Security Implementation using Biometric

Shrimadhav U K

Guided by: **Dr. Vinod Pathari**

November 28, 2016

***Abstract*—The project aims to develop a biometric security system, which can protect the user's device from unauthorized or unauthenticated access. The idea is inspired from Microsoft Windows Hello and Google Now, which allows us to speak our mind and the machine does it, through the profound advancement in machine learning and artificial intelligence. This project aims to implement an application which can recognize the face and the voice of the user, and accordingly allow or deny access to the system.**

***Keywords*—CMUSphinx, OpenCV, Real time, biometric, Face Recognition, PCA, Eigen faces, Yale Face DataBase, Principle Component Analysis,**

I. Introduction

Biometric Security is gaining more and more attention recently. This project attempts to implement an application which can take the voice input from a microphone, face input from a camera, and verify the authenticity of the user accessing the system.

II. Motivation

Human beings have reached a stage where it is no longer convenient to type the password when they want to be authenticated. This was the basic motivation of this project, i.e., to replace the password input using a keyboard, and instead ask the user to smile in front of their personal computer, and talk interactively to it. Then that personal computer unlocks, if it recognizes the integrity of the user.
Currently, no fool-proof solution exists which attempts to do both these tasks. There exists individual solutions for each of these individual tasks. But, these solutions are proprietary and requires specific licenses to use the offered services.

III. Problem Statement

To design a security system for GNU/Linux operating system using biometric of the user, i.e., the face and the voice of the user, that would replace the traditional password input using a keyboard.

IV. Related Works

1) Google Now https://www.google.com/search/about/learn-more/now/
2) Microsoft Windows Hello https://support.microsoft.com/en-in/help/17215/windows-10-what-is-hello

V. High Level Design

1) Design a function which takes the user voice through the microphone, and the name of the user and returns True or False, accordingly.
2) Design a function which takes an image of the user, using the camera, and the name of the user and returns True or False, accordingly.
3) Finally, design a system which unifies the functions designed above. The system should be able:
   - to override the default login screen in a GNU/Linux system.
   - to ensure the integrity of the confidential details created using the above functions.

VI. Literature Survey

A. Background and Related Work

Much of the work in computer recognition of faces have been approached by characterizing a face by a set of geometric parameters and performing pattern recognition based on the parameters.
Kanade's [6] face identification system was the first system in which all steps of the recognition process were automated, using a top-down control strategy directed by a generic model of expected feature characterstics. His system calculated a set of facial parameters from a single face image and used a pattern classification technique to match the face from a known set. This approach was a statistical based approach, which depended primarily on local histogram analysis and absolute gray-scale values.

B. The EigenFace Approach

Much of the previous work on automated face recognition has ignored the issue of just what aspects of the face stimulus are important for identification. This suggested that an information theory approach of encoding and decoding face images may give insight into the information content of face images, emphasizing the significant local and global features. Such features may or may not be directly related to our intuitive notion of face features such as the eyes, nose, lips, and hair.
In the language of information theory, the relevant information in a face image should be extracted, encode it as efficiently as possible, and compare one face encoding with a database of models encoded similarly.
A simple approach to extracting the information contained in an image of a face is to somehow capture the variation

in a collection of face images, independent of any judgment of features, and use this information to encode and compare individual face images.
In mathematical terms, we wish to find the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images, treating an image as a vector in a very high dimensional space. The eigenvectors are ordered, each one accounting for a different amount of the variation among the face images.
These eigenvectors can be thought of as a set of features that together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that the eigenvector is displayed as a sort of ghostly face, which is called an *eigenface*.
The approach to face recognition using the eigenface approach involves the following initiation operations:

1) Acquire an initial set of characterstic face images (the training set).
   This set should include a number of images for each person, with some variation in expression and in the lighting.
   (say **4** images of **10** people, so **M = 40**.)
2) Calculate the eigenfaces from the training set, keeping only the M images that correspond to the highest eigenvalues.
   These M images define the *face space*. As new faces are experienced, the eigenfaces can be updated or recalculated.
   a) Calculate the (M x M) matrix L, find it's eigenvectors and eigenvalues, and choose the M' eigenvectors with the highest associated eigenvalues.
      (Let M' = 10 in this example.)
   b) Combine the normalized training set of images (according to equation 1) to produce the (say, M' = 10) eigenfaces u_k.

$$u_l = \sum_{k=1}^{M} v_{lk}\Phi_k \qquad ,l = 1, 2, \ldots, M \quad (1)$$

   c) For each known individual, calculate the class vector Ω_k by averaging the eigenface pattern vectors Ω (from equation 2) calculated from the original images (four, in this example) of the individual.

$$\epsilon_k^2 = ||(\Omega - \Omega_k)||^2 \quad (2)$$

      Choose a threshold θ_ϵ that defines the maximum allowable distance from any face class, and a threshold θ_ϵ that defines the maximum allowable distance from face space. (according to equation 3)

$$\epsilon^2 = ||(\Phi - \Phi_f)||^2 \quad (3)$$

   d) For each new face image to be identified, calculate it's pattern vector Ω, the distance ϵ_i to each known class, and the distance ϵ to face space.
      If the minimum distance $\epsilon_k < \Theta_\epsilon$ and the distance $\epsilon < \Theta_\epsilon$, classify the input face as the individual associated with the class vector Ω_k.
      If the minimum distance $\epsilon_k > \Theta_\epsilon$ but distance $\epsilon < \Theta_\epsilon$, Then the image may be classified as unknown, and optionally used to begin a new face class.
3) If the new image is classified as a known individual, this image may be added to the original set of familiar face images, and the eigenfaces may be recalculated (steps 1-2). This gives the opportunity to modify the face space as the system encounters more instances of known faces.
4) Calculate the corresponding in M-dimensional weight space for each known individual, by projecting the face images onto the face space.

The above initialization operations can be performed from time to time whenever there is free excess computational capacity, available in the system.
Having initialized the system, the following steps are then used to recognize new face images:

1) Calculate a set of weights based on the input image and the M eigenfaces by projecting the input image onto each of the eigenfaces.
2) Determine if the image is a face (whether known or unknown) by checking to see if the image is sufficiently close to face space.
3) If it is a face, classify the weight pattern as either a known person or as unknown.
4) Update the eigenfaces and/or weight pattern.
5) If the same unknown face is seen several times, calculate its characterstic weight pattern and incorporate into the known faces.

¡Insert the image here ¿
In the prototype implemented, calculation of the eigenfaces is done as part of the training process.
The recognition currently takes about 30 milli seconds implemented in Python on an Intel Core i5, using face images of size 132 x 132.

VII. Work Plan for Next Semester

1) Develop an application which takes the voice input through the microphone, and train the application with multiple users.
2) Modify the above application to test that user's voice and return the status (Authorized or Not Authorized).
3) Learn how the GNU/Linux login screen works, and possible ways to overwrite it.
4) Implement the custom application as the new login screen in your GNU/Linux system.
5) Discover the ways to improve the application, and document the challenges faced during the implementation phase.

VIII. Conclusion

References

[1] Google Now, https://www.google.com/search/about/learn-more/now/

[2] Microsoft Windows Hello, https://support.microsoft.com/en-in/help/17215/windows-10-what-is-hello

[3] The CMU Audio Databases, http://www.speech.cs.cmu.edu/databases/

[4] UCSD Computer Vision, http://vision.ucsd.edu/content/yale-face-database

[5] Matthew Turk, Alex Pentland, Eigenfaces for Recognition

[6] Takeo Kanade, Computer recognition of human faces