4/23/2017

Face Recognition

Sam Milward and Joe Griffin

N0690641, N0680192

Contents

[Introduction 2](#_Toc480715188)

[How to use the program 3](#_Toc480715189)

[The GUI 4](#_Toc480715190)

[Creating a Profile 4](#_Toc480715191)

[Finding a Profile 4](#_Toc480715192)

[Edit Profile 4](#_Toc480715193)

[Delete Profile 4](#_Toc480715194)

[Functions within the program 5](#_Toc480715195)

[Flow Chart 5](#_Toc480715196)

[Testing 5](#_Toc480715197)

[Used Features of C++ 6](#_Toc480715198)

[Referenced Material 6](#_Toc480715199)

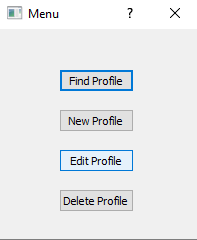
# Introduction

The following project has been created by Joe Griffin (N0680192) and Sam Milward (N0690641). We have created this program to create, store, delete and edit profiles of many users. These profiles will contain an individual’s information such a Name and Course Title, as well as a selection of images that will be used, to detect and recognise a person’s face to load a profile. The face recognition will be the main functionality of this program and has been achieved by using external libraries.

# How to use the program

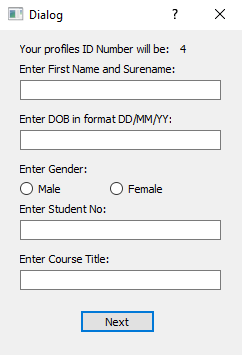
## The GUI

The GUI for this program has been made using an external library called Qt. The Qt library allowed me to create forms with widgets on them to navigate and input/output information to/from the user. The designer allowed me to easier link events, to functions within the program.



This is the menu form that I have used to navigate through the main functions of the program. Each button opens another form where a process of the project can be completed.

## Creating a Profile

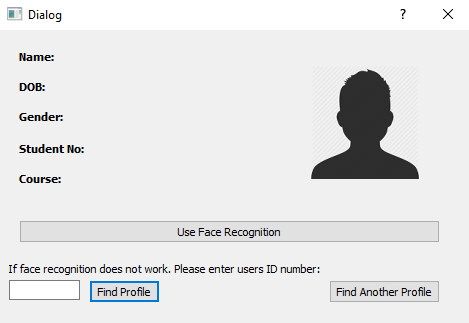


This form is used making a new profile on the system. The form consists of multiple text boxes, radio buttons, and command buttons for the user input. Output is displayed using labels and QMessageBoxes.

Clicking the next button on this form calls the function void CreateProfile::on\_cmdNext\_clicked() This function takes all of the information and stores, then loads the function to take 10 photos of the user, and then saves them in a newly created folder.

The Event of on\_cmdNext\_clicked() (When the next button is clicked) a selection of code is executed. This code extracts the information that has been entered the form, appends this information to the Profiles.txt and then closes it.

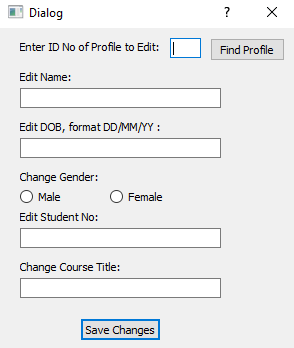
## Finding a Profile



This form is used to display information of a profile that has been found. A profile can be detected using two ways. A profile can either be loaded, from a prediction that has been made using the face recogniser. Or you can use a text box at the button of the file to input an ID number (Primary Key) to retrieve the information.

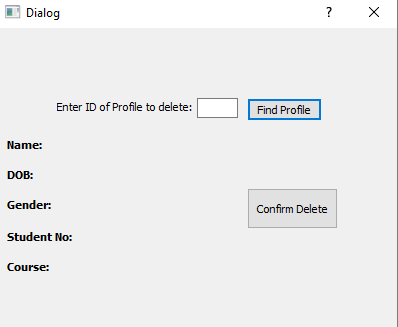
The output is displayed on the labels on the form so that the user can read the displayed information that is given to them.

## Edit Profile



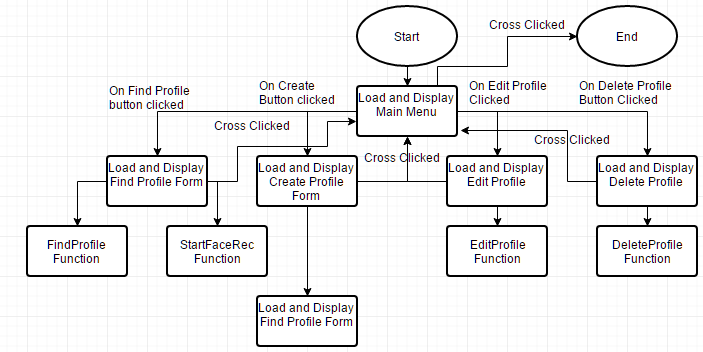
This form is for the user to edit any information about a profile. The user first needs to enter the ID number of the profile they wish to edit. This will then load up the information of the original profile in to text boxes. These can then be edited, and when the save button is clicked. The new text line will be entered in to the profile.txt in place of the original.

## Delete Profile



The form is designed to allow a used to remove a profile from the system. The used needs to input an ID number they wish to remove. They can click the Find profile button to load the information first to verify this is the correct profile to remove. Then the Delete profile button is used to remove that profile from the profiles.txt

# Flow Chart



# Functions within the program

**Find Profile**

FindProfile::FindProfile(QWidget \*parent) : //Constructor

This function is the constructor for the FindProfile form. It sets up the ui form so that ui can be referred to access all the widgets on the form. It also sets the Qlabel where the images are displayed, to the placeholder logo.

void FindProfile::on\_cmdFindProfile\_clicked()

This function takes the value of the entered ID number to find, searches through the textline line by line, if the desired ID matches the ID on the textline, then the textline is split using the delimintator (\*). Then finds the correct photo path for the profile and displays it on the label. Then outputs all the profiles information on to the form.

void FindProfile::on\_cmdUseFaceRec\_clicked()

This function launches the facial recognition section of the program by calling the container function RunRecogniser, and outputs the predicted ID to the find profile function which outputs the found profiles data.

void FindProfile::on\_cmdFindAnother\_clicked()

Clears the form of all profile information. Then displays the logo onto the label.

int CallVideoRecogniser(vector<Mat>& imagepaths, vector<int>& IDLabels, vector<String>& ImageFilePath)

This initialises the camera loop and trains a FisherFaces recogniser against the currently held profile images and associated labels, this then is used as a comparative base for the current video streams’ frames. It then finds what it believes to be the most likely match, and outputs the current image into the video stream with a box highlighting where it found a face and its prediction.

int RunRecogniser(vector<Mat>& Images, vector<int>& IDs, vector<String>& FilePaths) {

**Create Profile**

CreateProfile::CreateProfile(QWidget \*parent)

Calls the FindNewID function and displays the ID Number onto the label at the top of the form.

void CreateProfile::on\_cmdNext\_clicked()

Extracts all the information that has been entered into the text boxes and radio boxes. Converts the information to string, then writes all the information to a textfile, separating each piece of data with the delimiter (\*), Then calls the function UpdateCSV to write the new photo paths. Loads the camera loop to take ten photos of the user.

int FindNewIDNum()

Goes to the last line in Profiles.txt, takes the ID number of the textline, converts it to int, then increments the value by one, so it returns an unused ID.

int TakePhoto(int ID, vector<Mat>& imagepaths, vector<int>& IDLabels, vector<String>& ImageFilePath) {

Takes the vectors used to store data and imagefilepath and uses them as a filepath to save the amount of images we have decided is enough for the trainer.

**Delete Profile**

void DeleteProf::on\_cmdFindID\_clicked()

This function takes the value of the entered ID number to find, searches through the textline line by line, if the desired ID matches the ID on the textline, then the textline is split using the delimintator (\*). Outputs the text to the appropriate labels.

void DeleteProf::on\_cmdDelete\_clicked()

Takes the ID Number that is entered on form, opens profile.txt to read from, and opens a new file called NewProfiles.txt to write to. Loops through each line in the profiles.txt, takes the ID number, if the ID from the textline does not equal the ID number to delete then write the textline to the new profiles text file. If it does match then miss out the writing to a file. Once the loop has finished, close both files. Delete the original file, and rename the new file, to the same name as the original.

**Edit Profile**

void EditProfile::on\_cmdFindID\_clicked()

At first resets the whole form to blank. Then takes the value of the entered ID number to find, searches through the textline line by line, if the desired ID matches the ID on the textline, then the textline is split using the delimintator (\*). Outputs the text to the appropriate textboxes and selects the correct radio button for gender. If profile not in profiles.txt then sends error message.

void EditProfile::on\_cmdSave\_clicked()

Reads the entered text in the form, then creates a new txt file to store all the original profile data, apart from the line to be edited. Then adds in the newly edited line in the correct place in the textfile. Closes both files, then deletes the original file, and renames the new file, to the same name as the original.

**Video.cpp**

void CreateCSV(int ID)

This creates a csv file using the passed ID to create the set amount relative file paths for the pictures as well as the associated ID label.

void UpdateCSV(int ID)

This updates the csv file using the passed ID to create the set amount relative file paths for the pictures as well as the associated ID label.

void CreateRootFolder()

This creates the holder folder for the image folders, in our hierarchical folder file system. Uses windows own API to create the folder as we did not want to include another library that may have cause instabilities

void CreateFolder(int ID)

 This creates the holder folder for the images. Uses windows own API to create the folder as we did not want to include another library that may have cause instabilities

string ReadProfile(int id)

Pulls data from the created class that are held as pointers in a vector after they are dynamically created and saved to a .txt file.

fstream& GotoLine(fstream& file, int ID)

This takes an input file and locates the passed IDs file locations. Used to locate lines in csv file and and profile text file.

void ReadCSVDir(vector<Mat>& images, vector<int>& IDLabels, vector<String>& ImageFilePath)

This reads the directory of image paths, passes them into a vector as well as loads the images into a Mat vector which stores the actually loaded images. It uses the file paths to find the ID labels which it stores in another vector

vector<String> FindSavingPicDir(vector<String>& ImageFilePath, int ID)

This takes the vector which contains every file path and isolates the ones belonging to the ID that was passed and returns them in a String vector which can then be passed on to other functions such as the  TakePhoto function.

# Testing Plan

The primary function of the project that has been tested is the creation of a new profile, that then can be recognised via the built face reorganiser. The other functions such as editing, deleting profiles also work by string and text file manipulation. This project has been tested by multiple users creating their own profiles and then recognising their faces and searching the text files for the correct data.

The project is tested by opening the form CreateProfile. The testers then entered all their data, such as name, DOB etc. Once the data has been entered, the user then has their photos taken. These are stored in the correct file paths so that they can be accessed by the recogniser.

We then closed the Create Profile form and then opened the Find Profile form. Then used the button to access the loading of the camera and the face recogniser. The recogniser displays it prediction, and then when ESC is clicked, the camera is closed and the appropriate profile information and image is displayed on the Find Profile form.