BUSINESS COMPUTING PROJECT

FIND THE MEMBER

Pablo Garcia Arribas (714120290033)

Soohyun Lee (714120290013)

Begaiym Omurkanova (516120990004)

Hannah Rowe (516120990001)

Dastan Yusupov (516120990003)

Shanghai Jiao Tong University

INTRODUCTION

Our project 'Find the Member' was an extremely enjoyable project to have worked on together. Having used python language and the knowledge we gained from in-class lectures, we chose to incorporate machine learning algorithms and a cool idea to produce this project!

HOW IT ALL BEGAN

This group consists of five fun loving members who were thrilled at the idea of fulfilling this project. We first of all planned to gain a broad idea of the problem we aimed and then started writing the code. At first, it was extremely difficult, however, with constant research, often meeting up with one another and fellow team member encouragement we finally completed it.

CHALLENGES COMPLETING THE PROJECT

- While running the photo simulation program, we encountered several issues with our computers not being able to handle the data processors. So, we used the only team member with an Apple Macbook to run the program because it successfully ran there.

- Although a very popular idea among many people, writing the code deemed much harder than expected. There was a lot of information about the theory surrounding this program but, code-wise, we had to be very diligent in order to complete it.

<u>HOW DID WE COLLECT THE DATA TO TRAIN?</u>

To use the training method K-Neighbors we needed to create a data sample. Unlike other cases, data could not be found in the internet so we decided to create a program to create that data for us. This program would basically take 100 frames of each of us out of a video. Then all the pictures would be collected in one folder and then the second program will compare each face with the most 15 likeable of the ones we had previously taken.

<u>PYTHON IMAGING LIBRARY:</u>

The Python Imaging Library allowed us to develop this project as it increases the image processing capabilities of a Python interpreter.

This library provides extensive file format support, an efficient internal representation, and quite powerful image processing capabilities.

The core image library is designed for speedy access to data stored in a few basic pixel formats. It provides solid foundation for general image processing. Now, let's examine some possible uses of this library:

Image Archives

The Python Imaging Library is ideal for image archival and batch processing applications. You can use the library to do many things such as the creation of thumbnails, converting between file formats, the printing of images, etc.

Additionally, the current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.

Image Display

The current release includes "Tk PhotoImage" and "BitmapImage" interfaces, as well as a Windows DIB interface that can be used with PythonWin. For X and Mac displays, you can use Jack Jansen's imaging library.

Also for debugging, there is also a show method in the Unix version which calls xv to display the image.

Image Processing

The library contains some basic image processing functionality, including point operations, filtering with a set of built-in convolution kernels, and colour space conversions.

This library also supports image resizing, rotation and arbitrary transformations.

There is a histogram method that allows you to extract some statistics out of an image. This, for example, can be used for automatic contrast enhancement, and for global statistical analysis.

MULTI SCALE

This function normally detects objects but in the case of our program, it is detecting faces because we use it on face classifier. The function uses it on the grey scale image so we changed the colour of our photos to grey.

OPENCV AND CASCADE CLASSIFIER

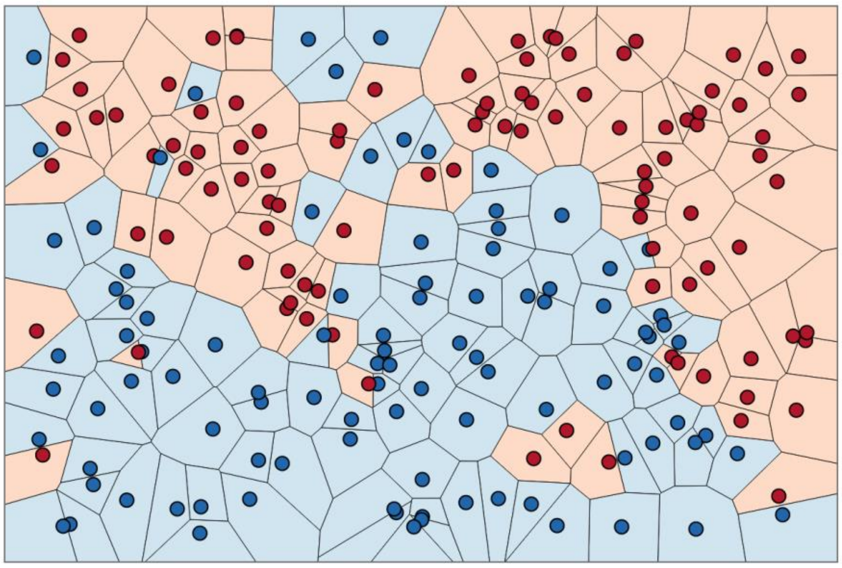
OpenCV (Open Source Computer Vision Library) was primarily designed for computational efficiency with a strong focus on real-time applications. The library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of different computation platforms. Before OpenCV and Cascade Classifier existed, detecting faces was definitely not an easy job provided the fact that there are billions of people in this world and everyone has a different face in structural patterns. However, all faces do have some of the same features such as one nose, two ears , two eyes and a pair of lips. Cascade Classifier was downloaded already containing data of faces. So, since it is already trained with many faces, we do not need to train it again. We just call on this classifier in the program and it will put boxes around the faces in the video.

FACE DETECTION OVERVIEW

Face detection is a computer technology that determines the locations and sizes of human faces in digital images. It detects facial features and ignores everything else, such as buildings, trees and even bodies! Face detection can be regarded as a more general idea or concept of face localization. The difference here, is that in face localization, the task is to find the locations and sizes of a known number of faces (this is usually one).

K-NEAREST NEIGHBORS OVERVIEW

The k-nearest neighbours algorithm is based around the simple idea of predicting unknown values by matching them with the most similar known values.



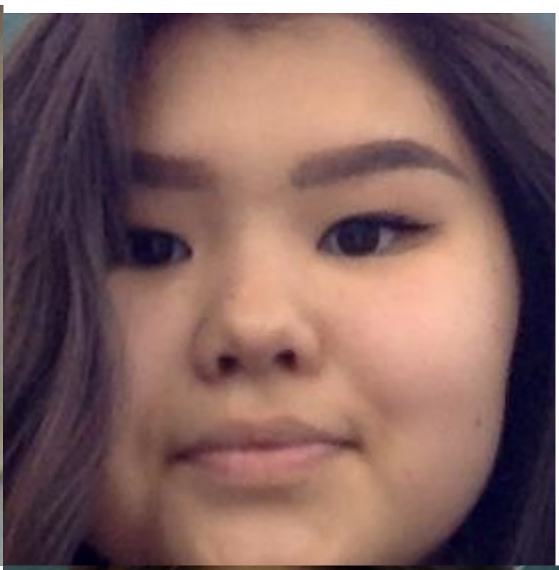
MIN NEIGHBOURS OVERVIEW

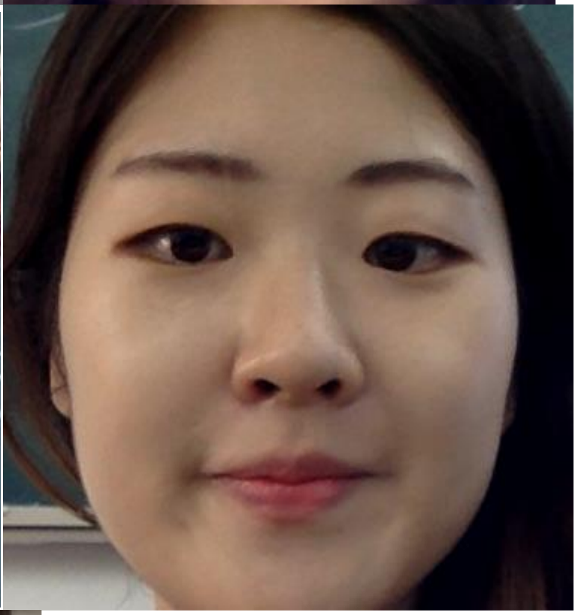
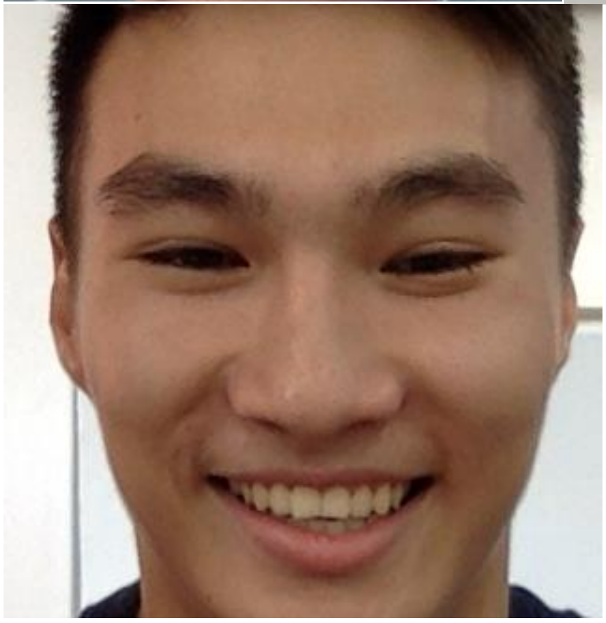
This is a parameter which we used to specify how many neighbors each candidate's rectangle should have to retain it. This will affect the quality of the detected faces meaning that there will be higher value results in less detections but with higher quality. In our code we use five of these because there are five member

ROI-COLOUR

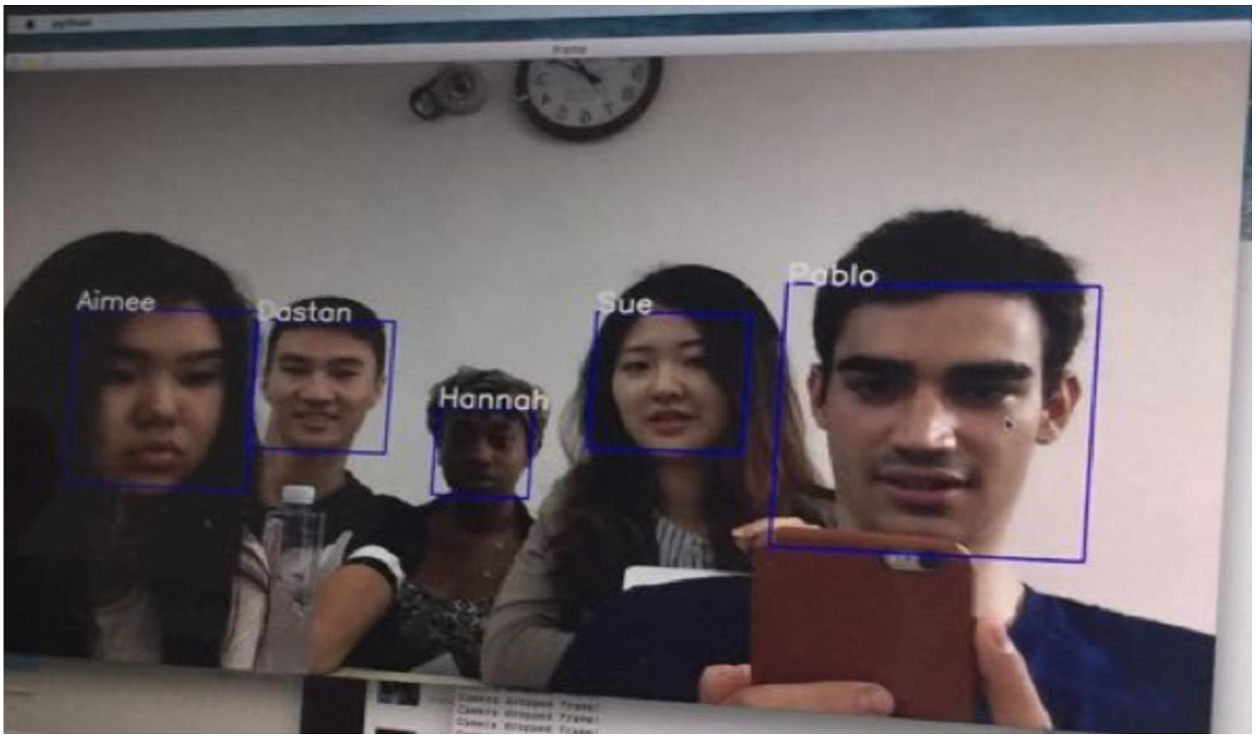
For eye detection in images using Roi-colour, face detection over the image is first performed until the face is found. Then, it searches within the face region for the eyes. This approach improves accuracy (because eyes are always on faces) and performance (because searching for a small area is more effective).

DATA SAMPLES-BEFORE



FINAL RESULT-AFTER



CONCLUSION

Besides working on a fun project, we learnt many things about Python computer language the most important of them all, we learnt how to work with one another efficiently. We mastered different communication skills while under pressure for our approaching exams. Also, we managed to divide the work equally. Finally, this project could not have come about without the effort and diligence of everyone on the team.