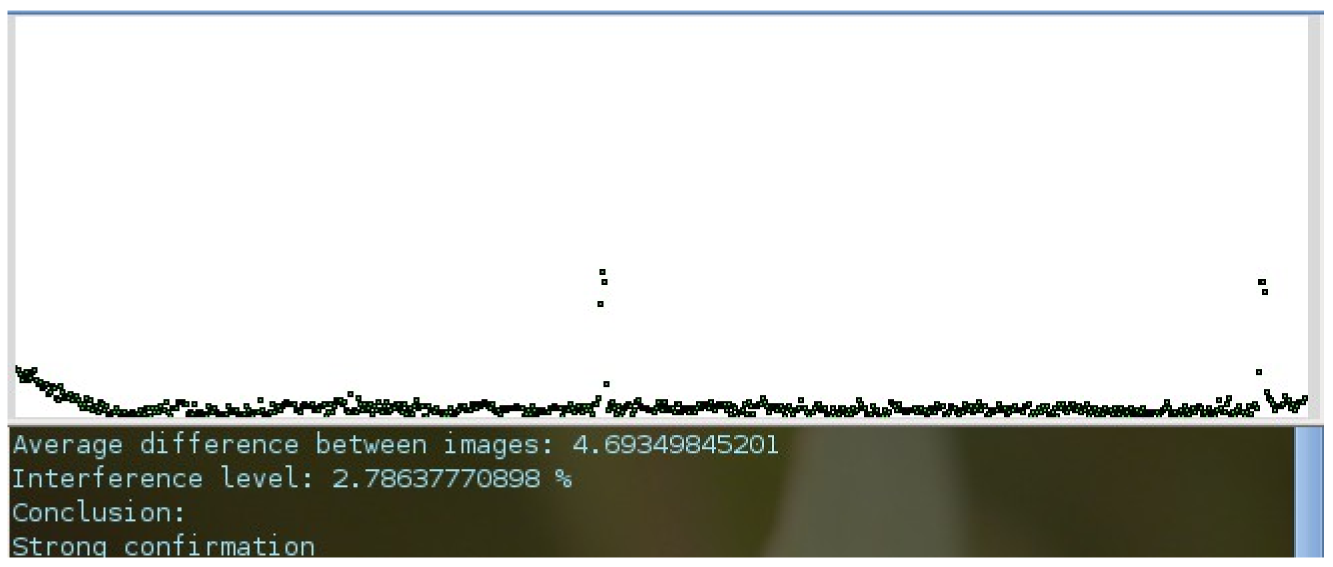
Biometric Security- Face Recognition

Kyle Ferris
Computer Systems Lab 2009-2010

Abstract

In the modern world, sensitive data or access to buildings can be protected by more than just a key or a password. Biometric data unique to every human can be used to allow or deny access. The purpose of this project is to be able to create a "key" for any person who wishes to use the program. An image of the client's face will be taken and used as the base biometric key. When the client wishes authorization, a new picture of their face will be taken and compared to the base image. The program should be able to recognize the client and authorize him or her, while denying access to those not recognized.



```
Average difference between images: 4.69349845201
Interference level: 2.78637770898 %
Conclusion:
Strong confirmation
```

Fig 1: sample program output

Algorithms

<u>Mean Pixel Area Comparison (MPAC):</u>
Given two grayscale images, it will in its internal processing split the image into different groups of pixels. It will then average the pixel intensities in those areas, and calculate the difference between average pixel intensity in equivalent areas in the two images.

<u>Connected Components Labeling:</u>
It will find groups of pixels of similiar intensity that are considered part of one component of the image, and then "label" them as one component. The output is a list of groups of connected pixels- the components.

Discussion

There are many different approaches to this type of problem. One of the most common is neural networks, but in this project I will be using Principal Component Analysis (PCA). PCA involves identifying the principal features of a face (aka, eyes, mouth, jaw structure) and comparing these features for different images. The algorithms listed to the left make up this PCA method.

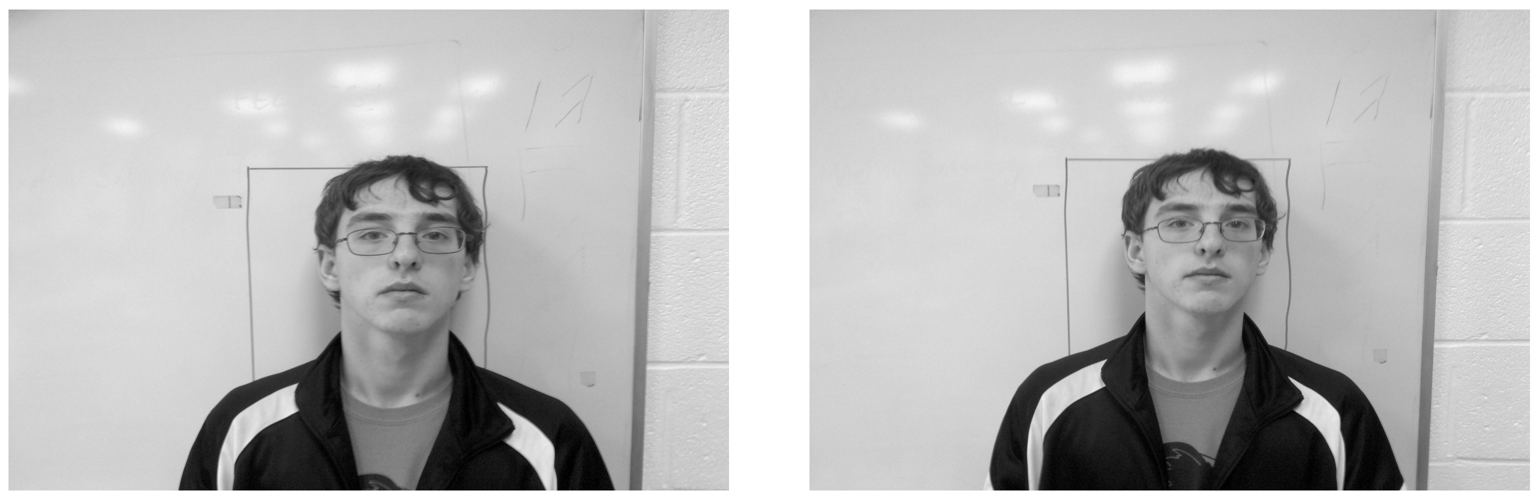


Fig 2: Sample Comparison images

Results and Conclusions

I ran into problems incorporating the CCL method. While I coded it correctly, the recursion process used presented problems. The code involves recursively going over almost every pixel in an image. This means that there is a new level of recursion for each pixel. However, python has a hard cap of 100 recursions and the images I am using are hundred of thousands of pixels. Therefore this could not be practically incorporated. It was still worthwhile as I used all the techniques involved in CCL, even if the finish was not quite there.
However, the program overall works well. Incorporating CCL successfully would have increased the reliability of the results. However, running the simple function without CCL still presents results comparable to the expectations I laid out at the beginning. Therefore, this project can be considered a success.