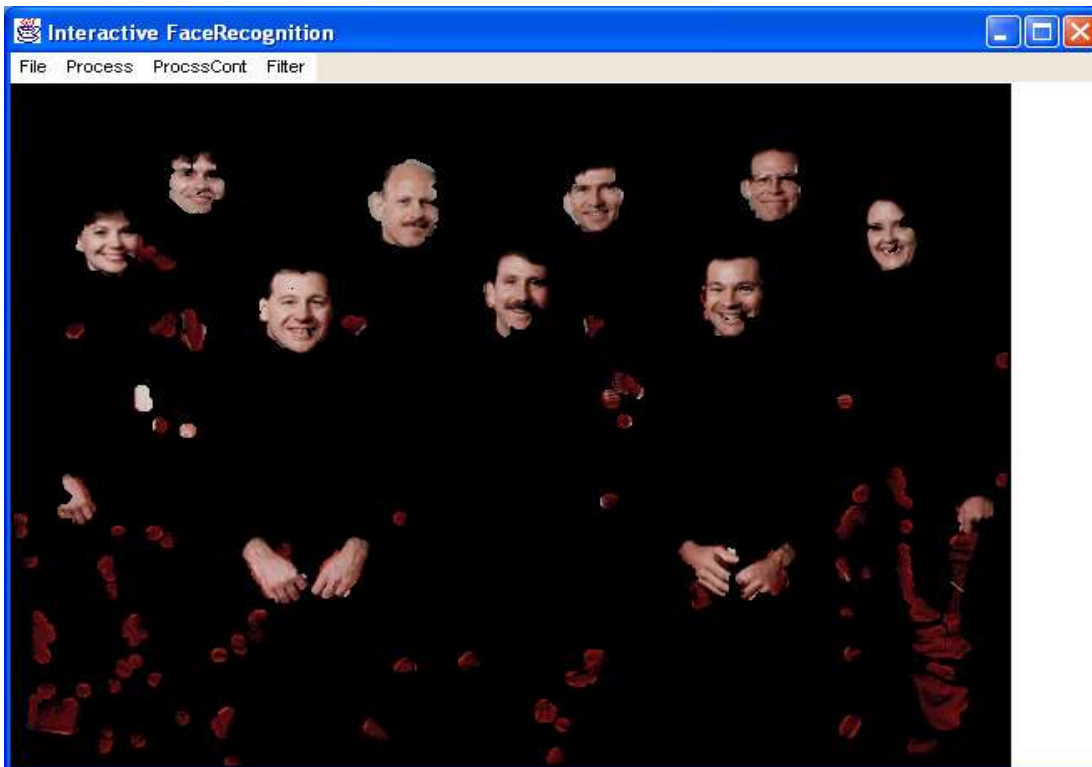


Fig 4.7.2 Skin detected image

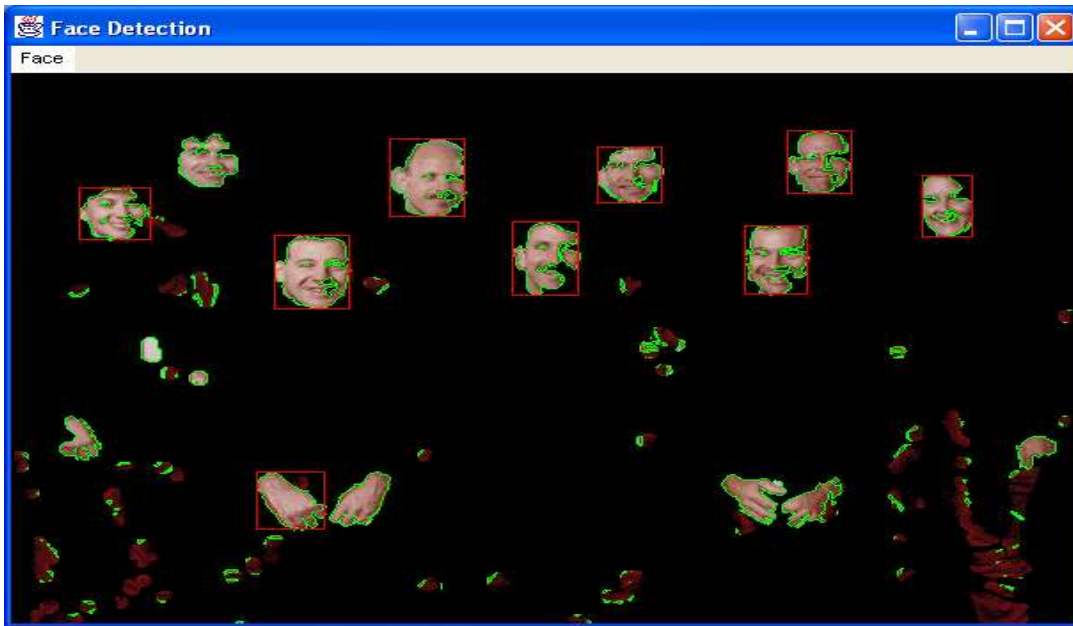


Fig 4.7.3 Face detected image



Fig 4.7.4 Input image

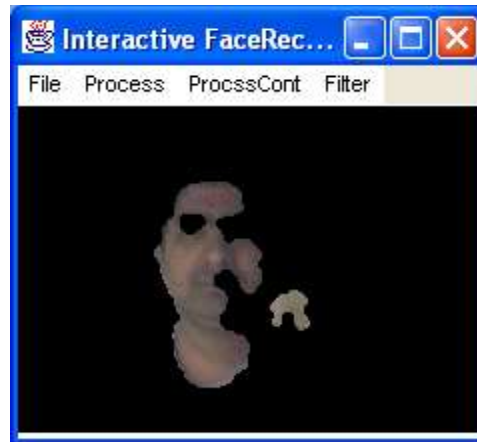


Fig 4.7.5 skin detection

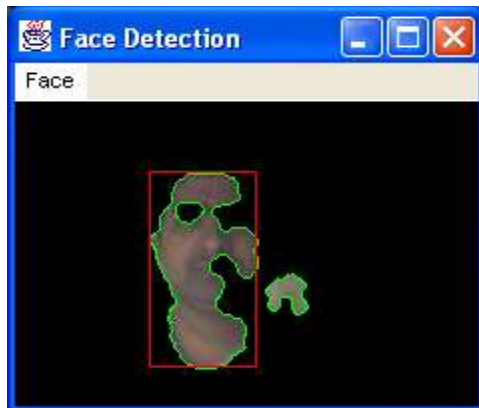


Fig 4.7.6 Face detection



Fig 4.7.7 Face recognition

## Implementation On The Zaurus

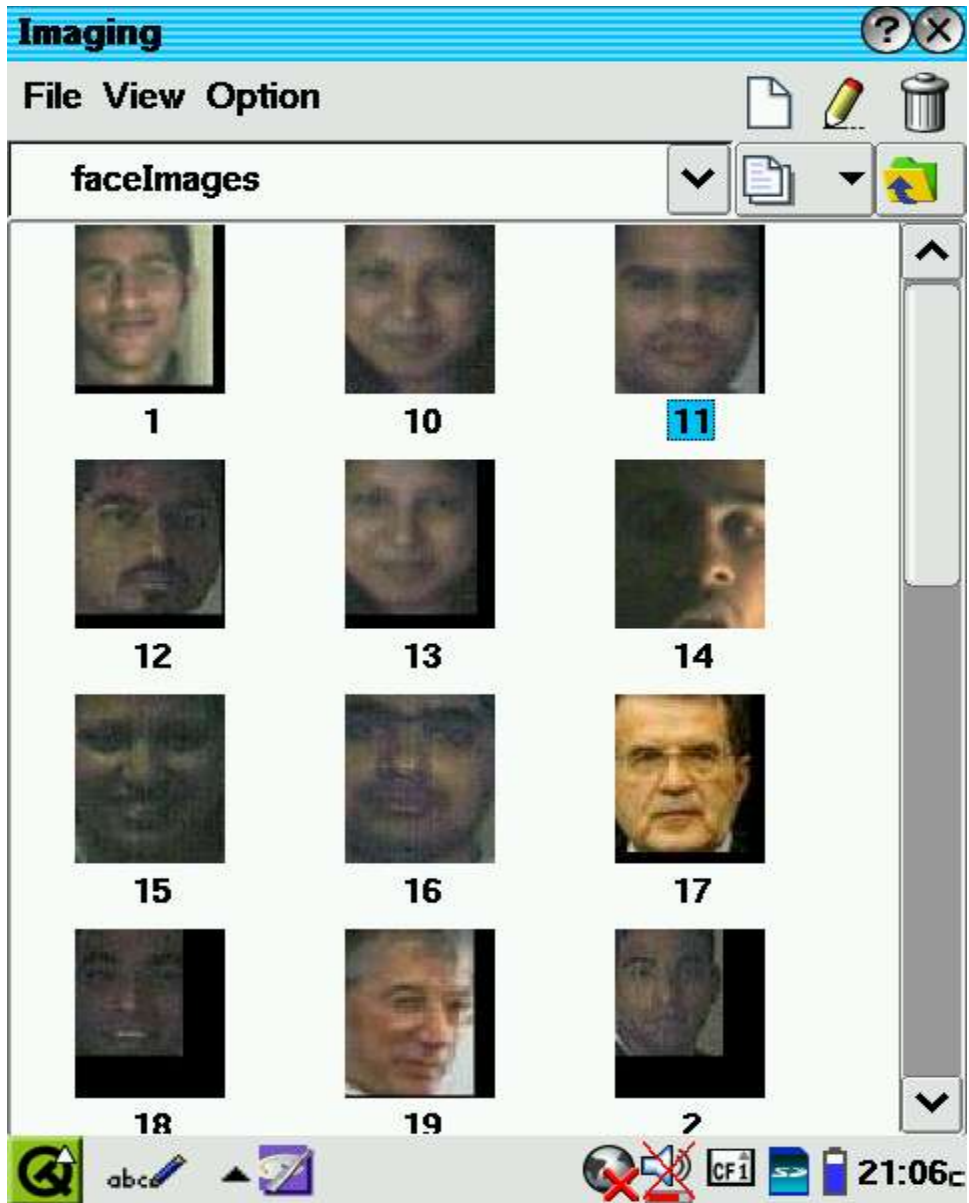


Fig 4.7.8 Face Recognition database created in the Zaurus.



Fig 4.7.9 Input image

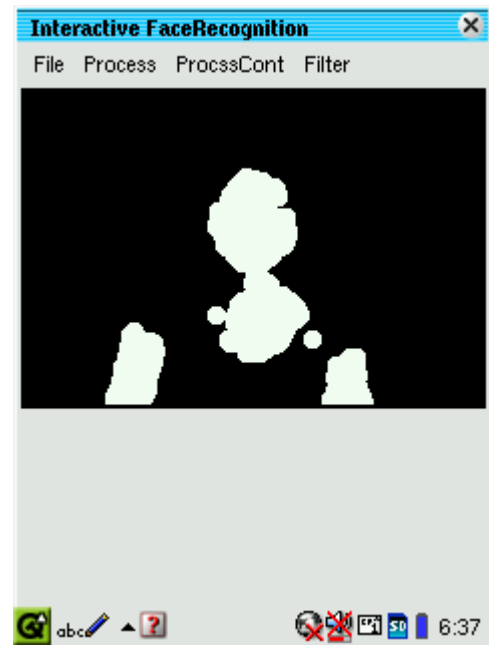


Fig 4.7.10 skin detection



Fig 4.7.11 Face detection



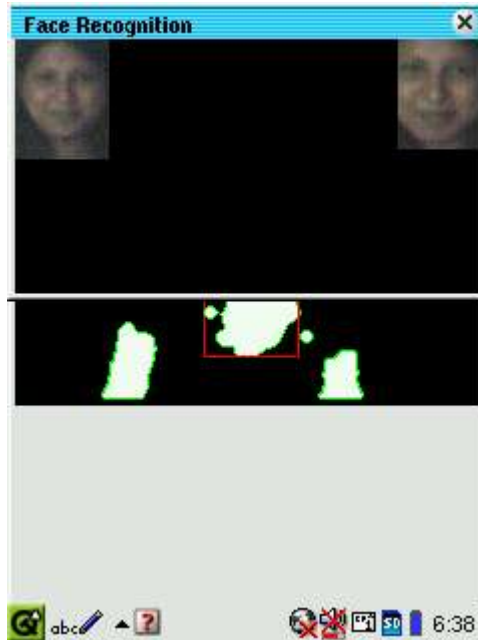


Fig 4.7.12 Face recognition

## Demo

For the demo we created a small database of my roommates along with some other pictures. The images are 50x60 pixels each. The demo will be shown on the laptop which will be VNC'd to the Zaurus so that everybody can see it on the projector.

## 5. Conclusion

In this paper we have presented an interactive face recognition algorithm on the embedded device. The device performs both face detection and recognition for color images. Face detection is performed in the YCbCr color space to provide a fast and accurate detection. The performance of the algorithm in detecting faces is above 85%. Face recognition is performed using PCA and the performance is found to be approximately 76%. Our work is significantly novel compared to the previous work for the fact that we are able to match the faces from the scene in an interactive time and that our algorithm is able to run on the given embedded hardware. Our future work will focus on improving the efficiency of the algorithm. Finally, we conclude saying that the Interactive Face Recognition device is a test bed for embedded face recognition research. As such, it contributes toward building a general infrastructure for research into embedded vision, further benefiting society.

## Literature Cited

- [1] M. Turk, A. Pentland. "Eigenfaces for Recognition". Journal of Cognitive Neuroscience. Vol 3, No. 1. 71-86, 1991.
- [2] Sanjay Kr. Singh<sup>1</sup>, D. S. Chauhan<sup>2</sup>, Mayank Vatsa<sup>3</sup>, Richa Singh, A Robust Skin Color Based Face Detection Algorithm, Department of Computer Science and Engineering Institute of Engineering and Technology Jaipur –222002, India. Tamkang Journal of Science and Engineering, Vol. 6, No. 4, pp. 227-234 (2003)
- [3] Paul Bourke, Autocorrelation -- 2D Pattern Identification, <<http://astronomy.swin.edu.au/~pbourke/other/correlate/>>
- [4] Boehme, H.-J., Brakensiek, A., Braumann, U.-D., Krabbes, M., and Gross, H.-M. Visually-Based Human-Machine Interaction in a Neural Architecture. In SOAVE'97 - Selbstorganisation von adaptivem Verhalten, pages 166--175. VDI Verlag, 1997. <http://citeseer.nj.nec.com/125077.html>
- [5] J. Sobottka and I. Pittas. Segmentation and tracking of faces in color images. In Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, vol. 4, pages 236-- 241, IEEE June 1996.
- [6] Wang, C., and Brandstein, M.S., "A hybrid real-time face tracking system" *Proc. ICASSP 1998*, Seattle, WA, May, 1998, pps. 3636-3740
- [7] G. Nagy, J. Zou, "Interactive Visual Pattern Recognition", *Proc. Int. Conf. Pattern Recognition XIV*, vol. 2, pp. 478-481, Quebec City, 2002.
- [8] Arthur Evans, John Sikorski, Patricia Thomas, Sung-Hyuk Cha, Charles Tappert, Computer Assisted Visual Interactive Recognition (CAVIAR) Technology. . IEEE transactions on Electro Information Technology, pages 1--6, May 2005
- [9] Renaud S'eguier, A vary fast adaptive face detection system, Institute of Electronics and Telecommunications of Rennes Sup'elec, Avenue de la Boulaie, 35511 Cesson-S'evign'e, France, Proceeding Visualization, Imaging, and Image Processing 452-456, April 2004
- [10] Peng Wang, Qiang Ji, Multi-View Face Detection under Complex Scene based on Combined SVMs, Department of Electrical, Computer and System. Engineering, Rensselaer Polytechnic Institute, Troy, NY, . IEEE transactions on [Pattern Recognition](#), vol. 4, pages 179—182, Aug. 2004
- [11] Minsick Park, Chang Woo Park, and Mignon Park], Algorithms for Detecting Human Faces Based on Convex hull, Yonsei University, Seoul, Korea, Optics Express, vol. 10,

Issue 6, p.274

[12.] H.-Y. S. Li, Y. Qiao, and D. Psaltis, "Optical network for real time face recognition," *Applied Optics* 32(26), pp. 5026--5035, 1993.

[13] Rein-Lien Hsuy, Student Member, IEEE, Mohamed Abdel Mottalebz, Member, IEEE, Anil K. Jain, Fellow, IEEE, Face Detection in Color Images, *IEEE transactions on pattern analysis and machine intelligence*, vol.24, no.5, may 2002

[14] Jianke Zhu, Mang Vai and Peng Un Mak, Face Recognition, a Kernel PCA Approach, Department of Electrical and Electronics Engineering, Faculty of Science & Technology, University of Macau, Macau SAR, China, Chinese Conference on Medicine and Biology (CMBE'03) at Wuxi, P. R. China, Oct. 24-26, 2003.

[15] [Terence, Rahul, Mathew, Shumeet] Terence Sim, Rahul Sukthankar, Mathew Mulin, Shumeet Baluja, Memory Based Face Recognition for Visitor Identification, The Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA, Proceedings of International Conference on Automatic Face and Gesture Recognition, 2000

[16] FernandoDe La Torre, Michael J.Black 2003 . Internatioal Conference on Computer Vision (ICCV'2001), Vancouver, Canada, July 2001. IEEE 2001

[17] Turk and Pentland, Face Recognition Using Eigenfaces, Method Eigenfaces", *IEEE CH2983-5/91*, pp 586-591 .

[18] Douglas Lyon. *Image Processing in Java*, Prentice Hall, Upper Saddle River, NJ. 1998.

[19] Matthew T. Rubino, *Edge Detection Algorithms*,  
<<http://www.ccs.neu.edu/home/mtrubs/html/EdgeDetection.html> >

[20] A. Pentland, B. Moghaddam, T. Starner, View-Based and Modular Eigenspaces for Face Recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 21-23 June 1994, Seattle, Washington, USA, pp. 84-91

[21] Liang Wang, Tieniu Tan, Weiming Hu, Face Tracking Using Motion-Guided Dynamic Template Matching, National Laboratory of Pattern Recognition. Institute of Automation, Chinese Academy of Sciences, Beijing, P. R. China, 100080, 5th Asian Conference on Computer Vision, 23--25 January 2002, pages 1--6, Melbourne, Australia.

[22] Douglas Lyon, The DocJava Home Page, <<http://www.docjava.com>>.

[23] "The Imperion Threading System" by Douglas A. Lyon, *Journal of Object Technology*, 2004 Vol. 3, No. 7, July-August 2004

- [24] "Project Imperion: New Semantics, Facade and Command Design Patterns for Swing", by Douglas A. Lyon, *Journal of Object Technology*, vol. 3, no. 5, May-June 2004, pp. 51-64.
- [25] "Asynchronous RMI for CentiJ", by Douglas A. Lyon, *Journal of Object Technology*. - vol. 3, no. 3, March-April 2004, pp. 49-64.
- [26] "On the use of a Visual Cortical Sub-band Model for Interactive Heuristic Edge Detection", by Douglas A. Lyon, *International Journal of Pattern Recognition & Artificial Intelligence (IJPRAI)*, Vol. 18, No. 4 (2004), pages 583-606.
- [27] *Java for Programmers*, Prentice Hall, Englewood Cliffs, NJ, 2004.
- [28] "Building Bridges: Legacy Code Reuse in the Modern Enterprise", By Douglas A. Lyon and Christopher L. Huntley, *Computer*, May, 2002, pp. 102-103.
- [29] McKenna, S.J.[Stephen J.], Jabri, S.[Sumer], Duric, Z.[Zoran], Rosenfeld, A. [Azriel], Wechsler, H.[Harry], Tracking Groups of People, *CVIU(80)*, No. 1, October 2000, pp. 42-56.
- [30.] Jeffrey M. Gilbert and Woody Yang. A real-time face recognition system using custom VLSI hardware. In *IEEE Workshop on Computer Architectures for Machine Perception*, pages 58--66, December 1993.  
<<http://citeseer.nj.nec.com/gilbert93realtime.html>>
- [31.] A. Pentland, B. Moghaddam, T. Starner, O. Oliyide, and M. Turk. View-Based and Modular Eigenspaces for Face Recognition. Technical Report 245, M.I.T Media Lab, 1993. <http://citeseer.nj.nec.com/pentland94viewbased.html>
- [32] H. Schneiderman, "Learning Statistical Structure for Object Detection", *Computer Analysis of Images and Patterns (CAIP)*, 2003, Springer-Verlag, August, 2003.
- [33] Yang and Huang 1994. "Human face detection in a complex background." *Pattern Recognition*, Vol 27, pp53-63
- [34] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001, <<http://citeseer.nj.nec.com/viola01rapid.html>>
- [35] Angela Jarvis, < <http://www.forensic-evidence.com/site/ID/facialrecog.html>>
- [36] Konrad Rzeszutek, <<http://darnok.com/projects/face-recognition>>



## APPENDIX A

### IMPLEMENTATION CODE

#### YIQ Color Model

```
public class Yiq extends FloatPlane {

    // Matrix used for conversion from RGB to YIQ
    double A[][] = {
        {0.2989, 0.5866, 0.1144},
        {0.5959, -0.2741, -0.3218},
        {0.2113, -0.5227, 0.3113}
    };

    //Constructor that takes an image
    public Yiq(Image img) {
        super(img);
    }

    // Mat3 is a math utility class for processing 3X3 matrices
    Mat3 rgbn2yiqMat = new Mat3(A);

    //Method used to convert RGB to YIQ color space
    public void fromRgb() {
        convertSpaceYiq(rgbn2yiqMat);
        System.out.println("yiq");
        rgbn2yiqMat.print();
    }

    /*Method that is used for skin detection in the YIQ color
    *space
    * if ((44 < Y < 223) && (0 < I < 64))
    * then we have skin
    */
    public void skinChromaKey() {
        for (int x = 0; x < r.length; x++)
            for (int y = 0; y < r[0].length; y++) {
                if (
                    (r[x][y] < 223) &&
                    (r[x][y] > 44) &&
                )
            }
    }
}
```

```

                (g[x][y] > 0) &&
                (g[x][y] < 64)
            )
            setPixel(x, y, 255);
        else
            setPixel(x, y, 0);
    }

}

//Method used to set pixel to binary in an image
public void setPixel(int x, int y, int v) {
    r[x][y] = v;
    g[x][y] = v;
    b[x][y] = v;
}

}

```

## YCbCr Color Model

```

public class Ycbcr extends FloatPlane{

//Matrix used for conversion from RGB to YCbCr
    double A[][] = {
        {0.299, 0.587, 0.114},
        {-0.16874, -0.33126, 0.50000},
        {0.50000, -0.41869, -0.08131}
    };

// Mat3 is a math utility class for processing 3X3 matrices
    Mat3 rgb2yuvMat = new Mat3(A);

//Constructor that takes an image
    public Ycbcr(Image img) {
        super(img);
    }

//Method used to convert RGB to YCbCr color space

```

```

public void fromRgb() {
    convertSpace(rgb2yuvMat);
    System.out.println("ycbcr");
    rgb2yuvMat.print();
}

//Method used for skin detection in the YIQ color
//space
public void skinChromaKey() {
    for (int x = 0; x < r.length; x++)
        for (int y = 0; y < r[0].length; y++) {
            if (
                (b[x][y] < 173) &&
                (b[x][y] > 133) &&
                (g[x][y] < 127) &&
                (g[x][y] > 77)
            )
                setPixel(x, y, 255);
            else
                setPixel(x, y, 0);
        }
    }

//Method used to set pixel to binary in an image
public void setPixel(int x, int y, int v) {
    r[x][y] = v;
    g[x][y] = v;
    b[x][y] = v;
}
}

```



## Morphological Operators

```
public class MorphUtils {

//Method to perform dilation in an image whose
functionality is to //is to gradually enlarge the
boundaries of regions of foreground //pixels

    public static short[][] dilate(short f[][], float k[[]
[]]) {
        int uc = k.length / 2;
        int vc = k[0].length / 2;
        int w = f.length;
        int h = f[0].length;
        short o[][] = new short[w][h];
        short sum;

        for (int x = uc; x < w - uc; x++) {
            for (int y = vc; y < h - vc; y++) {
                sum = 0;
                for (int v = -vc; v <= vc; v++)
                    for (int u = -uc; u <= uc; u++)
                        if (k[u + uc][v + vc] == 1)
                            if (f[x - u][y - v] > sum)
                                sum = f[x - u][y - v];
                o[x][y] = sum;
            }
        }
        return o;
    }

//Method to perform erosion in an image whose functionality
is //to erode away the boundaries of regions of foreground
pixels. //Thus the areas of foreground pixels shrink in
size and holes //within those areas become larger.

    public short[][] erode(short f[][], float k[[]]) {
        int uc = k.length / 2;
        int vc = k[0].length / 2;
        int w = f.length;
        int h = f[0].length;

        short o[][] = new short[w][h];
        short sum = 0;
```

```

        for (int x = uc; x < w - uc; x++) {
            for (int y = vc; y < h - vc; y++) {
                sum = 255;
                for (int v = -vc; v <= vc; v++)
                    for (int u = -uc; u <= uc; u++)
                        if (k[u + uc][v + vc] == 1)
                            if (f[x - u][y - v] < sum)
                                sum = f[x - u][y - v];
                o[x][y] = sum;
            }
        }
        return o;
    }
}
}

```

## Face Detection and Face Recognition

```

public class FaceDetectionFrame extends Frame implements
ActionListener {

//This object is used to display the graphics of the image
on the //frame
    Display d = new Display();

//vectors are declared to get the X-coordinate, Y-
coordinate, //height and width of the blob
    Vector locationX = new Vector();
    Vector locationY = new Vector();
    Vector imageWidth = new Vector();
    Vector imageHeight = new Vector();

//Blob detection object initialized to find the blobs in
the image //after processing

    BlobDetection theBlobDetection;

//constructor that takes two images as argument
    FaceDetectionFrame(Image img, Image image) {
        super("Face Detection");
        this.rawImg = img;
    }
}

```

```

        this.baseImage = image;

        Menu face = new Menu("Face");
        MenuItem mil = new MenuItem("Detect");
        mil.setShortcut(new MenuShortcut(KeyEvent.VK_0));
        mil.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {

                faceRecognition();

                                }
                        }
);
        face.add(mil);

        MenuBar mb = new MenuBar();
        setMenuBar(mb);
        mb.add(face);

        setSize(240, 320);
        setVisible(true);
        this.add(d);

        face.addActionListener(this);
        addWindowListener(

                new WindowAdapter() {
                        public void windowClosing(WindowEvent
e) {

                                theBlobDetection = null;
                                setVisible(false);
                                //terminate the program
                                }//end windowClosing()
                        }//end WindowAdapter
                )//end addWindowListener

        }

}

```

```

//Method used to detect blobs in the image areas whose
luminosity //is above or below a particular value. In our
case it is just a //binary image (black and white). It
computes their edges and //their bounding box

```

```

public void detectBlob() {

    ImageBean ib = new ImageBean(rawImg);
        theBlobDetection = new BlobDetection(ib.getWidth
            (),
            ib.getHeight());
        theBlobDetection.setPosDiscrimination(false);
// will detect bright areas whose luminosity < 0.38f;
        theBlobDetection.setThreshold(0.38f);
        theBlobDetection.computeBlobs(ib.getPels());
        blobNb = theBlobDetection.getBlobNb();
        rawImg = ib.getImage();
        d. update(this.getGraphics());
        this.add(d);
        setVisible(true);
        repaint();

}

```

//Method for performing face recognition uses principal component //analysis. This algorithm treats face recognition as a two-//dimensional recognition problem, taking advantage of the fact //that faces are normally upright and thus may be described by a //small set of 2-D characterisits views. Face images are projected //onto a feature space ('face space') that best encodes the //variation among known face images. The face space is defined by //the eigenfaces, which are the eigenvectors of the set of faces;

```

public void faceRecognition() {
    try {

        for (int i = 0; i < locationX.size(); i++) {

            ImageBean ib = new ImageBean
                (Integer.parseInt
                (String.valueOf(imageWidth.elementAt
                (i))),

                Integer.parseInt
                (String.valueOf(imageHeight.elementAt
                (i))));
            Image faceDetectedImage = ib.getImage();
            ImageBean ib1 = new ImageBean
                (faceDetectedImage, baseImage);
                ib1.imageRenewed
                (Integer.parseInt(String.valueOf

```

```

        (locationX.elementAt(i))),
                Integer.parseInt(
String.valueOf(locationY.elementAt
(i))),
                Integer.parseInt(
String.valueOf(imageHeight.elementAt
(i))),
                Integer.parseInt(
String.valueOf(imageWidth.elementAt
(i))));
ImageBean ib2 = new ImageBean(50, 60);
Image finalTemp = ib2.getImage();
ImageBean ib3 = new ImageBean(finalTemp
,ib1.getImage());
ib3.imageReCon();

try {
EigenFaceCreator creator = new
EigenFaceCreator();
creator.readFaceBundles
("/usr/mnt.rom/card/faceImages");

String result = creator.checkAgainstNew
(ib3.getImage());
System.out.println("Most closly
resembling: "+result+" with
"+creator.DISTANCE+" distance.");

if(result == null){

        new FaceRecognitionFrame(ib3.getImage
        ());
    }
else
    {

File f = new File
("/usr/mnt.rom/card/faceImages/" +
result);
Image faceRecognizedImage =
Toolkit.getDefaultToolkit().
getImage(f.toString());
MediaTracker mediatracker = new
MediaTracker(new Canvas());
mediatracker.addImage

```

```

(faceRecognizedImage, 1);
    try {
        mediatracker.waitForAll();
    }
    catch (
        InterruptedException e1) {
        e1.printStackTrace();
    }
    mediatracker.removeImage
(faceRecognizedImage);
    ImageBean ib4 = new ImageBean(240, 200);
    ImageBean ib5 = new ImageBean
        (ib4.getImage(), ib3.getImage());
    ib5.imageAddAndDisplay();
    ImageBean ib6 = new ImageBean
        (ib5.getImage(), faceRecognizedImage);
    ib6.imageAddFinalDisplay();
    new FaceRecognitionFrame(ib6.getImage());
    }
} catch (Exception e) { e.printStackTrace(); }

}

} catch (Exception e) {
    e.printStackTrace();
}
}

```

//Override the paint method to display the rawImg on the same //Canvas object, separated by one row of pixels in the background //color. Also draws a rectangle around the blob by calculating //blob information such as position and center.

```

public class Display extends Canvas {
    .
    public void paint(Graphics g) {

        g.drawImage(rawImg, 0, 0, this);
        PGraphics pg = new PGraphics();
        Blob b;
        float eA, eB, eC, eD = 0;
        ImageBean sib = new ImageBean(rawImg);
    }
}

```







```

//Method gets an image from the 2D red, green and blue
arrays
    public static Image getImage(short r[][], short g[][],
short b[][]) {
        int w = r.length;
        int h = r[0].length;
        int pels[] = new int[w * h];
        for (int x = 0; x < w; x++)
            for (int y = 0; y < h; y++)
                pels[x + y * w]
                    = 0xFF000000
                    | ((0xFF & r[x][y]) << 16)
                    | ((0xFF & g[x][y]) << 8)
                    | (0xFF & b[x][y]);
        return Toolkit.getDefaultToolkit().createImage(new
MemoryImageSource(w,
                    h,
                    ColorModel.getRGBdefault(),
                    pels, 0,
                    w));
    }

```

```

//method gets pixel array from an image
public int[] getPels(Image img, int width, int height) {
    pels = new int[width * height];

    PixelGrabber grabber =
        new PixelGrabber(img, 0, 0,
            width, height, pels, 0, width);

    try {
        grabber.grabPixels();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return pels;
}

```

```

//method converts the pixel array in to 2D array of red,
blue and //green

public void pelsToShort(short r[][], short g[][], short b

```

```

[[[]],
                                     int[] pels, int width,
int height) {
    int i;
    ColorModel cm = getRgbColorModel();
    for (int x = 0; x < width; x++)
        for (int y = 0; y < height; y++) {
            i = x + y * width;

            b[x][y] = (short) cm.getBlue(pels[i]);
            g[x][y] = (short) cm.getGreen(pels[i]);
            r[x][y] = (short) cm.getRed(pels[i]);

        }
    }
}

```

//method clips the image to 255 if the rgb value exceeds 255 and //makes it to 0 if it is negative

```

public void clip(ImageBean ib) {
    int w = ib.getWidth();
    int h = ib.getHeight();
    for (int x = 0; x < w; x++)
        for (int y = 0; y < h; y++) {
            if (r[x][y] > 255) r[x][y] = 255;
            if (g[x][y] > 255) g[x][y] = 255;
            if (b[x][y] > 255) b[x][y] = 255;
            if (r[x][y] < 0) r[x][y] = 0;
            if (g[x][y] < 0) g[x][y] = 0;
            if (b[x][y] < 0) b[x][y] = 0;
        }
    }
}

```

//method reconstructs the skin detected image from the binary //image

```

public void SkinReCon() {

    for (int x = 0; x < getWidth(); x++)

```

```

        for (int y = 0; y < getHeight(); y++) {
            if (r[x][y] == 255) {
                if (g[x][y] == 255) {
                    if (b[x][y] == 255) {
                        r[x][y] = r1[x][y];
                        g[x][y] = g1[x][y];
                        b[x][y] = b1[x][y];
                    }
                }
            }
        }
    }
}

```

//convert the image to gray scale by taking the average of the //red, green and blue colors.

```

public void gray() {
    for (int x = 0; x < getWidth(); x++)
        for (int y = 0; y < getHeight(); y++) {
            r[x][y] = (short)
                ((r[x][y] + g[x][y] + b[x][y]) / 3);
            g[x][y] = r[x][y];
            b[x][y] = r[x][y];
        }
}

```

//method used to negate an image

```

public void negate() {
    for (int x = 0; x < getWidth(); x++)
        for (int y = 0; y < getHeight(); y++) {
            r[x][y] = (short) (255 - r[x][y]);
            g[x][y] = (short) (255 - g[x][y]);
            b[x][y] = (short) (255 - b[x][y]);
        }
    clip(this);
}
}

```

//Method used to display the final facerecognized image on the //frame

```

public void imageAddFinalDisplay() {
    int m = getWidthOne()-1;

    for (int x = getWidth()-1; x > getWidth() -
getWidthOne(); x--) {

        for (int y = 0 ; y < getHeightOne() ;
y++){

            r[x][y] = r1[m][y] ;
            g[x][y] = g1[m][y] ;
            b[x][y] = b1[m][y] ;

        }

        m--;
    }
}

//Method tries to invoke the camera in the Zaurus
private void camera() {
    try {

        Runtime.getRuntime ().
            exec
            ("/home/QtPalmtop/bin/./sq_camera");
    } catch (IOException ioe) {
        ioe.printStackTrace ();
    }
}

//Getters and Setters declared for r,g and b
    public short[][] getR() {
        return r;
    }

    public void setR(short[][] r) {
        this.r = r;
    }

    public short[][] getG() {
        return g;
    }

    public void setG(short[][] g) {
        this.g = g;
    }

```

```
    }  
  
    public short[][] getB() {  
        return b;  
    }  
  
    public void setB(short[][] b) {  
        this.b = b;  
    }  
}
```