Face Identification System for Personalized Services

Ansu Mathew
M.Tech – Embedded Systems
NIELIT
Calicut, India

K. M. Martin
Scientist/Engineer - E
NIELIT
Calicut, India

Shiyas Shahabudeen
Embedded Software Specialist
Gadgeon Smart Systems Pvt. Ltd.
Cochin, India

Abstract— **The Face Identification System for Personalized Services is a reliable, low cost solution for real-time face recognition and verification. Proposed application is a greeting system, where the user will get timely personalized greeting messages once the system verifies that he is an authorized user. The system performs accurate and fast real-time face tracking and identification and the people will get timely greeting messages or personal messages by name. The system can be installed at the entry gate of an organization and the face images can be captured from a distance without touching the person being identified, and the identification does not require interacting with the person. The system monitors the people coming in the vicinity of camera, detect their faces and compare with the images stored in the database. If a matching entry is found, then the person will be greeted by name via speaker and strangers will be directed to the appropriate destination. Also an entry-log will be created with time-stamp. A web-based user-interface is provided for administrative tasks like enrolment and user management. It is implemented in Wandboard Quad which is a Freescale i.MX6 Cortex-A9 processor. Face recognition is performed using OpenCV (Open Source Computer Vision) [1] which is a library of programming functions mainly aimed at real time computer vision, developed by Intel.**

Keywords—OpenCV; Face; Detection; Recogntion; Wandboard;

I. Introduction

The human face plays an important role in our social interaction, conveying people's identity. Using the human face as a key to identification, the face recognition technology has received significant attention in the past several years due to its potential for a wide variety of applications. The Face Recognition system is a computer application for automatically identifying and verifying a person from video frames or digital images.

The problems of face recognition attracts researchers working in biometrics, pattern recognition field and computer vision. Several face recognition algorithms are also used in many different applications like biometrics, video compressions, indexing's etc. They can also be used to classify multimedia content, to allow fast and efficient searching for material that is of interest to the user. An efficient face recognition system can be of great help in forensic sciences, identification for law enforcement, surveillance, authentication for banking and security system, and giving preferential access to authorized users i.e. access control for secured areas etc. The problem of face recognition has gained even more importance after the recent increase in the terrorism related incidents. Use of face recognition for authentication also reduces the need of remembering passwords and can provide a much greater security if face recognition is used in combination with other security measures for access control. The cost of the license for an efficient commercial Face recognition system ranges from 30,000$ to 150,000$ which shows the significant value of the problem.

II. Project Description

Face Identification system for personalized service is a real-time implementation of face recognition technology in an embedded platform. The focused application is an automated greeting system, which delivers timely personalized greeting messages to the people approaching the system. The main advantage is that the system does not require any user interaction, i.e. the user need not touch a surface or stand in front of the system for quite long time. In most of the cases user may be unaware that he is being monitored by the system.

The system captures real-time video and do frame-by-frame analysis. It searches for valid faces in each frame and if it founds one then the system will identify that person from a set of faces stored in the database. Once the face is verified, that person will get a timely greeting voice message by name via the speaker connected to the system. Otherwise, he will be treated as a stranger and corresponding message will be delivered. The system also had a learning phase, where it collects multiple face images of the user for face training through a web interface.

A. OpenCV

OpenCV is an open source computer vision library available from http://opencv.org. The library is written in C and C++ and runs under Linux, Windows, Mac OS X, iOS, and Android. Interfaces are available for Python, Java, Ruby, Matlab, and other languages [2]. OpenCV was designed for computational efficiency with a strong focus on real-time applications: optimizations were made at all levels, from algorithms to multicore and CPU instructions.

B. Wandboard Quad

The Wandboard is an ultra-low power complete computer with high performance multimedia capabilities based around the Freescale i.MX6 Cortex-A9 processor [3]. Wandboard Quad is the newest upgraded version in the Wandboard family of development platforms and improves upon the previous Wandboard Dual, offering the latest Freescale i.MX6 ARM®

Cortex®-A9 Quad Core Processor at 1GHz performance. The Freescale i.MX 6Quad also integrates the Vivante GC2000 GPU, adding higher 3D graphics performance. It has Wi-fi (802.11n), Bluetooth, Gigabit LAN, SATA connector, 2GB DDR3, HDMI, Camera Interface and lot of other features.

III. Approaches to Face Detection

Face detection is a computer technology that identifies human faces in digital images. It detects human faces which might then be used for recognizing a particular face. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit.

Numerous methods have been proposed to detect faces in images. These methods can be organized in two categories: feature-based approaches and appearance-based approaches.

Feature-based approaches are usually based on the detection of local features of the face, such as the nose, the mouth or the eyes, and the structural relationship between these facial features. Whereas Appearance-based approaches consider face detection as a two-class pattern recognition problem. They rely on statistical learning methods to build a face/non-face classifier from training samples. Although both classes of methods do not deal with the same problems and environments, appearance-based approaches have recently received considerable attention and have proven to be more successful and robust than feature-based approaches.

In 2001, Viola and Jones introduced the first real-time frontal face detection system [4]. Instead of using pixel information, they proposed to use a new image representation and a set of simple features that can be computed at any position and scale in constant time. Boosting learning is both used for feature selection and classifier design.

A. *Local Binary Pattern (LBP)*

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number [5]. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications. It can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. Perhaps the most important property of the LBP operator in real-world applications is its robustness to monotonic gray-scale changes caused, for example, by illumination variations. Another important property is its computational simplicity, which makes it possible to analyze images in challenging real-time settings.

IV. APPROACHES TO FACE RECOGNITION

According to the new technical era, some advancement has taken place and some techniques of facial recognition have achieved popularity. Some of the popular face recognition algorithms are EigenFace, Fisherface, Local Binary Pattern Histogram (LBPH) etc [6].

Eigenface is a practical approach for face recognition. PCA reduces the dimension size of an image greatly in a short period of time. The accuracy of Eigenface is also satisfactory (over 90 %) with frontal faces. However, as there has a high correlation between the training data and the recognition data. Preprocessing of image is required in order to achieve satisfactory result. The drawback is that it is very sensitive for lightening conditions and aging effects.

Fisherface is similar to Eigenface but with improvement in better classification of different class images. With Fisher Linear Discriminating approach, we could classify the training set to deal with different people and different facial expression. We could have better accuracy in facial expression than Eigen face approach. Besides, Fisherface removes the first three principal components which is responsible for light intensity changes, it is more invariant to light intensity. Fisherface is more complex than Eigenface in finding the projection of face space.

We are using Fisherface algorithm for face recognition because it gives satisfactory results in real-world conditions compared to others.

V. SYSTEM ARCHITECTURE

The system has a modular architecture. Fig 1 shows the block representation of the system.

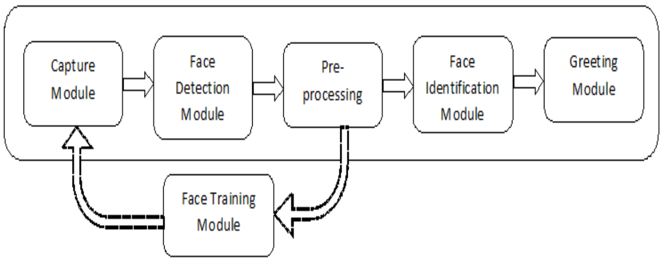


Fig 1. Block diagram

Capture module deals with the configuration of video interface and performs the real-time video capture. Face Detection module analyses each captured frame and extracts valid faces from each frame. Face Identification deals with face recognition and verification of the detected face. Once a face is verified, speaker module checks the last access time of that user and deliver a voice greetings if he is coming for the first time on that day. The strangers are also greeted accordingly.

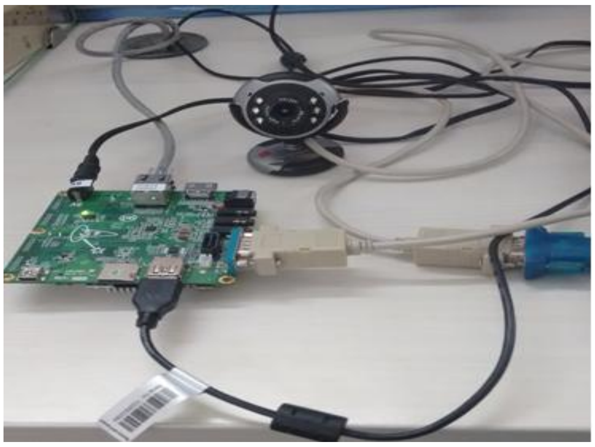


Fig 2: Experimental set-up

VI. IMPLEMENTATION

The implementation was done on Wandboard Quad. This system is developed using the C++ interface of OpenCV [1]. The system has two mode of operation: Face Identification mode, which is the default one and the Face Training mode, which is running on a web server. The implementation starts with the creation of a custom Linux distribution for Wandboard Quad using Yocto Project [7] [8]. Generated image will be copied to a bootable SD card and the Wandboard is made to boot from SD card.

A. Face Identification System Implementation

The face identification process starts with a real-time frame capture. The USB webcam connected with Wandboard does the video capture. OpenCV has a VideoCapture class that provides C++ API for capturing video from cameras or for reading video files. VideoCapure object grabs, decodes and returns the next video frame. The grabbed frame is stored in a Mat object, which is the basic image container of OpenCV. This image matrix is given to the Face Detection module, for extracting the faces in the frame.

1. Face Detection Module

OpenCV comes with pre-trained LBP detector for frontal faces that can be used directly in our applications just by loading cascade classifier XML file.

After loading the classifier (just once during initialization), it can be used to detect faces in each new camera frame. But first we should do some initial processing of the camera image just for face detection, by performing the following steps:

a). Grayscale color conversion: Face detection only works on grayscale images. So the color camera frame should be converted to gray scale.

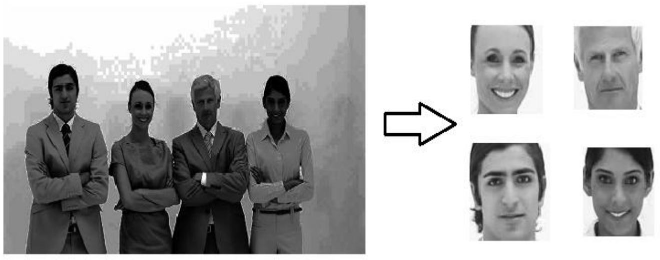


Fig 3. Face Detector output

b). Shrinking the camera image: The speed of face detection depends on the size of the input image (it is very slow for large images but fast for small images), and yet detection is still fairly reliable even at low resolutions. So the camera image is shrunk to a reasonable size by keeping the aspect ratio (the ratio of width to height) of the output same as the input.

c). Histogram equalization: Face detection is not as reliable in low-light conditions. So histogram equalization is performed to improve the contrast and brightness.

After the initial frame processing, the face can be detected by using the CascadeClassifier::detectMultiScale( ) function. The parameters for this function is chosen to customize the face detector for our application such as the minimum feature size, Search scale factor, search for multiple faces etc.

The output of the detectMultiScale() function will be a std::vector of the cv::Rect type object. For example, if it detects two faces then it will store an array of two rectangles in the output. Fig 3 illustrates the Face Detector output to a sample frame after initial frame processing

Since the face detector is given a shrunken image, it should be resized in order to get the face regions for the original image. Also make sure that faces on the border of the image stay completely within the image. Now the detected face is extracted from the original image using the face rectangles and then taken for face pre-processing.

2. Face Pre-processing

Face recognition is extremely vulnerable to changes in lighting conditions, face orientation, face expression and so on, so it is very important to reduce these differences as much as possible. Otherwise the face recognition algorithm will often think there is more similarity between faces of two different people in the same conditions than between two faces of the same person.

Apart from the basic frame processing, we require many sophisticated techniques, including facial feature detection (for example, detecting eyes, nose, mouth and eyebrows) for a reliable face recognizer in real-world conditions. We are using only eye detection and ignore other facial features such as the mouth and nose, which are less useful.

a). Eye Detection

Eye detection can be very useful for face preprocessing, because for frontal faces we can always assume a person's eyes should be horizontal and on opposite locations of the face and should have a fairly standard position and size within a face, despite changes in facial expressions, lighting conditions, camera properties, distance to camera, and so on. It is also useful to discard false positives when the face detector says it has detected a face and it is actually something else. It is rare that the face detector and two eye detectors will all be fooled at the same time, so if you only process images with a detected face and two detected eyes then it will not have many false positives (but will also give fewer faces for processing, as the eye detector will not work as often as the face detector).

We are using some pre-trained eye detector that comes with OpenCV v2.4 that can detect open or closed eyes.

b). Face Processing

With the face and both eyes detected, the following pre-processing steps can be applied:

1. Geometrical transformation and cropping: This process would include scaling, rotating, and translating the images so that the eyes are aligned, followed by the removal of the forehead, chin, ears, and background from the face image.

2. Separate histogram equalization for left and right sides: This process standardizes the brightness and contrast on both the left- and right-hand sides of the face independently.

3. Smoothing: To reduce the effect of pixel noise, a bilateral filter is used to smoothen the face image while keeping edges sharp.

4. Elliptical mask: The elliptical mask removes some remaining hair and background from the face image. To create the mask, we will draw a black-filled ellipse onto a white image. Fig 4 shows the creation of elliptical mask.

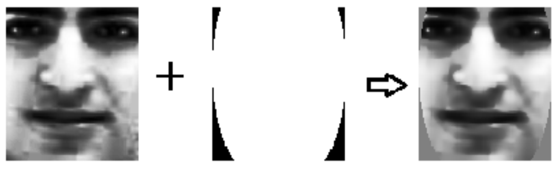


Fig 4. Elliptical mask creation

Fig 5 shows the face preprocessing steps 1 to 4 applied to a detected face. The final image shows a sample result from all the face preprocessing stages. Notice it is much more consistent for face recognition in different brightness, face rotations, angle from camera, backgrounds, positions of lights, and so on. This preprocessed face will be used as input to the face recognition stages, both when collecting faces for training, and when trying to recognize input faces.

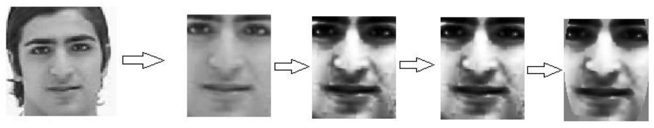


Fig 5: Face pre-processing steps

3. Face Recognition Module

The face recognition algorithms are available through the FaceRecognizer class in OpenCV's contrib module. To use the face recognition algorithm, we must create a FaceRecognizer object or Fisherface algorithm. This will give access to that algorithm. Also, we should select a threshold for prediction. This parameter is closely linked with the recognition accuracy. The threshold value is automatically adjusted based on the number of users in face database. The training data is stored in a XML file and for recognition process we should load the training model from this file.

Face Recognizer module predicts the identity of the face in the pre-processed face image using OpenCV API. This identity value will be the label number that we originally used when collecting faces for training. To get an accurate result, we are not relying on single prediction, rather do prediction on multiple images and finally more accurate identity will be given. This is done in face verification.

a). Face Verification

Face verification gives the correct ID of the person standing in front of the system. It takes 5 seconds time to find the identity of a person once he comes in the vicinity of the camera. During this time, system continuously grabs camera frame, search for valid faces, and predict the identity. After 5 seconds, system analyses all the predicted identities during this interval and the ID with maximum prediction will be identified as the recognized person. It can be a person who is already known to the system or can be an unknown.

The system also predicts multiple identities, if more than one person approaches the system. The verified user ID is then passed to the speaker module, to activate the greeting delivery.

4. Greeting Module

This module is responsible for delivering greeting message to the identified users. First the system find the current date and time using *ntpdate* [9]. If the Identified person is an unknown, then the system will immediately deliver a timely greeting message to the stranger like “Good Morning Stranger”. We are using a text-to-speech converter utility called *espeak* [10] for converting the text message to a voice output.

User will get greetings only once in a day. So if the identified person is a known user who is already trained by the face algorithm, system fetches his last access (date and time of last greeting) from the database. If the last access was not on the same day, then he will get timely greetings by name. For example: “Good Evening XXX”. Along with that, the last access field of the user should be updated with current date and time.

B. Face Training System Implementation

During training phase, the face recognition algorithm will learn how to distinguish between faces of different people. The collected faces for training are referred to as training set. After the face recognition algorithm has finished training, we could then save the generated knowledge to a file or memory and later use it to recognize which person is seen in front of the camera.

It is important that we provide a good training set that covers the types of variations we expect to occur in our testing set. For example, if we only test with faces that are looking perfectly straight ahead, then we only need to provide training images with faces that are looking perfectly straight ahead. But in real-time scenario the person might be looking to the left or up, then we should make sure the training set will also include faces of that person doing this, otherwise the face recognition algorithm will have trouble recognizing them, as their face will appear quite different. This also applies to other factors such as facial expression (for example, if the person is always smiling in the training set but not smiling in the testing set) or lighting direction (for example, a strong light is to the left-hand side in the training set but to the right-hand side in the testing set), then the face recognition algorithm will have difficulty recognizing them. The face preprocessing steps that we just saw will help reduce these issues, but it certainly won't remove these factors, particularly the direction in which the face is looking, as it has a large effect on the position of all elements in the face.

The face training system is supported with a web application. The training phase requires more computational power and time, because each time when we are adding a new user to the face database we have to train the entire face database. This will take several minutes in an ARM platform. For example, to train a face database of 30 user, each having 30 face images Wandboard Quad took 20 minutes to complete. So we offloaded the training process to a separate PC, which act as a Web client. It communicates with the server (Wandboard) via a socket mechanism.

1. Face Database

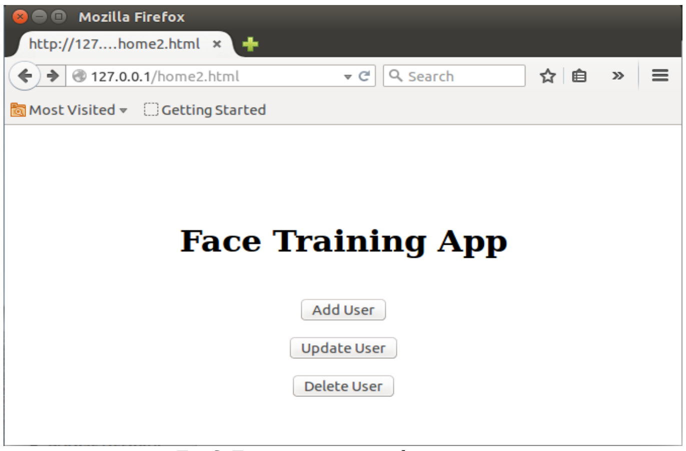
Face database is the collection of the pre-processed faces of all the authorized users for face training. It is stored in the device file system. The face database stores 30 face images of each user. The system access this database during the face training.

2. Web App

A web application is developed for face training which is intended to do three functions: adding new user to face database, update the database of existing user and delete the existing user entry.

The app is implemented using Lighttpd webserver running on admin PC. Only admin has access to the face training system. Fig 6 shows the home page of web app.

Fig 7. Shows the new user form to enter the user details. After getting user details, the system will start collecting face images of the person in front. After collecting required images, a message will be displayed in the web page and now the person can leave. The system then performs the training of entire face database. After training, generated data will be written to an xml file and send to the Wandboard server. Now onwards system could identify the new user along with previously trained users.



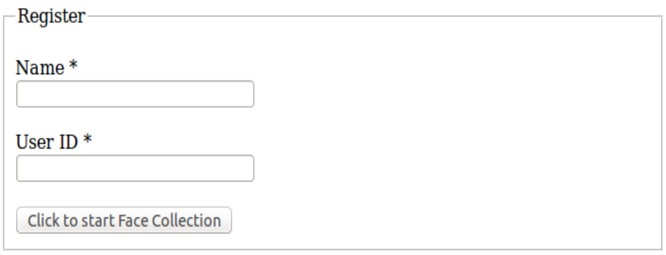
Face Training App

Add User

Update User

Delete User

Fig 6. Face training app – home page


Register

Name *

User ID *

Click to start Face Collection

Fig 7. Add User form

Similarly, we can update or delete the training data set of an existing user by selecting the appropriate field.

VII. TESTING AND RESULTS

The system is tested on several use cases and it succeeded in all cases. The system is able to identify multiple users approaching same time and could able to distinguish between trained and unknown users. Face Recognition accuracy of the system is evaluated by creating a face database with 30 users. Training and testing was done on different days with similar lighting conditions and we got a recognition accuracy of 89%.

VIII. CONCLUSION AND FUTURE SCOPE

Realized a reliable, real-time face recognition system on embedded platform. The current system gives a descent result under constrained environment. Similar performance can be achieved on more realistic scenarios with further research on this field. Also we can include some more techniques like 3-D face recognition, skin detection etc. for getting higher recognition rates.

ACKNOWLEDGEMENT

I sincerely acknowledge the immense contribution of Mr. K. M. Martin and Mr. Shiyas Shahabudeen for their guidance, encouragement and supervision during the period of this work.

REFERENCES

[1] http://opencv.org/
[2] Kenneth Dawson-Howe, “A Practical Introduction to Computer Vision with OpenCV”, 1st ed. Wiley, 2014.
[3] http://www.wandboard.org/
[4] P Viola and M. Jones, “Fast and robust classification using asymmetric adaboost and a detector cascade” in Advances in Neural Information Processing Systems (NIPS), MIT Press, pages 1311–1318, 2002.
[5] T. Ahonen, A. Hadid, and M. Pietikäinen, “Face recognition with local binary patterns” in Proc. 8th European Conference on Computer Vision (ECCV), pages 469–481, Prague, Czech Republic, 2004.
[6] P. Belhumeur, J. Hespanha, and D. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711--720, 1997.
[7] http://www.yoctoproject.org/docs/current/ref-manual/ref-manual.html
[8] Otavio Salvador, Daiane Angolini, “Embedded Linux Development with Yocto Project”, Kindle Edition, Packt Publishing, July 2014.
[9] http://www.pool.ntp.org/en/
[10] http://espeak.sourceforge.net/