

Embedded Face Identification System Design

Yuan-Wei Tseng* and Chong-Chi Wang*

ABSTRACT

With the technical advance, embedded systems have been widely applied to many applications of humans' daily lives. Security system based on face identification is one of the popular applications of the embedded systems because faces consist of many biological features so it is much safer than the security systems that only take passcodes.

In this paper, a simple and low cost embedded system dedicated for real-time human face identifications is constructed. The comprehensive procedures in building up an embedded system such as setup environment for cross-compilation, migration of Bootloader, migration of *Linux* kernel, fabrication and migration of root document system and setup of peripheral driving devices have been presented. The face identification program is developed based on *Haar-like* feature training and *AdaBoost* classifier. When the program is migrated to and run on the developed embedded system, it can identify the registered face in average credibility 0.85. Therefore, the real-time performance of the embedded system is assured.

Keywords: *real-time embedded system, computer vision, human face identifications, Haar-like features, Adaboost classifier.*

INTRODUCTION

With the progress of technology, the performance of single chip system becomes more and more powerful while the cost becomes lower and lower. Many applications have been developed. To design an embedded system, the selections of microprocessor and operating system are of vital importance.

An operation system coordinates, commands and control both hardware and software resources for the whole system and peripheral devices to achieve real-time performance.

Considering the cost down and compatibility, usually, de facto microprocessor and operating system are designer choices in embedded system designs. For example, *ARM* series processors [1] and *Linux* operating system [2] are very popular in embedded systems. In this paper, an embedded system for particular persons' face identification is developed. *ARM II* processor and *Linux* operating system are selected based on trade-off between performance and cost. This system can be further integrated into smart home applications [3, 11] and security applications. In the past, many similar *FPGA* based embedded face recognition systems have been developed and a good survey of such systems is given in [4, 11]. However, some *FPGA* based systems are suffered from the low processor speeds. That's why *ARM* based system is selected in this research.

Face is an important biological feature that consists of lot of information. Therefore, particular persons' face identification has great deal of security related applications. Furthermore, this application is ideal to test the performance of the designed embedded system.

Generally speaking, the tasks of particular persons' face identifications include collecting face images, positioning faces, preprocess of face images for identifications, searching for identity against database and

* Department of Electrical Engineering, I-Shou University No. 1, Sec. 1, Syuecheng Rd., Dashu District, Kaohsiung City 84001, Taiwan, R.O.C., E-mail: yuanwei@isu.edu.tw

finally confirming identity. To achieve face identifications, *OpenCV* (Open Source Computer Vision) [5], developed by Intel, an open source image process library is used. The *GUI* (graphic user interface) of the embedded system is developed using powerful cross-platforms native *C++* libraries *Qt* [6]. Therefore the designed embedded system integrates *ARM II* processor, *Linux* operating system, *Qt GUI* and *OpenCV* to meet performance requirement at very low cost.

APPLICATION SCENARIOS

In this paper, the architecture of designed embedded face identification system is shown in Fig. 1.

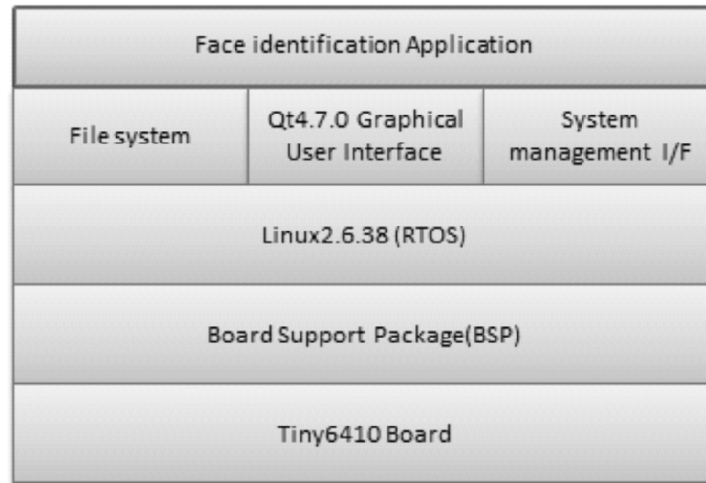


Figure 1: Architecture of the System

As shown in Fig.2, the *Tiny 6410* board [13], equipped with *ARM II* processor and multi-media interfaces was selected as hardware platform to construct the embedded system in this research.

In this design, *Linux2.6.38* was chosen as the operating system of the designed embedded face identification system. Image acquisition application program interface, *Video4Linux (sV4L)* which can setup camera and acquire images is integrated in the *Linux* system. Working with cross platform *UI* and software application development framework *Qt 4.7.0*, graphical user interface of face identification application program has been developed.

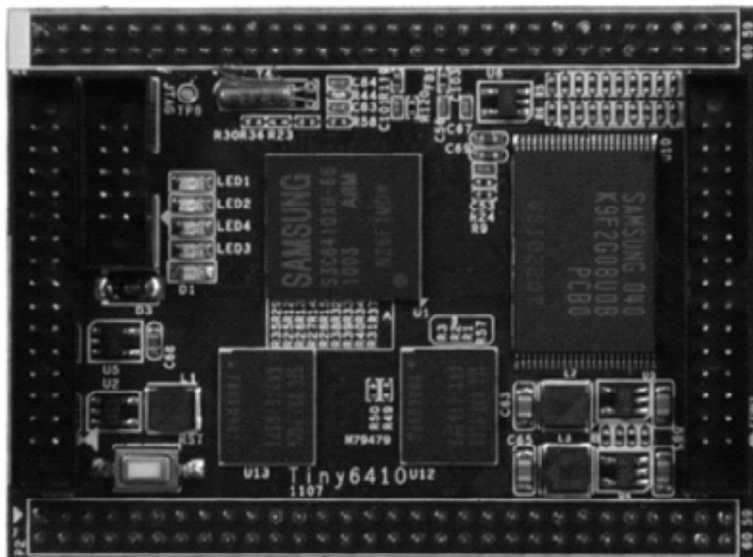


Figure 2: Target board Tiny 6410

The design stages of this embedded face identification system are summarized as follows.

At the first stage, the architecture of hardware platform should be studied. At the second stage, the development environment is set up. The so called *minicom* which is a *Linux* communication program is installed and configured at the third stage. Cross compiler tools are used to generate executables for embedded system or multiple platforms. It is used to compile for a platform upon which it is either not feasible or lack of resources to do the compiling and debugging, just like this embedded face identification system in this research. Therefore, *Linux* cross compiler development environment has to be configured in a personal computer to compile and to debug application programs with link of device drivers to obtain the *.exe* executable file at the fourth stage so that application programs can be quickly developed with abundant PC resource. In this research, *Fedora Linux* is used. Once the application programs are successfully verified in a PC, the initial program of the embedded system, boot loader is compiled and implemented in the fifth stage. In this research, open-source *U-boot* (Universal Boot Loader) is used. Boot loader is in charge of the initialization of system hardware, memory mapped and I/O addressing, setting parameters of *Linux* kernel and loading *Linux* operating system. In the sixth stage, *Linux* kernel is configured and compiled. Before kernel is being compiled, one can select the internet protocol. The kernel to be migrated to the hardware platform consists of modules of process scheduler, memory management, virtual file system, networking interface, process communication. After kernel is compiled, tools, application programs, *init* programs, device drivers such as Ethernet and USB Video Class and necessary library are saved in the root file system in the sixth stage. Then bootloader, kernel image and root file system are written into flash memory so that the *Linux* kernel is migrated to the experimental board in the final stage.

The details of above procedure are available in [7] and the design flow is shown in Fig. 3.

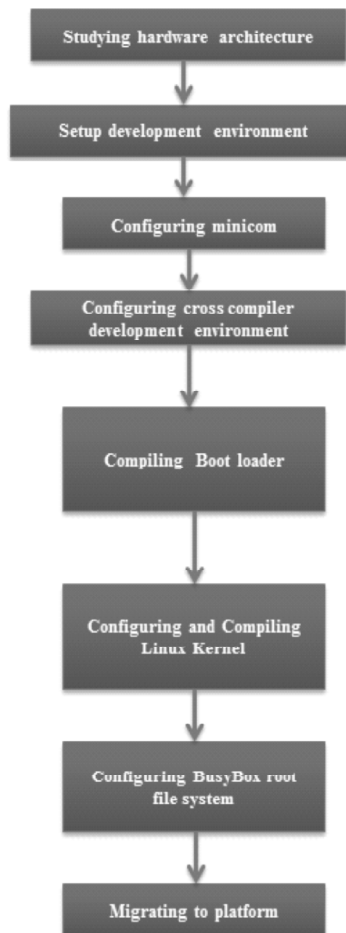


Figure 3: Linux Kernel Configuration and Compiling Flow

FACE IDENTIFICATION

In this paper, *haar-like* features based Face identification utilities from *OpenCV* are applied so that the embedded system can quickly identify its object, a particular person's face. In 2001, Paul Viola and Michael Jones proposed integral image [8]. Based on integral images, a fast face detection system that utilized *AdaBoost* and *Cascade* algorithms [8] was proven to be very effective. Rainer Lienhart *et al.* [9] further proposed an extended set of *haar-like* features of four edge features, eight line features, and two center-surround features.

Viola and Jones use *AdaBoosting* as their basic classifier, which combines a collection of cascade weak classification functions to form a stronger classifier that often outperforms most 'monolithic' strong classifiers such as support vector machines [10] and Neural Networks. A weak classifier is only required to be better than chance, and thus can be very simple and computationally inexpensive.

The *AdaBoost* algorithm and mathematic background are briefly summarized as follows.

First of all, the following training set with n training samples is prepared as input for running *AdaBoost* process.

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad (1)$$

where x_i are training samples and corresponding $y_i \in \{0, 1\}$ are used to indicate x_i as positive samples when $y_i = 1$ while x_i as negative samples when $y_i = 0$. The number of positive samples is m and the number of negative samples is l where $n = l + m$.

Secondly, initializing the individual sample weighting by

$$w_{m,i} = \frac{1}{2m} : \text{positive samples' weightings} \quad (2)$$

$$w_{l,i} = \frac{1}{2l} : \text{negative samples' weightings} \quad (3)$$

At the third stage, the sample weightings are further adjusted. The idea of the adjustment is as follows. Every weak classifier has a feature. Suppose that T features are used in building a strong classifier, the sample weightings are normalized as

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}, t = 1, 2, \dots, T. \quad (4)$$

For every feature, a weak classifier h_j is trained. The weighted error rate of every feature is given by

$$\varepsilon_i = \sum_{i=1}^n w_t(x_i) |h_j(x_i) \neq y_i| \quad (5)$$

When $\varepsilon_i = 0$, the feature is correctly classified while when $\varepsilon_i = 1$, the feature is unclassified.

The weak classifier which has the smallest error rate ε_t is picked up as the best weak classifier h_t . Based on the best weak classifier, the sample weightings are adjusted by

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\varepsilon_i} \quad (6)$$

where

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t} \quad (7)$$

Finally, connecting all weak classifiers in series, the following strong classifier is built.

$$h(x) = \begin{cases} 1 & \sum_{i=1}^T \alpha_i h_i(x) \geq \frac{1}{2} \sum_{i=1}^T \alpha_i, a_i = \log \frac{1}{\beta_i} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$



Figure 4: Image Training Flow

The image blocks are fed through a series of cascade weak classifiers. If it is not matched with any weak classifier, it is immediately categorized as a non-feature block. Only the image blocks that pass all weak classifiers are the blocks with features. The training flow is given in Fig. 4.

OpenCV provides *createsmaple* and *haartraining* for training classifier. The program *createsmaple.exe* is first applied to form training vector from different folders that contain positive samples of a particular person’s face and negative samples of non-objects. The created training vector in *.dat* file is then imported into program *haartraining.exe* for training a classifier cascade on a training set. After running *haartraining.exe*, a classifier description in an xml file that can be deployed or tested is created. In this research, the xml file is recognizable face database.

The *OpenCV* functions [5] that handle face identification in this research are as follows.

1. `fileStorage = cvOpenFileStorage ("facedata.xml", 0 , CV_STORAGE_READ):`
Read in the xml file of recognizable face database [3]

2. `findNearestNeighbor(float *projectedTestFace):`

Extract the face features of the input image and compare them with features of recognizable faces in database to determine if there is a match based on average creditability $fConfidence$, where

$$fConfidence = 1.0f - \sqrt{(\text{leastDistSq} / (\text{float} (nTrainFaces * nEigens))) / 255.0f}.$$

When $fConfidence$ is greater than threshold, face is successfully identified and the corresponding ID will be displayed on screen.

3. `cvEigenDecomposite (IplImage* obj, int nEigObj, void* eigInput, int ioFlags, void* userData, IplImage* avg, float* coeffs):`

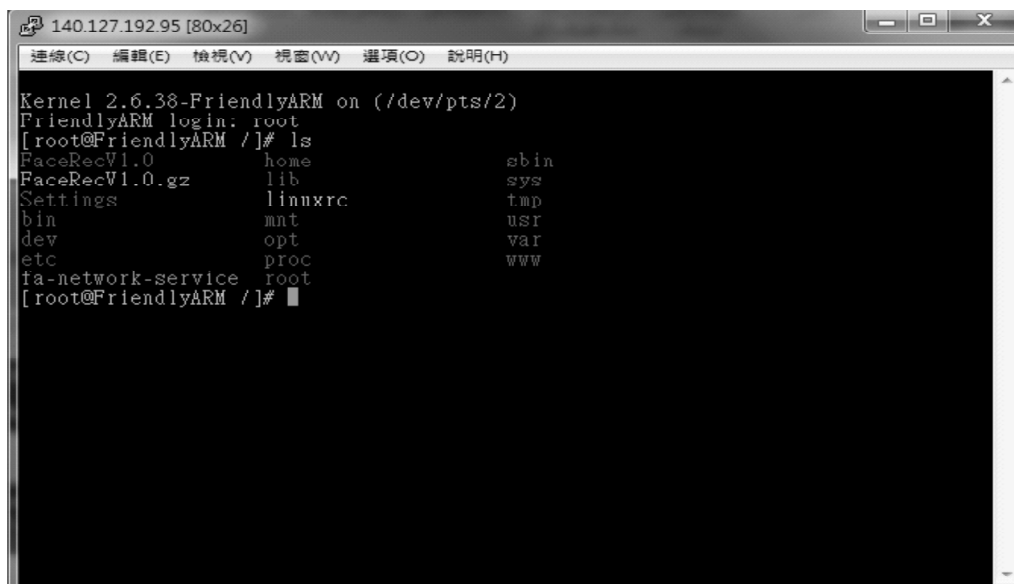
The function `cvEigenDecomposite` can generate image projections in vector form for both candidate objects and training samples. `cvEigenDecomposite` calculates all decomposition coefficients in a row vector for the input image object in matrix form using the previously calculated Eigen objects basis and the averaged object. Since the projected image is not a matrix but a vector, both classification and identification of image become easier. To generate projections of training samples, the following switches are used.

`cvEigenDecomposite(faceImagArr[i], nEigens, eigenVectArr, 0, 0, pAvgTrainImg, projectedTrainFaceMat -> data.fl + i*offset)`, where i is from 0 to $nTrainFaces$.

4. `cvCalcEigenObject():` This function is used in training phase. Based on the training samples, it calculates Eigenface transfer matrix.

EMBEDDED SYSTEM OPERATIONS AND EXPERIMENT RESULTS

There are two main tasks in this research. The first one is to design and configure the platform of embedded system. The second one is to develop a particular persons' face identification program that runs on the embedded system. Through USB network camera, consecutive images in YUV format are collected, transferred to *Qimage* format and displayed on LCD. The images are further transferred to *IpImage* format. Consequently, *Haar Cascade Face Detector* or so-called *Viola-Jones* method in *OpenCV* is utilized to carry out particular persons' face identifications.



```

140.127.192.95 [80x26]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
Kernel 2.6.38-FriendlyARM on (/dev/pts/2)
FriendlyARM login: root
[root@FriendlyARM /]# ls
FaceRecV1.0      home      sbin
FaceRecV1.0.gz  lib       sys
Settings        linuxrc   tmp
bin             mnt       usr
dev            opt       var
etc            proc      www
fa-network-service  root
[root@FriendlyARM /]#

```

Figure 5: Linux Terminal Command Window



Figure 6: Qtopia Graphic Interface

Both embedded *Linux* system and *Qt* interface called *Qtopia* were first compiled on a PC and then ported to the embedded platform. Linux terminal command window is shown in Fig. 5 while *Qtopia* graphic interface is shown in Fig. 6.

For more flexible applications, the designed embedded system has internet access ability to use IP camera. With internet, it also can be controlled by a remote PC host if necessary. Internet can be set up in *Qtopia* graphic interfaces as shown in Fig. 7.

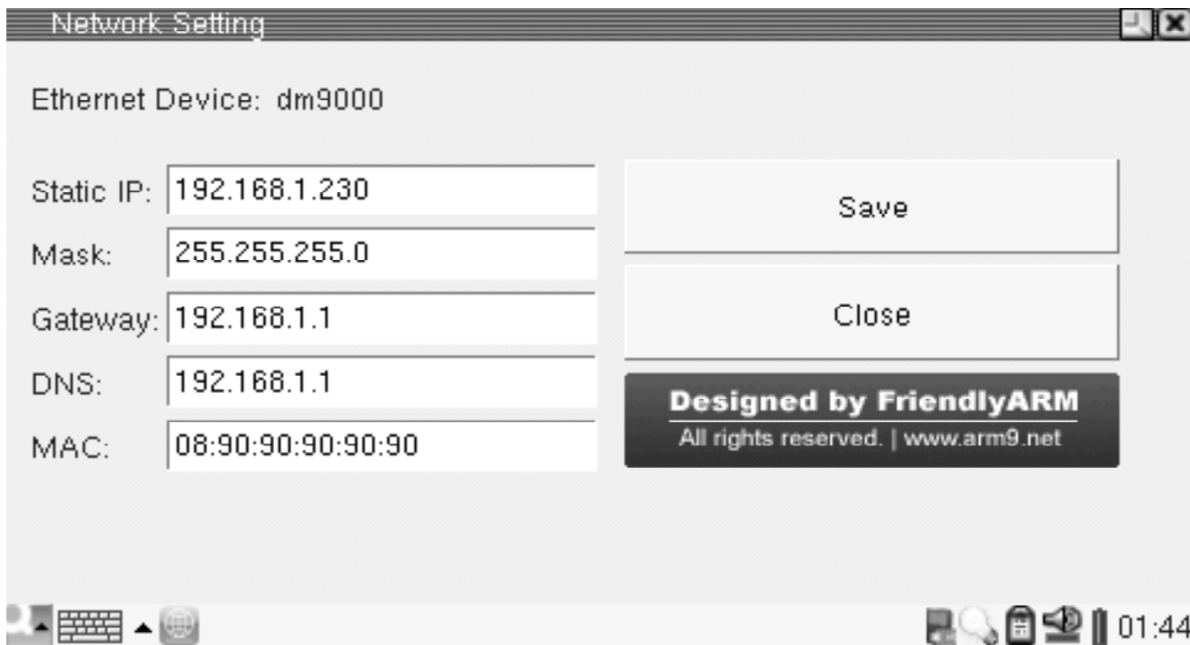


Figure 7: Setting Up Internet Access in Qtopia Interface

Before executing face identification application, we need to shut down *Qtopia* by clicking "Terminate Server" in *Qtopia* interface as shown in Fig. 8.

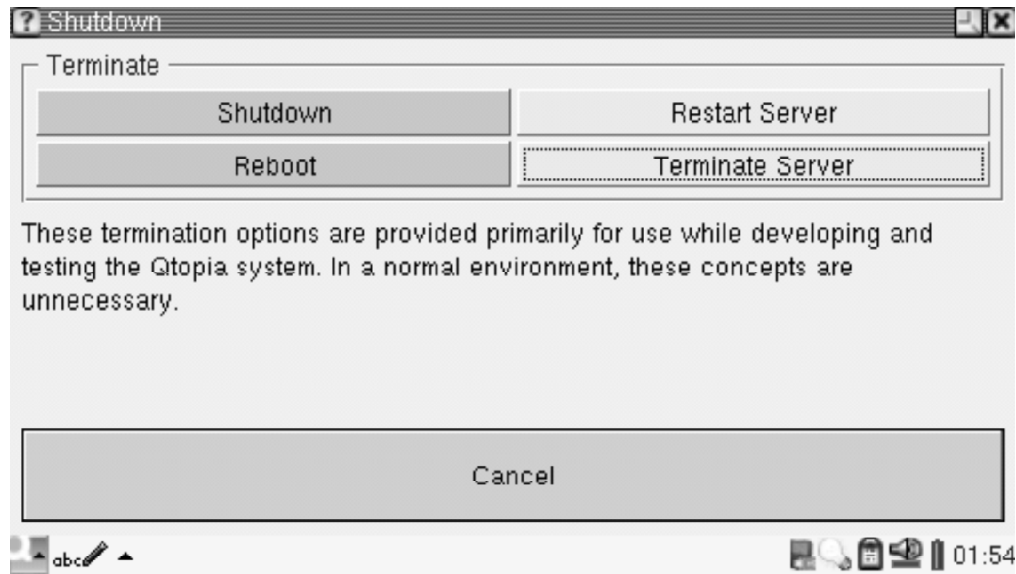


Figure 8: *Qtopia* Interface to Terminate Server

After quitting *Qtopia* interface, PC can remotely control the embedded platform via internet. The protocol between host PC and embedded platform is TFTP (Trivial File Transfer Protocol), which is simpler than FTP but perform good enough.

OpenCV provides some useful classifiers for detecting facades of human faces such as *haarcascade_frontalface_alt.xml* and *haarcascade_frontalface_alt2.xml* which can be directly applied in application program to improve the detecting speed.

When executing the face identification program in the embedded system, the face area in an image can be enclosed by a green rectangle and displayed on LCD. The face images within the green rectangle are processed with histogram equalization to get better training and identification results. The processed face images are checked against the database of recognizable faces with PCA (Principal Components Analysis) method [12]. Once the creditability is higher than threshold, a matched occurs. Consequently, the corresponding ID of the recognizable person is displayed under the green rectangle. On the other hand, for unrecognizable faces, no ID will be displayed. The face identification flow is given in Fig. 9.

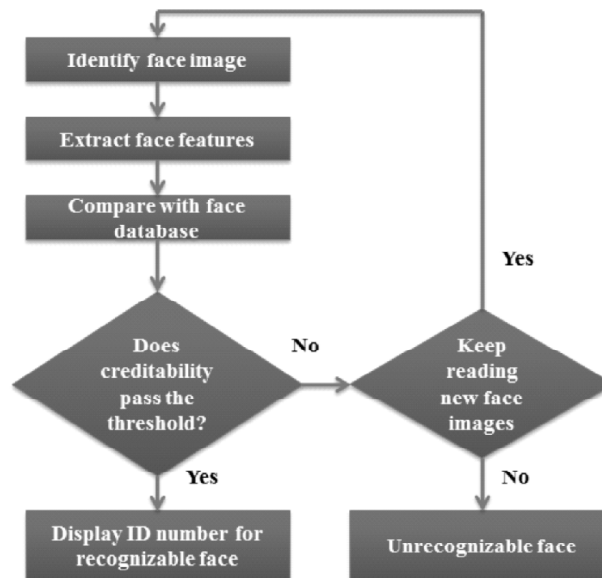


Figure 9: The Face Identification Flow

When the compiled programs are migrated to the embedded platform through TFTP, the executing interface is shown in Fig. 10.

In Fig. 10, a recognizable face example is given. The face is successfully identified and the corresponding ID is displayed as “Jeff”. For comparison, an unrecognizable face example is given in Fig. 11. The face cannot be identified and the corresponding ID cannot be displayed because his face is not in the database of recognizable faces. In general cases, the results come out in 1-2 seconds.

Using more positive and negative image samples in training resulted in improvement of the correctness of face identifications in experiment. On the other hand, the improvement was also obtained by averaging the creditability of several images. A match occurs only when the average of the creditability exceeds threshold.

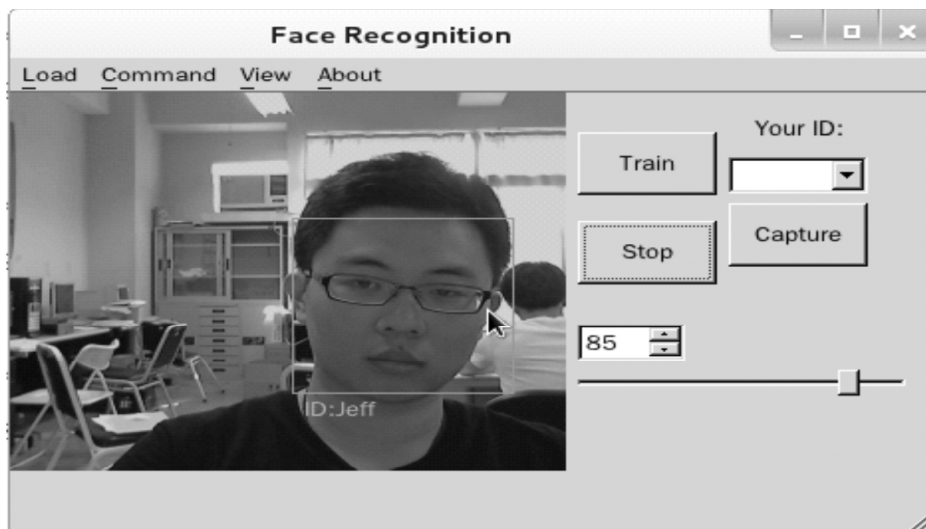


Figure 10: Recognizable Face and Displayed ID

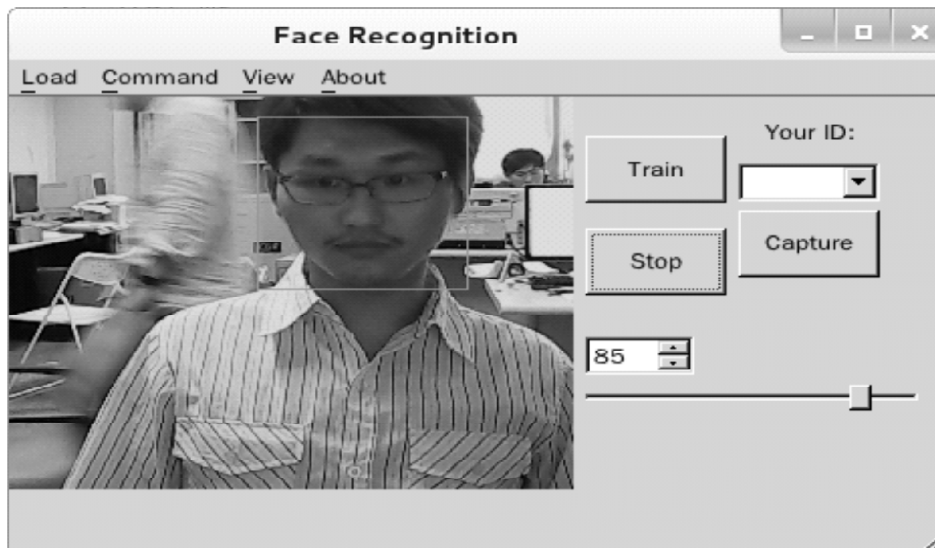


Figure 11: Unrecognizable Face

CONCLUSIONS

ARM based embedded systems have more and more applications recently, especially in consumer electronics including PDA and mobile phones. In this paper, the procedure of design and configuration of ARM 11

based embedded system such as system architecture, establishing cross compiler, setting *Linux* kernel, configuring Boot loader, configuring root file system and device drivers are introduced.

The performance of the designed embedded system with graphic user interface is verified by the application program of particular persons' face identifications and real-time results are successfully obtained. The *ARM* microprocessor used in this experiment is S3C6410 which runs at 533-677MHz. Speed wise, it is fast enough. The total cost of the hardware and camera is less than \$120. Therefore, a very cost effective embedded face identification system is constructed. More applications and features can be added to this system and those will be our future works.

REFERENCES

- [1] <http://arm.com/products/processors/index.php>
- [2] http://www.operating-system.org/betriebssystem/_english/bs-linux.htm
- [3] Fei Zuo and de With, P. H. N. "Real-time Embedded Face Recognition for Smart Home," *IEEE Transactions on Consumer Electronics*, Vol. 51, No.1, pp. 183-190, 2005.
- [4] Al-Shebani, Qasim; Premaratne, Prashan; Vial, Peter "Embedded Door Access Control Systems based on Face Recognition: A Survey," pp. 1-7, The 7th International Conference on Signal Processing and Communication Systems (ICSPCS), 16-18 Dec. 2013.
- [5] *OpenCV*, <http://opencv.org/>
- [6] *Qt*, <http://qt.digia.com/product/>
- [7] Chong-Chi Wang, *Embedded System Design with Application on Face Identification*, Master thesis, Dept. of Electrical Engineering, I-Shou University, Kaohsiung, Taiwan, June, 2013.
- [8] Paul Viola, and J. Michael Jones. "Rapid Object Detection Using a Boosted Cascade of Simple Features," *IEEE CVPR*, Vol. 1, No. 2, pp 511-518, Dec. 2001.
- [9] Rainer Lienhart, Jochen Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," MRL Technical Report, May 2002, revised December 2002.
- [10] C. C. Chang and C. J. Lin. *LIBSVM: A Library for Support Vector Machines*, *ACM Transactions on Intelligent Systems and Technology*, 2:27:1-27:27, 2011.
- [11] Pan Chao, "Design and Implementation of Face Recognition System based on Linux", Master Thesis, Indian University, Xi'an, Shaanxi, China, 2012.
- [12] Principal Component Analysis: http://en.wikipedia.org/wiki/Principal_component_analysis
- [13] FriendlyARM, <http://www.friendlyarm.net/products/tiny6410>