Simple Face Recognition Implementation For Computer Authentication

Josh Easton, Tin-Yau Lo

Introduction

The topic of face recognition for video and complex real-world environments has garnered tremendous attention from governments for crime fighting as well as airport terrorism deterrence. However, there has been little attention towards consumer use of face recognition products. The aim of this project is to come up with a simple implementation for computer authentication to replace the popular pass-phrase based authentication on personal computers. The popularity of high-resolution cameras on market made the possibility of face recognition based computer authentication possible.

Humans can recognize face even when the matching image is distorted, such as a person wearing glasses, and humans can perform the task fairly easy. Understanding how humans decipher and do matching is an important research topic for medical and neural scientists.

Approach

Facial recognition is an easy task for a human to perform; it is nearly automatic, and requires little mental effort. A computer, on the other hand, has no innate ability to recognize a face, and must be programmed with an algorithm to do so. Even the best algorithms available today are not even close to perfect, and rely a lot on statistically probability.

There are a few algorithms that can be used, one of which is the Eigenfaces algorithm. It first must be trained by being given several images of the same face. These images are used to train the computer to recognize several features of a person's face. Since the face will not be in the exact same pose every time, the face in the picture must be centered and scaled. Generally the person's eyes are used to center the images. After this minor preprocessing, over a hundred signatures are taken of the face. In order to get a good set of signatures, the face needs to be completely exposed, without any form of obstruction.

After the computer has been trained to recognize a certain face, it can then look at any picture of a face, calculate a set of signatures on it, and compare it to every face it has been trained to recognize. It then will compute a set of probabilities for each trained face, and whichever is the most probable will be considered a match. If none of the faces appear to be very probable, then no match will be returned.

The Hidden Markov Models method works in much the same way. A set of signatures are taken of a face to train it, and are used to compare to a target face to find a match.

WORK PERFORMED

The first step in our project consisted of researching the various facial-recognition algorithms, and determining which would be the most fit for our application. There are many open source libraries and applications available, each with a different level of ease-of-use and completeness. Intel OpenCV and CSU Face Recognition engine are very featured-rich but they are overly bloated. We decide to use Eigenface engine by Konrad Darnok. As the engine itself was written in Java, we used Java to implement the video grabbing and the rest of the front-end. The result is a program that can be used in conjunction with a video camera. The capture program is shown at *figure #1*.

The "FaceRecognitionCap" program in figure #1 is written in Java which utilizes Quicktime Java library sequence grabber library, running on Mac OS X and Windows (with Quicktime installed) to take Snapshot into gray scale image. Therefore we simulate the authentication by taken a series of trained image using the firewire camera and in our

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

figure #1

case, the Apple iSight to capture a 320x240 image from sequence grabber (using Quicktime Java) and convert it into grayscale (with java.awt framework) and finally save it into JPEG format using Sun's JIMI image library.

Then the "FaceRecognition" Java program is to run the resultant test image against a trained library. The result is printed out in a text window. In a real system, this would be passed to a program that wanted to authenticate a user, such as a screen saver or login screen.