Hardware Implementation of Facial Recognition on Android Platform

Jonathan Hoffpauir and Andy Nguyen
EE 4780 – Introduction to Computer Vision
May 4, 2012

I. OBJECTIVES

The goal of this project is to implement face recognition on the Android platform for use on mobile devices. The software should be able to take an image of a person through the built-in camera and recognize the individual based on previously stored samples. The implementation of facial recognition was initially planned to be performed with OpenCV and then integrated into the Android platform. After difficulties were experienced while attempting to integrate the two components, a MATLAB implementation was accomplished instead. The MATLAB implementation used a cropped and resized image on a computer and compared the image with those in a database. The face detection algorithm was successfully implemented on Android, which involved programming in Java. While the complete integration of both algorithms was not completed, each component successfully performed. Future work on this project would include implementing both face recognition and face detection on the Android platform, which could then be used as a used as a security feature to grant users access to personal files and applications, or even the initial security lock screen of the device. The following report discusses the methods and results of this face recognition project.

II. ANDROID PLATFORM

Development for an Android application requires downloading the Android SDK. Programming and implementation for this project was completed using the Eclipse IDE for Java. After some research was performed, it was discovered that OpenCV code can be executed on the Android platform, as well. OpenCV offers many features that facilitate both facial recognition and face detection. The implementation of this project involves devices running Android version 2.0, for which access to devices is readily available. The availability of such devices running this software version facilitated better testing and demonstration. One of the reasons that this implementation is desirable on the Android platform is the availability of a front facing camera on mobile Android devices. Images of faces can be easily taken and stored. Also, video feed offer samples of the user's face to provide a database of images to be compared. A user interface can be developed with the touchscreen of the device to allow simple user commands.

III. FACE DETECTION

One of the main issues to overcome with facial recognition, or object recognition in general, is getting consistent object sizes. The first solution to this problem involved a simple approach of building an Android app from the ground up that could capture photographs from the front facing camera. A user interface template element was placed over the camera preview of the Android application. This template appeared as a green oval that the user could roughly align to his or her face when taking the photograph, which would provide more consistent results when trying to recognize the image. After successfully completing this implementation, a more advanced method of face capture was attempted.

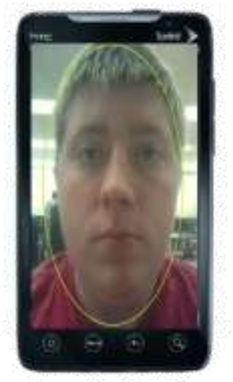

Figure 1: Image Capture Android Application

A facial detection algorithm, specifically the Haar cascade classifier, was used to detect faces on an Android application. The cascade classifier works in two phases. In the first phase, called the learning or training phase, the algorithm is supplied with a large number of images. These images are both positive examples (desirable objects to be detected) or negative examples. The algorithm runs a series of feature detecting filters over the examples and generates data based on the shape of the object. Once it has gathered this data, it can be used in the detection phase. In this phase a new object is introduced, and the same series of filters is passed over the object. Based on the values the filters generate, the algorithm can determine whether or not the object is present. This algorithm can be improved upon since the features of the desired object are known to be a face. As a result, hard-coded training is used, for which the algorithm already knows the desired features. This can reduce, or even in some cases such as this one, completely eliminate the need for the training phase. The Haar cascade classifier algorithm was implemented by modifying sample code provided by OpenCV. A working app was created that could detect the user's face and place a green box around it.

IV. FACE RECOGNITION

After successfully executing a face detection algorithm, an appropriately sized and scaled image of a face could be used for face recognition. The initial approach to solving the face recognition problem involved a common computer vision algorithm known as Eigenface. This algorithm can be easily implemented in OpenCV. The algorithm basically works by comparing the unknown face to a database of faces. After computing a general "difference" between the database faces and the unknown face, the closest matching face is chosen. Determining the difference between to different images can be a complicated task; however, with the use of Principal Components Analysis (PCA), the task can be simplified. Rather than comparing every single pixel of an image to that of another, all points and data can be projected onto a subspace. By using Principle Components Analysis, data can be reduced into a single point in multi-dimensional space. Therefore, the distance between two points in a PCA subspace can be measured to determine the overall difference of an image.

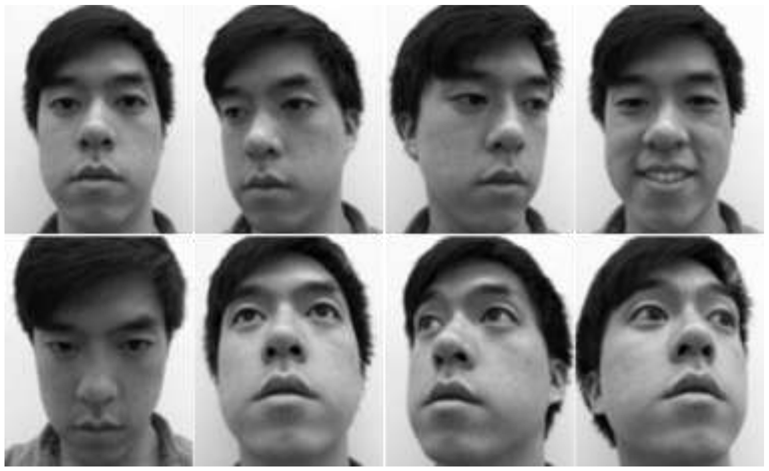

Figure 2: Sample Database Images

After several unsuccessful attempts at porting the Eigenface code in OpenCV to the Android platform, an alternative method was explored. A MATLAB implementation based on the Eigenface algorithm was performed. The implementation involved using a database containing over 400 faces, which included 40 individuals with 10 images per person. This database was obtained online from the "AT&T Database of Faces." In order to compare the unknown image to those in the database, a mean image of all database images was computed, and the values were subtracted from the unknown image. After calculating the eigenvectors of the images, the closest matching face could be obtained. Additional images were appended to the database to included personal individual faces.

V. Results

An Android app that captured face images was successfully created from the ground up. Furthermore, the OpenCV face detection algorithm was executed on Android, as well. After encountering difficulty integrating OpenCV code with the Android platform, the face recognition algorithm was implemented with MATLAB. The separate components of face recognition and face detection worked successfully independently; however, they were not both integrated together on the Android platform. Since there were difficulties implementing face recognition on an Android device, face recognition could not be used as a security feature for the device.

VI. Conclusion

In the future, it would be useful to improve on this project by successfully implementing the Eigenfaces algorithm on the Android platform. With this integration, a stand-alone application that can perform facial recognition can be created. As suggested by the audience during the presentation, it is also possible to send the captured images to a remote server to be analyzed. A response could then be received to determine whether or not the image was a match. This technique would move much of the processing off of the mobile device; however, it would require extra outside resources. Once a user's face could be successfully recognized on the Android device, the face recognition can implemented as a security mechanism for the device.

VII. References

Android Image
http://blog.galvintech.com/wp-content/uploads/2011/11/Android-Waving.jpg

AT&T Database of Faces
http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Cascade Classification
http://opencv.itseez.com/modules/objdetect/doc/cascade_classification.html

Face Recognition With Eigenface
http://www.cognotics.com/opencv/servo_2007_series/part_4/index.html

MATLAB Implementation
http://www.mathworks.com/matlabcentral/fileexchange/16760